Wybrane zagadnienia projektowania systemów informatyki przemysłowej

Piotr Gaj

SPIS TREŚCI

CONTENTS

PRZEDMOWA

Niniejsza monografia dedykowana jest osobom zainteresowanym tematyką projektowania systemów informatycznych działających na potrzeby automatyzacji procesów przemysłowych, w szczególności inżynierom informatykom i automatykom, a także studentom specjalności informatyczne systemy przemysłowe, informatyka przemysłowa oraz innych pokrewnych. Autor ma również nadzieję, że niejeden inżynier innych specjalności, zaangażowany w tworzenie rozwiązań informatycznych w tej dziedzinie, znajdzie w niniejszej monografii interesujące go treści.

W zamyśle książka ma charakter monograficzny i dotyczy systemowych aspektów wykorzystania urządzeń i sieci komputerowych. Nie dotyczy konkretnych rozwiązań i produktów komercyjnych oraz języków programowania. Tym samym nie stanowi instrukcji obsługi czy formy dokumentacji, a autor nie faworyzuje wybranych idei, pozostawiając wybór i ocenę czytelnikowi. Książka ma na celu usystematyzowanie zagadnień związanych z tworzeniem systemów komputerowych na styku z układami automatyki. Przedstawiono w niej opis standardowych pojęć i modeli, typowych zjawisk i problemów, jak i nietypowe i oryginalne punkty widzenia, które mogą pobudzić kreatywność czytelnika.

Monografia zawiera 7 rozdziałów, ale zawartość merytoryczna jest podzielona na cztery części. W pierwszej omówione są podstawy, stanowiące tło pojęciowe dla dalszych rozważań, w części drugiej modele systemów, w trzeciej typowe komponenty, a w czwartej zagadnienia bezpieczeństwa. Zatem, kompozycja monografii jest zbudowana tak, aby na wstępie zapoznać czytelnika z używanymi terminami, następnie zaprezentować, jak systemy mogą być konstruowane, następnie z czego, a na koniec, na co zwracać uwagę względem ich bezpieczeństwa. Całość jest wsparta kilkoma przykładami rzeczywistych rozwiązań oraz bibliografią zawierającą zarówno pozycje książkowe i artykuły, jak i dokumentacje oraz raporty, a także źródła online.

Poszczególne rozdziały dotyczą:

Rozdział 1. pojęć podstawowych związanych z technikami i technologiami[1] informatycznymi, z technikami i technologiami przemysłowymi i środowiskiem pracy.

Rozdział 2. modeli w ujęciu klasycznym, zagadnienia rozproszenia, definicji abstrakcji, a także ukazuje podejścia zgodne z normami.

Rozdział 3. opisu podstawowych rodzajów elementów stosowanych do budowy systemów z podziałem na urządzenia, oprogramowanie, infrastrukturę komunikacyjną i integrację.

Rozdział 4. aspektów bezpieczeństwa projektowania systemów, zarówno w ujęciu bezpieczeństwa funkcjonalnego, jak i bezpieczeństwa zasobów.

Rozdział 5. podsumowania poruszanych zagadnień.

Rozdział 6. prezentacji źródeł bibliograficznych, na które składają się pozycje literatury światowej, dokumentacje i raporty firmowe, standardy, źródła webowe oraz inne źródła istotne dla prezentowanych treści.

Rozdział 7. uzupełnienia przedstawianych zagadnień o przykłady. Rozdział ten zawiera również dodatkowe spisy oraz indeks istotnych pojęć.

Autor książki jest pracownikiem Zakładu Urządzeń Informatyki Instytutu Informatyki Wydziału Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Ma ponad-dwudziestoletnie doświadczenie zarówno dydaktyczne i badawcze, jak również aplikacyjne. Specjalizuje się w projektowaniu informatycznych systemów przemysłowych, komputerowych sieciach przemysłowych i szeroko pojętym oprogramowaniu aplikacyjnym.

Jest autorem wielu publikacji naukowych oraz redaktorem, współredaktorem, redaktorem pomocniczym (ang. associate editor), jak również recenzentem w polskich i zagranicznych czasopismach, a także organizatorem i recenzentem na konferencjach krajowych i międzynarodowych. Autor jest członkiem IEEE (SM'13) oraz Industrial Electronics Society.

Autor dziękuje za istotny wkład w powstanie tej książki prof. dr. hab. inż. Andrzejowi Kwietniowi, pracownikom Zakładu Urządzeń Informatyki, pracownikom firmy PPHW Proloc Sp. z o. o. oraz recenzentom.

[1] Warto rozróżnić te dwa pojęcia. Słownik wyrazów obcych podaje, że grecki wyraz *techne* oznacza „sztukę, rzemiosło" i oznacza „ogół środków i czynności wchodzących w zakres działalności ludzkiej, związanej z wytwarzaniem dóbr materialnych". Natomiast termin „technologia" pochodzi od greckich słów *techne* i *logos* (zbiór, rozum i in.) i oznacza „metodę przetwarzania dóbr materialnych w dobra użyteczne", w tym także w wiedzę o tym procesie. Zatem technika dotyczy ogółu środków i czynności, a technologia metod lub wiedzy. [M80].

1. INFORMATYCZNE SYSTEMY PRZEMYSŁOWE – WPROWADZENIE

Układy mające styczność z procesem przemysłowym kojarzą się przede wszystkim z systemami automatyki przemysłowej (układy automatyki, układy automatyzacji, ang. automation systems) [332]. Z jednej strony, teoria i opisanie takich systemów wiąże się z obszernym zakresem wiedzy określanej przez dyscyplinę naukową, jaką jest automatyka [108]. Z drugiej strony istnieją komputerowe systemy informatyki, których wiedzę definiuje dyscyplina nauki, jaką jest informatyka [377], [132]. Należy zauważyć, że współczesne systemy automatyki tworzone są za pomocą technik komputerowych. Innymi słowy, jedną z wielu dzisiejszych dziedzin zastosowań systemów informatycznych są różnego rodzaju układy obsługujące procesy przemysłowe. Konstruowanie takich układów w oderwaniu od zagadnień, jakimi zajmuje się informatyka, jest błędne i prowadzi do rozwiązań nieciekawych z punktu widzenia informatyki lub niepoprawnych z punktu widzenia obsługiwanego procesu. Można na to spojrzeć również z innej strony. Wymagania, jakie generuje współczesny przemysł, skutkują koniecznością umieszczania systemów informatycznych zarówno w warstwie biznesowej, jak i procesowej procesów produkcyjnych. Rezultat finalny działania takich układów względem różnego rodzaju miar (np. cena, jakość, nowatorstwo, czas, koszty, wydajność, wpływ na środowisko itp.) określony jest przez efekt synergiczny wielu podprocesów związanych z różnymi dziedzinami nauki i techniki. Bez systemów informatycznych byłby on niemożliwy do osiągnięcia dla współczesnych wymagań rynku.

Odpowiedź na pytanie, czy informatyk powinien posiadać wiedzę o Informatycznych Systemach Przemysłowych (ISP[2] czy też jak kto woli SIP – Systemach Informatyki Przemysłowej, lub KSP – Komputerowych Systemach Przemysłowych), lub wręcz specjalizować się w tej dziedzinie, jest zatem prosta i oczywista: powinien posiadać. ISP są to systemy komputerowe, które pracują na rzecz właściwego funkcjonowania procesów produkcyjnych (ang. shop/floor level). W ogólnej definicji, a szczególnie z informatycznego punktu widzenia, odnosi się to zarówno do systemów pracujących blisko procesu technologicznego, jak i do systemów pracujących bliżej procesów biznesowych. Każdy element układu uczestniczy

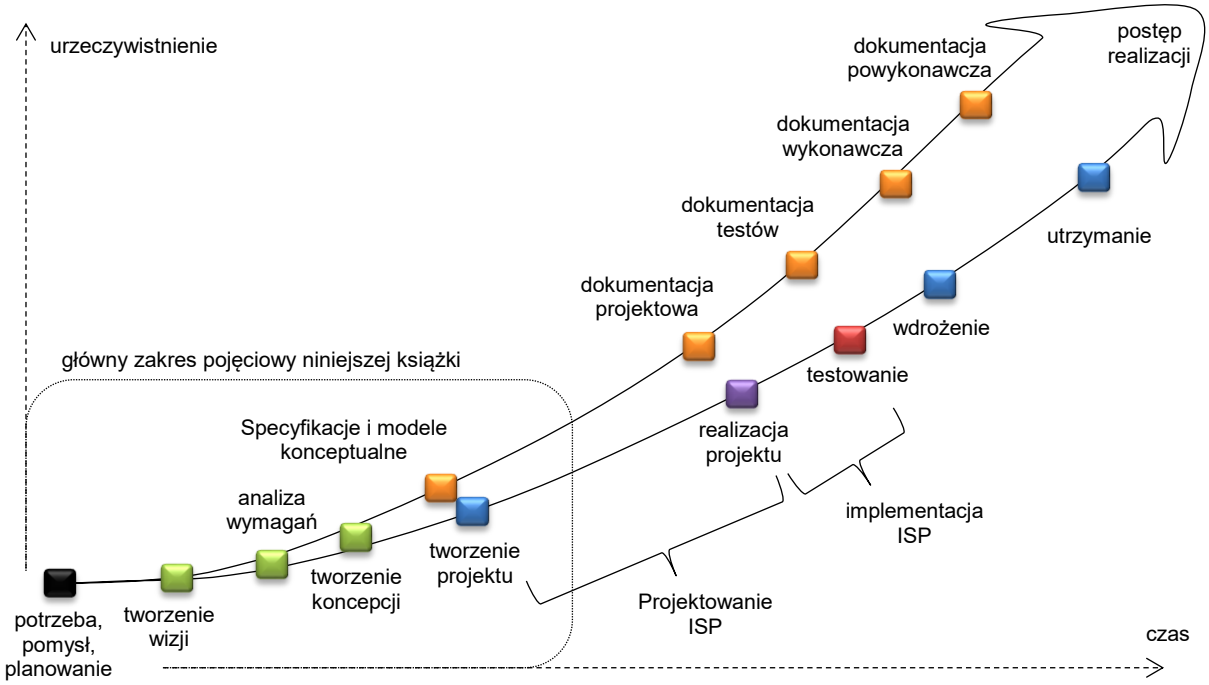
[2] Skrót nie jest unikalny. Występuje np. pojęcie ang. Interoperable System Project związane z rozszerzeniem modelu komunikacyjnego sieci Profibus, czy też tematycznie niezwiązane Instytut Spraw Publicznych, Internet Service Provider i inne.

w pracy na rzecz efektu końcowego, jakim jest produkt. Praca taka wymaga przetwarzania, kodowania, przesyłania oraz składowania informacji i w takich dziedzinach informatyk jest niezbędny. Jego działalność nie może się jednak ograniczać tylko i wyłącznie do działań programistycznych i tworzenia kodu według jakiejś specyfikacji, stanowiącej rodzaj czarnej skrzynki, oderwanej od rzeczywistości. Tworzenie ISP nie sprowadza się tylko do algorytmizacji procesu i programowania. Konstrukcja ISP wynika z formalnych i nieformalnych definicji struktur, funkcji i współzależności związanych z obsługą procesu technologicznego z jednej strony oraz z wiedzy płynącej z różnych dziedzin informatyki z drugiej. Nieodzowna jest zatem szeroka wiedza informatyczna, umożliwiająca przełożenie wspomnianych definicji na poprawnie działający i współdziałający system informatyczny, nierzadko złożony, składający się z wielu rozproszonych urządzeń, programów i sieci. Dodatkowo potrzebna jest ogólna wiedza dziedzinowa, związana ze specyfiką systemów automatyki i technologią procesów przemysłowych.

Proces tworzenia ISP składa się z wielu etapów zazębiających się i wymagających współpracy. Punktem wyjścia jest ogólna koncepcja i analiza wymagań dokonywana na podstawie specyfikacji. Następnie realizowany jest proces projektowania, którego efektem nie jest działający system a dokumentacja, według której zespół integratorów jest w stanie taki system urzeczywistnić. Wykonanie dobrego projektu jest kluczowe dla rozwiązań technicznych, w tym informatycznych, a w szczególności dziedziny ISP. Nierzadko projektowanie zajmuje więcej czasu i wymaga więcej wiedzy, doświadczenia oraz wysiłku umysłowego niż sama realizacja. Typowe etapy realizacji ISP wraz z efektami kolejnych działań zostały przedstawione na rysunku 1. Zwykle postęp w tworzeniu następuje od pojęć ogólnych i abstrakcyjnych, przez projekty i prototypy do realnych rozwiązań urzeczywistnianych w sprzęcie i oprogramowaniu. Niniejsza monografia zawiera omówienie pojęć, od których zaczyna się tworzenie rozwiązań i których dobre zrozumienie jest wymogiem sukcesu kolejnych etapów.

W tworzeniu takich rozwiązań informatyk nie działa sam. Do realizacji ISP, tak jak w przypadku wielu innych systemów, potrzebni są fachowcy z różnych dziedzin, a efekt końcowy jest zawsze efektem pracy zespołu a nie jednostki. Żaden inżynier informatyk czy automatyk, niewykształcony w zakresie procesów technologicznych danej klasy, nie będzie w stanie stworzyć poprawnego systemu. Stąd zachodzi potrzeba współdziałania technologów, elektryków, mechaników, automatyków, informatyków i innych osób o wymaganej w danym przypadku specjalizacji i stworzenia z nich zespołu realizującego daną aplikację. Zatem, na ile informatyk musi wnikać w dziedziny, o których wie niewiele? Wiedza interdyscyplinarna jest niezbędna. Wskazane jest, aby każdy członek zespołu poznał podstawy funkcjonowania procesu i jego obsługi w sposób ogólny lub w potrzebnej części. Jest to konieczne do rozu-

mienia rozwiązywanych problemów w kontekście na tyle szerokim, aby możliwa była współpraca między członkami zespołu, i aby współpraca ta była kreatywna.



Rys. 1. Ilustracja procesu tworzenia ISP
Fig. 1. Illustration of the ISP creation process

Z punktu widzenia informatyka może się rodzić wątpliwość dotycząca kwestii istnienia jakiś specyficzności ISP względem systemów informatycznych pracujących w innych obszarach, jak np. administracja czy rozrywka. Można by się obawiać, że te specyficzności są na tyle duże, że informatyk niewiele może zdziałać w dziedzinie ISP. Obecnie tak nie jest. Dużych różnic w zastosowaniu wiedzy informatycznej właściwie nie ma. Podstawy technologiczne zarówno w dziedzinie konstrukcji sprzętu, jak i oprogramowania są spójne. Specyfika dotyczy implementacji tych podstaw w kontekście dziedziny zastosowań, głównie interfejsów wejścia/wyjścia i silnej zależności działania ISP od czasu. Interfejsy elementów ISP są różnorakie. Można tu znaleźć wszelkie powszechnie rozpoznawalne interfejsy komunikacyjne i interfejsy nietypowe. Nietypowość ta wynika z charakterystyki ich pracy względem czasu oraz z ich dostosowania do urządzeń procesowych. W dziedzinie interfejsów użytkownika trudno doszukać się istotnych różnic względem klasycznych interfejsów GUI (interfejs graficzny użytkownika, ang. Graphic User Interface) i aplikacji IT (ang. Information Technology). Zatem oprócz dobrze znanych technik wspomagających interakcję komputerów i dialog z człowiekiem (np. sieć, monitor, klawiatura itp.) wykorzystuje się techniki i technologie specjalizowane, wspomagające dialog z układami automatyki (np. sieci polowe, panele, układy wejścia/wyjścia, odpowiednie algorytmy i metody itp.). Podstawowe zadania ISP wyko-

nywane są na rzecz obsługi procesu produkcyjnego, a nie na rzecz interakcji z człowiekiem. Zatem, ogólnie przyjmuje się, że podmiotem, którego dotyczy działanie systemu, jest proces. Niemniej jednak, człowiek jest zawsze nadrzędnym użytkownikiem takiego systemu, określanym często jako nadzorca (ang. supervisor).

Dialog z procesem prowadzony jest przez urządzenia mające bezpośredni z nim kontakt i nazywa się je aparaturą kontrolno-pomiarową (AKP) i aparaturą automatyki (AKPiA), (p. strona 19). Oddziaływanie z człowiekiem ma z reguły charakter nadzorczy i prowadzone jest przez klasyczne interfejsy użytkownika implementowane na klasycznych komputerach, co najwyżej w wykonaniu dostosowanym do warunków innych niż biurowe. Obecnie wiele urządzeń AKP jest integrowanych z urządzeniami komputerowymi [379]. Nie zmienia to jednak funkcjonalności zintegrowanego urządzenia ani od strony informatycznej, ani od strony automatyki. Zatem jeśli rozważać specyficzności ISP, to trzeba je odnieść do odpowiedniej konstrukcji systemu, wykorzystania technologii i implementacji algorytmów.

Współczesny informatyk profesjonalista powinien dostrzegać wszechobecność komputerów, nawet tych prostych, jednoukładowych wbudowywanych w wiele urządzeń nie tylko stricte przemysłowych, ale także otaczających nas urządzeń powszechnego użytku. Dużo ostatnio mówi się o internecie rzeczy (IoT – ang. Internet of Things) [148], [22], [156], systemach otaczających i sieciach sensorowych (ang. ambient, WSN – Wireless Sensor Networks) [143], [398], [151], [398], technologiach chmurowych (ang. cloud computing & storage) [159], [65], [4], wirtualizacji [407], [130], [35] oraz systemach cybernetyczno-fizycznych (CPS – ang. Cyber-Physical Systems) [111], [375]. Może aktualnie ich zastosowania w systemach przemysłowych są niszowe, ale następuje rozwój, który przynależy informatyce. Zatem informatyk powinien nie tylko potrafić projektować sprawdzone rozwiązania i aplikować współczesne techniki i technologie, ale również dostrzegać i kształtować środki, wiedzę i metody dla zastosowań futurystycznych. Powinien widzieć powiązania informacyjne tam, gdzie inni ich nie dostrzegają i umieć je wykorzystać na każdym etapie konstruowania systemu informatycznego. Dzięki temu w dynamicznie rozwijającym się świecie społeczeństwa informacyjnego będzie miał szanse funkcjonować zawodowo jeszcze przez wiele lat.

W wielu książkach autorzy skupiają się na konkretnych urządzeniach i sposobach ich programowania, pomijając aspekty różnorodności rynku i zagadnienia systemowe. W niniejszej książce skupiono się zarówno na środowisku przemysłowym, urządzeniach komputerowych, jak i na modelach oraz oprogramowaniu w kontekście tworzenia systemu informatycznego, ale w oderwaniu od konkretnych konstrukcji i implementacji. Systemy przemysłowe nie kończą się na urządzeniach i specyficznych językach, a właściwie się na nich zaczynają. Nad nimi znajdują się zagadnienia algorytmiczne, systemowe, integracyjne,

komunikacyjne, bezpieczeństwa, bazodanowe, webowe, interfejsów użytkownika i wiele, wiele innych. Zostanie to pokazane w dalszej części książki.

Istnieje ciekawy aspekt aktywności zawodowej w tej dziedzinie. Projektowanie i aplikowanie ISP niesie ze sobą dużą różnorodność dostępnych rozwiązań technologicznych zarówno od strony informatyki, jak i procesu przemysłowego. Powoduje to, że systemy informatyki przemysłowej rzadko kiedy są powielane, a co za tym idzie osoba zajmująca się tą tematyką będzie stawiana wielokrotnie przed nowymi problemami lub klasami problemów projektowych i programistycznych. Istotne jest zgłębienie pewnego obszaru wiedzy, którego wycinek jest przedstawiony w niniejszej książce, i który można stosować jako narzędzie do tworzenia rozwiązań, a nie ich powielania. Z tego powodu opisywana dziedzina wiedzy daje duże możliwości zawodowego spełniania się dla inżynierów, a autor szczerze namawia do jej zgłębiania.

1.1. Pojęcia podstawowe

Podrozdział zawiera systematyzację pojęć pochodzących z automatyki, informatyki i okolic związanych z poruszaną tematyką. Określenie i zrozumienie abstrakcji z nią związanych jest bardzo przydatne z punktu widzenia konstruktora i programisty systemów, umożliwiając spojrzenie na system jako na całość, a nie jako na zestaw odrębnych, choć połączonych, elementów. Omawiane terminy są czasami trudne do jednoznacznego nazwania i zdefiniowania. Wynika to z faktu, że jednostki komercyjnie (integratorzy, producenci, organizacje, grupy wsparcia itp.) zajmujące się tą tematyką często używają własnej terminologii, która pochodzi z języka angielskiego, zaszłości historycznych albo jest lepszym, lub gorszym spolszczeniem. Dlatego ze względu na często spotykaną w literaturze różnorodność nazewniczą – zarówno polsko, jak i anglojęzycznych terminów w nawiasach podane są synonimy nazw. Dla wygody czytelnika jest to czynione wielokrotnie. Terminy angielskojęzyczne podawane są w liczbie pojedynczej.

Niniejsza publikacja nie jest opisem lub wykładnią międzynarodowych norm dotyczących układów czy systemów sterowania. Jednak zastosowane opisy, podziały i nazewnictwo bezpośrednio nawiązują do norm IEC61131 [M26] oraz IEC61499 [M34] i nie stoją z nimi w sprzeczności. Ponadto, w podrozdziałach 2.5 i 2.6 przedstawiono istotne informacje związane z tymi normami.

Poniżej zawarto podstawowe pojęcia pogrupowane tematycznie wokół głównych zagadnień, takich jak proces, system, środowisko, aplikacja i komunikacja. Rozdział jest przydatny dla osób, które zaczynają przygodę z informatyką przemysłową lub chcą uporządkować znane sobie pojęcia.

1.1.1. Technologia procesów przemysłowych

Dla spójności rozumienia dalszych treści niezbędne jest zdefiniowanie pojęcia procesu przemysłowego. **Proces przemysłowy** jest to przebieg zjawisk fizykochemicznych, mających związek przyczynowo-skutkowy i odnoszący się do obróbki oraz przeróbki materiału w celu uzyskania produktu. Zjawiska mogą następować po sobie lub występować współbieżnie. Określenia podstawowych pojęć użytych w powyższej definicji czytelnik znajdzie w literaturze. Innymi słowy, z procesem przemysłowym mamy do czynienia, gdy jakiś **materiał** jest przetwarzany w celu uzyskania jakiegoś **produktu** i wykorzystuje się do tego celu znane i rozumiane zjawiska naturalne, przy czym zarówno materiał, jak i produkt nie muszą być materialne (np. prąd, ciepło itp.). Działalność ta postrzegana jest jako **produkcja** lub **wytwarzanie** (przetwarzanie). Opis tych zjawisk, ich kolejności, wzajemnych zależności i sposobu kontroli nazywany jest **technologią**. Można skojarzyć technologie z przepisem na uzyskanie produktu[3].

Synonimem dla procesu przemysłowego może być **proces produkcyjny**, choć jest to pojęcie szersze, gdyż w procesie produkcji może działać wiele jednostkowych procesów przemysłowych. Dla przykładu, w procesie produkcji piwa stosuje się procesy przemysłowe związane z produkcją butelek, drukowaniem etykiet, warzenia napoju, napełnianiem i etykietowaniem, pasteryzacją, paletyzacją i inne.

Procesy fizykochemiczne biorące udział w procesie przemysłowym mogą być **wolnozmienne** lub **szybkozmienne**. Nie istnieje precyzyjna definicja tych pojęć i zwykle przyjmuje się, że procesy wolnozmienne to takie, które mogą być postrzegane przez człowieka z użyciem jego zmysłów, natomiast szybkozmienne to takie, gdzie zmysły zawodzą i do ich obserwacji wymagane jest wsparcie techniczne.

Ciekawa może być dyskusja na temat produktów niematerialnych, jak oprogramowanie lub sztuka ulotna, lecz jest to poza zakresem niniejszej książki, a ponadto twórcy mogliby się poczuć urażeni traktowaniem ich pracy jako proces przemysłowy. Wymiennie z procesem przemysłowym będzie stosowane tożsame pojęcie **procesu technologicznego**. Efektem funkcjonowania procesu przemysłowego będzie produkt finalny przeznaczony do użytkowania lub półprodukt przeznaczony dla dalszego przetwarzania przez inne procesy przemysłowe.

W nazewnictwie technologii produkcji występuje pojęcie **węzła technologicznego**. Jest to zespół urządzeń technicznych realizujących proces przemysłowy lub operacje jednostkowe danego procesu. Sprzęgnięcie węzłów w związki przyczynowo-skutkowe prowadzi do powstania **linii technologicznej**, nazywanej również linią produkcyjną, ciągiem technologicz-

[3] Słowo technologia pochodzi od greckich słów „techne" i „logos" (zbiór, rozum i in.) i oznacza metodę przetwarzania dóbr materialnych w dobra użyteczne, w tym także wiedzę o takim procesie przetwarzania.

nym, ciągiem produkcyjnym lub nitką technologiczną. Jest to zespół węzłów realizujących dany proces technologiczny. Z punktu widzenia zarzadzania dobrze jest również wprowadzić pojęcie **jednostki produkcyjnej**, która określa zarówno zestaw technicznych elementów procesu technologicznego (fizycznych, np. maszyna, linia), jak i zestaw elementów nietechnicznych (biznesowych, ekonomicznych, np. zlecenia, transport, magazyn) uczestniczących w realizacji **procesu wytwórczego**. Więcej o procesach można znaleźć w [361].

Kontrolowanie przebiegu procesu nazywane jest prowadzeniem procesu lub w kontekście zakładu produkcyjnego **utrzymaniem ruchu**. Na taką kontrolę składają się przeglądy, niezbędne modernizacje, serwis, utrzymywanie rezerw, testy urządzeń obsługujących proces, jak i dbanie o ich poprawne funkcjonowanie, a także zaopatrzenie w materiał, odbiór produktu i wiele innych czynności. Wydajna produkcja powinna zachodzić w sposób ciągły bez niespodziewanych przerw, czyli bez tzw. postojów nieplanowanych. Podczas prowadzenia procesu możliwe są **postoje** planowane. Wynikają one z danej technologii i z reguły są niezbędne lub wynikają z procesu planowania produkcji i są konsekwencją złożonego procesu zarządzania przedsiębiorstwem. Postoje nieplanowane skutkują stratami dla zakładu i najczęściej są spowodowane awariami.

Prowadzenie procesu dotyczy także zagadnień **bezpieczeństwa**. Zapewnianie bezpieczeństwa dla procesu przemysłowego wiąże się z zapewnieniem prawidłowego dostępu do procesu oraz zapewnieniem jego prawidłowego działania. Przekłada się to bezpośrednio na systemy informatyczne i mówi o bezpieczeństwie **zasobów** (ang. security) oraz o bezpieczeństwie **funkcjonalnym** (ang. safety). Więcej na ten temat znajduje się w podrozdziałach 1.2.5 i 3.4.

Na styku pomiędzy procesem technologicznym a systemem informatycznym pracują urządzenia **aparatury kontrolno-pomiarowej** (AKP). Czasami można spotkać określenie AKPiA, co oznacza AKP i automatyki. Takie urządzenie lub zestaw powiązanych ze sobą urządzeń z punktu widzenia systemu informatycznego można opisać zbiorem informacji i nazwać **obiektem przemysłowym**. Jest to abstrakcja wykraczająca poza znany z automatyki **obiekt sterowania** czy **regulacji**, gdyż zakres działań z nim związanych wykracza poza domenę automatyki. Zbiór informacyjny obiektu musi zawierać definicje struktur i powiązań danych, ich związki semantyczne i czasowe, a także algorytmy i zasady ich przetwarzania oraz przekazywania.

Obiekty takie mogą posłużyć jako abstrakcyjne elementy (modele, komponenty) definiujące interfejs procesu przemysłowego z systemem informatycznym oraz stanowić podstawę do opisu przepływu danych w systemie. Abstrakcja obiektu jest urzeczywistniana przez układy AKP, które są niepodzielne lub stanowią funkcjonalny zestaw (więcej w 2.4.1). Uzyskanie modelu procesu w wyniku dokonania klasycznej jego algorytmizacji [363] wydaje się być

zbyt wąskim działaniem przy tworzeniu rozwiązań klasy ISP. Istotne jest, aby tam, gdzie od strony technologicznej widać proces i aparaturę, a od strony automatyki zasady ich działania, zbudować model opisujący struktury danych i programy. Zatem, informatyzacja procesów, oprócz utworzenia modelu w wyniku algorytmizacji, wymaga dodatkowo utworzenia modelu informacyjnego, na bazie którego można poprawnie dokonać wyboru środków i metod przetwarzania, kodowania, przesyłania oraz składowania informacji.

Elementarna funkcjonalność interfejsu informacyjnego urządzenia AKP sprowadza obiekt do producenta (źródła) lub konsumenta (ujścia) informacji. Zatem względem przepływu takiej informacji, obiekty w swym podstawowym zakresie funkcjonują jako:

- układy **inicjatorów** – (ang. sensors) są to układy pobierające informację o stanie procesu, tzw. producenci czy też dostarczyciele informacji. Można do nich zaliczyć np.: układy pomiarowe, czujniki, sondy, detektory, krańcówki, styczniki, przyciski itp.;
- układy **wykonawcze** – (ang. actuators) są to układy modyfikujące stan procesu, tzw. konsumenci czy też odbiorcy informacji. Można do nich zaliczyć np.: układy napędowe, zawory, pompy, wentylatory, mieszadła, grzałki, siłowniki itp. Z punktu widzenia przepływu informacji do tej kategorii można również zaliczyć urządzenia prezentujące stan procesu, typu elementy sygnalizacyjne.

W praktyce, urządzenia AKP związane z danym obiektem mają zwykle funkcjonalności obu powyższych kategorii, i stanowią:

- układy **mieszane** – (ang. mixed). Urządzenia te zarówno pobierają z procesu, jak i oddziałują na proces (dostarczają informację do procesu). Zatem obiekty zbudowane za pomocą takich urządzeń stanowią jednocześnie producenta i konsumenta informacji. Większość przytoczonych powyżej przykładów stanowi w praktyce układy mieszane.

Ponadto, istnieją układy stanowiące zarówno źródło, jak i ujście informacji, ale są to układy niezwiązane bezpośrednio z procesem, a z tworzeniem interfejsu zrozumiałego dla użytkownika nadzorującego ten proces. Są to wszelkiego rodzaju komputerowe i niekomputerowe układy interfejsowe przeznaczone dla człowieka. Komputerowe bazują na konsolach, panelach oraz innym sprzęcie i oprogramowaniu opisanym szerzej w rozdziale 2.7. Natomiast klasycznym, niekomputerowym interfejsem pomiędzy procesem przemysłowym a człowiekiem są elektryczne **układy synoptyczne**. Synoptyka w tym przypadku dotyczy zbiorczego wglądu w stan procesu i oddziaływania. Obraz procesu jest prezentowany przez różnorodne urządzenia sygnalizacyjne (lampki, kontrolki, wskaźniki itp.), natomiast oddziaływanie na proces przez różne zadajniki (stacyjki, przełączniki, załączniki itp.). Całkowita eliminacja układów synoptycznych na rzecz komputerowych

interfejsów użytkownika wydaje się niemożliwa ze względu na aspekty zapewniania i ergonomii dostępu do kluczowych informacji. Trudno sobie wyobrazić, aby przycisk zatrzymania awaryjnego wymagał kliknięcia myszą, a sygnalizator zagrożenia był na n-tym oknie wizualizacji.

1.1.2. Prowadzenie procesu

Współcześnie, komputerowe (czytaj informatyczne) wspomaganie działania przedsiębiorstwa produkcyjnego dotyczy zarówno podejścia i od strony obsługi procesu technologicznego, jak i od strony działalności gospodarczej, czyli tzw. biznesu.

Sukces wytwarzania zależy od działania urządzeń, systemów i technologii, ale również od ich obsługi. Obsługa wiąże się zarówno z wiedzą i umiejętnościami ludzi, jak i z procedurami, zasadami i politykami obowiązującymi w przedsiębiorstwie. Dlatego do prowadzenia produkcji niezbędne są tzw. procesy biznesowe, jak i procesy techniczne.

Proces biznesowy dotyczy ustalonej sekwencji działań zmierzających do osiągnięcia zdefiniowanego celu. Dla przedsiębiorstw produkcyjnych głównym celem jest oczywiście produkcja, przy czym narzucone są na nią pewne założenia i wymagania. Procesy biznesowe z reguły dzielą się na zarządzające (ang. management process), operacyjne (ang. operational process) i pomocnicze (ang. supporting process). Współczesne systemy informatyczne są w stanie zapewnić zaawansowane wsparcie dla wszystkich trzech rodzajów procesów biznesowych [361], [285], [135].

Proces techniczny stanowi sekwencje działań związanych z obsługą procesu technologicznego. Można go utożsamiać z procesami produkcyjnymi (por. 1.1.1) rozważanymi w kontekście fizycznym, czyli maszyn, urządzeń i innych środków, w tym oprogramowania (algorytmy, programy, definicje, struktury, dane itp.) i wiedzy (procedury, receptury, parametry itp.).

Zagadnienia związane z prowadzeniem produkcji są bardzo szerokie i wykraczają daleko poza tematykę niniejszej książki. Warto jednak wspomnieć o niektórych wskaźnikach opisujących jakość funkcjonowania danego procesu, a związanych z działaniem ISP [344]. Powiązanie dotyczy wykorzystania danych. W tabeli 1 przedstawiono wskaźniki przykładowe, które z jednej strony korzystają z danych dostarczanych z ISP, a z drugiej dają obraz przebiegu procesu i jego jakości na poziomie kontroli produkcji. Wskaźniki te warto znać, gdyż tworząc ISP często zaistnieje potrzeba dostarczania odpowiednich danych do systemów kontrolnych celem ich wyliczenia. Są one czasem nazywane narzędziami opisu wydajności lub osiągów jakościowych produkcji (ang. performance).

Wszystkie dane potrzebne do obliczenia powyższych wskaźników można uzyskać z ISP, o ile istnieją urządzenia odpowiedzialne za fizyczne zliczanie (AKP) lub manualne wprowa-

dzanie (np. HMI zob. 3.1.10, 3.2) takich danych do węzłów oraz stosowne zmienne, algorytmy i sieci umożliwiające parametryzacje procesu oraz przetwarzanie i dostarczanie wartości w żądane miejsce.

Tabela 1

Przykładowe wskaźniki opisujące jakość produkcji

Skrót	*Nazwa*	*Opis*
FTQ	First Time Quality	Jest to tzw. jakość za pierwszym razem, czyli potencjał danej linii produkcyjnej do produkowania bez wad. Wskaźnik określa niejako „wydajność jakościową" i można go obliczać jako iloraz: $$ftq = \frac{n-(s+r)}{n}$$ gdzie: *n* – liczba produktów produkowanych w procesie, *s* – liczba produktów niespełniających wymagań jakości (braki), *r* – liczba produktów naprawionych, Wskaźnik zwykle wyrażany w przedziale <0, 1> lub w procentach. Wskaźnik osiąga wartość *zero*, gdy suma liczby braków i produktów naprawionych jest równa *n*. Oznacza to, że cała partia produkcji nie spełnia wymagań jakości „za pierwszym razem", czyli po wyprodukowaniu. Wskaźnik osiąga wartość *jeden*, gdy wspomniana suma wynosi zero. Oznacza to, że cała partia produktów spełnia wymagania jakości i nie wymaga ani naprawy, ani odrzutu. W praktyce, zwykle nie osiąga się wartości skrajnych.
FPY TPY	First Pass Yield Throughput Yield	Są to inne nazwy wskaźnika FTQ.
RTY	Rolled Throughput Yield	Wskaźnik określa zdolność wieloetapowego procesu do wyprodukowania produktu bez wad. Oblicza się go przez wymnażanie wskaźników jakościowych (FTQ, FPY) uzyskanych dla poszczególnych podprocesów. $$rty = \prod_{i=1}^{m} ftq_i$$ gdzie: *m* – liczba podprocesów.
PPM	Parts Per Million	Jest to wskaźnik określający liczbę braków (defektów, skaz itp.) na milion produktów wytworzonych.
DPU	Defects Per Unit	Jest to wskaźnik określający liczbę defektów w określonej partii produktów. Można go obliczyć z ilorazu: $$dpu = \frac{d_s}{n}$$ gdzie: *n* – liczba produktów wyprodukowanych, d_s – liczba wszystkich stwierdzonych defektów w *n* produktach.
DPMO	Defects Per Million Opportunities	Wskaźnik określający liczbę stwierdzonych defektów w milionie możliwych do stwierdzenia defektów. Można go obliczyć ze wzoru: $$dpmo = \frac{d_s}{nd_p}$$ gdzie: d_p – liczba potencjalnych (możliwych do wystąpienia) defektów w produkcie.

Przykładowa prezentacja wskaźników FTQ została przedstawiona w raporcie z systemu MES na rysunku 115 (b, c).

1.1.3. Środowisko przemysłowe

Środowisko przemysłowe rozumiane jest jako ogół warunków otoczenia, w jakich zachodzi proces oraz jakie ten proces stwarza w swoim otoczeniu. Warunki takie zdecydowanie odbiegają od warunków biurowych. Różnica polega na zwiększonej uciążliwości wynikającej z występowania różnych **zaburzeń** środowiskowych:

- otoczenia – duża rozpiętość temperatur pracy, wilgotności, agresywność chemiczna,
- mechanicznych – wstrząsy, wibracje, zapylenie, bryzgi, uszkodzenia,
- przewodzonych – zaburzenia wartości prądu i napięcia zasilania urządzeń, sieci komputerowych, transmisji pomiarów itp.,
- elektromagnetycznych – (ang. EMI – Electromagnetic Interferences) sprzężeń indukowanych i pojemnościowych (dot. występowania obcych pól elektrycznych, magnetycznych i elektromagnetycznych oddziałujących na sieci komputerowe, transmisje pomiarów, urządzenia pomiarowe, urządzenie komputerowe itp.).

Wszystkie te czynniki stanowią zagrożenie dla urządzeń oraz samego procesu. Oczywiście rozważanie wpływu środowiska na zagadnienia technologiczne nie stanowi tematu książki, ale inżynier musi być świadom, że występowanie niekorzystnych czynników może wpływać nie tylko na same urządzenia.

Nigdy nie zakładaj, że środowisko pracy ISP będzie przyjazne.

Załóż, że zaburzenia środowiskowe są zmienne z racji zmienności prowadzonego w nim ruchu, jak i z powodu możliwości jego modyfikacji.

Problemem jest występująca duża różnorodność środowisk. Nie można określić, że środowisko procesów przemysłowych zawsze czymś się będzie charakteryzowało, a czymś nie. Zależy to silnie od procesu, lokalnych zasobów technicznych oraz polityki danej firmy. Określając środowisko, nie bez znaczenia jest też wpływ człowieka. Człowiek jest elementem środowiska i może mieć istotny wpływ na działanie układu. Sprowadza się to do niestanowiących zagrożeń oddziaływań neutralnych, gdy człowiek jest świadomym użytkownikiem, powodującym co najwyżej naturalne zużycie elementów interfejsu, oraz do stanowiących zagrożenia oddziaływań negatywnych. Pracownik może niekorzystnie wpływać na pracę i zużycie elementów systemu niecelowo lub celowo. Dla przykładu, podłączenie urządzeń dużej mocy do sieci zasilającej komputery, używanie urządzeń zakłócających (telefony, krótkofalówki itp.) w pobliżu urządzeń wrażliwych,

podłączanie/odłączanie urządzeń do komputerów, instalowanie/zmiana oprogramowania itp. akcje mogą skutkować zaburzeniami pracy systemu. Celowe działania negatywne to dewastacja, sabotaż, kradzież, szpiegostwo itp., a nawet wyłączanie i odłączanie urządzeń czy brak należytej dbałości i ignorowanie instrukcji. Dobierając elementy systemu, należy brać to wszystko pod uwagę.

Dodatkowo, poza wpływem środowiska na proces, istotny jest wpływ procesu na środowisko. Ten problem jest z reguły bardziej postrzegany i rozumiany, gdyż wiąże się z tematami ochrony środowiska i bhp. To zagadnienie jest dość szeroko poruszane w literaturze i w mediach [152].

1.1.4. Zagadnienie czasu

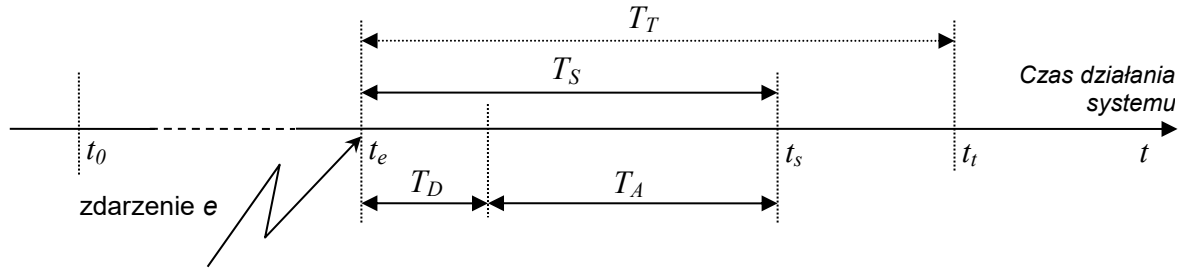
Proces przemysłowy, aby przebiegał prawidłowo, czyli aby zostały zachowane wszystkie wymagane związki przyczynowo-skutkowe musi przebiegać w sposób, gdzie każde zdarzenie i zjawisko zachodzi w sposób jednoznaczny w stworzonych dla niego warunkach. ISP musi zatem stwarzać warunki determinujące prawidłowy przebieg procesu.

❖ **Determinizm działań**

Specyfiką funkcjonowania systemu informatycznego w środowisku przemysłowym jest **determinizm** zachodzących zjawisk oraz ograniczenia czasowe z tym związane [220]. Każdy system kontrolujący proces powinien zachowywać się w sposób deterministyczny. Ogólne **pojęcie determinizmu** działania rozumiane jest jako twierdzenie, że wszystkie występujące zjawiska w rozpatrywanym układzie podlegają nieuchronnym prawidłowościom. Oznacza to, że każde wystąpienie zdarzenia w układzie jest jednoznacznie określone przez ogół warunków, w jakich zachodzi. Dla danych warunków odpowiedź układu jest zawsze taka sama. Działanie procesu przemysłowego jest teoretycznie deterministyczne. Wynika to z jednoznaczności praw fizyki i chemii. W praktyce, należy jednak brać pod uwagę czynniki, których przewidzenie nie jest łatwe bądź niemożliwe. W efekcie proces może zachować się zgodnie z prawami natury, ale niezgodnie z założeniami, jakie przyjęto do jego kontroli. Aby uniknąć problemów z utrzymaniem ruchu, system obsługujący taki proces powinien obsługiwać sytuacje awaryjne. W obsługę taką wchodzą zarówno awarie systemu, jak i „sytuacje niemożliwe" w procesie. Działanie systemu musi być deterministyczne. Jest to niezbędne, aby dotrzymać warunków prowadzenia procesu, wynikających z technologii. Innymi słowy, dla określonych warunków zaistniałych w procesie system musi zawsze zareagować w taki sam sposób.

Ponadto, zachowanie systemu obsługującego proces musi być zdeterminowane względem czasu. **Determinizm czasowy** w funkcjonowaniu obiektu definiuje się jako zgodne z działaniem deterministycznym reakcje obiektu na zdarzenia, w określonym i skończonym

czasie. Oznacza to, że system musi wykazać się **nadążnością**, czyli musi obserwować otoczenie i reagować na zjawiska w określonym i skończonym czasie. Przedział czasu, w którym dane działanie ma się wykonać, określa się jako **ograniczenie czasowe.** Jest to przedział czasu w zakresie większym od zera, a mniejszym bądź równym jakiemuś **czasowi granicznemu** przyjętemu dla danego przypadku (ang. timeout, deadline). Określenie zachowania jakiegoś zadania względem czasu nazywa się **charakterystyką czasową** wykonania tego zadania. W praktyce może to dotyczyć np. opóźnienia transmisji, czasu obliczeń, cyklu działania itp., rozpatrując w wartościach chwilowych, średnich, pesymistycznych lub rozkładu wartości. Na rysunku 2 przedstawiono ilustrację zjawisk czasowych przy ogólnej obsłudze jakiegoś hipotetycznego zdarzenia w dowolnym układzie.



Rys. 2. Ilustracja wystąpienia i obsługi zdarzenia
Fig. 2. Illustration of event occurrence and service

gdzie:

t_0 – moment początku działania systemu,

t_e – moment wystąpienia zdarzenia w systemie (ang. event),

t_s – moment zakończenia obsługi zdarzenia (ang. end of service),

t_t – maksymalny czas zakończenia obsługi zdarzenia (ang. time limit),

T_D – czas trwania detekcji zdarzenia, reakcji na zdarzenie (ang. detection),

T_A – czas trwania odpowiedzi systemu, czas akcji (ang. answer, action),

T_S – czas trwania obsługi zdarzenia (ang. service),

T_T – maksymalny dopuszczalny czas trwania obsługi zdarzenia (ang. timeout).

W większości systemów biurowych i domowych charakterystyka wystąpień jest probabilistyczna i w wystarczająco dużym czasie obserwacji można ją opisać jakimś znanym rozkładem prawdopodobieństwa. W systemach z ograniczeniami czasowymi sytuacja jest podobna, jednak ograniczenia sprawiają, że dziedzina czasu staje się ewidentnie dyskretna. Należy tu wyraźnie zwrócić uwagę, że wszelkie komputery cyfrowe oddziałujące z rzeczywistością, w której zdarzenia zawsze są w ciągłej dziedzinie czasu, oddziałują z nią w dziedzinie dyskretnej, dzieląc czas na pewne kwanty, poniżej wartości których system cyfrowy nie oddziałuje. To mniej więcej tak jak z wrażeniem ruchu w obrazie filmowym. Zarejestrowane klatki są statyczne, ale odtworzone wystarczająco szybko dają efekt ciągłości ruchu. Działanie takie

wynika z konstrukcji komputerów, a szczególnie ich układów przetwarzających i układów wejścia/wyjścia (ang. input-output, IO) [233], [188]. Rozważania o ciągłym przetwarzaniu w domenie czasu jest umowne i sprowadza się do określenia czasów reakcji wynikających z szybkości rozpatrywanych układów cyfrowych, a w właściwie z rozdzielczości, z jaką dany układ cyfrowy jest w stanie informacje rejestrować i przetwarzać. Jeśli rozdzielczość jest większa od zmienności układów zewnętrznych, to można mówić o przetwarzaniu ciągłym, choć w rzeczywistości takie nie jest.

Momenty, w których występują zdarzenia w systemach z ograniczeniami czasowymi, mogą mieć charakter [M88]:

- cykliczny – (periodyczny, ang. periodic, cyclic)
  Powtarzalne w stałych odstępach czasu ze stałym okresem *C*. Za cykliczne przyjmuje się również zdarzenia jednokrotne. Momenty wystąpienia zdarzeń są znane (przewidywalne).

- acykliczny – (aperiodyczny, incydentalny, sporadyczny, ang. aperiodic)
  Momenty wystąpień są nieprzewidywalne przed zadziałaniem układu. Często wyróżnia się zdarzenia:
  - sporadyczne (sporadyczny, ang. sporadic) jako zdarzenia z określoną maksymalną częstotliwością wystąpień f_{max} (minimalnym czasem powtórzeń, minimalnym okresem) i z określonym T_S, zachodzą one stale, lecz z niestałym odstępem (można by rzec cyklicznie, z tym że okres cyklu jest zmienny). Częstotliwość wystąpień limitowana jest granicznie przez czas obsługi T_S oraz maksymalny dopuszczalny odstęp czasowy między zdarzeniami,
  - aperiodyczne (aperiodyczny, ang. aperiodic), w których znany jest czas T_S, a charakterystyka czasowa wystąpień nie jest znana. Zdarzenia pojawiają się w nieregularnych odstępach określonych co najwyżej rozkładem prawdopodobieństwa.

Jeśli czas T_S nie jest znany, to nie można gwarantować determinizmu czasowego obsługi takiego zdarzenia i wówczas mówi się o obsłudze bez ograniczeń czasowych (ang. non-real-time). Na czas T_S może składać się wiele innych czasów związanych z operacjami obsługi zdarzenia, zależnie od stopnia szczegółowości dekompozycji takiej obsługi. Na rysunku 2 pokazano tylko typowe dwa czasy: czas detekcji, w którym zdarzenie jest wykrywane i identyfikowane, oraz czas odpowiedzi, w którym zidentyfikowane zdarzenie jest obsługiwane.

Pojęcie zdarzeń można łatwo rozszerzyć do pojęcia zadań. Zadanie jest wykonywane w układzie i ma swój początek w czasie (moment wyzwolenia, aktywowania) oraz czas trwania. Należy dopuścić możliwość uruchamiania zadań przez wiele różnych zdarzeń, o różnej

charakterystyce czasowej. Istnieje zatem skończony zbiór E zawierający k zdarzeń e mogących wystąpić w systemie.

$$E=\{e_q|q=1,2,\ldots,k\}$$

Źródłem takich zdarzeń mogą być przerwania lub relacje między danymi. Z danym zadaniem x powiązany jest podzbiór m zdarzeń e, które mogą je aktywować.

$$E_x=\{e_j|j=1,2,\ldots,m\}$$

Można uogólnić, że w systemie występuje zbiór zadań Z składający się z n zadań x ograniczonych czasowo i wyzwalanych zdarzeniami e ze zbioru zdarzeń E_x.

$$Z=\{x_i|i=1,2,\ldots,n\}$$

gdzie zadanie x_i jest opisane parametrami:

$$x_i=<E_x,\ T_S,\ T_T>$$

$$e_j=<t_e>$$

gdzie dla wszystkich $i > 1$ oraz $j > 1$ czas t_e może przyjmować wartości:

- dla zdarzeń cyklicznych

$$t_e = t_e(j) - t_e(j\text{-}1) = t_e(j) + C = const,$$

- dla zdarzeń sporadycznych

$$t_e(j\text{-}1)+T_S(i\text{-}1) \le t_e < t_e(j\text{-}1)+1/f_{max},$$

- dla zdarzeń aperiodycznych

$$t_e = t_e(j\text{-}1)+P(e_j),$$

gdzie $P(e_j)$ jest wartością zmiennej losowej opisującej wystąpienia zdarzenia j.

Dla pierwszego wystąpienia zadania (x_1) czasy te mogą przyjmować wartości:

- dla zdarzeń cyklicznych

$$0 < t_e \le C,$$

- dla zdarzeń sporadycznych

$$0 < t_e < 1/f_{max},$$

- dla zdarzeń aperiodycznych

$$0 < t_e = P(e_1).$$

Aby system zachował determinizm czasowy działania, to dla każdego zadania ograniczonego czasowo musi zachodzić nierówność:

$$t_e(j\text{-}1)+T_S(i\text{-}1) \le t_e(j\text{-}1)+T_T \le t_e(j)$$

czyli

$$0 < T_S(i) \leq T_T$$

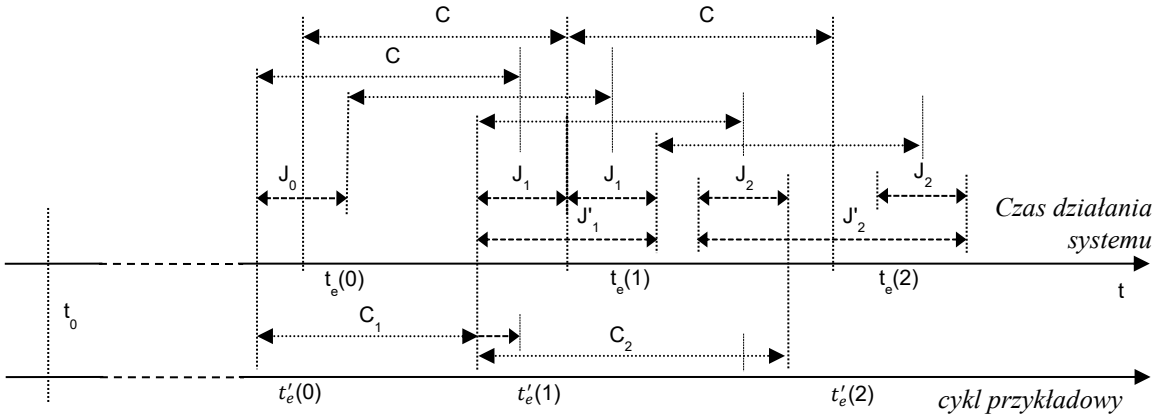
przy czym w praktyce

$$0 + \Delta t < T_S(i) \leq T_T \pm \Delta t$$

gdzie Δt jest pewną niestałością zachowań układu i wynika z jego fizycznej inercji. Jeśli Δt jest dużo mniejsze od T_T, to przy analizie czasowej systemu może być pomijane. Wówczas pesymistyczną (maksymalną) wartość czasu obsługi zdarzenia można określić jako czas graniczny

$$max(T_S) = T_T$$

Sytuacja, gdy $T_S > T_T$ jest błędna i powinna być obsłużona jako błąd. Sposób obsługi takiego błędu zależy od wymagań układu względem ograniczeń czasowych. Spotykane rodzaje wymagań omówione są w kolejnym podrozdziale (strona 33).



Rys. 3. Ilustracja wystąpienia niestałości cyklu zdarzeń
Fig. 3. Illustration of jitter of events

Przy opisywaniu charakterystyk czasowych w układach złożonych, w których występuje wiele zdarzeń, a czasy obsługi zadań są określane w przedziale, pojawia się kolejna niestałość (na rysunku 3 oznaczoną przez *J*) wynikająca, oprócz inercji, ze współpracy zadań cyklicznych, które oddziałując na siebie wzajemnie wprowadzają niestałości czasów realizacji (kolebanie, ang. jitter). Współwystępowanie nieprzewidywalnych zdarzeń acyklicznych również zaburza stałość czasu obsługi zdarzeń. Problem niestałości wynika z konstrukcji układu, a nasila się, szczególnie gdy zadania konkurują o wspólne zasoby lub muszą być wstrzymywane z innych powodów (np. synchronizowane). W układach deterministycznych wartość niestałości podaje się jako wartość maksymalną dla określonej aktywności układu.

Zaburzenia stałości cyklu zdarzeń cyklicznych zilustrowane zostały na rysunku 3. Wartość niestałości *J* w danym cyklu nie jest znana, jest jednak określona wartością maksymalną, czyli w praktyce w granicach.

$$J_i \neq const$$
$$max_{0<i\leq k}(J_i) = J_{max}$$
$$t'_e(i) = t_e(i) \mp \frac{J_{max}}{2}$$

gdzie $t_e(i)$ jest teoretycznym czasem w i-tym cyklu, a $t'_e(i)$ czasem rzeczywistym przy założeniu braku niestałości w cyklach poprzednich. Uwzględniając wcześniejsze cykle, można określić minimalne i maksymalne wartości czasów $t'_e(i)$.

$$t_e(i) - i\frac{J_{max}}{2} \leq t'_e(i) = t_e(i) + i\frac{J_{max}}{2} \qquad \text{dla } 0 < i \leq k$$

Z każdym kolejnym okresem cyklu wartość odstępu czasu pomiędzy zdarzeniami może być różna, czyli dla przykładu z rysunku $C_1 \neq C_2 \neq C \neq const$. Wraz ze wzrostem czasu, odstępstwa od zadanego cyklu mogą narastać, gdyż rosną potencjalne niestałości (J') względem tego cyklu. W przykładzie przyjęto generację zdarzeń w momentach granicznych niestałości, co uwypukla problem. W praktyce wartość realna niestabilności nie przyjmuje zawsze wartości granicznych, lecz jest wartością zmiennej losowej o danym rozkładzie, przeważnie normalnym, z przedziału czasu niestałości danego zdarzenia, czyli w odpowiednio długim czasie działania średnia wartość przeważnie wypada na wartości czasu oczekiwanego. Nie zmienia to jednak faktu, że niestałość cyklu prowadzi do zaburzeń funkcjonowania układu, jeśli wymaga on wystarczająco dużej precyzji. Aby rozwiązać problem jej utraty, stosuje się synchronizację względem wymaganego w danym przypadku zegara.

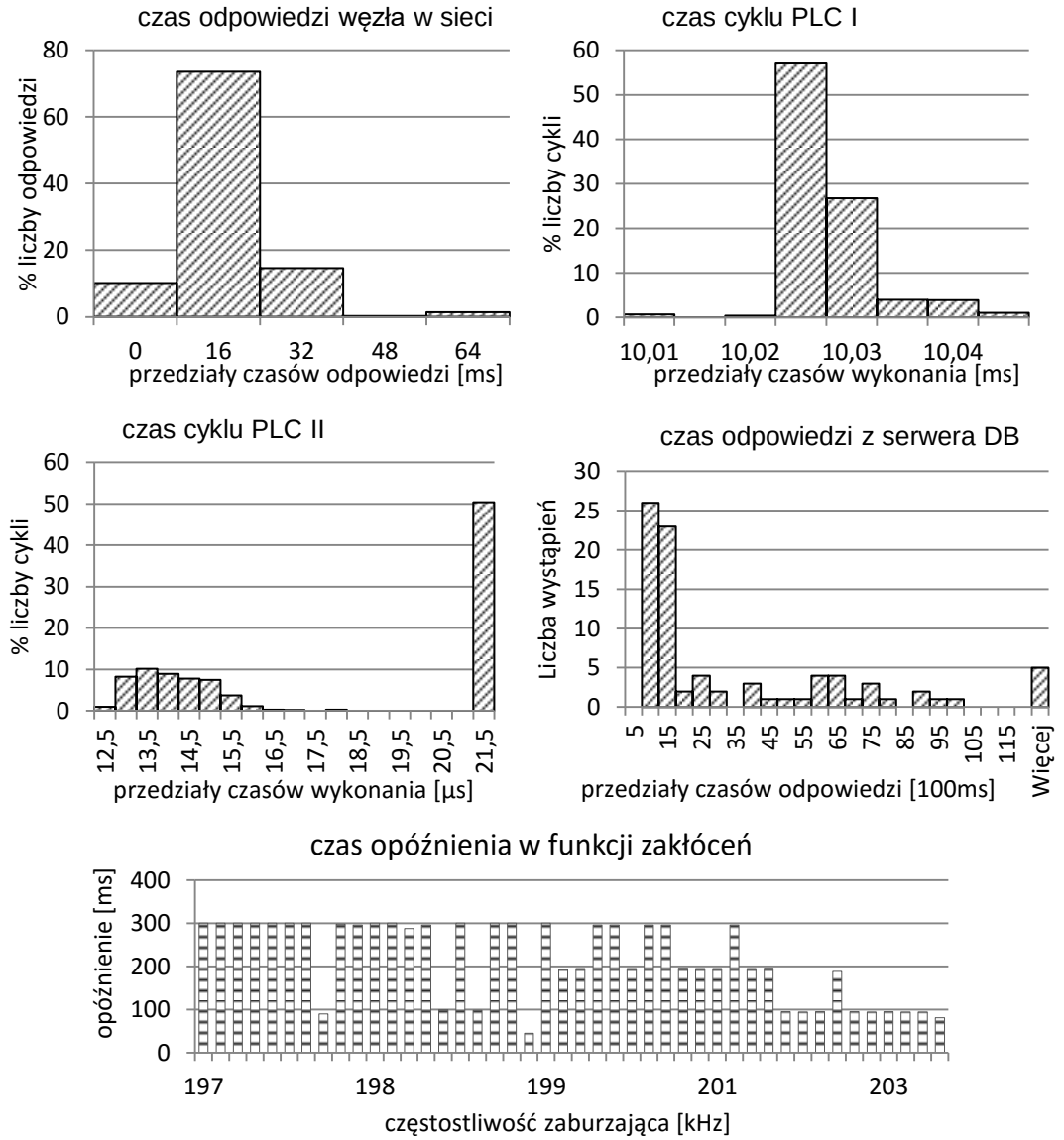
Jitter jest bardzo istotną cechą charakteryzującą działania systemów z ograniczeniami czasowymi. Określa ona zakres realizowalności interakcji czasowej układu z elementami własnymi oraz z jego otoczeniem (ze wszelkimi źródłami zdarzeń). Do podstawowych interakcji należą działania cykliczne, a także działania sporadyczne (acykliczne). Rozważania o niestałości cyklu zdarzeń można łatwo przenieść na wszelkie inne zjawiska czasowe, a w szczególności na czas wykonania zadań T_S.

W praktyce T_S nigdy nie może być zerem, zatem jego wartość optymistyczna (minimalna) jest zawsze większa od zera i stanowi minimalny czas potrzebny dla obsługi danego zdarzenia. Wartość T_S może być opisana albo wartością minimalną i maksymalną (optymistyczną i pesymistyczną), albo rozkładem liczebności, lub prawdopodobieństwa wystąpienia danych wartości z przyjętego zbioru wartości. Rozkłady mogą być tworzone na podstawie teoretycznej analizy czasowej układu lub z pomiarów T_S dla układu rzeczywistego.

Na rysunku 4 przedstawiono histogramy przykładowych rozkładów liczebności czasów obsługi klasyfikowanych w pewnych przedziałach. Ilustrują one zjawiska o różnych wymaganiach względem limitowania czasu obsługi i obrazują problem niestałości czasów T_S oraz konieczności określania czasu T_T. Przedstawione histogramy stworzone są na podstawie zbio-

rów wartości zmiennych T_S pochodzących z rzeczywistych układów, gdzie zmienna T_S związana jest odpowiednio z: pomiarem czasu odpowiedzi abonenta w sieci komputerowej, pomiarami czasu wykonania programu dla dwóch rodzajów PLC oraz pomiarem czasu cyklicznego zapisu pomiarów do bazy danych.

Przy obserwacji działającego systemu i pomiarach czasów T_S nie ma gwarancji na zebranie pełnej charakterystyki czasowej opisującej jego działanie w każdym możliwym przypadku. Aby tego dokonać, należałoby mierzyć T_S dla wszystkich możliwych zdarzeń oraz ich wszelkich koincydencji lub prowadzić pomiary w czasie nieskończonym. Oba przypadki są w praktyce niemożliwe do realizacji.



Rys. 4. Przykłady rozkładów czasów T_S
Fig. 4. Examples of T_S distribution

Występuje zatem problem rozkładu brzegowego. W powyższych przykładach pomiary były dokonywane w seriach po kilka tysięcy. Mimo to nie ma gwarancji, że w jakiś specyficznych warunkach pracy układu charakterystyka czasu odpowiedzi, czasu wykonania programu, czy też rozpatrywanych przedziałów czasu nie uległaby zmianie. Dlatego wyznaczenie parametru T_T na podstawie obserwacji systemu może być zawodne i wskazane jest, aby dobierać go na podstawie wartości pesymistycznej T_{Smax} wyliczonej analitycznie lub na podstawie założeń wynikających z przyjętego rozkładu prawdopodobieństwa. Wyznaczona wartość pesymistyczna powinna być zweryfikowana z maksymalną wartością uzyskaną przy obserwacji systemu. Rozkład teoretyczny powinien być weryfikowany przez porównanie z rozkładem pozyskanym z pomiarów praktycznych. Rozkłady powinny być podobne. Podobieństwo można zbadać np. wyliczając dywergencję Kullbacka-Leiblera [14] i przyjmując dla niej akceptowalny próg. Rozbieżność sugeruje błąd w analizie lub błąd w metodzie pomiarowej.

W praktyce omawiane zależności dotyczą wszelkich zdarzeń w układzie zdeterminowanym czasowo, od wykonania kodu w komputerach do przekazywania danych w sieciach. Natomiast gdy w systemie występują zadania, które nie mają ograniczeń czasowych, wówczas do modelowania ich zgłoszeń i obsługi należy przyjąć rozkłady najlepiej pasujące do rzeczywistości. Projektując taki system, warto zadbać, aby czas t_e spełniał zależność

$$t_e(i) \geq t_e(i-1) + T_S(i-1)$$

W przeciwnym razie będą rosły kolejki obsługi, a tym samym odpowiedź układu będzie coraz bardziej oderwana od rzeczywistości. W systemach biurowo-domowych wiąże się to z dobrze znanym wzrostem irytacji użytkownika. W ISP doprowadzi to do utraty danych lub awarii. Systemy tego typu nie będą omawiane, a Czytelnika autor odsyła do prac [239], [81], [73], [95].

Więcej o analizie charakterystyk czasowych w kontekście cyklu PLC znajduje się np. w [381], [382], a opis podobnego problemu w [382].

❖ Systemy czasu rzeczywistego

Pojęcie **czasu rzeczywistego** stanowi swoiste kryterium podziału zastosowań systemów informatycznych i ich elementów względem czasu. Ich zdeterminowana w czasie zdolność do działania w interakcji z otoczeniem definiowana jest przez ograniczenia czasowe i określa rzeczywistość (zakres realizowalności, nadążność) takiej interakcji.

Ogólnie, systemy z ograniczeniami czasowymi nazywane są **systemami czasu rzeczywistego** (SCR, ang. RTS – Real Time Systems) [206]. Systemy czasu rzeczywistego są to systemy, które **na bieżąco** (terminowo) pracują w interakcji z elementami niestanowiącymi składowych tych systemów. SCR jest to system komputerowy, w którym przetwarzanie (ang. *real-time processing*) jest wykonywane współbieżnie z procesem zewnętrznym – otoczeniem

systemu, i odznacza się nadążnością. Wszystkie reakcje elementów systemu współdziałających w czasie rzeczywistym muszą być zdeterminowane czasowo. Poprawność działania SCR zależy od:

- poprawnej obsługi zdarzenia (prawidłowa deterministyczna odpowiedź),
- obsługi w żądanym czasie (deterministyczny czas odpowiedzi).

Zatem podstawowym celem działania SCR jest terminowa realizacja zadań. Do poprawnego funkcjonowania ISP czasu rzeczywistego niezbędne jest:

- przetwarzanie danych w określonym czasie (działanie węzła),
- przekazywanie informacji w określonym czasie (działanie sieci).

Analiza działania SCR w kontekście przemysłowym przeważnie sprowadza się do **analizy najgorszego przypadku** (ang. worst-case), czyli rozpatrywania wystąpienia przypadków pesymistycznych – określających wykonywanie zadań z maksymalnym dopuszczalnym czasem. Zostało to omówione w poprzednim podrozdziale. Czasem warto się jednak pokusić o analizę statystyczną takich systemów [380], [381].

W systemach czasu rzeczywistego występują dwa podstawowe rodzaje współzależności czasowej zadań:

- działania synchroniczne – (ang. synchronous)
  Działania statyczne i predefiniowane, wykonywane w kontrolowanym czasie, znanym i określonym (ang. time-driven). Kontrola czasu wykonywania jest realizowana cyklicznie z synchronizacją cyklu (ang. periodic) lub według predefiniowanego wzorca (ang. schedulability) obsługiwanego przez mechanizm szeregujący (ang. scheduler). Działanie podlega ograniczeniom czasowym, a do analizy stosuje się najgorszy przypadek.

- działania asynchroniczne – (ang. asynchronous)
  Wystąpienia dynamiczne nie są znane i nie są określone w czasie, wynikają ze zdarzeń acyklicznych (ang. event-driven). Znane są natomiast te zdarzenia, czasy ich obsługi oraz minimalny czas powtórzeń (ang. repetition time). Na ich obsługę alokowane są zasoby i czas, jednak ich wykorzystanie można opisywać tylko rozkładem prawdopodobieństwa wystąpienia danych zdarzeń lub przypadkiem pesymistycznym, czyli powtarzalnym występowaniem zdarzeń z minimalnym czasem powtórzeń. W SCR działania asynchroniczne tak samo jak synchroniczne podlegają ograniczeniom czasowym.

Czasami spotyka się rozpatrywanie przetwarzania w kontekście systemów synchronicznych z ograniczeniami czasowymi i asynchronicznych bez ograniczeń. Nie jest to dobre podejście. W SCR w obu przypadkach stosowane są ograniczenia czasowe. W praktyce systemów przemysłowych najczęściej spotyka się zadania, które mają charakter zarówno cykliczny, jak

i acykliczny. Dla takich przypadków istnieje pojęcie systemów półsynchronicznych (ang. mesosynchronous), w których stosuje się działania (procesy, zadania) synchroniczne razem ze zdarzeniową obsługą procesów asynchronicznych. Obsługa synchroniczna jest obwarowana ograniczeniami bazującymi na cyklu lub szeregowaniu, natomiast działania asynchroniczne również podlegają ograniczeniom, ale wynikającym tylko z czasów granicznych. Gwarancja obsługi takich działań wynika z alokacji zasobów i ogólnych reguł określających ich wykorzystanie. Czas realizacji zadań jest zatem różny w różnych momentach czasu, ale zawsze podlega ograniczeniom. Budowanie SCR tylko z działań synchronicznych prowadzi do tworzenia skomplikowanych szeregowań i nadmiarowości prowadzącej do niepotrzebnego wykorzystywania zasobów.

Można też spojrzeć na ten problem od strony ograniczeń czasowych, które można podzielić na ostre i łagodne. Ograniczenia ostre wymagają pracy synchronicznej, czyli zadbania, aby rozpoczęcie i zakończenie zadania było zgodne z konkretnym ograniczeniem (np. okresem cyklu, szeregowaniem). Ograniczenia łagodne sprowadzają się tylko do zadbania o nieprzekraczanie czasów granicznych, czyli rozpoczęcia i zakończenia zadania najszybciej jak się da, ale nie później niż określony limit. Formalny podział SCR opisano w następnym podrozdziale. Określa on sposób, w jaki należy traktować przekroczenia czasu granicznego.

W praktyce spotyka się również systemy hybrydowe (mieszane), w których funkcjonują zarówno zadania czasu rzeczywistego (ang. real-time task), jak i zadania bez ograniczeń czasowych (ang. non-real-time task).

Należy zwrócić uwagę, że praca w czasie rzeczywistym nie wiąże się z szybkim działaniem jakiegokolwiek elementu systemu wg jakiejkolwiek miary. Istotne jest dotrzymanie ograniczeń czasowych. W praktyce spotyka się systemy pracujące z ograniczeniami na poziomie mikrosekund, ale są też SCR z ograniczeniami na poziomie wielu minut.

Przykładami systemów obsługujących procesy szybkozmienne mogą być systemy sterowania napędami, np. w taśmociągach albo produkcji papieru, obsługa detekcji błędów przy produkcji kabli czy obsługa zabezpieczeń technologicznych. Natomiast działania wolnozmienne to np. regulacja przepływu, temperatury czy też wszelakie procesy, gdzie parametry pracy nie ulegają gwałtownym zmianom i mogą być postrzegane zmysłami człowieka.

❖ **Podział SCR**

W teorii systemów czasu rzeczywistego przyjmuje się ich podział wg kryterium zastosowań, gdzie określa się przydatność wyników obsługi zdarzeń względem upływu czasu. Przeważnie wyróżnia się trzy kategorie systemów.

- Soft Real – Time Systems (SRTS); są to systemy z tzw. miękkimi (łagodnymi) ograniczeniami czasowymi
  - Z założenia, w tego typu systemach dopuszczalne są przekroczenia czasów granicznych podczas ich normalnego, poprawnego działania, do określonej wartości.
  - Przekroczenia nie powodują awarii systemu oraz nie wpływają negatywnie na otoczenie.
  - Rezultatem opóźnień w obsłudze zadań jest obniżająca się funkcjonalność systemu. W skrajnym przypadku, dla opóźnień dużo większych od przyjętego czasu granicznego, zachodzi niemożność zrealizowania zamierzonych funkcji. W teorii można mówić, że przy opóźnieniu zmierzającym do nieskończoności działanie systemu jest niemożliwe. W praktyce system przestaje być przydatny, gdy opóźnienia powodują dostarczenie nieprzydatnych rezultatów jego działań, gdy zanika interakcja z otoczeniem lub gdy irytacja użytkownika rośnie ponad miarę.
- Firm Real – Time Systems (FRTS); są to systemy z tzw. mocnymi (silnymi) ograniczeniami czasowymi
  - W tego typu systemach przekroczenia czasów granicznych są identyfikowane i obsługiwane, nie są jednak traktowane jako błąd krytyczny. Zakłada się tymczasowość występowania zaburzeń.
  - Wyniki związane ze spóźnionym wykonaniem zadania (np. odpowiedź sieci, obliczenia) są traktowane jako błędne i nie brane pod uwagę przy przetwarzaniu. Do detekcji takich błędów wykorzystuje się czas graniczny. Zatem dana funkcjonalność nie maleje z upływem czasu, tak jak w SRTS, lecz tymczasowo przestaje działać wraz z przekroczeniem czasu granicznego.
  - Rezultatem niedotrzymania ograniczeń czasowych jest tymczasowa niemożność zrealizowania założonej funkcjonalności, czyli obniżenie jakości usług oferowanych przez system. Możliwe jest, że skutki opóźnień będą negatywnie oddziaływać na otoczenie.
- Hard Real – Time Systems (HRTS); są to systemy z tzw. twardymi (ostrymi) ograniczeniami czasowymi
  - W systemach tego typu przekroczenia czasów granicznych są z założenia niedopuszczalne, natomiast jeśli występują, to tylko z powodu awarii systemu.
  - Przekroczenia podczas normalnej, bezawaryjnej pracy powodują awarię systemu i/lub jego otoczenia. Zatem, awaria może dotyczyć zarówno niemożności realizowania zadań systemu, jak i nieprawidłowego działania obiektu.

– Wystąpienie błędu przekroczenia czasu realizacji może wystąpić tylko z powodu awarii. Nieterminowe wypracowanie odpowiedzi jest błędem w równym stopniu co podanie odpowiedzi błędnej. Jeżeli występuje on przy normalnej, bezawaryjnej pracy, oznacza to, że system został błędnie zaprojektowany.

Formalny opis powyższego podziału bazujący na tzw. funkcji zysku przedstawiony jest w następnym podrozdziale.

Aby zwiększyć bezpieczeństwo funkcjonalne, systemy FRTS, a szczególnie HRTS powinny mieć mechanizmy obsługujące sytuacje awaryjne. Najczęściej są to układy redundantne [228], [199], [201], [202] lub niezależne układy bezpieczeństwa (ang. safety) [314] (zob. 3.4).

Dla poprawnego działania systemy typu HRTS muszą być odporne pesymistycznie, czyli odporne na tzw. lawinę zdarzeń [220]. Zjawisko to zachodzi, gdy w systemie jest zgłaszana do obsługi maksymalna liczba zdarzeń w oknach czasowych związanych z nadążnością systemu, a obsługa każdego z nich zajmuje maksymalny dopuszczalny czas im przydzielony. W praktyce łatwo jest stworzyć pseudo RTS, gdyż w normalnej pracy większość urządzeń komputerowych ma wystarczająco dużo mocy obliczeniowej, a sieci wystarczająco dużo wolnego pasma (ang. bandwidth), aby nadążać za otoczeniem. Problem pojawia się właśnie w sytuacjach stresowych, gdy rośnie liczba zdarzeń lub występuje niekorzystna koincydencja współzależności.

RTS może być konstruowany z podsystemów, przy czym podsystemy, w tym zagnieżdżone, mogą być różnego typu. Do określania typu danego systemu istotny jest charakter czasowy współpracy jego elementów (usług), a nie ich działanie jako takie. Systemy na bazie PLC oraz komputerowych sieci przemysłowych są klasyfikowane jako FRTS lub HRTS.

❖ Funkcja zysku

Istnieje użyteczne pojęcie formalnie opisujące podział systemów. Jest to tzw. funkcja zysku. Określa ona korzyść ze zrealizowania zadania przez system w określonym momencie czasu. Funkcja jest nierosnąca, czyli korzyść nigdy nie rośnie z czasem, natomiast może maleć.

Dla systemów Soft Real-Time funkcja zysku przyjmuje wartości:

$z(t) = u$ dla $t_0 < t < t_1$

$z(t) = u(t_t\text{-}t/t_t\text{-}t_s)$ dla $t_1 < t < t_1$

$z(t) = 0$ dla $t \geq t_t$

Dla systemów Firm Real-Time funkcja zysku przyjmuje wartości:

$z(t) = u$ dla $t_0 < t < t_t$

$z(t) = 0$ dla $t \geq t_t$

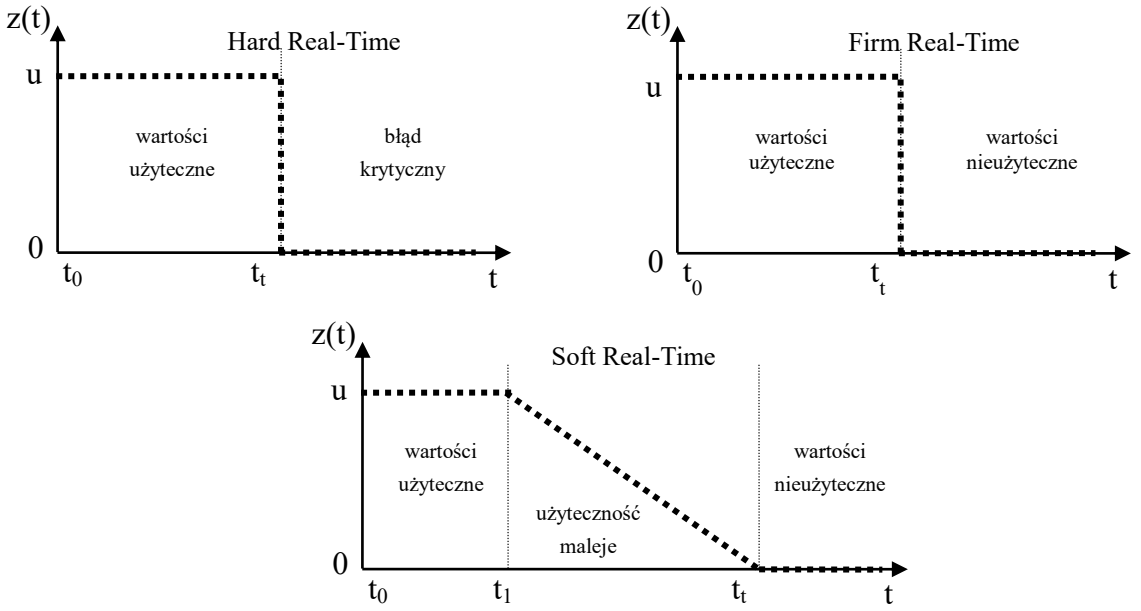
Dla systemów Hard Real-Time funkcja zysku przyjmuje wartości:

$z(t) = u$ dla $t_0 < t < t_t$

$z(t) = -\infty$ dla $t \geq t_t$

Ze względu na ograniczenia fizyczne t musi być większe od t_0, a przypadek $t \leq t_0$ jest w praktyce nierealny.

Funkcja zysku została zilustrowana na rysunku 5. Określenie wartości funkcji oznacza umowny wynik uzyskany w wyniku wykonania zadania. Może to być wartość zmiennej uzyskana jako wynik przetwarzania lub jako wartość dostarczona siecią itp.



Rys. 5. Ilustracja funkcji zysku dla Systemów Czasu Rzeczywistego
Fig. 5. Illustration of the gain function in RTS

Dla wszystkich systemów wartość funkcji zysku jest stała poniżej pewnego czasu granicznego. W systemach FRTS powyżej tego czasu funkcja przyjmuje wartość zero, co oznacza, że wynik wykonania zadania staje się nieprzydatny. W systemach SRTS powyżej pewnej wartości t_1 przydatność wyników zaczyna maleć aż do osiągnięcia zera w czasie t_t. Czasem przyjmuje się, że malenie funkcji nie jest liniowe, a wartość dąży do zera z upływem czasu. Rozpatrywanie przypadku ogólnego, gdy zadania wykonane szybciej dają wyniki bardziej przydatne, jest rozpatrywaniem systemów typu SRTS, gdy $t_1=t_0$.

Dla systemów typu HRTS wartość funkcji zysku po czasie t_t przyjmuje wartość nieskończenie małą. Należy to interpretować zarówno jako uzyskanie wyniku błędnego, i zatem jego nieprzydatność, jak i jako błąd działania. Sytuacja taka w normalnych warunkach pracy z założenia jest niemożliwa do osiągnięcia. Jej wystąpienie sygnalizuje błąd krytyczny działania systemu, porównywalny np. do dzielenia przez zero, i uniemożliwia jego dalszą po-

prawną pracę. Zabezpieczenia przed takimi sytuacjami realizują systemy bezpieczeństwa [124], [39] opisane w 3.4.

Dodatkowo w systemach z ograniczeniami czasowymi istnieją metryki związane z czasem granicznym. Są to miary:

- spóźnienia (ang. lateness) definiowanego jako czas opóźnienia realizacji zadania względem zdefiniowanego limitu
  $L = T_S - T_T$,
  dla przypadku idealnego spóźnienie wynosi $-T_T$,
- spóźnienia rzeczywistego (spóźnienia pozytywnego, ang. tardiness, positive lateness) definiowanego jako nieujemna wartość spóźnienia
  $max[0, L]$,
  gdy realizacja zadania jest terminowa, to miara przyjmuje wartość zero,
- terminowości (spóźnienia negatywnego, ang. earliness, negative lateness) definiowanego jako wartość „zapasu czasowego", czyli czasu pozostałego do czasu granicznego
  $max[-L, 0]$,
  dla przypadku idealnego spóźnienie to wynosi T_T.

❖ **Zarządzanie czasem**

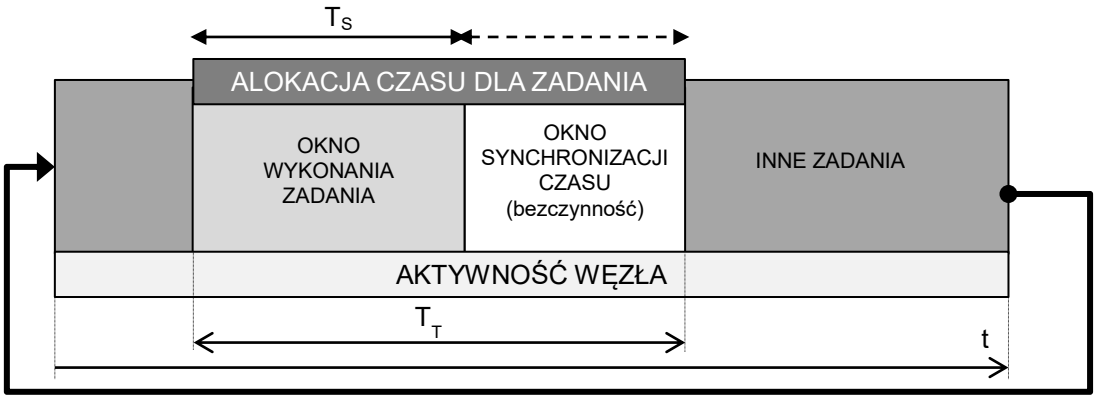
Zarządzanie czasem realizacji zadań z ograniczeniami czasowymi sprowadza się do zapewnienia mechanizmów uruchamiających zadania w odpowiednim porządku oraz kontrolujących czas ich wykonania. Istnieją dobrze znane mechanizmy zarządzania wykonywaniem zadań, pochodzące z teorii systemów rozproszonych [66]. W kontekście ISP do najistotniejszych można zliczyć:

- zadania uruchamiane w cyklach,
- kontrolę czasu uruchomień,
- kolejkowanie zadań,
- szeregowanie zadań,
- prioretyzację zadań,
- wywłaszczanie zadań.

Przy programowaniu PLC czy też przy obsłudze zdeterminowanej w czasie wymiany danych w sieciach komputerowych spotyka się i wykorzystuje wszystkie z powyżej wymienionych mechanizmów. Najważniejszym mechanizmem jest jednak praca w cyklach. Uogólniony cykl działań przedstawiono na rysunku 6.

Dane zadanie może się wykonywać w różnym czasie zależnie od warunków w danym momencie czasu. Dla każdego zadania alokowany jest czas, który może być synchronizowa-

ny do stałej wartości lub nie. Dzięki temu okres cyklu wykonywania zadań jest stały lub zmienny. Gdy brakuje synchronizacji choćby dla jednego zadania składowego, to zmienność jego czasu realizacji w każdym przebiegu cyklu (ang. sweep) skutkuje zmiennością okresu cyklu.



Rys. 6. Uogólnione umiejscowienie zadania w oknie działań cyklicznych
Fig. 6. Generalized placement of a task in the window of periodic actions

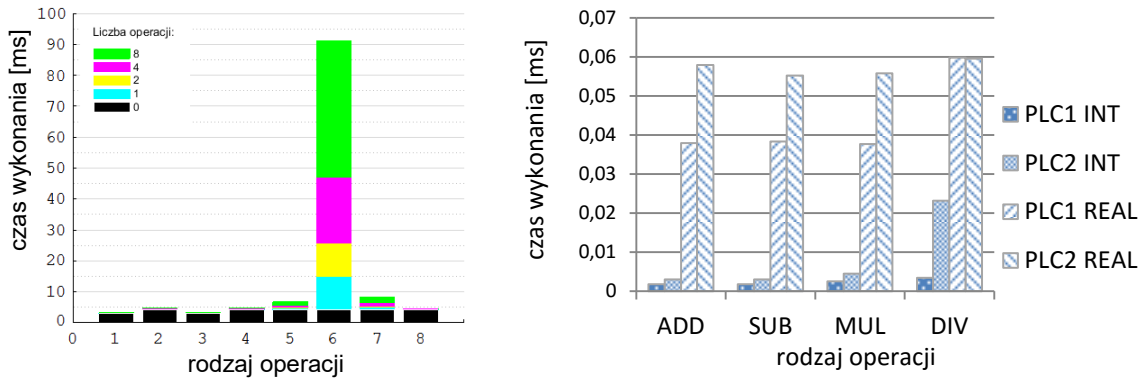
Dlatego w zarządzaniu cyklami stosuje się często synchronizację każdego przebiegu cyklu, a nie tylko jego elementów składowych. Pokazano to na rysunku 7. Sumaryczny czas realizacji *m* zadań jest stały i wynosi:

$$T_{S\_total} = \sum_{i=1}^{m} T_i + T_{SYN} = const$$

przy czym T_{SYN} jest wartością zmienną z zakresu $\left[0, T_{S_{total}} - \sum_{i=1}^{m} T_i\right]$. Oznacza to, że suma czasów zadań składowych $\sum_{i=1}^{m} T_i$ jest zmienna i zależy od zmienności czasów realizacji każdego z zadań. W praktyce zmienność ta wynika z czasów przetwarzania (np. czasu realizacji instrukcji i ścieżek przejścia w programie, czasów odpowiedzi i czasów oczekiwania w sieciach itp.).

Rys. 7. Ogólna idea synchronizacji, w każdym pojedynczym przebiegu cyklu
Fig. 7. General idea of synchronization in each single pass of the cycle

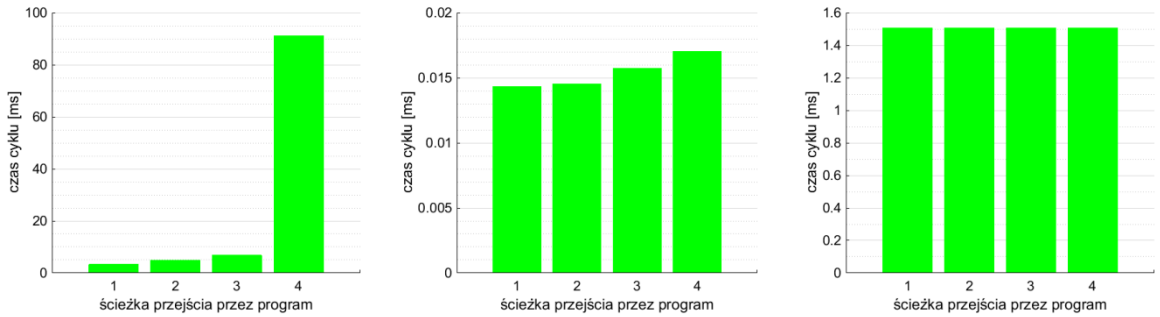
Na rysunku 8 przedstawiono różnice w czasach realizacji różnych instrukcji dla przykładowego sterownika klasy PLC (ang. Programmable Logic Controller). Ilustruje to, że w rzeczywistych programach czas wykonania może istotnie się różnić w zależności od danego przejścia w cyklu.



Rys. 8. Przykładowe czasy wykonywania operacji
Fig. 8. Sample commands execution times

Gdy realizacja zadania wymaga uwzględnienia choćby jednego warunku, wówczas czas realizacji w danym przebiegu będzie zależny od czasu realizacji względem wartości tego warunku w tym przebiegu. Przykładowe rozkłady czasów operacji dla tego samego programu i różnych sterowników przedstawiono na rysunku 9.

Więcej na tematy związane z pracą cykliczną i wykorzystaniem innych mechanizmów analizy i kontroli czasu wykonania zadań można znaleźć w [226], [229], [220] (zob. też 3.1.3).



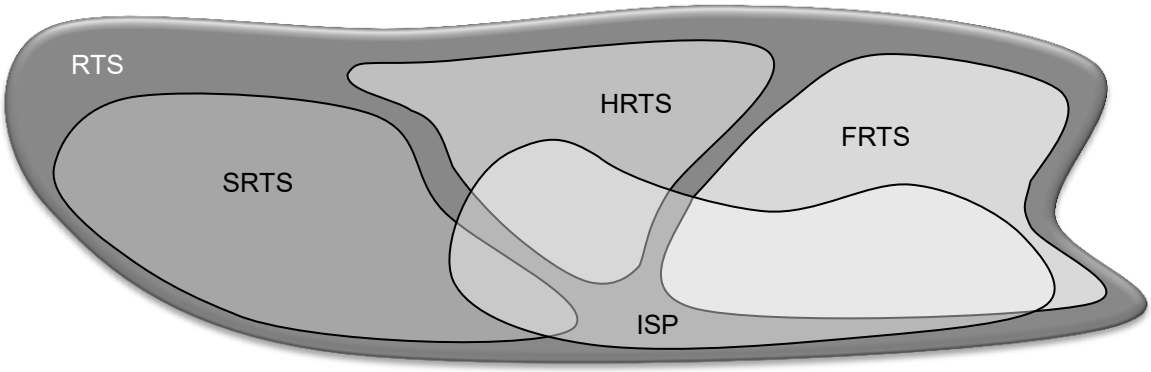
Rys. 9. Przykładowe rozkłady czasu wykonania prostego programu
Fig. 9. Sample distributions of simple program execution time

❖ Przykłady

Systemy przemysłowe są systemami klasy RTS, lecz są tylko jednym z ich rodzajów. Systemy przemysłowe mogą być klasyfikowane zarówno jako SRTS, FRTS oraz HRTS, zależnie od aplikacji. Najczęściej spotyka się systemy przemysłowe typu FRTS. Bardzo wymagające instalacje obsługujące niebezpieczne procesy tworzone są na bazie HRTS. Rzadziej

w przemyśle spotyka się systemu SRTS. Przykładowa ilustracja przynależności ISP do klasyfikacji RTS została przedstawiona na rysunku 10.

W praktyce mało jest informacji opisującej procesy, która z upływem czasu traci na ważności. Przeważnie jej bieżąca wartość jest istotna do pewnego momentu, po którym staje się niewiarygodna i nieprzydatna. Dla prostej ilustracji: wartość temperatury w podgrzewanym zbiorniku zmienia się w czasie z jakimś maksymalnym przyrostem w czasie związanym z wydajnością układu grzewczego, wpływem otoczenia i stratami. Pobieranie (akwizycja) takiej wartości może być prowadzone w cyklu. Cykl można dobrać do okresu, w którym zaistnieje możliwość wystąpienia zmiany na przetworniku pomiarowym. Po takim czasie wartość fizyczna temperatury na obiekcie może być inna niż wartość zmiennej ją reprezentującej, zaalokowanej w pamięci urządzenia. Zatem jej użyteczność do dalszego przetwarzania jest wówczas dyskusyjna i dla zachowania poprawnego obrazu procesu w pamięci należy ponownie pobrać jej wartość. Występuje zatem próg ważności informacji w czasie.

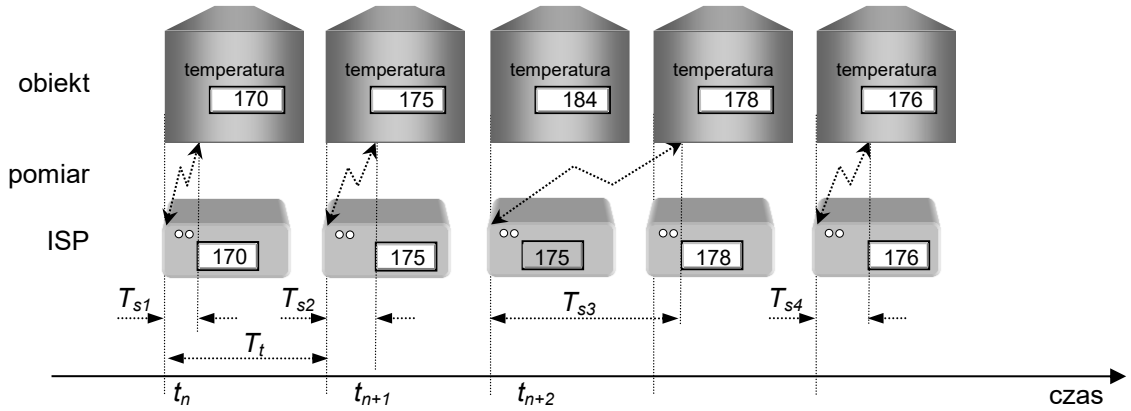


Rys. 10. Ilustracja ISP względem rodzajów systemów RT (proporcje orientacyjne)
Fig. 10. Illustration of ICS in relation to the types of RTS (approximate proportions)

Czas niezerowej użyteczności nie musi wynikać z czasów charakteryzujących zjawiska fizyczne. Może wynikać z technologii. Dla powyższego przypadku czas użyteczności pobranej wartości temperatury nie musi wiązać się z jej minimalnym a mierzalnym przyrostem w czasie, lecz może wynikać z wytycznych technologicznych określających jej tolerancję. Nie zmienia to faktu, że po przekroczeniu czasu, po którym tolerancja reprezentacji wartości może zostać przekroczona, użyteczność „starego" pomiaru, z punktu widzenia technologii, jest zerowa. Wówczas obraz procesu w systemie, w danym momencie czasu, nie odpowiada przyjętemu w założeniach obrazowi rzeczywistości. Zostało to zilustrowane na rysunku 11.

Maksymalne okno czasowe obserwacji systemu wynosi T_t. Abstrahuje się tutaj od wartości tego czasu. Może to być okres liczony w milisekundach, ale równie dobrze może to być okres wielominutowy. Spóźniony trzeci pomiar powoduje, że po czasie spóźnienia o wartości T_t wartość reprezentująca temperaturę na obiekcie nie odzwierciedla stanu tego obiektu w zadanej tolerancji czasowej. Zatem zapisana w pamięci wartość staje się nieprzydatna

z punktu widzenia technologicznego. W przykładzie jest to podkreślone, gdyż ISP w czasie $t_{n+2}+T_{s3}$ nie pozyskuje informacji o wartości 184. Dopiero po tym czasie pozyskuje wartość 178. Jeżeli w algorytmach przetwarzania znajduje się instrukcja warunkowa z warunkiem na wykrycie przekroczenia wartości 180 i specjalnej obsługi takiej sytuacji, to spóźniony odczyt spowoduje, że ani wykrycie sytuacji, ani jej obsługa nie nastąpi. W systemie HRTS może to prowadzić do awarii procesu, np. przegrzania surowca, zaburzenia procesu itp. Nie tylko brak wartości danych może powodować problem, ale i spóźnione uzyskanie takich wartości. Wówczas system będzie reagował z opóźnieniem wykraczającym poza określone normy technologiczne. W systemie FRTS zakłada się, że tego typu chwilowy brak poprawnej wartości lub jej incydentalne spóźnienie nie doprowadzi do awarii procesu.



Rys. 11. Ilustracja utraty ważności spóźnionej informacji
Fig. 11. Illustration of the loss of validity of delayed information

Przy okazji tego przykładu warto zwrócić uwagę, że irracjonalne zmniejszanie czasu pobrania kolejnych pomiarów do niczego pozytywnego nie prowadzi. Algorytmy przetwarzające nie będą przez to lepiej działać i generowane wyniki będą takie same jak dla okresów dłuższych. Otrzyma się natomiast gęstszy strumień informacji, który będzie niepotrzebnie zajmował zasoby. Bywa, że początkujący projektanci ISP definiują ograniczenia czasowe nie na podstawie wymagań technologicznych tylko na podstawie możliwości sprzętowych lub, co gorsza, nie zastanawiają się nad tym w ogóle, bazując na granicach oferowanych przez dane techniki informatyczne. Takie podejście powoduje, że w systemie przetwarzana przekazywana i składowana jest pewna ilość informacji nadmiarowej, która do niczego nie służy. Objawia się to negatywnie przy eksploatacji baz danych oraz przy ewentualnej rozbudowie systemu. Nadmiar danych w bazach powoduje ich niepotrzebny rozrost oraz zwiększenie czasu wyszukiwań. Natomiast podczas rozbudowy zapewnienie ograniczeń czasowych dla obsługi nowych danych może wymagać rekonstrukcji istniejących definicji.

W życiu codziennym systemy czasu rzeczywistego otaczają nas ze wszystkich stron. Większość ich użytkowników nie jest tego świadoma. W tabeli 2 przedstawiono kilka przy-

kładów, wykraczających poza ISP. W praktyce, ISP mogą być konstruowane na podstawie różnych wymagań względem reakcji na przekroczenie ograniczeń czasowych. Dlatego w tabeli znajdują się one w każdej z kategorii systemów RT. Jednak najczęściej są to systemy klasy FRTS i HRTS. Sporadycznie buduje się ISP na podstawie SRTS, gdy ich funkcjonalności sprowadzają się do wspomagającego raportowania, zarządzania lub diagnozowania.

Zachęca się Czytelnika do poszukania w swoim otoczeniu systemów informatycznych, które są i nie są zależne od czasu. Wnioski mogą być interesujące.

Tabela 2

Przykłady zastosowań SCR

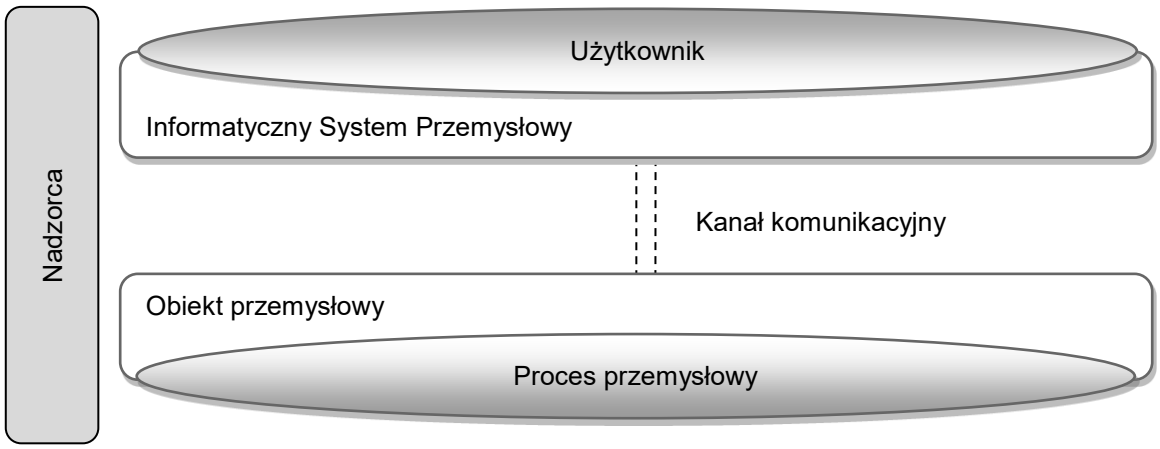
Systemy FRT/HRT	*Systemy SRT*
• systemy sterowania • systemy telefoniczne • systemy pomiarowe • monitoring on-line • teletransmisja istotnych danych • systemy pokładowe pojazdów, statków, samolotów, rakiet itp. • systemy kontroli ruchu • systemy militarne • systemy monitorowania pacjentów • multimedia/radio/TV • wiele innych…	• niektóre systemy automatyki • sieci bankowe/bankomatowe • systemy rezerwacyjne • systemy bazodanowe • systemy informacyjne • systemy nawigacyjne • wspomaganie handlu, pracy biur • telefonia internetowa • automatyka budynków • urządzenia powszechnego użytku • wiele innych…

1.1.5. System informatyczny

Dla uporządkowania pojęć warto zdefiniować, co to jest system, a w szczególności **system informatyczny**. Systemem nazywamy układ elementów powiązanych ze sobą różnymi zależnościami, a utworzony w celu spełnienia określonych funkcji. Natomiast wg słownika informatycznego przez system informatyczny rozumie się komplet programów i (lub) urządzeń pozostających w relacji komponentów, a przeznaczonych na ogół do realizacji wspólnego celu w zakresie przetwarzania danych [2]. Na potrzeby zastosowania systemu informatycznego w przemyśle powyższa definicja jest wystarczająca. Systemy informatyczne nie muszą być konstrukcyjnie i funkcjonalnie jednorodne. Mogą one być dekomponowane na mniejsze podsystemy, jak i integrowane w większe, obsługując odpowiednio węższy i szerszy aspekt prowadzenia procesu. Istnieje cały szereg różnego rodzaju systemów informatycznych, znajdujących zastosowanie w różnych dziedzinach obsługi produkcji przemysłowej (por. 1.2). Należy również rozróżnić pojęcie systemu informacyjnego od systemu informa-

tycznego. Dogłębna analiza tych pojęć przedstawiona została w [216]. W dalszych rozważaniach zajęto się systemami informatycznymi.

Rozważania w niniejszej książce dotyczą systemów informatycznych, mających kontakt z procesem przemysłowym. Ze względu na wspomnianą powyżej niejednorodną budowę systemów informatycznych rozważane systemy można traktować jako podsystemy większej całości. Rozważania nad bezpośredniością lub pośredniością kontaktu systemu z procesem domyślnie przywołują strukturę warstwową systemów. Współcześnie jednak może nie być to podstawą rozważań ze względu na trendy odchodzenia od separacji warstw architektury na rzecz jej ujednolicania. Separacji ulegają natomiast funkcjonalności systemu oraz przepływy informacji względem czasu. Dlatego omawiane systemy można potraktować jako skrót myślowy odnoszący się do wszelkich systemów informatycznych, mających bliższy lub dalszy kontakt z procesem przemysłowym. Na rysunku 12 przedstawiono ogólny schemat rozważanego układu. Jego główne elementy to **informatyczny system przemysłowy** (ISP) i obiekt przemysłowy.



Rys. 12. Podstawowe elementy rozpatrywanego środowiska
Fig. 12. Basic elements of the considered environment

ISP składa się z różnych elementów i powiązań między nimi zrealizowanych przez **kanał komunikacyjny**. Przez pojęcie kanału komunikacyjnego można rozumieć ogół środków danego typu, służących do przekazania informacji między urządzeniami. Kanał może być logiczny, bazujący na abstrakcyjnym połączeniu i środkach, oraz fizyczny związany z konkretnym połączeniem i realnymi środkami technicznymi. Rozważany system ma charakter **lokalny**, oznacza to, że obsługuje on określony i skończony zbiór informacji opisujący konkretny proces przemysłowy, przy czym nie ma znaczenia jego złożoność. Lokalność systemu można też rozpatrywać względem lokalności środków komunikacyjnych stosowanych w systemie. Do systemu lokalnego można zapewnić dostęp zdalny, czyli obsługę informacji

spoza powyżej określonego zbioru. Otrzymuje się wówczas system złożony (zintegrowany). Więcej na ten temat znajduje się w podrozdziałach 1.2.2 oraz 2.4.3.

Funkcjonalnie ISP może spełniać rolę systemu sterowania, choć jego zadania mogą daleko wykraczać poza taką rolę [31], [362], [304], [240], [343], [274] (por. 1.2). Jeśli kanałem komunikacyjnym jest **sieć komputerowa** (zob. 2.4.4), to mówi się o usieciowionych **systemach sterowania** (ang. NCS – Networked Control Systems) [19], [292] lub szerzej o rozproszonych **systemach sterowania** (ang. DCS – Distributed Control Systems) [289], [397], a w kontekście informatycznym o rozproszonych systemach komputerowych (ang. DCS – Distributed Computer Systems) [125], [105], [127], [138]. W dalszej części mówiąc o ISP traktuje się domyślnie o usieciowionych ISP.

Tworzenie systemów automatyki i informatyki ma na celu przede wszystkim zwiększenie jakości produktu i bezpieczeństwa produkcji oraz obniżenie kosztów i zwiększenie wydajności jego wytwarzania. W ISP czasu rzeczywistego każdy fizyczny sygnał stosowany do sterowania i regulacji układami wykonawczymi lub do zbierania danych z czujników musi być na bieżąco przetwarzany przez jakąś „inteligencję" czy to naturalną, lub sztuczną.

Obecnie człowiek działa jako projektant i budowniczy takich systemów oraz jako ich użytkownik. Jego rola jako użytkownika oddziałującego z infrastrukturą i aplikacją ISP jest zróżnicowana i zależna od przyznanego zakresu kompetencji. Role mają charakter nadzorczy i dotyczą pewnych ściśle zdefiniowanych obszarów funkcjonowania systemu. Są to zwykle funkcje powiązane z obsługą **procesu**, **nadzorem** (kontrolowaniem) produkcji i zarządzaniem procesem biznesowym. Zatem, z każdym systemem lub jego wydzieloną funkcjonalnością jest bezpośrednio powiązany jego lokalny **nadzorca** (ang. supervisor) lub grupa nadzorców wielospecjalistycznych, zainteresowanych zrealizowaniem wspomnianymi powyżej celami i poprawnością działania całości. Nie oznacza to jednak, że użytkownikiem nie może być inny system lub byt informatyczny. Im bardziej infrastruktura sprzętowo-programowa systemów jest zintegrowana, tym częściej zachodzi potrzeba współpracy procesów i maszyn, w tym również automatycznej kontroli poprawności ich działania. Mimo to nadrzędnym nadzorcą systemu jest zawsze człowiek. Nawet systemy bezobsługowe wymagają nadzoru prowadzonego przez człowieka względem wymaganych przeglądów i obsługi awarii.

Współcześnie, mimo istotnej i niepodważalnej roli człowieka jako nadzorcy, rolę głównego kontrolera (sterownika), czyli układu oddziałującego bezpośrednio z procesem technologicznym, sprawują komputery. Wynika to z faktu, że człowiek nie jest w stanie kontrolować przebiegu rzeczywistych zjawisk w stopniu wymaganym przez współczesną technologię.

Podstawowym elementem ISP jest tzw. **węzeł** systemu. Pełni on niepodzielną rolę w procesie przetwarzania informacji i stanowi odrębne urządzenie uczestniczące w wymianie informacji. Jeśli węzeł jest elementem systemu lokalnego, określa się go jako **węzeł lokalny**.

W przeciwnym razie jest to **węzeł zdalny**. Węzły lokalne odpowiadają za wykonywanie zadań funkcjonalnych systemu, natomiast zdalne stanowią zwykle interfejs dla prezentacji danych. Wybrane węzły lokalne mogą pełnić rolę **węzłów interfejsowych** (dostępowych, integracyjnych), realizujących przekazywanie danych do użytkowników i nadzorców. Z punktu widzenia sterowania i regulacji niektóre węzły lokalne mogą stanowić tzw. **źródła sterowania**, czyli ich aplikacje mają odpowiednie środki i prawa do wydawania rozkazów dla urządzeń wykonawczych. W ISP nie wszystkie węzły muszą pełnić rolę źródeł sterowania. W praktyce, najczęściej węzłami lokalnymi ISP są specjalne komputery (kontrolery, sterowniki, np. PLC por. 3.1) połączone siecią przemysłową (np. Profibus, EtherCAT itp. por. 3.3).

Obiekt przemysłowy jest związany z procesem technologicznym. Obsługa obiektu przemysłowego stanowi podstawowe zadanie funkcjonalne ISP. Dlatego, aby system informatyczny klasyfikować jako ISP, musi istnieć przynajmniej jeden element systemu komunikujący się z obiektem przemysłowym. Dla dalszych rozważań przyjęto, że elementami systemu IT mogą być urządzenia komputerowe, programy oraz inne systemy, stanowiące podsystemy rozważanego systemu.

1.1.6. Przetwarzanie danych

W celu realizacji zadań funkcjonalnych systemu musi zostać zdefiniowane, dedykowane dla danego ISP przetwarzanie danych w postaci tzw. aplikacji. Pojęcie to jest bardzo wieloznaczne, mimo że dla systemów informatycznych ma zawsze wspólne źródło, wywodzące się z warstwowego podziału funkcji oprogramowania. Przez **aplikację** w kontekście przetwarzania danych w ISP rozumie się konkretne oprogramowanie użytkowe, pracujące na rzecz danego obiektu, odpowiedzialne za oferowanie i realizację funkcjonalności wymaganej przez użytkownika. Na aplikacje może składać się jeden lub wiele programów działających na jednym lub wielu urządzeniach. Przez **program** rozumie się zbiór logicznie powiązanych elementów języka programowania. Program może być zapisany w postaci źródłowej, wymagającej odpowiedniej translacji w celu jego wykonania lub w postaci wykonywalnej. Programy działające na rzecz funkcjonowania aplikacji nazywa się programami aplikacyjnymi lub czasami programami użytkownika. Więcej na temat pojęcia aplikacji znajduje się w podrozdziale 2.4.5, a o programach dla ISP w 3.2.

W dziedzinie ISP, oprócz pojęcia procesu przemysłowego, pojawia się również pojęcie procesu w rozumieniu przetwarzania informacji. Proces przemysłowy i proces przetwarzania nie są w żadnym stopniu tożsame. **Proces** w kontekście **przetwarzania danych** jest rozumiany jako uruchomiony program lub zbiór logicznie powiązanych i uruchomionych programów składających się na wykonywaną przez dany zbiór zasobów aplikację. Zatem pojęcia

procesu używa się w tym wypadku do określenia programu uruchomionego, czyli posiadającego urzeczywistnienie (instancję) w zasobach komputera i wykonywanego przez te zasoby.

❖ **Zasoby**

Przez **zasoby** komputera rozumie się ogół środków sprzętowo-programowych umożliwiających wykonanie programów aplikacyjnych. Zasoby niektórych urządzeń dedykowanych do pracy w ISP, szczególnie sterujących (sterowników, kontrolerów, zob. 3.1), są wyraźnie uwypuklane. Stąd występują typowe pojęcia, określające zasoby na ogólnym poziomie rozróżnialności funkcyjnej sprzętu. Są to:

- **jednostki centralne** (przetwarzające, ang. CPU – Central Processing Unit). **Jednostka przetwarzająca** jest jednostkowym zestawem zasobów wykorzystywanym do uruchamiania programu. Innymi słowy, jest to komputer z procesorem, pamięcią, magistralami, układami sterującymi i innymi układami wewnętrznymi wynikającymi z jego architektury;
- **układy zasilające** (zasilacze, ang. PS – Power Supply). Zasilacze dostarczają odpowiedniego zasilania dla jednostek centralnych oraz innych modułów sprzętowych. Przeważnie nie biorą udziału w przetwarzaniu informacji, ale w niektórych konstrukcjach uczestniczą w zadaniach diagnostyki;
- **układy IO** (wejścia/wyjścia, ang. Input-Output, SM – Standard Module). Zajmują się obsługą wejść i wyjść dyskretnych oraz analogowych, czyli tzw. standardowych interfejsów wejścia i wyjścia;
- **układy specjalizowane** (funkcyjne, FM – Function Module, IM – Interface Module). Zajmują się obsługą niestandardowych interfejsów wejścia i wyjścia, czyli portów komunikacyjnych, interfejsów sieciowych oraz interfejsów dla urządzeń procesowych (np. termopary, impulsatory, enkodery, serwonapędy itp.).

Dostępność i wymiar ilościowy takich zasobów określa możliwości urządzenia zarówno względem ogólnego przetwarzania, jak i specjalizacji w kontekście aplikacji przemysłowych. W pierwszym przypadku zasoby określają:

- przestrzeń pamięci dla programu i danych (np. rozmiar programu, organizacja i liczba programów, zachowywanie, alokacje, adresacje),
- szybkość wykonania programu (np. operacji na sekundę),
- zakres przetwarzania (np. operacje, typy, zegar RTC, zmienny przecinek) itp.

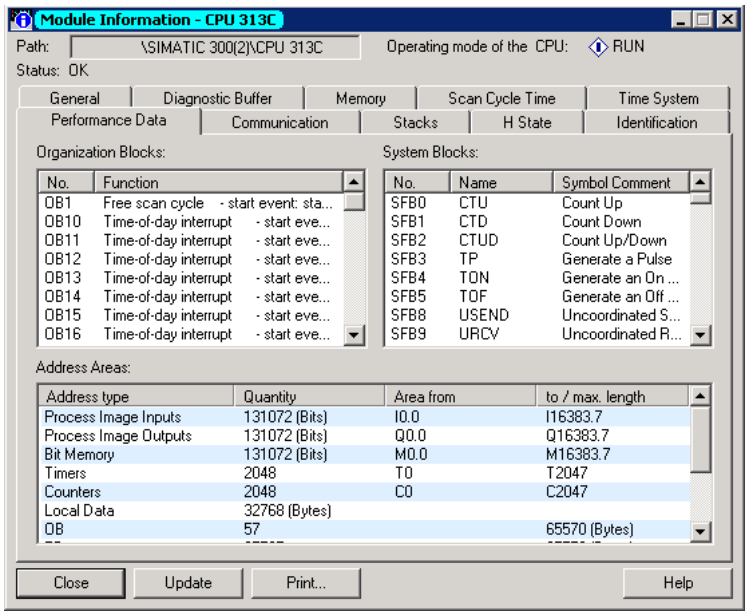
W drugim możliwości dotyczą:

- przestrzeni wejść/wyjść (np. dyskretnych, analogowych, cyfrowych, adresacji, alokacji, zachowywania itp.),

- instancji prealokowanych obiektów programowych lub rozmiaru pamięci dla ich alokacji (np. liczników, timerów, markerów, stref systemowych),
- programowych obiektów funkcyjnych (np. regulatory, obsługa funkcji specyficznych),
- interfejsów specjalizowanych i wszelkich peryferiów (np. sieci polowe, protokoły RT, obsługa modułów funkcyjnych).

Zestawienie takich cech jest charakterystyczne dla danego zestawu zasobów i stanowi prezentację urządzenia (ang. performance), określającą jego dostępne możliwości w rozumieniu maksymalnym. Przykład takiej prezentacji przedstawiono na rysunku 13.

Formalna definicja wszystkich zasobów danego sterownika wraz z jego peryferiami (układy IO, koprocesory itp.) określona jest jako **konfiguracja** i jest przechowywana w specjalnej strefie pamięci, tzw. pamięci konfiguracji (ang. HWC, HC – Hardware Configuration). Konfiguracja zasobów jest niezbędna ze względu na specyfikę budowy urządzeń. Komputery sterujące są z reguły modułowe z konfigurowalną przestrzenią IO. Zarówno system operacyjny urządzenia, jak i narzędzie deweloperskie musi posiadać informację, z czym w danym przypadku mają do czynienia, aby poprawnie wykonać czynności diagnostyczne, walidacyjne i translacyjne związane z jego funkcjonowaniem i oprogramowaniem.



Rys. 13. Zestawienie możliwości urządzenia sterującego[4]
Fig. 13. Performance characteristics of a controller

[4] Prezentowane okno pochodzi z programu narzędziowego Step7 oraz CPU serii Simatic S7-300.

❖ **Kod i zadania**

Przez **kod** rozumie się zestaw poleceń dla jednostki przetwarzającej zapisanych przy użyciu języka programowania. Na kod składają się słowa kluczowe języka (elementy języka, ang. keyword), operandy, operatory, funkcje, bloki funkcyjne, literały, etykiety, parametry i inne jednostki leksykalne. Pojęcie kodu może dotyczyć również użycia języków formalnego opisu działania tzw. języków modelowania, które operują na abstrakcjach oderwanych od fizyczności zasobów sprzętowo-programowych urządzeń. Więcej na temat pojęć związanych z językami programowania można znaleźć w [249], [40], a modelowania w [46], [113], [29]. W dalszej części książki pojęcie kodu będzie związane zawsze z językiem programowania, a nie modelowania. Zatem kod nie będzie utożsamiany z opisem modelu działania aplikacji dokonanym w jakimś języku formalnym np. UML [404], a będzie związany z zapisem wykonywalnego programu lub jego fragmentu.

Sekwencja kodu mająca wyróżniony początek, koniec oraz identyfikowana przez nazwę lub numer będzie nazwana **jednostką kodu** lub **POU** (ang. Program Organization Unit). POU może mieć formę funkcji, bloku funkcyjnego lub programu, a samo pojęcie może dotyczyć zarówno źródła kodu, jak i instancji.

Do sterowania wykonywaniem programów często stosuje się pojęcie **zadania**. Jest to abstrakcyjny element kontrolny, umożliwiający uruchamianie powiązanych z nim wykonywalnych jednostek kodu (programu, programów, bloków), zgodnie ze zdefiniowanym sposobem wyzwalania. Zależnie od tego sposobu programy mogą być uruchamiane cyklicznie lub zdarzeniowo, a także wstrzymywanie i wznawiane, jeśli kontrola przewiduje użycie mechanizmu wywłaszczania (por. 2.1.2). Pojęcie zadania wykorzystywane jest w normie IEC61131 [M26] (zob. 2.6).

❖ **Informacja, zmienne i dane**

Proces przemysłowy na potrzeby obsługi przez ISP wymaga określenia zbioru informacji cyfrowej opisującej obiekty przemysłowe. Informacja ta stanowi przedstawienie własności obiektów i relacji między nimi. Do reprezentacji informacji w procesie tworzenia kodu (programowania, kodowania, ang. programming) używa się pojęcia **zmiennej** (ang. variable). Zmienna jest pojęciem abstrakcyjnym wiążącym cyfrową reprezentację informacji z jej identyfikacją używaną w kodzie. Identyfikacja może być wyrażona w postaci nazwy symbolicznej lub adresu. Zmienne identyfikowane przez podanie bezpośredniego adresu do elementu pamięci nazywane są zmiennymi **bezpośrednimi** (ang. direct variables). Każda zmienna w działającym systemie musi mieć swoją instancję, czyli musi wiązać się fizyczną alokacją identyfikowaną przez adres i rozmiar w pamięci urządzenia. Zmienna posiada swój **typ** prosty (elementarny) określający sposób, w jaki informacja jest reprezentowana w pamięci. Na

sposób ten składa się użyte **kodowanie** i rozmiar alokacji. Kodowanie określa rodzaj użytego zapisu binarnego, rozmiar określa przestrzeń bitową, na której dokonywane jest kodowanie. Ogólniej, typem danych określa się dostępny zbiór wartości wraz ze zbiorem operacji dozwolonych do wykonania na tym zbiorze wartości. Zmienne typów prostych mogą być agregowane w typy złożone, jak tablice lub struktury [377], [112].

Zmienne przyjmują wartości, które zmieniają się w czasie na skutek działania aplikacji i obiektu. Wartości zmiennych w danym momencie czasu to **dane**, na podstawie których wykonuje się program. Dana jest konkretną wartością podlegającą procesowi przetwarzania i jest zawsze związana ze sposobem jej kodowania, i co za tym idzie interpretacji. Zatem takie same dane w danej interpretacji mogą dostarczać różnych informacji w innej. Dla przykładu, ciąg binarny 1001 1000 może oznaczać liczbę 152 (kodowanie na 16 bitach) jak również -104 (kodowane na 8 bitach ze znakiem), ale równie dobrze jest to wartość 98 w kodzie BCD lub też informacja w postaci ciągu bitów (ang. bitstring), że czwarte, piąte i ósme wejście dyskretne jest aktywne. Wszystko zależy od interpretacji danych, czyli typu zmiennej. Również konkretna informacja może być reprezentowana przez różne dane. Dla przykładu, temperatura o wartości 17,51 może być reprezentowana jako 0000 0110 1101 0111 (typ integer na 16 bitach), jako 0001 0111 0101 0001 w kodzie BCD lub jako 0100 0001 1000 1100 0001 0100 0111 1011, czyli liczba zmiennoprzecinkowa kodowana cechą i mantysą wg IEC60559 [M74], (zob. też IEC24732 [M72], IEEE754 [M60]) na 32 bitach.

Ponadto, dane mają swój kontekst użycia, który wiąże się z przypisaniem do funkcjonalności systemu. Określenie kontekstu nie wpływa bezpośrednio na zasady przetwarzania, ale może pomóc przy tworzeniu struktur danych. Dane mogą opisywać chwilowy stan procesu, systemu lub całej produkcji. Do opisu informacyjnego procesu przemysłowego wykorzystuje się tzw. **dane procesowe**, do opisu stanu wewnętrznego systemu **dane systemowe,** a do opisu stanu produkcji **dane produkcyjne** związane z prowadzeniem produkcji, na którą może składać się działanie wielu procesów i systemów.

Dobór typu zmiennej w ISP jest istotny w takim samym stopniu jak w innych rodzajach systemów informatycznych. Szczególnie należy zwracać uwagę na fizyczny zakres wartości zmiennych procesowych oraz ich istotność w dziedzinie rozdzielczości możliwych zmian. Często urządzenia ISP mają zdecydowanie bardziej ograniczone zasoby niż urządzenia systemów biurowych, dlatego programista powinien dbać o dobre wykorzystanie pamięci. Dla przykładu, pomiaru z przetwornika 12-bitowego nie ma potrzeby zapisywać w słowach 32-bitowych, a liczba reprezentująca jego wartość niekoniecznie musi być typu real, jeśli zmiany są w zakresie ±10 jednostek i dwóch miejsc po przecinku.

W uogólnieniu systemowym zmienna i związana z nią w czasie wartość są nazywane **jednostką danych** (ang. PDU – Process Data Unit) i może się odnosić zarówno do danych

z procesu przetwarzania, jak i danych z procesu technologicznego. Informacja w postaci zbioru wartości zmiennych związanych z procesem przemysłowym, urządzeniem, programem lub innym elementem systemu w danym momencie czasu stanowi **stan** informacyjny tego elementu. Na stan elementu wpływa jego działanie jak i oddziaływania zewnętrzne innych elementów, w tym nadzorcy. Nadzorca wpływa na stan przez **parametryzację**, czyli zmianę wartości wybranych zmiennych w czasie działania elementu, bądź przez **konfigurowanie** elementu przed jego użyciem.

Ze zmiennymi binarnymi (dyskretnymi, boolowskimi) związane są pojęcia tranzycji (zmiany, przejścia) stanu w dziedzinie czasu. Wyróżnia się zmianę z zera na jeden oraz z jeden na zero i określa się odpowiednio jako **zbocze narastające** i zbocze **opadające**. Analogia do zboczy wynika z porównania zmian stanu zmiennych do zmiany stanów sygnałów reprezentowanych przez te zmienne w czasie. W praktyce odniesienie to stosuje się również do zmiennych typu boolean i innych elementów dyskretnych (np. zdarzeń, progów), niemających bezpośrednich powiązań z realnymi sygnałami.

❖ **Zakres funkcjonalny przetwarzania**

ISP mają specjalizowaną dziedzinę zastosowań, a co za tym idzie aplikacje mają specyficzny i charakterystyczny zakres funkcjonalny. Wśród typowych aktywności aplikacji ISP znajdują się takie zadania, jak:

- **sterowanie** – (ang. control)
  Oddziaływanie na proces przemysłowy umożliwiające kontrolowanie jego przebiegu i zmianę stanu obiektu. W praktyce oddziaływanie realizowane jest przy użyciu nośników informacji, jakimi są sygnały (dyskretne, ciągłe i cyfrowe, por. 1.2.4), najczęściej o charakterze elektrycznym. Do realizacji zadań sterowania przeważnie wykorzystuje się działania wyzwalane zdarzeniami cyklicznymi.

- **obserwacja** – (ang. acquisition)
  Zbieranie informacji, które dotyczą bieżącego stanu procesu. W praktyce, tak jak sterowanie, akwizycja informacji realizowana jest przy użyciu sygnałów.

- **monitorowanie** – (ang. monitoring)
  Obserwacja przebiegu procesu z jednoczesnym śledzeniem jego nieprawidłowości zdefiniowanych według określonych reguł. Przeważnie monitorowanie dotyczy obserwacji wielu wielkości w długim okresie czasu i jest realizowane bez przerw w działaniu, nawet w sytuacji awarii bądź wyłączenia monitorowanego obiektu. Monitorowanie dotyczy wartości bieżących i umożliwia ich logowanie. Przeważnie wykorzystuje się zdarzenia asynchroniczne.

- **wizualizacja** przemysłowa – (ang. visualization)
  Prezentacja stanu procesu przy użyciu środków umożliwiających percepcję i zrozumienie informacji przez człowieka. Najczęściej do wizualizacji wykorzystuje się środki multimedialne kontrolowane przez układy komputerowe. Dość często stosuje się również układy synoptyczne. Funkcjonalność wizualizacji jest często integrowana z funkcjami nadzorczymi w oprogramowaniu typu SCADA (por. 3.2).

- **raportowanie** – (ang. reporting)
  Prezentacja danych statystycznych przebiegu procesu, trendów i podsumowań oraz zgłaszanie sytuacji nieprawidłowych. Wiąże się ono przeważnie z analizą danych zalogowanych z przebiegu procesu w jakimś okresie. Raportowanie może być traktowane jako warstwa prezentacyjna dla monitorowania długoterminowego, działająca w trybie off-line.

- **zarządzanie** – (ang. management)
  Jest to pojęcie bardzo szerokie, a znaczenie zależy od kontekstu. W ISP funkcje zarządzania dotyczą oddziaływania na elementy systemu celem zapewnienia poprawnego bieżącego i przyszłego funkcjonowania. Wchodzą w to funkcjonalności administracyjne, usługi aktualizacji, bieżącego utrzymania, w tym diagnostyki, wsparcia dotyczącego pielęgnacji środowisk programowych i ich rozwoju, wsparcia związanego z obsługą serwisową urządzeń, konfiguracji integracji oraz wiele innych.

- **nadzór** – (ang. supervision)
  Kontrolowanie poprawności działania procesu i systemu, zapewnianie jego parametryzacji, konfiguracji i diagnostyki. Funkcjonalność często jest integrowana z wizualizacją w oprogramowaniu typu SCADA (por. 3.2).

- **diagnostyka** – (ang. diagnostics)
  Detekcja nieprawidłowości działania elementów procesu i systemu, wskazywanie ich źródeł i przyczyn.

Należy pamiętać, że tworzenie aplikacji wiąże się ze sztuką projektowania oraz sztuką programowania. Wobec tego bardzo istotne jest kreacyjne postępowanie w obu przypadkach, ale zgodne z techniczną wiedzą inżynierską. Niestety nie ma jednego idealnego przepisu na stworzenie poprawnej aplikacji. Jest to sztuka, a zatem postaci finalnych może być wiele. Dlatego ważne jest, aby stosować się do zasad inżynierii oprogramowania [166], [181], [256], reguł, norm i wypracowanych postępowań [197], [145] oraz zatrudniać kreatywnych twórców.

1.1.7. Przekazywanie danych

Komunikacja w ISP ma za zadanie utrzymanie spójności danych w systemie, czyli zapewnienie przekazywania danych pomiędzy aplikacjami węzłów. Bardzo istotne jest podkreślenie, że zadaniem funkcjonalnym komunikacji jest przekazywanie danych właśnie pomiędzy aplikacjami pracującymi w węzłach, a nie pomiędzy jakimiś bliżej nieokreślonymi elementami typu pamięć, karta sieciowa, interfejs, komputer, abonent, stacja itp. Informacja musi być rozprowadzona na fizycznym obszarze działania systemu i z ograniczeniami czasowymi. Wiąże się to z koniecznością zachowania cech spójności informacyjnej przestrzennej i czasowej opisanych w 2.3.

Do realizacji przekazywania danych potrzebne są elementy sprzętowo-programowe różnego rodzaju. Jednak, z punktu widzenia funkcjonalnego ISP komunikują się aplikacje, a nie np. komputery. Fizycznie przekazywanie danych realizowane jest przy użyciu sieci komputerowych. Przez **sieć komputerową** rozumie się ogół środków sprzętowo-programowych umożliwiających przekazanie informacji pomiędzy podłączonymi do niej komputerami. Sieci komputerowe wykorzystywane w ISP są nazywane lokalnymi **sieciami systemowymi**, a potocznie komputerowymi **sieciami przemysłowymi** lub przemysłowymi **sieciami informatycznymi**. Węzły ISP podłączone do danej sieci są **abonentami** tej sieci lub węzłami usieciowionymi. Pojęcie sieci przemysłowych w szerszym kontekście jest wyjaśnione w [353] oraz w standardach ANSI/ISA. W niniejszej publikacji pojęcie sieci przemysłowej będzie rozumiane tylko jako komputerowe sieci przemysłowe.

Na poziomie abstrakcji używa się pojęcia kanału komunikacyjnego (por. 2.2), w którym zachodzą określone wymiany zapewniające pożądane przekazanie informacji. Przez **wymianę** rozumie się przekazanie dowolnej informacji ze stosu protokołów danego węzła do stosu protokołów drugiego węzła, wykonane z użyciem pojedynczej aktywności warstw stosów i medium. Wymiany sieciowe **poziome** są to wymiany, które zachodzą między warstwami stosu protokołów węzłów stanowiących lokalny ISP. W praktyce są to wymiany zachodzące pomiędzy węzłami systemu lokalnego. Wymiany sieciowe **pionowe** są to wymiany zachodzące pomiędzy różnymi warstwami stosu węzłów lokalnych a węzłów interfejsowych, i to one realizują komunikację aplikacji użytkowników i nadzorców z aplikacjami węzłów lokalnych, czyli w praktyce np. integrację systemów (por. 3.5) i interfejsy dla człowieka.

Z punktu widzenia warstw aplikacji zadaniem sieci jest przekazanie wartości zmiennych. Dane takie w kontekście sieciowym są nazywane **danymi użytecznymi** (ang. payload, useful data). Pozostałe dane przekazywane w wymianach są danymi pochodzącymi od poszczególnych warstw stosu protokołów i służą tym warstwom. Zależnie od protokołu do przekazania danych użytecznych może być potrzebnych wiele wymian. Nieprzerywalna sekwencja wymian prowadząca do przekazania danych użytecznych między aplikacjami węzłów nazywana

jest **transakcją** sieciową. Transakcja może być zrealizowana w całości albo wcale. Niezakończona poprawnie transakcja nie może zmienić stanu węzła.

W dziedzinie komunikacji w ISP istnieje, tak jak w przypadku aplikacji, kilka charakterystycznych aktywności. Wśród typowych znajdują się takie, jak:

- **Transakcje cykliczne** – (ang. cyclic transactions, cyclic exchanges, periodic)
  Bazują one na predefiniowanym **schemacie** zwanym **scenariuszem wymian** (ang. schema, scenario). Przyjmuje się traktować transakcje cykliczne jako podstawowy mechanizm utrzymywania spójności danych w systemie zarówno pod względem czasowym, jak i przestrzennym. Wynika to z faktu, że transakcje typu cyklicznego działają z ograniczeniem czasowym swojego cyklu oraz mają naturalną cechę powtarzalności, co eliminuje potrzebę repetycji w razie stwierdzenia problemów z transmisją.

- **Transakcje acykliczne** – (ang. acyclic, on request, sporadic, aperiodic)
  Transakcje tego typu nie są predefiniowane w scenariuszu, lecz są zlecane do wykonania przez aplikacje w nieokreślonych momentach czasu, wynikających z działania algorytmów. Realizacja takich wymian nie odbywa się jednak w dowolnych momentach czasu, a podlega pod szeregowanie wynikające z działania scenariusza.

Powyższe dwie aktywności są podstawowe w systemach przemysłowych i w praktyce, mimo stosowania wielu kompletnie różnych sieci i pracujących w nich protokołów, główna idea działania sieci zawsze bazuje na tych aktywnościach.

Scenariusz wymian działa na bazie cyklicznego aktywowania zadań sieciowych i jest budowany na podstawie znanych modeli działania sieci, takich jak Master-Slave, Token Passing, Producer-Distributor-Consumer, Time Slicing, czy innych stanowiących większą lub mniejszą wariację lub hybrydę wymienionych. Wszelkie aktywności sieci muszą być ograniczone czasowo i dlatego predefiniowany schemat działania w postaci scenariusza wymian stanowi podstawę do ich uruchamiania. Wszelkie aktywności w tym przypadku zachodzą zawsze w określonych elementarnych oknach czasowych, w tym w powtarzalnych oknach cyklicznych lub w ich seriach, w i prealokowanych oknach przeznaczonych na aktywność acykliczną. Realizacja transakcji w oknach zależy od predefinicji aktywności cyklicznych i alokacji czasu dla aktywności acyklicznych zawartych w scenariuszu. Powtarzany i stały zestaw okien elementarnych stanowi **cykl sieci** (ang. network cycle) i gwarantuje zdeterminowane w czasie przekazywanie danych. Aktywności w każdej sieci przemysłowej zachodzą ściśle według takiego cyklu.

1.2. Dziedzina funkcjonowania ISP

ISP stosowane są w różnych obszarach zastosowań. Są to zarówno obszary automatyki przemysłowej (ang. automation), automatyzacji procesów (ang. process automation), sterowania napędami (ang. drive control), automatyki budynków (ang. building automation), jak i obszary systemów samochodowych (ang. automotive systems), dystrybucji energii (ang. power distribution), metropolitalnych (ang. metropolitan) i innych. Obserwując przestrzeń ostatnich kilkudziesięciu lat, można stwierdzić, że zaangażowanie ISP w różnych dziedzinach gospodarki stale rośnie.

Z historycznego punktu widzenia komputery od dawna były używane w automatyce [222]. Wykorzystywano wówczas dość potężne, jak na tamte czasy, urządzenia używane jako centralne stacje przetwarzania lub jako węzły systemów zdecentralizowanych, powszechnie zwanych DCS (por. 3.1). Główną ich rolą w tym przypadku była ciągła kontrola procesu przemysłowego. Układy dedykowane dla sterowań dyskretnych były projektowane na podstawie przekaźników, a wraz z rozwojem technologii zostały przekształcone proste układy mikroprocesorowe znane jako pierwsze PLC [28], [210]. Dalsze prace rozwojowe skutkowały adaptacją bieżących technologii, zarówno sprzętowych, jak i programowych, do konstrukcji urządzeń. Aktualnie, różnice między komputerami DCS oraz PLC nie są zbyt wyraźne z punktu widzenia konstrukcyjnego. Czasami w obydwu przypadkach wykorzystuje się dokładnie tę samą bazę sprzętową. Inne grupy komputerów, takie jak PC, IPC, HMI i inne, wyposażone w interfejsy dla człowieka, są najczęściej traktowane jako punkty wymiany informacji między światem cyfrowym a człowiekiem [110]. Stosowanie takich urządzeń na poziomie procesu przemysłowego przysparza problemów z właściwym dotrzymywaniem warunków czasowych (ang. timelining) [54] (zob. 2.7).

Środowisko przemysłowe jest dość specyficzne, jeśli je porównywać względem środowisk biurowych. Charakteryzuje je niezmienność oraz duża bezwładność w adaptacji nowych technik i technologii z dziedziny IT. Niezmienność wynika z faktu, że proces przemysłowy jest stały w opisie i działaniu dla danej technologii produkcji. Nie ulega on modyfikacji, dopóki nie ulega zmianie technologia lub nie następuje modernizacja infrastruktury. Wiąże się z tym stałość opisu informacyjnego procesu i stałość obsługujących go algorytmów. Przekłada się to na niezmienność liczby zmiennych i ich reprezentacji, a także niezmienność działającej aplikacji i w konsekwencji na niezmienność ISP. Bezwładność w modernizacji systemów informatycznych wynika z podobnych przyczyn. Poprawna obsługa procesu przemysłowego skutkuje oczekiwanymi rezultatami w postaci produktu wytwarzanego z żądanymi parametrami. Dopóki nie istnieje potrzeba zmiany parametrów produkcji, zmiany interakcji systemu z otoczeniem lub nie występują powody wynikające z utrzymania ruchu, to nie ist-

nieje potrzeba zmian w danym ISP. Nieuzasadnione modernizacje generowałyby zbędne koszty inwestycyjne i straty z przestojów.

Wiele produktów można wytworzyć przy użyciu systemów automatyki sterowanych analogowo, ręcznie lub wręcz przy użyciu manufaktury. Jednak do uzyskania bardziej wyrafinowanych produktów, o lepszej jakości, z większą wydajnością i przy niższych koszach niezbędne jest stosowanie systemów automatyki wspomaganych komputerowo. Większość przypadków aplikowania ISP odnosi się do systemów sterowania działających w czasie rzeczywistym. Oprócz tego, istnieją jednak dziedziny zastosowań, dla których systemy ISP nie muszą wykazywać wszystkich cech systemów czasu rzeczywistego. Są to dziedziny niezwiązane bezpośrednio z procesem technologicznym lub w których szczególny stosunek do ograniczeń czasowych zależy od wymagań aplikacji. Jest to szczególnie widoczne, gdy weźmie się pod uwagę integrację międzysystemową (por. 3.5) i możliwość zapewnienia zdalnego dostępu w środowisku heterogenicznym. Dlatego w usieciowionych ISP rozwiązania klasy Soft-RT są jak najbardziej pożądane, szczególnie w tych, które działają w strefie komunikacji intranetowej i publicznej.

1.2.1. Przedsiębiorstwo

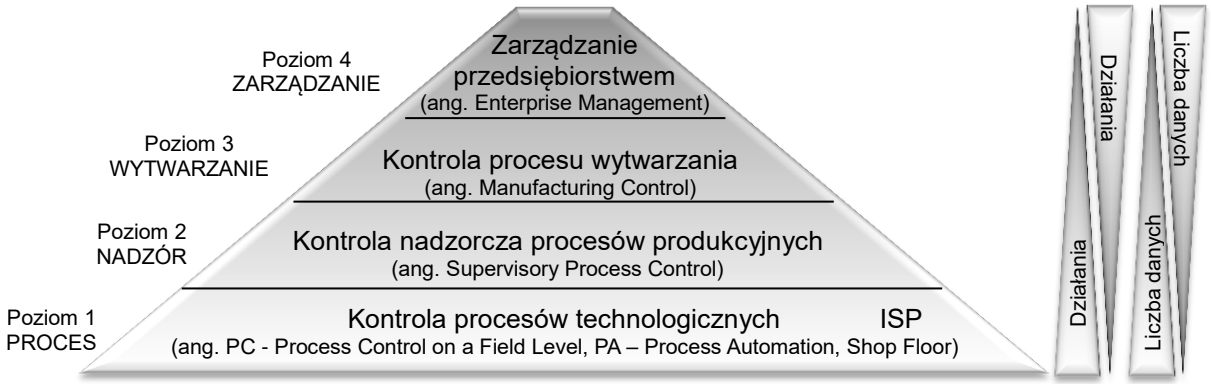
Związki komputerów z funkcjonowaniem zakładu przemysłowego istnieją na poziomie produkcji, jak również na dowolnie innym poziomie wynikającym z przepływu informacji w przedsiębiorstwie. Dotyczy to związków pomiędzy systemami informatycznymi obsługującymi produkcję oraz ogólną i szczegółową organizację biznesu, w tym zbyt, zarządzanie, planowanie, dostawy, współpracę działów itp. Modelowanie komputerowych systemów pracujących z zakładach produkcyjnych jest trudne ze względu na złożoność i różnorodność rozwiązań. Dla potencjalnej standaryzacji pojęciowej jest ono dużym wyzwaniem. Aktualnie istnieją proste modele obrazujące różne obszary działania systemów informatycznych w przedsiębiorstwie, ich zadania i usługi w tych obszarach oraz połączenia i przepływy informacyjne między nimi. Celem niniejszego podrozdziału jest przedstawienie podstawowego wglądu w podejścia do modelowania systemów informatycznych w przedsiębiorstwie i zjawisk z nimi związanych. W literaturze można spotkać:

- model piramidowy (rysunek 14) – jest to typowy model opisujący hierarchiczną wymianę danych [292], [319], [338], [317], [368]. Zakłada wyraźne rozseparowanie funkcji systemów i komunikację między nimi przez ściśle zdefiniowany interfejs;
- model odwróconej piramidy (rysunek 15) – model bazujący na zastosowaniach. Zakłada istnienie repozytorium danych i zbioru obsługujących go usług uruchamianych na rzecz konkretnego elementu systemu w kontekście jego przeznaczenia;

- model bazujący na usługach (rysunek 16) – model płaski znany jako Diabolo lub model podwójnego stożka (ang. double cone)[5] [229], [318], [317], [368], [214]. Zakłada istnienie centralnego modelu informacyjnego i działających na jego bazie rozproszonych funkcjonalności.

Na rysunku 14 przedstawiono model piramidowy. Liczba warstw (poziomów funkcjonalnych) w takim modelu może być różna, zależnie od możliwości ich wydzielenia względem odrębnych i odseparowanych funkcjonalności. Najczęściej przyjmuje się istnienie czterech ogólnych warstw w hierarchii, tak jak pokazano na rysunku 14.

W modelu hierarchicznym każda warstwa odpowiada za określone funkcjonalności i wymienia dane z innymi przez urządzenie pośredniczące. W modelu zakłada się ujednoliconą komunikację w podsystemach i jej heterogeniczność w obrębie całości. Dlatego wymiana danych między systemami wymaga zmiany ich reprezentacji zgodnie z potrzebami, np. zmiany protokołu, adresacji, kodowania itp. Ogranicza to konstrukcję podsystemów do urządzeń należących do tej samej rodziny produktów lub konstrukcji wspieranych przez zamknięte grupy producentów.



Rys. 14. Model hierarchiczny wymiany informacji w przedsiębiorstwie
Fig. 14. The pyramid model of data exchanging in factory

Dla każdej z warstw istnieją przypisane systemy programowe przeznaczone do obsługi zadań danej warstwy. W podstawowym zakresie są to systemy klasy:

- Dla poziomu 1: systemy sterowania – klasa oprogramowania umożliwiająca sterowanie i regulacje procesu dla urządzeń polowych, np. klasy PC, IPC, PA, PLC, PAC, DCS, HMI.

[5] Nazwy pochodzą od kształtu graficznej reprezentacji modelu przypominającej Diabolo, czyli bryłę w kształcie klepsydry. Używa się jej w grze zręcznościowej polegającej na obracaniu bryły na lince łączącej dwa patyki. Nazwa może być też rozkodowana jako akronim od ang. Distributed Information Architecture to Bolster Lifecycle Optimization.

- Dla poziomu 2: systemy nadzorcze procesu – klasa oprogramowania umożliwiająca sprawowanie nadzoru nad procesem technologicznym, np. typu SCADA (ang. Supervisory Control and Data Acquisition).
- Dla poziomu 3: systemy nadzorcze produkcji – klasa oprogramowania umożliwiająca kontrolę procesu produkcyjnego, np. typu MES (ang. Manufacturing Execution System).
- Dla poziomu 4: systemy wspomagania zarządzania – klasa oprogramowania wspomagająca zarządzanie przedsiębiorstwem, np. typu ERP (ang. Enterprise Resource Planning).

Zadania tych systemów są określone i zdefiniowane w 3.2. W praktyce, konkretne implementacje wprowadzają wzajemne przenikanie funkcjonalności i dlatego nie należy zbyt sztywno szufladkować dostępnych produktów czy nawet klasy tych produktów. Ponadto, istnieje cały szereg podsystemów i systemów towarzyszących wspomagających pracę wyżej wymienionych (por. 3.2.11). Kwestia istnienia i przynależności organizacyjnej systemów informatycznych względem takiego czy innego modelu jest dość umowna i zależy od konkretnego przypadku. Klient zawsze może potrzebować systemu, który nie jest łatwo klasyfikowalny do modelu, a integrator może taki system dostarczyć. Dla przykładu, można przywołać system przygotowania produkcji dla maszyn dziewiarskich z dodatku 7.1.5, który ma cechy zarówno systemu HMI, SCADA, jak i MES. Można oczywiście dekomponować moduły systemu i określać ich przynależności do podstawowych klas, ale nie przynosi to żadnej wymiernej korzyści przy projektowaniu, implementacji, testach, utrzymaniu itp. Więcej na temat podsystemów pomocniczych znajduje się w 3.2.11.

Model hierarchiczny jest stosowany od lat i przez to bardzo popularny. Jednak ze względu na dynamiczny rozwój technologii cyfrowych oraz wprowadzenie „inteligentnych" układów AKP i standardowych technik komunikacyjnych model ten jest coraz mniej odpowiedni do modelowania systemów informatycznych w zakładach przemysłowych. Jest to spowodowane tym, że stosując komputerowo wspomagane i usieciowione elementy systemów automatyki, następuje zanik warstw modelu hierarchicznego na rzecz płaskiej struktury bezpośrednio współpracujących elementów i bazy wiedzy określającej zasady takiej współpracy. Z tego powodu powstały wymienione powyżej dwa inne modele, bardziej oddające rzeczywiste przepływy informacji w nowoczesnych architekturach systemów.

Model odwrócony (rys. 15) w literaturze spotyka się dość rzadko. W modelu tym zakłada się istnienie wspólnej struktury danych, opisującej informacje podlegające obsłudze w przedsiębiorstwie. Informacja jest zbierana z uruchomionych procesów zarówno technicznych, jak i biznesowych, zabezpieczana względem poprawności i reguł dostępu, a następnie udostępniana tylko tym systemom i ich elementom, dla których jest potrzebna. Model jest zorientowany na gromadzenie informacji z wielu źródeł i precyzyjną jej dystrybucję. Dlatego nadaje się lepiej do modelowania procesów biznesowych niż do procesów technicznych.

Rys. 15. Model odwróconej piramidy
Fig. 15. Reversed pyramid model

Przy tego typu zarządzaniu informacją, zapewnianie twardych zależności czasowych jest technicznie trudne do realizacji. Wąskim gardłem przepływu informacji jest warstwa składowania danych oraz warstwa dostępowa. Są one realizowane na bazie serwerów baz danych i serwerów danych, które z założenia nie są orientowane na prace w trybie HRT. Inaczej ujmując, systemy zbudowane na podstawie takiej architektury, bazują na wymianach pionowych, co nie sprawdza się przy wymianie informacji procesowej.

W modelu Diabolo (rys. 16) dolny stożek reprezentuje funkcjonalności związane z obsługą procesu technologicznego, natomiast górny reprezentuje funkcjonalności zarządzania i organizacji produkcji (np. wg modelu MESA [W30]). Zatem, istnieją dwa ogólne procesy: techniczny i biznesowy, komunikujące się przez wspólną definicję obsługiwanej informacji.

Rys. 16. Model Diabolo
Fig. 16. The Diabolo model

W modelu zakłada się ujednoliconą komunikację zarówno w podsystemach, jak i między nimi. Dzięki ujednoliceniu interfejsów komunikacyjnych możliwe jest tworzenie systemów składających się z różnorodnej bazy elementowej. Dlatego klasy (typy) systemów przedstawione powyżej nie pasują tak dobrze do tego modelu jak do modelu piramidowego.

W modelu Diabolo, zamiast rozpatrywać istnienie odseparowanych funkcjonalnie systemów rozsądniej jest rozpatrywać istnienie ich rozdzielonych, ale współdziałających funkcjonalności lub traktować je jako podsystemy równorzędne, dostarczające pewnych usług na rzecz całości.

Przepływ danych w systemie może być definiowany niezależnie dla każdego urządzenia, dla każdego zadania czy też wręcz dla dowolnego elementu logicznego jego struktury. Określenie powiązań opisujących taki przepływ jest niezbędne dla zaprojektowania i stworzenia każdego systemu. Sprowadza się to do zidentyfikowania i zdefiniowania łączonych elementów, zależności (połączeń, powiązań) aplikacyjnych między nimi i sposobu ich obsługi (por. 2.4.5 i 2.4.7). Definicje tworzą model wymiany informacji (model informacyjny), opisujący zależności aplikacyjne i sposób realizacji usług pobrania i dostarczenia danych zarówno w warstwie biznesowej, jak i technologicznej, a także pomiędzy nimi [368]. Dla systemu hierarchicznego model taki jest rozproszony w strukturze systemu. Natomiast dla modelu Diabolo powiązania mogą być definiowane na bazie modelu wspólnego. Umożliwia to łatwiejsze utrzymanie systemu i jego prostszą rekonfigurację względem operowania na zależnościach rozproszonych i nieujednoliconych w formie. Ponadto, centralny model ułatwia tworzenie rozwiązań multiplatformowych, czyli pochodzących od różnych producentów.

Podobnie jak w modelu omawianym poprzednio, przepływ informacji bazuje tu na wymianach pionowych. Jednak nie zakłada, że tylko i wyłącznie mogą funkcjonować takie wymiany. Elementami procesów mogą być elementy złożone (np. maszyny) lub systemy (np. ISP), w ramach działania których funkcjonują wymiany poziome i zadania wykonywane w dowolnym trybie RT. Przepływy przez model informacyjny dotyczą skomunikowania i zapewnienia współpracy tych elementów. Podobnie można założyć przy modelu odwróconym. Jednak model podwójnego stożka jest bardziej przejrzysty i uniwersalny względem odwróconej piramidy i dlatego zyskał większe uznanie.

We współczesnych podejściach projektowania znanych z idei „fabryki cyfrowej” (ang. digital factory, smart factory) [241], [276], [100], [412] model ze wspólnym modelem informacyjnym spotyka się częściej niż model hierarchiczny czy odwrócony. Jednak w systemach już działających, osadzonych w technikach polowych (zob. 2.4.4), model hierarchiczny jest bardziej popularny.

Warto na koniec wspomnieć o podejściu eliminującym strukturę, w którym system jest płaski (ang. direct management, direct IT)[6]. Zakłada się w nim istnienie urządzeń (węzłów) z wbudowanymi funkcjami IT. Chodzi tu o rozbudowane funkcje przetwarzania, udostępniania i integracji znane z oprogramowania zarządzającego typu SCADA czy MES. Tym samym system zbudowany z takich węzłów nie potrzebuje oprogramowania pośredniczącego między

[6] Na przykład Automation Server wbudowywany w urządzenia Saia-Burgess.

ISP a użytkownikami. Funkcjonalności nadzorczo-zarządzające są implementowane w rozproszonej architekturze węzłów ISP, wówczas tworzenie warstw i hierarchii modeli piramidowych traci sens, gdyż budowana jest tylko jedna warstwa o kompleksowej funkcjonalności. Jednak ze względów technicznych, takich jak ograniczona moc i zasoby węzłów, zastosowanie takiego podejścia jest w praktyce możliwe tylko dla małych i ewentualnie średnich systemów.

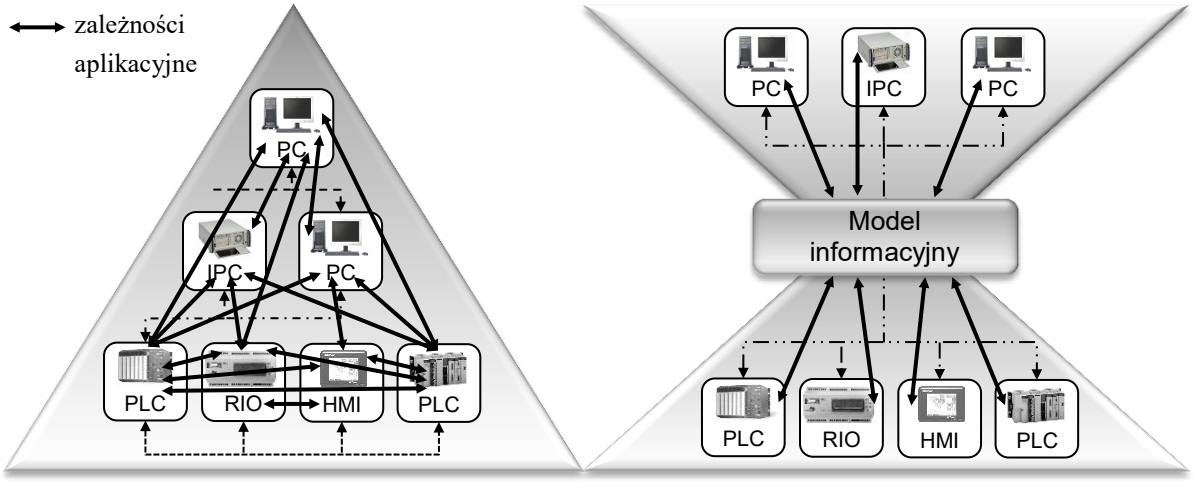
1.2.2. Przepływ informacji

W modelu hierarchicznym przepływ informacji można podzielić na pionowy odnoszący się do realizacji usług międzywarstwowych i poziomy odnoszący się do usług wewnątrzwarstwowych. W modelu płaskim sprawa przepływu informacji wygląda podobnie, z tym że istnieją tylko dwa poziomy niepozostające w hierarchii: procesu i zarządzania. W tym modelu tym większy nacisk kładzie się na modularność i przepływy z tym związane. Modularność umożliwia wydzielenie współpracujących równorzędnych podsystemów. Współpraca może być rekonfigurowana, a moduły ponownie użyte bez konieczności istotnych modyfikacji. W tym świetle mówi się o integracji **poziomej,** dotyczącej przepływu informacji między modułami, i pionowej dotyczącej komunikacji z procesami zarządzania. Konfiguracja obejmuje zasady mapowania informacji znajdującej się w jednej lokacji (np. moduł) do innej (np. jednostka centralna).

Mapowanie informacji to proces dynamicznego grupowania logicznych lokalizacji względem określonego jej przepływu. Określa się również dodatkowy wymiar integracyjny związany z interfejsem do zarządzania cyklem życia systemu produkcyjnego (ang. life cycle), w którym wykorzystywana jest wspomniana modularność.

W modelu hierarchicznym wymiana informacji zachodzi w interfejsach specyficznych dla użytych w przypadku danego podsystemu środków i metod. Skutkuje to mnogością interfejsów i niejednolitym sposobem dostępu do danych. Biorąc pod uwagę model płaski oparty na usługach, wszelkie dane procesowe i dane z przetwarzania są udostępniane przez ustandaryzowany interfejs we wspólnym modelu informacyjnym. W praktycznych realizacjach dane są udostępniane aplikacjom przez serwer lub zestaw serwerów danych (np. OPC). Aplikacje te stanowią klientów uzyskujących dostęp do danych przez ustandaryzowane usługi. Podejście takie jest nazywane SOA (ang. Service Oriented Architecture) [412], [214] (zob. 2.1.1), która to jest dobrze znana z innych zastosowań IT [189], [310]. Należy zauważyć, że podejście takie jest dobre dla celów przekazywania danych do i z procesów biznesowych (górny stożek). Natomiast dla ISP (dolny stożek) wprowadza on znaczący narzut czasowy lub nawet prowadzi do utraty parametrów czasu rzeczywistego. Dlatego w wymianach poziomych

i w integracji poziomej na poziomie ISP należy szczególną troskę wykazać w kwestii zapewnienia parametrów czasowych.



Rys. 17. Wymiana informacji w modelach systemów automatyzacji
Fig. 17. Information exchange within the models of automation

Ogólną strukturę wymiany informacji dla obu modeli przedstawiono na rysunku 17. Zaprezentowane typy węzłów i zależności aplikacyjne są przykładowe.

Niezależnie od przyjętego modelu należy rozpatrywać przynajmniej dwa poziomy aktywności systemów informatycznych odzwierciedlających różne funkcjonalności. Jeden dotyczy prowadzenia produkcji, a drugi zarządzania (por. 3.5).

❖ **Poziom procesu**

Na poziomie automatyzacji procesu, postrzeganym jako najniższy w strukturze wymiany informacji, sygnały z binarnych czujników i elementów wykonawczych transmitowane są przez sieć sygnałową (AS-I, Hart itp.). Informacja taka dociera do układów przetwarzających typu PLC lub ogólnie komputerów. Daje to prostą i stosunkowo tanią technikę przesyłania danych i zasilania tym samym kablem. Przy braku takich rozwiązań informacja do komputerów dociera kanałami analogowymi.

Na poziomie prowadzenia procesów technologicznych, nazywanym często polowym, rozproszone elementy obiektowe, takie jak moduły I/O, przetworniki, napędy, zawory i panele operatorskie komunikują się ze sobą, z jednostkami przetwarzającymi i ogólnie z elementami systemu, przez deterministyczny system komunikacji. Wymiana informacji dotyczy abonentów mających bezpośredni kontakt z urządzeniami AKP obsługującymi proces przemysłowy (ang. *sensors*, *actuators*) (por. 1.1.1), urządzeń AKP stanowiących abonentów oraz urządzeń przetwarzających, jak np. PLC.

Transmisja danych procesowych odbywa się cyklicznie, podczas gdy dodatkowe przerwania, dane konfiguracyjne i diagnostyczne przesyłane są acyklicznie. Komunikacja bazuje na sieciach komputerowych, np. sieciach polowych lub sieciach opartych na Ethernecie. Zakłada się, że jest to komunikacja zdeterminowana czasowo.

❖ **Poziom zarządzania**

Na poziomie zarządzania, postrzeganym często jako nadrzędny względem procesu, zachodzi dwukierunkowa wymiana danych pomiędzy:

- podsystemami informatycznymi pracującymi na tym poziomie, czyli systemami IT klasy MES, ERP, CMMS (por. 3.2.9, 3.2.10),
- urządzeniami z poziomu procesu a ww. podsystemami zainteresowanymi danymi z produkcji.

Wymiana danych między poziomami dobywa się poprzez pośredników. Są to najczęściej bramy (ang. gateway) łączące różne sieci w przypadku komunikacji heterogenicznej lub serwery i bazy danych przy podejściu usługowym.

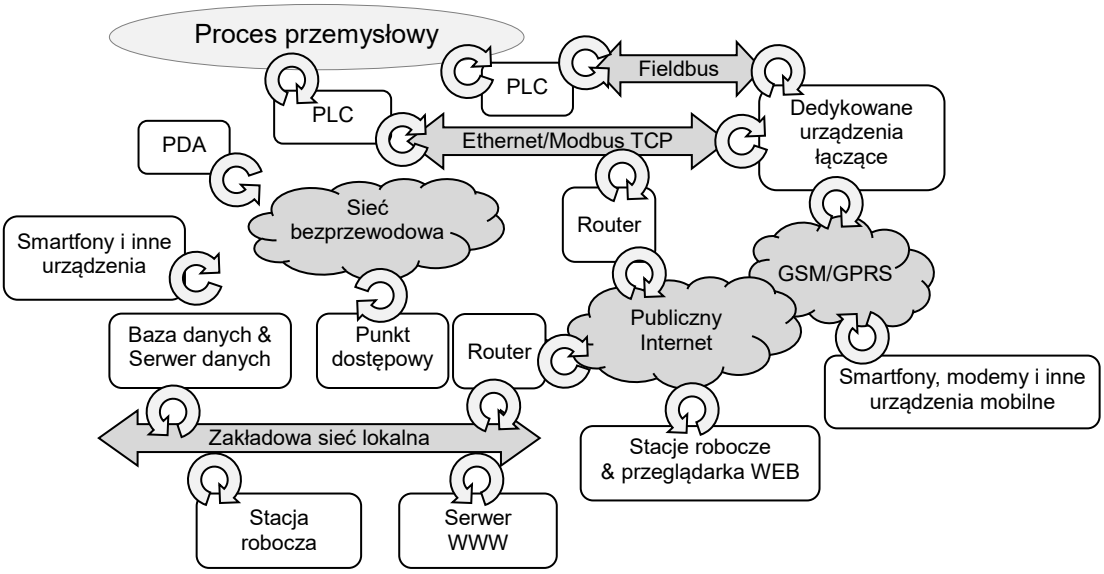
Liczba przekazywanych danych zawsze maleje, im bardziej niezwiązane funkcjonalnie są łączone podsystemy. Dlatego w modelu hierarchicznym nie ma potrzeby wykorzystywania sieci o dużych przepustowościach i wydajnościach. Jednak dla modelu płaskiego liczba danych dystrybuowana przez pośrednika jest duża, gdyż obejmuje stan całego procesu. Dlatego dla tego typu podejścia wymiana informacji wymaga wydajnej komunikacji. Wykorzystuje się tu protokoły oparte na bazie sieci Ethernet, np. standard Profinet lub inne tego typu, jak Powerlink, EtherCAT itp. Na wyższym poziomie abstrakcji stosuje się tu ustandaryzowaną komunikację typu Klient-Serwer, np. na bazie usług OPC, SuiteLink, DDE, RSTP itp. Zakłada się, że komunikacja nie jest nacechowana ograniczeniami czasowymi.

Liczba podsystemów lokalnych na danym poziomie nie jest określana i zależy od wymaganych funkcjonalności i złożoności obsługiwanych procesów. Należy zauważyć, że przekazywanie danych między nimi może mieć charakter zarówno poziomy, jak i pionowy, zależnie od przyjętego modelu.

❖ **Środowisko heterogeniczne**

Budowa współczesnych systemów przemysłowych potrafi być stosunkowo złożona i wykraczać poza modelowanie przepływu danych w przedsiębiorstwie rozumianym jako ograniczony terytorialnie obiekt. W modelach opisanych wcześniej może istnieć ujednolicona infrastruktura komunikacyjna lub częściowo, czy też znacząco heterogeniczna. W praktyce pełne ujednolicenie jest rzadko spotykane. Częściej integratorzy muszą projektować i aplikować przepływ danych w strukturach zróżnicowanych zarówno pod względem produ-

centów, jak i użytych technik. Przykład systemu z różnorodnymi mechanizmami komunikacyjnymi przedstawiono na rysunku 18 [125].



Rys. 18. Przykład heterogenicznej struktury wymiany danych
Fig. 18. An example of heterogeneous structure of data exchange

Przepływ danych wykraczający poza obszar zakładu jest możliwy dzięki zastosowaniu sieci rozległych, intersieci [64] oraz sieci bezprzewodowych [389], w tym komórkowych. Żaden z wymienionych modeli nie ogranicza zastosowania tego typu sieci.

Rzeczywistość jest zwykle różna od idealnych modeli, dlatego często nawet w przypadku systemu opartego na modelu płaskim pewne jego podsystemy są budowane w strukturze hierarchicznej z użyciem różnorodnej bazy elementowej. W efekcie nie należy liczyć na pełne ujednolicenie i ustandaryzowanie wymiany danych w systemach złożonych. W dalszej części niniejszej książki nie rozpatruje się systemów w całościowym kontekście jakiegokolwiek modelu automatyzacji. Rozważany jest ISP w rozumieniu definicji z rozdziału 2 i modelu warstwowego opisanego w rozdziale 2, abstrahując od sposobu realizacji modelu informacyjnego.

1.2.3. Wymagania i ograniczenia

Aplikacja systemu musi mieć kilka istotnych właściwości, które wymuszają sposób, w jaki systemy są konstruowane i w jaki działają. Oczekiwania względem działania ISP muszą wynikać i być ograniczane przez te właściwości. W tym podrozdziale tylko zasygnalizowano podstawowe własności, a ich rozwinięcie znajduje się w dalszej części książki.

Środowisko przemysłowe, z punktu widzenia ilości obserwowanej, przetwarzanej i wypracowywanej informacji w zadanym oknie czasowym, ma **charakter statyczny**, w przeciwieństwie do środowisk biurowych czy biznesowych. Prowadzi to do sytuacji, gdzie

właściwości reprezentacji informacji (kodowania), jej składowania i przekazywania (mapowania) oraz algorytmizacji jej przetwarzania są stałe dla danej instalacji technologicznej.

W większości przypadków systemy działające na poziomie automatyzacji, w tym ISP, nie wymagają dużej szybkości pracy, niezależnie od punktu widzenia i definicji pojęcia szybkości. Wymagane jest natomiast przetwarzanie w czasie rzeczywistym (por. 1.1.4) oraz praca z odpowiednim dla procesu poziomem bezpieczeństwa (por. 3.4). Oba te atrybuty wymuszane są przez **wymogi procesu** (aplikacji). Technologiczny opis procesu dostarcza wymagań, które muszą być spełnione, gdyż zgodne z nimi działanie stanowi sens istnienia ISP. Wszelkie odstępstwa i ograniczenia uniemożliwiające dotrzymanie wymagań procesu będą skutkowały nieprawidłowym jego prowadzeniem, i w konsekwencji stwierdzeniem nieprzydatności systemu. Z wymaganiami procesu związane są również wymagania środowiska (por. 1.1.2), w którym ISP ma funkcjonować. Projektowane rozwiązanie musi być odporne na negatywne wpływy środowiska, a zignorowanie ograniczeń z nich płynących może skutkować zaburzeniami funkcjonowania części lub całości systemu.

Oprócz wymagań procesu istnieją jeszcze **wymagania klienta,** w tym wymagania współczesnego rynku produktów. Rynek wymaga produktów wysokiej jakości, dostosowanych do bieżących potrzeb klienta, krótkoseryjnych, i co za tym idzie z krótkim cyklem produkcyjnym. Istnieje duże prawdopodobieństwo, że tworzony system będzie obsługiwał całkiem nową lub mocno zmodyfikowaną linię technologiczną oraz że w wyniku konieczności relatywnie częstych modyfikacji produktu, linia taka będzie równie często modyfikowana. Dlatego pomimo statyczności danego procesu, projektanci ISP często w ostatnich latach spotykają się z wymogiem zapewnienia konstrukcyjnej i funkcjonalnej **elastyczności** systemu (dynamiczność, sprężystość, ang. dynamism, flexibility, resilience). Pociąga to za sobą konieczność dynamicznej zmiany jego aplikacji i informatycznego opisu (modelu i mapowania informacji) [322]. Do istotnych problemów, z którymi projektant szybko zmieniających się wymagań produkcyjnych musi sobie radzić w większości istniejących systemów, należą:

- Długi czas i duża złożoność projektowania, przeprojektowania i uruchamiania. Koszt tych procesów wraz z przestojami stanowi jeden z większych składników całościowych kosztów wytwarzania.
- Mało elastyczne architektury systemów, szczególnie architektur scentralizowanych, nieskalowalnych lub skalowalnych wysokim kosztem.
- Częsty brak mechanizmów bezpieczeństwa i tolerancji awarii lub problemy integracji takich podsystemów.
- Heterogeniczność środowiska często związana z niekompatybilnością technik i technologii informatycznych.

- Izolacja ISP od innych systemów informatycznych, szczególnie systemów i technologii biznesowych klasy ICT (ang. Information and Communications Technologies). Izolacja dotyczy nie tylko technicznego zagadnienia integracji, lecz i izolacji mentalnej. Bardzo często działy IT w przedsiębiorstwach nie postrzegają systemów automatyzacji jako systemów informatycznych i w mniejszym lub większym stopniu odcinają się od działań w tym obszarze (wspomniane również w 3.1).

Wymagania klienta dotyczą również usług oferowanych przez ISP, na których istnienie ma wpływ tylko projektant i nie są niezbędne z punktu widzenia procesu. Wymagania takie z reguły są zbieżne z technologią, choć zdarza się, że klient niekoniecznie jest ekspertem z dziedziny informatyki, komunikacji czy automatyki, i generuje wymagania bezzasadne lub prowadzące do zmniejszenia bezpieczeństwa. Z jednej strony, przy tworzeniu systemów wskazane jest być otwartym na życzenia użytkownika, jednak z drugiej należy zachowywać wszelkie **pryncypia techniczne**.

Istnieje jeszcze **aspekt ekonomiczny** tworzenia ISP. Duża różnorodność technik i technologii informatycznych oraz dostępnych rozwiązań wzorcowych może skłaniać do szukania rozwiązań wyrafinowanych i kosztownych. W dziedzinie ISP nie ma zależności, że najnowsze i najdroższe rozwiązania są najlepsze. Tworzenie ISP jest dziedziną, gdzie sztuką jest dobranie technologii do aktualnych potrzeb i możliwości rozwoju. Dla przykładu, sieci typu RTE (Ethernet przemysłowy, ang. Real-Time Ethernet) niekoniecznie muszą wypierać sieci polowe (ang. fieldbus) (por. 2.4.4). Dla większości rozwiązań sieci polowe będą stanowić w zupełności wystarczające techniki komunikacyjne jeszcze przez długie lata. Choć pojawiają się systemy, których usługi wymagają nowszych. Podobnie ze sterownikami PLC. Do obsługi pomiarów np. z kilkudziesięciu liczników nie potrzeba stosowania urządzeń z dużą mocą obliczeniową.

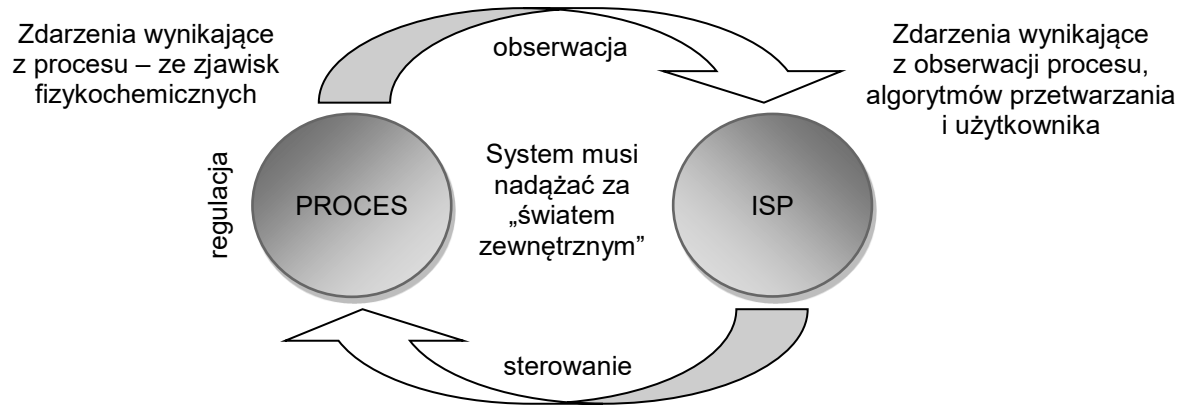
Kolejnym aspektem projektowania ISP jest **aspekt rozwoju**, który pośrednio wiąże się z poprzednim. Przy projektowaniu ISP koniecznie należy brać pod uwagę możliwość jego potencjalnego utrzymania i rozwoju przez okres wynikający z czasu życia podobnych systemów. Wiąże się to z cechą systemów, jaką jest trwałość (ang. sustainability) (por. 2.3.2, [76]). W praktyce, czas życia ISP wynosi kilkanaście lub nawet kilkadziesiąt lat. Nie jest dobrą praktyką tworzenie rozwiązań, które z założenia za kilka lat będą musiały zostać zdemontowane i zastąpione innymi. Autor zna rozwiązania, które pomimo doskonałych parametrów technicznych, musiały zostać wyłączone z eksploatacji i zastąpione innymi. Stało się tak tylko z tego powodu, że w momencie projektowania nie przewidziano, iż w przyszłości potrzebne będzie rozszerzenie funkcjonalności, które jest niemożliwe do zrealizowania z powodu ograniczeń technologicznych lub rynkowych. Wynika z tego, że poza zagadnieniami tech-

nicznymi, tworzenie ISP powinno się odznaczać dość nietechniczną cechą tzw. „future proof" – odporności na przyszły rozwój.

Stworzenie „dobrego" ISP jest zatem nie tylko kwestią czysto techniczną, ale i kwestią rozeznania potrzeb klienta, elementarnego bilansu ekonomicznego oraz wizji przyszłości.

1.2.4. Zadania ISP

Połączone ze sobą elementy automatyki tworzą układ, na którego stan oddziałuje obiekt przemysłowy, jak również układ przez swoje elementy oddziałuje na stan tego obiektu. Jest to podstawa oddziaływania z obiektem w układach automatyki. Więcej na ten temat od strony teorii automatyki można znaleźć w [108], [12]. Podobnie sprawa wygląda, gdy po stronie układów automatyki znajduje się system komputerowy. Na rysunku 19 przedstawiono ideę takiego oddziaływania. W ISP wykorzystuje się układy automatyki do obsługi procesu, a do obsługi informacji węzły komputerowe i sieci komputerowe.



Rys. 19. Schemat oddziaływań nadążnych System-Proces
Fig. 19. Schema of System-Process RT effect

Występują zatem dwa główne zadania systemu:

- obserwacja procesu,
- sterowanie działaniem procesu.

Gdy oba zadania są wykonywane równocześnie lub na przemian, można wówczas realizować zadanie **regulacji**. Regulacja jest to automatyczne sterowanie procesem w funkcji wartości danych pozyskanych z procesu, zachodzące w czasie rzeczywistym.

❖ **Zapewnienie opisu informacyjnego**

Sterowanie i regulacja odbywają się na podstawie:

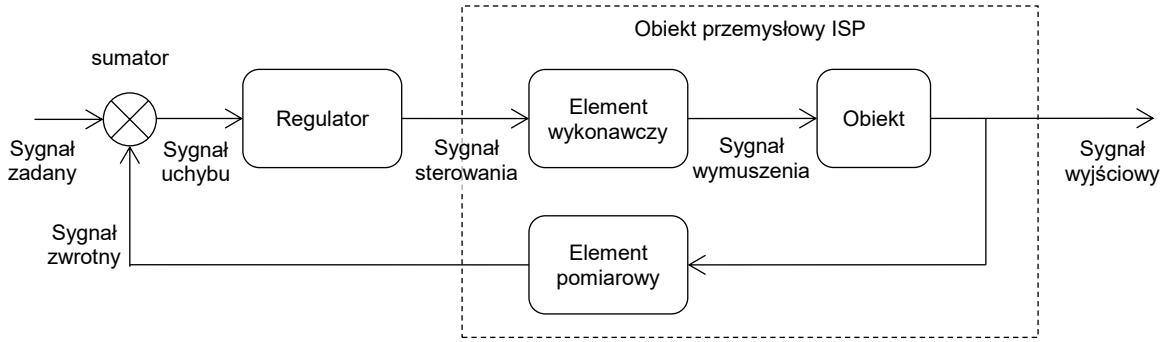
- zbioru informacji stanowiącej dane wejściowe – informacji pozyskanych z procesu i opisujących jego stan (obraz wejściowy procesu, ang. process input image),

- zbioru informacji stanowiącej stan systemu – informacji opisujących stan wewnętrzny systemu (obraz systemu),
- zbioru informacji stanowiącej dane wyjściowe – informacji wyprowadzanych na zewnątrz systemu, oddziałujących na proces (obraz wyjściowy procesu, ang. process output image).

Wymienione wyżej zbiory stanowią swoiste **wektory informacji** zorientowane w czasie i odzwierciedlające rzeczywisty stan układu z dokładnością do czasu ważności okien czasowych, w których mogą uchodzić za ważne, a wynikających z wymaganych ograniczeń czasowych dla danego przypadku. W dalszej części książki będzie używane pojęcie wektora informacji jako struktury danych przechowującej dane określonego rodzaju i typu (por. 2.4.6). W większości przypadków będzie to jednowymiarowa tablica bitów, słów lub podwójnych słów, ale również może to być wielowymiarowa tablica dowolnych struktur danych. W zależności od zasięgu, kodowania i interpretacji wektor może też stanowić dane zserializowane, np. wektory informacyjne na poziomie transakcji sieciowych. Opis informacyjny można rozpatrywać zarówno w kontekście całego systemu, jak i jego elementów składowych. Węzły i ich elementy przechowują lokalne alokacje podzbiorów z nimi związanych. Obsługa przepływu danych pomiędzy takimi wektorami jest jednym z zadań ISP zarówno w wymiarze lokalnym urządzenia, jak i całego systemu.

❖ **Tworzenie układu regulacji**

Automatyka rozpatruje przedstawiony na rysunku 20 układ jako **zamknięty układ regulacji** (pętla regulacji, ang. control loop, closed loop).



Rys. 20. Schemat układu regulatora z obiektem regulacji
Fig. 20. Schema of a regulator with its controlled object

Jest to typowy, formalny schemat układu regulacji składającego się z sumatora, regulatora, elementu wykonawczego, obiektu i elementu pomiarowego, przez który realizowane jest sprzężenie zwrotne. Przerywając pętlę sprzężenia zwrotnego, można mówić o samej obserwacji procesu lub o samym sterowaniu procesu. Z pojęciem regulacji spotyka się jednak tylko wówczas, gdy istnieje pętla sprzężenia.

❖ **Obsługa sygnałów**

Sygnały na obiektach przemysłowych są nośnikiem informacji dostarczanej do budowania wektorów wejścia (do układów przetwarzających) i pobieranej z wektorów wyjścia celem oddziaływania na obiekt (do układów wykonawczych). Do praktycznej realizacji kanałów sygnałowych używa się sygnałów elektrycznych obsługiwanych przez odpowiednie interfejsy i transmitowanych przez odpowiednie przewody (por. 3.4.2). Teoria sygnałów jest szeroką dziedziną, której niniejsza publikacja nie dotyczy. Warto sięgnąć po odpowiednią literaturę [177], [250], [349]. Sygnały obiektowe w ISP należy klasyfikować według ich dziedziny, którą jest czas, oraz według przeciwdziedziny, czyli zbioru wartości, które przyjmuje sygnał. Względem dziedziny można wyróżnić:

- Sygnały **ciągłe**, w których zarówno dziedzina, jak i przeciwdziedzina są ciągłe. Oznacza to, że w dowolnym momencie czasu znana jest wartość sygnału oraz wartość ta może przybierać dowolną wartość z określonego zakresu.
- Sygnały **dyskretne**, w których dziedzina sygnału jest dyskretna, a jego przeciwdziedzina ciągła. Oznacza to, że wartości sygnału są zadane w dyskretnym zbiorze chwil czasu, natomiast wartości, które sygnał przyjmuje, są ciągłe.
- Sygnały **cyfrowe**, w których zarówno dziedzina, jak i przeciwdziedzina są dyskretne. Oznacza to, że tak jak dla sygnału dyskretnego wartości są próbkowane w określonych momentach czasu, natomiast przenoszona przez sygnał informacja jest reprezentowana przez wartości ze zbioru dyskretnego.

Rozważając sygnał fizyczny, można stwierdzić, że wielkości fizyczne sygnałów są zawsze ciągłe. Warto przy tym pamiętać, że na wartość sygnałów elektrycznych wpływa zawsze[7] szum termiczny oraz że każde urządzenie techniczne czy nawet organizm biologiczny może przetwarzać tylko takie sygnały, których widmo ogranicza się do pasma o określonej i ograniczonej szerokości. W praktyce, niezależnie od rodzaju „użytkownika" sygnału, wartości ciągłe ze świata zewnętrznego (obiektu) są zawsze przekształcane w pewne kwanty informacji, czyli w sygnał cyfrowy. Można zauważyć, że komputery[8] są w stanie działać tylko na podstawie sygnałów cyfrowych zdigitalizowanych z sygnałów ciągłych dostarczonych do interfejsów. Stąd, w układach cyfrowych sygnały są zawsze próbkowane, a ich przetwarzanie dokonywane jest na bazie modelu, którym jest sygnał cyfrowy dostępny

[7] Poza nadprzewodnikami.

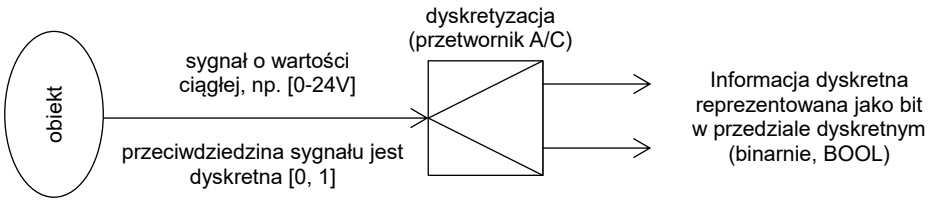
[8] Warto dokonać eksperymentu myślowego, że sprawa dotyczy również naszego mózgu. Nie działa on synchronicznie, ale informacja docierająca do nas jest próbkowana w czasie oraz reprezentowana przez zbiory rozmyte. Stąd twierdzenie starożytnych Pitagorejczyków, że „wszystko jest liczbą" wydaje się być w tym kontekście zasadne.

w urządzeniu w postaci zmiennych aktualizowanych i przetwarzanych w pewnych momentach czasu.

Dlatego w ISP częściej dokonuje się podziału sygnałów IO względem ich przeciwdziedziny, zakładając, że dziedzina jest „niemal ciągła". Oczywiście ciągłość dziedziny jest umowna. W rzeczywistości, dziedzina sygnałów obsługiwanych w węzłach ISP jest zawsze dyskretna, a skwantowanie wartości przenoszonej informacji zależy od konstrukcji przetworników oraz funkcjonowania systemu operacyjnego węzła. Dla podziału według przeciwdziedziny sygnałów cyfrowych można wyróżnić następujące układy wejścia i wyjścia węzłów:

- Wejścia/wyjścia **binarne**,
  dla których wartość sygnału jest ciągła, dziedzina jest dyskretna, a przeciwdziedzina odzwierciedla wartości w zbiorze dyskretnym, dwuwartościowym [0, 1] (prawda, fałsz, ang. true, false). Sygnały takie nazywa się często sygnałami binarnymi, natomiast układy IO nazywane są też cyfrowymi lub dyskretnymi (ang. digital/discrete/binary). Układy wejścia oznaczane są zwykle przez DI (ang. digital input), a wyjścia przez DO (ang. digital output). Reprezentacją cyfrową są bity (zmienne typu boolean).

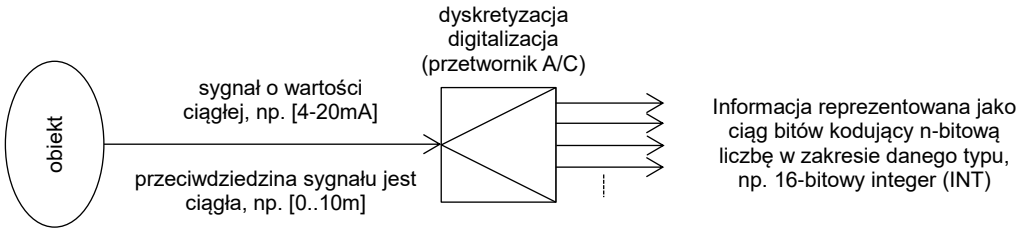
  W układach wejść binarnych węzła wartości ciągłe sygnału obiektowego są digitalizowane do wartości binarnych. Ponadto, sygnał jest próbkowany w określonych momentach czasu, wynikających z działania przetwornika oraz z działania systemu operacyjnego węzła. Zobrazowano to na rysunku 21.



Rys. 21. Ilustracja wejścia binarnego
Fig. 21. Illustration of binary input

W kontekście ISP poprawność określenia wejść i wyjść binarnych jako dyskretnych lub cyfrowych jest dyskusyjna. Sygnały obsługiwane przez te układy mają wartość ciągłą i przenoszą informację kodowaną w dyskretnej dziedzinie czasu oraz dyskretnej przeciwdziedzinie wartości binarnych. Cyfrowa jest reprezentacja tej informacji w pamięci. Jednak nazwy te mogą być mylące, ze względu na fakt, że wszystkie układy IO są dyskretne oraz że istnieją sygnały cyfrowe przenoszące informację zakodowaną synchronicznie w ciągu bitów (sieci, porty komunikacyjne), których interfejsy IO mogą być również nazwane cyfrowymi.

- Wejścia/wyjścia **analogowe**,
  dla których wartość sygnału jest ciągła, dziedzina jest dyskretna, a przeciwdziedzina jest pozornie ciągła. Reprezentacja ciągłej wartości w pamięci komputera nie jest możliwa ze względu na ograniczenia kodowania i alokacji, wynikające z typu zmiennej reprezentującej tę wartość w pamięci. Układy wejścia oznaczane są zwykle przez AI, a wyjścia przez AO. Reprezentacją cyfrową są ciągi bitów (liczby i słowa, zmienne typu np. WORD, INT, REAL itp.).



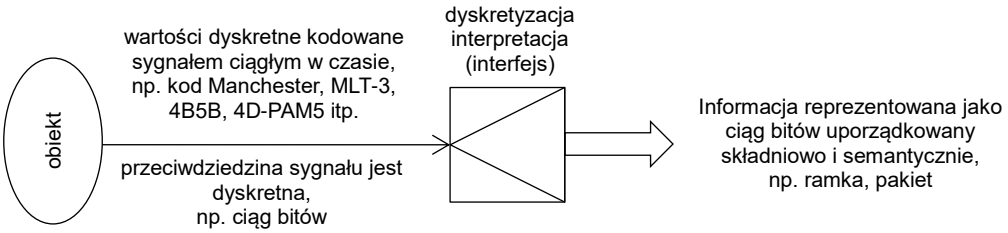
Rys. 22. Ilustracja wejścia analogowego
Fig. 22. Illustration of analogue input

W układach wejść analogowych węzła wartości ciągłe sygnału obiektowego są digitalizowane do wartości o określonej dokładności odwzorowania, wynikającej z rozdzielczości przetwornika oraz sposobu reprezentacji informacji w pamięci. Ponadto, sygnał jest próbkowany w określonych momentach czasu, wynikających z działania przetwornika oraz z działania systemu operacyjnego węzła. Zostało to zobrazowane na rysunku 22.

- Interfejsy **cyfrowe**,
  dla których wartość sygnału jest ciągła, a dziedzina jak i przeciwdziedzina są dyskretne. Zwykle zbiór wartości przeciwdziedziny jest zbiorem binarnym. Wynikałoby z tego, że wejścia/wyjścia cyfrowe są tym samym co binarne. Różnica jednak polega na sposobie przekazywania informacji przez sygnał.

  W interfejsach cyfrowych informacja jest kodowana z użyciem ciągu wartości binarnych w określonym czasie, co powoduje, że informacja jest znana dopiero po zebraniu i zinterpretowaniu wartości szeregu bitów w określonym czasie. Natomiast interfejsy binarne i analogowe dostarczają kompletnej wartości z każdym krokiem dyskretyzacji. Reprezentacją cyfrową są ciągi bitów, kodujące struktury opisujące przenoszoną informację.

  W praktyce, interfejsy cyfrowe są to najczęściej porty komunikacyjne (np. RS232, 422, 485, USB, Bluetooth itp.) i interfejsy różnego rodzaju sieci informatycznych i pomiarowych (np. Ethernet, Profibus, CAN, DeviceNet, ASI itp.). Zobrazowano to na rysunku 23.

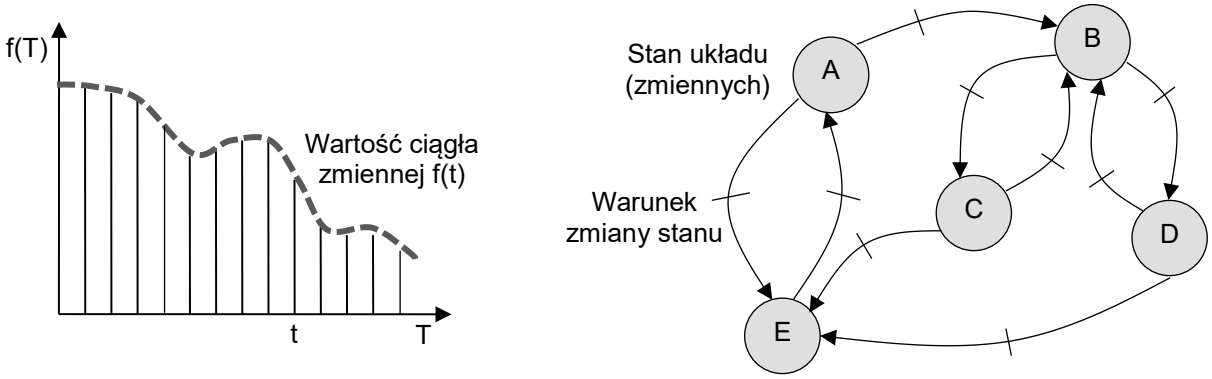


Rys. 23. Ilustracja wejścia cyfrowego
Fig. 23. Illustration of digital input[9]

Podobnie jak sygnały, procesy technologiczne można podzielić na:

- **procesy ciągłe**, w których na dowolnym skończonym odcinku czasu można wyróżnić nieskończoną liczbę zmian, a różnice między nimi są dowolnie małe oraz
- **procesy dyskretne**, w których można wyróżnić określoną i skończoną liczbę stanów (i zdarzeń), a proces znajduje się zawsze co najmniej w jednym z nich.

Działanie procesów ciągłych modelowane jest przez równania różniczkowe, transformaty itp., natomiast dyskretnych przez równania różnicowe, maszyny stanu, grafy przejść, sieci Petriego itp. Rozwiązanie to zobrazowano na rysunku 24.



Rys. 24. Ilustracja podejścia do obsługi procesów ciągłych i dyskretnych
Fig. 24. Illustration of continuous and discrete process service

W ISP można wyróżnić różne sposoby przetwarzania informacji, biorąc pod uwagę:

- rodzaj sygnałów przenoszących informacje pomiędzy obiektem a elementami ISP, i co za tym idzie typ reprezentujących ją zmiennych,
- zależności czasowe interakcji z otoczeniem, czyli zagadnienia wprowadzania i wyprowadzania informacji w funkcji czasu,
- zależności czasowe przetwarzania, czyli działanie algorytmów obsługujących proces względem czasu.

[9] Przez *digital input* (*output*) często określa się wejścia (wyjścia) binarne, p. uwaga strona 69.

Istnieją zatem dwa podejścia:

- przetwarzanie ciągłe (z czasem ciągłym, ang. continuous) – w którym zmienne obiektowe przyjmują dowolne wartości z przedziału zmienności wynikającej z ich typu, a zmiany następują z rozdzielczością czasową wynikającą z cyklu przetwarzania. Można tu dodatkowo wydzielić podejścia:
  - linearne (logiczne, sekwencyjne, np. wykorzystuje się regulacje z użyciem regulatorów ciągłych np. PID, P, I, D, PD, PI),
  - nielinearne (wielokierunkowe, wielowątkowe, np. regulacje rozmyte, neuronowe),
- przetwarzanie dyskretne (z czasem skwantowanym, ang. discrete) – w którym zmienne procesowe przyjmują wartości dyskretne, a zmiany następują w określonych stanach układu. Można tu dodatkowo wyróżnić podejścia:
  - warunkowe (np. przejścia boolowskie, ekspertowe),
  - sekwencyjne (np. przejścia czasowe, zdarzeniowe).

Podczas przetwarzania informacji przy obsłudze procesów ciągłych można tolerować sporadyczne błędy na wejściach danych (np. szum z układów pomiarowych). Nie wpływa to na odzwierciedlenie stanu procesu w układzie przetwarzania. Natomiast zakłócenia w przetwarzaniu dyskretnym mogą powodować przekłamanie stanu lub wyjście procesu poza obszar zdefiniowany. Odtworzenie stanu dyskretnego układu fizycznego jest trudne. W praktyce spotyka się najczęściej układy hybrydowe, stanowiące połączenie ciągłych i dyskretnych.

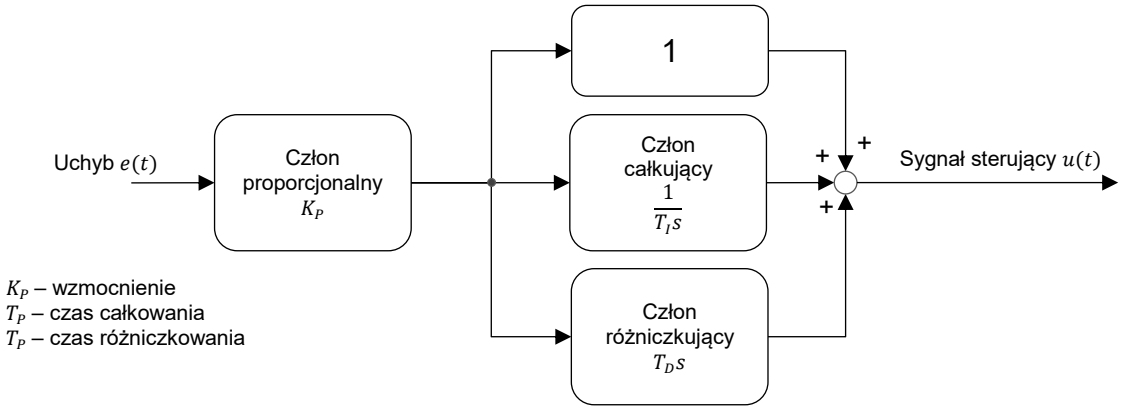
Niezależnie od rodzaju procesu, sygnału czy przetwarzania należy zauważyć, że wartości zmiennych przyjmują w układach cyfrowych wartości zdyskretyzowane w czasie, zgodnie z cyklami pracy tych układów. Dość często okres dyskretyzacji jest na tyle mały, że rozdzielczość czasowa reprezentacji informacji w pamięci komputera jest w zupełności wystarczająca do odwzorowania procesów ciągłych bez istotnej utraty informacji o ciągłości procesu.

Jeśli ktoś chciałby wyróżnić również procesy cyfrowe, to najlepsza definicja bazuje na definicji procesu w rozumieniu informatycznym (por. 1.1.6), czyli jako przetwarzanie danych przez instancje wykonywanego programu. Choć można tu się doszukiwać innych znaczeń związanych z technologiami cyfrowych wydruków produktów czy zastosowań nanotechnologii [24], [378].

We współczesnym przemyśle systemy sterowania muszą radzić sobie ze wszelkimi rodzajami rzeczywistych procesów i związanych z nimi sygnałów. Dla procesów ciągłych podstawowe i niezwykle ważne są układy regulatorów, natomiast dla procesów dyskretnych układy maszyn abstrakcyjnych, jak maszyny stanu (ang. FSM – Finite State Machine) czy automaty (ang. automaton).

❖ **Tworzenie algorytmów regulacji**

Podstawowym regulatorem, tzn. najczęściej wykorzystywanym w ISP, jest regulator typu PID. Regulator **PID** jest to tzw. regulator proporcjonalno-całkowo-różniczkujący (ang. Proportional-Integral-Derivative controller). Schemat blokowy typowego regulatora PID pokazano na rysunku 25.

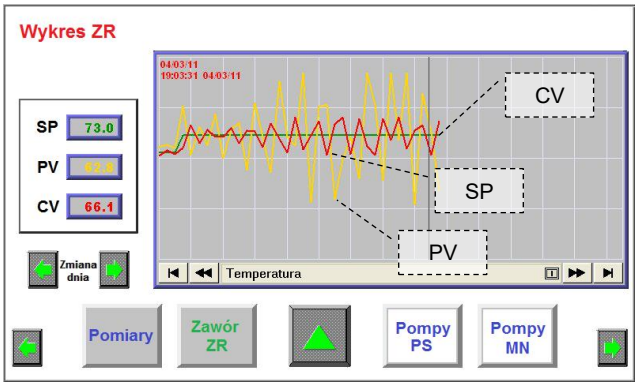


Rys. 25. Schemat zastępczy typowego, idealnego regulatora PID
Fig. 25. Equivalent circuit of a typical, ideal PID regulator

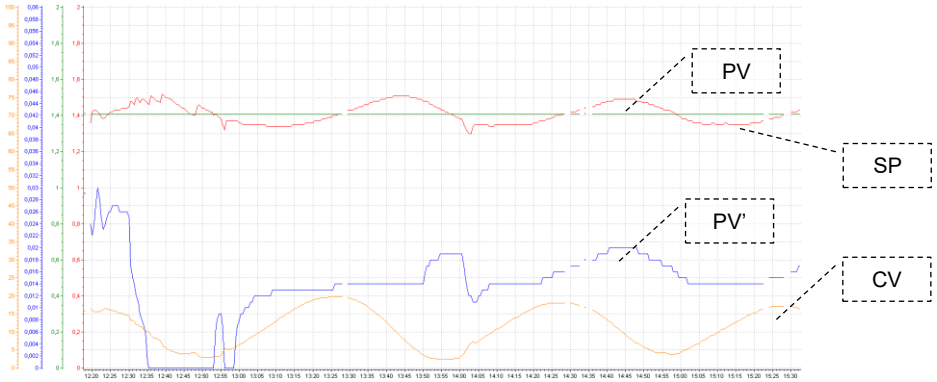
Zgodnie z rysunkiem 20 na wejściu regulatora pojawia się wartość tzw. uchybu, czyli różnicy pomiędzy wartością zadaną (ang. set point) a wartością zmierzoną z obiektu. Dlatego potocznie za wejście programowego bloku regulatora przyjmuje się wartość zadaną i wartość zmierzoną. Działanie regulatora PID ma na celu utrzymanie danej wartości w układzie na poziomie zadanym, na podstawie jej pomiaru na wejściu i oddziaływania na nią na wyjściu. Istotnym czynnikiem jest czas próbkowania wyjścia regulatora, który powinien być jak najkrótszy, aby obiekt regulowany nie pozostawał w stanach przejściowych.

Regulator jest złożeniem trzech członów regulacji pracujących ze sprzężeniem zwrotnym. W członie proporcjonalnym (P) regulator wyznacza wartość tzw. uchybów pomiędzy wartością zadaną a bieżącą wartością zmierzoną i ją kompensuje. Człon całkowy (I) kompensuje akumulację uchybów z poprzednich cykli regulacyjnych. Natomiast człon różniczkujący (D) kompensuje uchyby przewidywane. Suma ważona wartości wypracowanych w członach PID stanowi sygnał oddziałujący na obiekt. Regulator taki stosuje się wówczas, gdy obiekt jest dynamiczny, nie jest znana jego reakcja i nie można jednoznacznie wyliczyć odpowiedzi układu na pobudzenie. Przez odpowiednią parametryzację pracy algorytmów w członach regulatora można zapewnić oddziaływanie na obiekt, zgodne z zamierzonym celem. Wartości K_P, T_I i TD stanowią podstawowe nastawy regulatora. Regulator działa poprawnie, gdy jest poprawnie sparametryzowany. Błędna parametryzacja i wynikające z niej błędne użycie regulatora może prowadzić do niestabilności układu regulowanego. Parametryzacja regulatora, czyli tzw. strojenie, musi być zawsze dopasowana do danego obiektu.

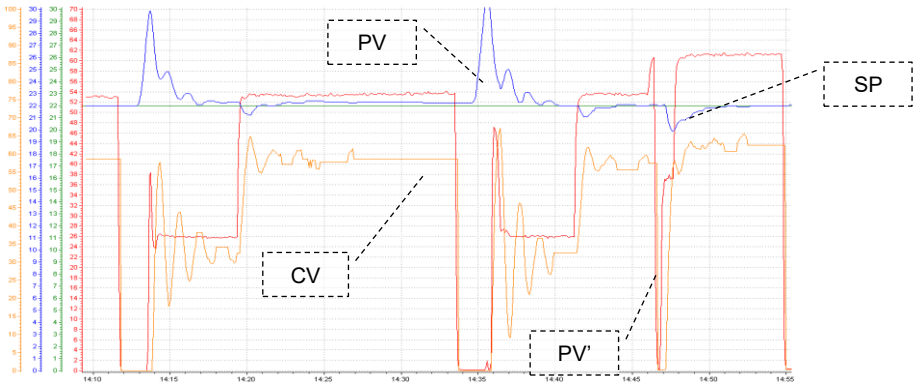
a)



b)



c)



Rys. 26. Ilustracja działania regulatorów PID w praktyce[10]
Fig. 26. Illustration of practical operation of PID

Regulatory PID są najbardziej popularnymi regulatorami w systemach automatyki i dla większości zastosowań są wystarczające. Na rysunku 26a) przedstawiono przykładowy ekran HMI (zob. 3.2.8), na którym prezentowany jest wykres działania regulacji temperatury przy użyciu zaworu regulacyjnego. Na wykresach prezentowana jest wartość zadana (SP, zadana, ang. setpoint), wartość zmierzona z obiektu (PV, procesowa, ang. process value/variable) oraz wartość oddziaływania (CV, sterująca, ang. control value/variable).

Na wykresie z rysunku 26b) przedstawiono regulacje poziomu w zbiorniku (PV) przez sterowanie stopniem otwarcia zaworu (CV). Wyświetlane są dwie wartości procesowe. War-

[10] Przedstawiono dzięki uprzejmości PPHW Proloc Sp. z o. o.

tość PV pochodzi ze zmiennej reprezentującej pomiar poziomu i stanowi sprzężenie zwrotne regulatora. Wartość PV' odzwierciedla bieżący pomiar ciśnienia nad medium i nie bierze udziału w przetwarzaniu w algorytmie regulacji. Mimo że nie stanowi wartości, na którą regulator oddziałuje bezpośrednio, wartość PV' jest zależna od pracy regulatora. Zależność ta wynika z praw fizyki wiążących zmiany ciśnienia w zbiorniku w funkcji zmian poziomu medium. Oczywiście wartość ta zależna jest również od innych czynników wynikających z innych elementów wykorzystywanego układu automatyki. W przykładzie (c) pokazano podobnie jak w (a) regulację temperatury od przepływu sterowanego stopniem otwarcia zaworu. Wartość PV reprezentuje temperaturę, PV' przepływ, a CV steruje otwarciem zaworu.

Wszystkie trzy przykłady ilustrują działanie rzeczywistych regulatorów. Zaprezentowanych przykładów nie należy traktować jako wzorcowych, ich parametryzacja nie jest idealna, jednak wyraźnie ilustrują one działanie algorytmu regulacji w regulatorze PID.

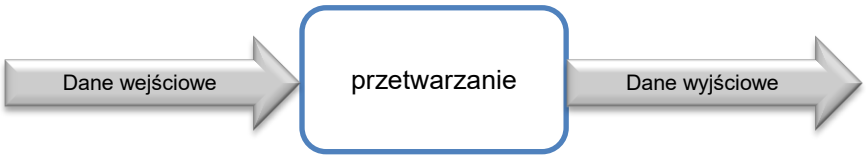
Regulatory rozmyte (ang. fuzzy logic controller) działają na bazie logiki rozmytej, czyli wielowartościowej. Tego typu regulatory znajdują zastosowanie tam, gdzie występują niejednoznaczności logiczne w opisie zjawisk, gdzie działanie w logice dwuwartościowej (TRUE, FALSE) prowadzi do wystąpienia sprzeczności logicznych oraz gdy dane są niepewne i nieprecyzyjne. Regulator działa na zasadzie przetwarzania danych (wnioskowania, interpretacji) ze zbiorów rozmytych według przyjętej bazy reguł. Dane wejściowe są przypisywane do zbiorów rozmytych w trakcie tzw. rozmywania (fuzyfikacji). Wydobycie danych wyjściowych ze zbiorów rozmytych do fizycznego oddziaływania na obiekt odbywa się w procesie tzw. wyostrzania (defuzyfikacji). Oba procesy sprowadzają się najczęściej do serii testów logicznych na progowanie wartości i liczenia średnich ważonych [193].

❖ **Obsługa maszyn abstrakcyjnych**

Regulacje dyskretne wiążą się z teorią automatów [259], [77]. Zadania dotyczą oprogramowania automatów sekwencyjnych i kombinacyjnych oraz wszelkich układów logicznych w postaci maszyn stanów (por. 2.1.2). Zagadnienia te powinny być dobrze znane informatykom. Dla przypomnienia:

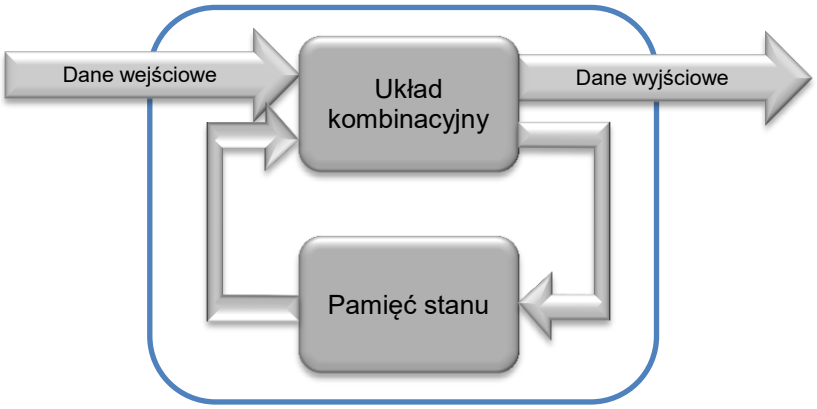
- Logiczne układy kombinacyjne, to wszelkie układy logiczne implementowane jako funkcje. Rezultat ich działania zależy od danych wejściowych i nie zależy od stanu. Stan układu, czy też jego pamięć, instancja, alokacja nie występuje. W analogii do obiektów programistycznych układ taki stanowi funkcja lub ogólnie rozkaz, niewymagający tworzenia instancji. Układy kombinacyjne nie wymagają języków specjalistycznych, choć

niektóre z nich mogą zdecydowanie podnieść czytelność kodu, w szczególności języki graficzne opisujące logikę tzw. drabinkową[11] lub języki diagramowe[12].



Rys. 27. Idea układu kombinacyjnego
Fig. 27. Idea of combinational circuit

- Logiczne układy sekwencyjne, to układy z tzw. pamięcią, czyli mające swój lokalny stan wewnętrzny. Ich działanie zależy od danych wejściowych oraz od stanu układu. Przy programowaniu ich odpowiednikami będą wszelkiego rodzaju elementy programowe wymagające utworzenia instancji w celu ich użycia, np. obiekty, bloki funkcyjne itp. Do oprogramowania układów mogą być wykorzystane dowolne języki zorientowane na akcje, czyli przetwarzanie danych.



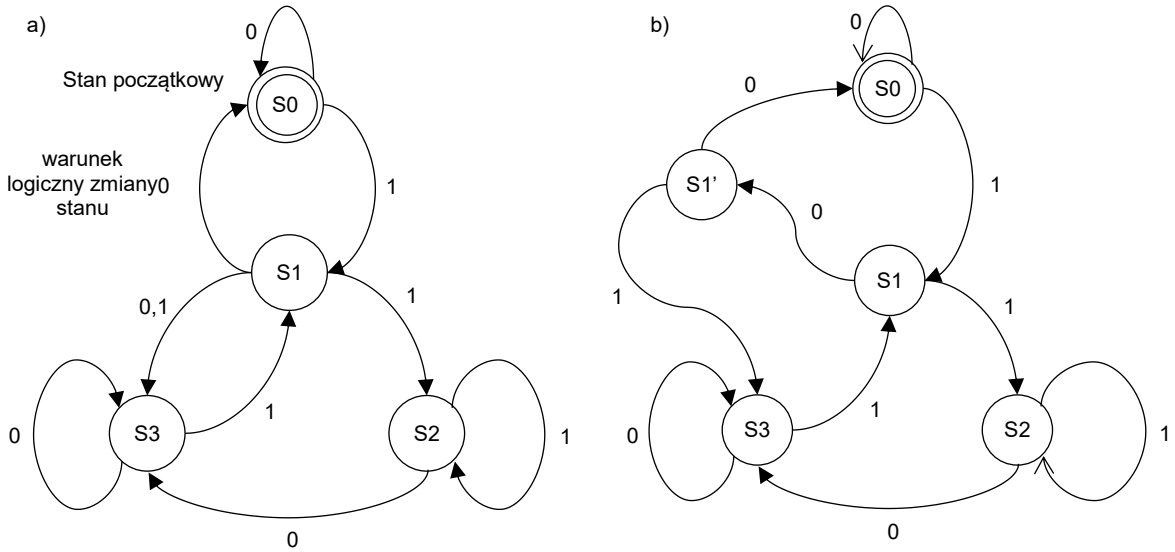
Rys. 28. Idea układu sekwencyjnego
Fig. 28. Idea of sequential circuit

- Maszyny stanów (automat skończony, ang. FSM – Finite State Machine) są abstrakcyjnym pojęciem ogólnym, określającym układ umożliwiający opis dynamiki działania układu przez wyodrębnienie statycznych i powiązanych ze sobą stanów oraz określenie dyskretnych warunków przejścia między nimi. Na rysunku 29 zamieszczono przykład maszyny niedeterministycznej (a) i deterministycznej (b). Stany *S* odzwierciedlają stany sterowanego układu. Przejścia międzystanowe mogą być dowolne, są realizowane przez badanie warunków i nie oddają chronologii działań. Więcej na temat maszyn stanów można znaleźć w publikacjach [259], [294]. W zastosowaniach ISP często spotyka się konieczność oprogramowania maszyny stanów. Najczęściej obiektem programistycznym

[11] Na przykład IL, LD, STL, LAD itp.
[12] Na przykład FBD, CFC, FUPLA itp.

realizującym maszynę stanów jest program lub jakiś rodzaj bloku kodu tworzący instancję i pozwalający na cykliczne uruchomienie.

Utworzenie maszyny stanów wymaga języka programowania umożliwiającego użycie przynajmniej instrukcji warunkowych, skoków i sekwencji rozkazów przetwarzających, tzw. akcji powiązanych z danym stanem. Dedykowane języki umożliwiające opis maszyny stanów powinny być klasyfikowane bardziej jako języki modelowania[13] niż programowania. Z natury swego przeznaczenia są orientowane na opis procesu.

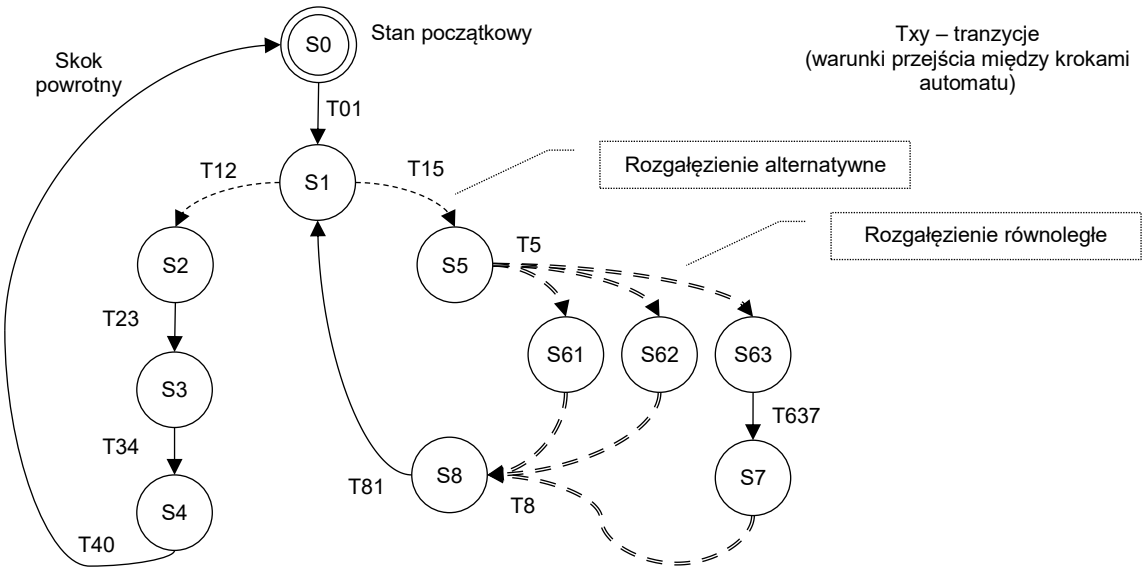


Rys. 29. Ilustracja przykładowych skończonych maszyn stanów
Fig. 29. Illustration of examples of finite state machines

- Rozszerzenie pojęcia FSM do grafu stanów i przejść między nimi stanowi bardzo użyteczne narzędzie do modelowania obiektów sekwencyjnych (zob. rysunek 24b). W zastosowaniach ISP często spotyka się obiekty, których działanie można opisać nie tyle przez maszynę stanów co przez automat sekwencyjny. Przykładowy graf opisujący taki automat przedstawiono na rysunku 30.

Stany *S* odzwierciedlają kolejne kroki automatu, reprezentując jednocześnie chronologię (następstwo) działań. Tranzycie *Txy* definiują warunki przejść z danego kroku do następnego. W opisie sekwencji można rozważać rozgałęzienia i skoki. Rozgałęzienia (dywergencje) i złączenia (konwergencje) mogą opisywać działania wykonywane alternatywnie lub równolegle. Skoki umożliwiają przejście do dowolnego kroku w sekwencji.

[13] Na przykład UML, STD, HiGraph itp.



Rys. 30. Ilustracja przykładowego grafu automatu sekwencyjnego
Fig. 30. Illustration of a sample sequential graph

Przy programowaniu, do tworzenia opisu automatów sekwencyjnych wykorzystuje się specjalne języki opisujące sekwencje stanów i przejścia[14] (zob. 3.2.13, rys. 120). Języki te słabo nadają się do opisu klasycznej maszyny stanów. Języki te są zorientowane na opis działań w kontekście ich sekwencji i warunków przejść między kolejnymi krokami w tejże sekwencji, a nie na definicje stanów i wzajemnych przejść między nimi. W praktyce, pozwala na to użycie rozgałęzień i skoków, choć kosztem czytelności opisu. Języki takie są zorientowane na opis procesu, co zdecydowanie podnosi czytelność całości kodu danego programu.

Więcej na temat zadań spotykanych w systemach przemysłowych oraz innych powiązanych zagadnień można znaleźć w [268]. Również w przestrzeni webowej istnieje dużo informacji na temat takich systemów i problemów z nimi związanych, np. [W1], [W11].

Podsumowując kwestię zadań, można stwierdzić, że współczesne wymagania stawiane komputerowym elementom sterującym i regulującym, w szczególności związanych z zadaniami opisanymi powyżej, mogą być wypełnione przez rozwiązania systemowe klasy ISP. Ewentualne ograniczenia nie wynikają z idei czy możliwości technicznych, lecz z doboru danego sprzętu i oprogramowania.

1.2.5. Zakres funkcjonalny

Podstawowym zadaniem ISP jest wspomaganie układów automatyki w realizacji wyżej wspomnianej pętli regulacji, czyli w obsłudze procesu przemysłowego. Jednak możliwości, jakie dają współczesne technologie, rodem m. in. z elektroniki, informatyki, teleinformatyki

[14] Na przykład SFC – Sequential Function Chart, Grafcet, Graph itp.

i komunikacji, umożliwiają realizację wielu innych zadań, które polepszają wskaźniki produkcyjne oraz zwiększają komfort i bezpieczeństwo pracy. Zadania ISP nie ograniczają się zatem tylko do komputerowego wspomagania monitorowania, sterowania czy regulacji. Do innych zadań ISP należą:

❖ **Zapewnienie interfejsu dla człowieka**

Wykorzystując zasoby systemowe i środki multimedialne, ISP umożliwia zbudowanie interfejsu z procesem, zrozumiałego dla człowieka. Możliwości z tego wynikające daleko wykraczają poza standardowe układy synoptyczne i stale się poszerzają wraz z rozwojem technik mobilnych oraz metod obrazowania danych. Wykorzystując taki interfejs, człowiek może dokonać **oddziaływania** na inne elementy systemu, jak i procesu. Więcej na ten temat w podrozdziałach 3.2 i 4.1.

Należy w tym miejscu wystosować swoisty apel do projektantów, aby przy projektowaniu i modernizacji układów automatyki nie usuwać „na siłę" synoptyki na rzecz komputerów. Układy synoptyczne są proste, niezawodne i przejrzyste. Jeżeli istnieje informacja wymieniana z człowiekiem, która jest kluczowa z punktu widzenia prowadzenia procesu i bezpieczeństwa, to wskazane jest, aby dostęp do niej był szybki i jednoznaczny. Należy minimalizować liczbę elementów, które pośredniczą w oddziaływaniu, i tym samym nie narażać użytkownika na np. konieczność szukania właściwego okna, manipulację myszą bądź czekanie na restart komputera czy obsługę innych zadań, w sytuacji gdy każda sekunda ma znaczenie. Wyraźnymi przykładami mogą być tu np. wyłączniki awaryjne lub sygnalizatory zagrożeń.

❖ **Zapewnienie interfejsu dla innych systemów**

Istotną cechą systemów jest zdolność do integracji z innymi systemami (por. 3.5). Czasy, kiedy komputer pracował co najwyżej w połączeniu z urządzeniami zewnętrznymi, lub systemy stanowiły hermetyczne wyspy w przedsiębiorstwie, już dawno odeszły w niepamięć. Poszerzanie funkcjonalności systemów wiąże się nieodzownie z koniecznością ich wzajemnej integracji, a co za tym idzie, budowania interfejsów taką integrację umożliwiających. Dotyczy to zarówno interfejsów komunikacyjnych, reprezentacji danych, jak i zarządzania ich przepływu. Zagadnienia integracji generują szeroki wachlarz problemów [320], [286], [49]. Współcześnie, jednym z największych wyzwań jest heterogeniczność integrowanego środowiska sprzętowo-programowego, w tym łączenie systemów przez sieci publiczne. Celem jest umożliwienie przepływu danych w fabrykach i ich porozrzucanych oddziałach oraz zwiększenie dostępności tych funkcji systemu, które mogą być używane zdalnie.

❖ **Zapewnienie obsługi historii przebiegu procesu i działania systemu**

Obserwacja procesu i sterowanie są funkcjami podstawowymi umożliwiającymi obsługę procesu wykonywaną na bieżąco. W praktyce bardzo często istnieje potrzeba analizy działania systemu po fakcie lub nawet bez komunikacji z procesem (ang. off-line). Do tego celu system musi zapewnić:

- składowanie zdarzeń i danych wraz ze znacznikami czasowymi,
- odczyt i jednoznaczną identyfikacje zachowanych danych,
- prezentację i analizę zachowanych danych,
- odtworzenie kolejności zdarzeń,
- zidentyfikowanie zdarzeń w czasie względem odczytu względnego lub bezwzględnego zegara systemowego.

Analiza zachowanych informacji może być prowadzona przez człowieka lub automatycznie przez funkcje systemowe. Do gromadzenia danych wykorzystuje się bazy danych (por. 3.2.3).

❖ **Informowanie o stanach nieprawidłowych**

Wspomniana powyżej funkcjonalność automatycznej analizy danych wraz z odpowiednimi definicjami wzorców analizy (model, progi, charakterystyka czasowa itp.) umożliwia wykrywanie stanów nieprawidłowych przebiegu procesu i działania systemu. ISP zapewnia funkcje dostarczania takich informacji dla użytkownika. Prezentacja może być w postaci:

- alarmów – sygnalizacji stanów nieprawidłowych; najczęściej alarmy są stanami zmiennych dyskretnych (binarnych), stanowiącymi wynik operacji logicznej (warunku),
- raportów – zbiorczej prezentacji zebranych informacji w jakimś konkretnym kontekście; najczęściej dotyczą czytelnej prezentacji w języku naturalnym, tabeli, wykresu itp., dotyczącej wybranego okresu czasu i/lub zakresu danych,
- komunikatów – krótkich informacji opisujących stan funkcjonowania; najczęściej są to krótkie raporty o stanie jakiegoś układu,
- innych form zrozumiałych dla człowieka, w tym multimedialnych.

Stosowane formy zależą od dostępnych środków, najczęściej są to prezentacje graficzne, tekstowe i dźwiękowe dostarczane jako aplikacje, dokumenty, SMSy itp.

❖ **Przewidywanie zdarzeń**

Na bazie analizy danych pozyskiwanych z procesu można również utworzyć funkcjonalność przewidywania pewnych stanów układu, np. awarii, zużycia, konieczności przeglądu, remontu itp. Funkcji takich dostarczają zarówno proste aplikacje analizujące w czasie rze-

czywistym dane pomiarowe i ruchowe, jak również bardziej złożone bazujące na zgromadzonych danych historycznych, algorytmach sztucznej inteligencji, a nawet algorytmach eksploracji danych (ang. data mining) [213].

❖ **Wspomaganie zarządzania obsługą procesu**

Oprócz prowadzenia procesu, ISP może zapewnić dostarczenie danych do zarządzania takim prowadzeniem. Dotyczy to planowania produkcji, określania parametrów wydajnościowych, limitów, zakupów, dostaw itp. (np. wskaźniki oszczędnego gospodarowania, zob. 3.2.9). Aplikacje realizujące takie funkcje są przeznaczone dla człowieka, a obszarem działania jest przestrzeń biurowa. Nie mają one bezpośredniego kontaktu z procesami technologicznymi, ale do realizacji rzeczywistego wspomagania działania muszą mieć wgląd w bieżące i historyczne dane pracy ISP. Więcej na temat wspomagania zarządzania można znaleźć w 3.2 oraz w [262], [231].

❖ **Zwiększenie bezpieczeństwa funkcjonalnego**

Podsystem bezpieczeństwa może stanowić element ISP jak również ISP może współpracować z niezależnym systemem bezpieczeństwa. Funkcje zapewniania poziomu bezpieczeństwa prowadzenia i przebiegu systemu przemysłowego są jednymi z kluczowych funkcji systemów kontrolujących procesy przemysłowe. Komputerowe wsparcie, jakie istnieje w ISP, daje duże możliwości w tej dziedzinie, gdyż oprócz mechanicznych, elektrycznych czy elektronicznych zabezpieczeń w ISP można zastosować wsparcie „inteligentne", wynikające z działania programów. Zapewnianie bezpieczeństwa staje się przez to procesem dynamicznym, a nie tylko zapewnieniem środków statycznych. Istotne jest, aby ISP w sposób ciągły zwiększał bezpieczeństwo układu, czyli aby mechanizmy bezpieczeństwa stanowiły wartość dodaną do cech całości układu. Szerzej o bezpieczeństwie traktuje podrozdział 3.5.4.

❖ **Kontrola działań człowieka**

Oprócz zapewniania bezpieczeństwa od technicznej strony działania systemu, jednym z zadań ISP jest również kontrola nad działaniami układów, które nie są deterministyczne i mogą celowo bądź nieświadomie wprowadzać niepoprawne dane do systemu. Typowym źródłem niepoprawnej informacji i niedeterministycznych zachowań jest człowiek. W każdym rozwiązaniu pewne działania w obsłudze procesu przemysłowego muszą być prowadzone bądź nadzorowane przez człowieka. Dotyczy to głównie obsługi sytuacji szczególnych oraz działań diagnostyczno-serwisowych i wymaga manualnego oddziaływania na obiekt. Zadaniem ISP jest niedopuszczenie do niewłaściwego oddziaływania (por. 4.1.2).

1.2.6. Standaryzacja ISP

Z dziedziną ISP tak jak z każdym innym obszarem technicznym wiąże się pytanie, czy środki i metody działania w tym zakresie nie należy poddać normalizacji. Odpowiedź jest pozytywna, a dla osób mających wątpliwości w niniejszym podrozdziale znajduje się kilka zdań wyjaśnienia.

Ustalenie standardów znacząco ułatwia prace integratorom. Produkty zgodne z daną normą muszą spełniać jej wymogi. Zatem, w zakresie definiowanym przez normę można się spodziewać zgodności cech konstrukcyjnych i funkcjonalnych z tym zakresem. Standaryzacja może dotyczyć zarówno urządzeń, oprogramowania, jak i metod ich wytwarzania, testowania itp. Umożliwia to stosowanie urządzeń, oprogramowania i narzędzi od różnych producentów, jak i zdecydowanie upraszcza przejście z działań powiązanych z danymi rozwiązaniami na inne.

Istnieje jednak inne spojrzenie na sprawę standaryzacji. Można obawiać się, że ujednolicenie rozwiązań spowoduje stagnację w rozwoju oraz ograniczy konkurencyjność dostępnych rozwiązań. Jest to prawdą, jeśli założy się niezmienność standardów. W realiach rynkowych istotnie nowe konstrukcje i podejścia są z reguły wprowadzane wbrew istniejącym rozwiązaniom. Jeśli dane rozwiązanie jest wystarczająco dobre i popularne, to efektem będzie z pewnością rozszerzenie normy oraz jego ustandaryzowanie.

Z obserwacji praktycznych można stwierdzić, że techniki, które nie zostały ustandaryzowane, potrafią również dobrze funkcjonować na rynku. Przykładem jest protokół Modbus, który przez długie lata był implementowany w urządzeniach, zgodnie z opisami pochodzącymi od jednostek komercyjnych, a nie od organizacji standaryzujących. Mimo to zyskał ogromną popularność i jest nadal stosowany[15]. Jednak efektem braku standaryzacji były implementacje różniące się na tyle, że można znaleźć urządzenia, które, mimo że obsługują Modbus, nie są w stanie współpracować ze sobą.

Zanim zaczniesz coś wymyślać od podstaw, sprawdź, czy ktoś już tego nie wymyślił i czy nie ma rozwiązań standardowych. Niekompatybilność i niestandardowość rozwiązań nie przysparza klientów.

Warto jeszcze zwrócić uwagę na problem unikania pełnej zgodności z normami. Często zdarza się, że producenci reklamują produkt jako zgodny ze standardem. Jednak w praktyce zgodność jest tylko częściowa, a informacji o tym niełatwo się doszukać w podstawowych materiałach reklamowych. Dotyczy to często wspomnianego protokołu Modbus i implementacji jego funkcji. O ile funkcje o niskich numerach są zwykle implementowane,

[15] Dotyczy roku publikacji.

o tyle funkcje o wyższych numerach już nie zawsze. Podobnie jest np. ze zgodnością z normami EMC lub nawet z implementacją niektórych dyrektyw kompilatora. Dlatego przed zakupem należy sprawdzać dokumenty określające zakres standaryzacji.

W Polsce obowiązują normy zatwierdzane i publikowane przez Polski Komitet Normalizacyjny. Ponieważ Polska jest członkiem Unii Europejskiej, wiele specjalistycznych norm aktualnie obowiązujących pochodzi od wymagań unijnych, a także z ogólnoświatowych ustaleń normalizacyjnych. Międzynarodową organizacją normalizacyjną w dziedzinie technik elektrycznych i elektronicznych oraz powiązanych jest od 1992 organizacja IEC – International Electrotechnical Commission (Międzynarodowa Komisja Elektrotechniczna). Jest to pozarządowa organizacja typu non profit. Została stworzona w wyniku międzynarodowych spotkań i dyskusji w Wielkiej Brytanii w 1906 roku. W 1948 roku jej siedziba została przeniesiona do Szwajcarii [W23].

W temacie ISP można doszukać się wielu związanych norm. Jednak do najważniejszych należą cztery:

- IEC/PN-EN 61131 – Sterowniki programowalne (ang. Programmable controllers)

Norma IEC61131 [M26] jest standardem międzynarodowym dla urządzeń komputerowych klasy PLC. Oznacza to, że odnosi się do wszelkich urządzeń komputerowych pracujących z funkcjonalnością PLC, czyli wykorzystywanych do sterowania i kontrolowania maszyn oraz procesów technologicznych, w tym do: sterowników, powiązanych urządzeń peryferyjnych, urządzeń programujących i debugujących, urządzeń HMI i innych. Składa się z dziewięciu części. Dla integratorów i programistów PLC najważniejsze części to: pierwsza – dotycząca definicji pojęć [M19] oraz część trzecia [M21] – dotycząca programowania. Ponadto dla deweloperów istotna jest część ósma [M16] – związana z zagadnieniami aplikowania i implementowania języków. Pozostałe to: część druga [M20] dotycząca sprzętu, czwarta [M15] dotycząca wytycznych dla użytkownika, piąta [M22] związana z komunikacją, szósta [M23] bezpieczeństwa funkcjonalnego, siódma [M24] opisująca zagadnienia programowania rozmytego i dziewiąta [M25] dotycząca interfejsów dla komunikacji z małymi układami obiektowymi (ang. SDCI – Single-drop Digital Communication Interface, IO-Link [M65]). Pierwsza wersja normy ukazała się w 1998 roku. Od tamtej pory wystąpiło kilka aktualizacji, m. in. część trzecia w 2004 i 2013. Więcej na temat zagadnień z normy IEC61131 znajduje się w podrozdziale 2.6.

- IEC/PN-EN 61499 – Bloki funkcjonalne (funkcyjne, ang. Function blocks)

Norma dotyczy systemów rozproszonych, w tym sterowania i automatyzacji procesów technologicznych. Norma wprowadza pojęcie bloku funkcjonalnego, które umożliwia zamodelowanie systemu jako całości w oderwaniu od konkretnej implementacji sprzętowo-programowej. Opis dotyczy przetwarzania, przepływu danych oraz przekazywania zdarzeń.

Kodowanie algorytmów może być realizowane zgodnie z IEC61131. Norma składa się z czterech części, z czego część trzecia obecnie nie jest publikowana. Pierwsze wersje normy pojawiły się w latach 2000-2001, zostały one uaktualnione w 2005, a ostatnie modyfikacje zostały wydane w roku 2013 jako wersja druga. Część pierwsza normy dotyczy architektury systemów [M34], część druga dotyczy narzędzi programowych [M35], a część czwarta reguł zgodności urządzeń oraz przenośności oprogramowania i konfiguracji [M36]. Więcej na ten temat znajduje się w podrozdziale 2.5.

- IEC/PN-EN 61158 – Przemysłowe sieci komunikacyjne – Specyfikacje magistrali miejscowej (ang. Industrial communication networks – Fieldbus specifications)

Dotyczy wykorzystywania sieci polowych w przemysłowych systemach sterowania i definiuje ich typy. Zawiera sześć części i opisuje trójwarstwowy model sieci polowych (ang. media, data-link, application, por. 2.4.2). Część pierwsza [M27] dotyczy przeglądu sieci i wytycznych dla pozostałych części oraz normy profili IEC 61784. Część druga [M28] to opis usług warstwy fizycznej. Część trzecia [M29] dotyczy usług, a część czwarta [M30] protokołów warstwy łącza. W części piątej [M31] znajduje się opis usług, a w części szóstej [M32] protokołów warstwy aplikacji. IEC61158 stanowi podstawową normę związaną z sieciami przemysłowymi. Pierwsza wersja pochodzi z roku 1993, a najnowsze uaktualnienia z roku 2014. Części 3-6 są mocno rozbudowane i podzielone na kilkanaście dokumentów z opisem poszczególnych podtypów sieci polowych względem konkretnych warstw protokołów i usług.

- IEC/PN-EN 61784 – Przemysłowe sieci komunikacyjne – Profile (ang. Industrial Communication Networks – Profiles)

Norma zawiera pięć części i dotyczy identyfikacji i opisu protokołów oraz definicji podzbiorów usług, tzw. profili, dla protokołów zdefiniowanych w IEC61158. Część pierwsza dotyczy definicji profili sieci polowych [M39]. W części drugiej [M40] rozszerzono profile polowe o profile dla sieci przemysłowych opartych na Ethernecie [M75]. Część trzecia stanowi uzupełnienie o zagadnienia bezpieczeństwa funkcjonalnego [M41]. Część czwarta nie jest aktualnie publikowana. Natomiast część piąta [M42] traktuje o zagadnieniach instalacyjnych dla poszczególnych profili. Pierwsza wersja normy pochodzi z 2003 roku. Ostatnie modyfikacje podobnie jak IEC61158 zostały opublikowane w 2014 roku. Części trzecia i piąta są podzielone na wiele dokumentów, dotyczących konkretnych profili sieci (ang. CPF – Communication Profile Familes). Więcej na temat sieci i ich standaryzacji znajduje się w podrozdziałach 2.4.4 oraz 3.3.

Informacje z ww. norm będą wykorzystywane w dalszej części książki. Szczególnie części pierwsza i trzecia normy IEC61131 okażą się przydatne przy omawianiu zagadnień pro-

gramowania. Ich tłumaczenia na język polski zostały wykonane, jednak w 2004 roku zostały wycofane i aktualnie obowiązują tylko wersje angielskojęzyczne.

W organizacji IEC normy są tworzone przez grupy ekspertów zgrupowanych w komitetach technicznych (TC, ang. Technical Committees). W ramach komitetów działają podkomitety (SC, ang. Subcommittees), a w nich grupy robocze (WG, JWG, ang. Working Groups, Joint WG), grupy projektowa (PT, Project Teams) i grupy utrzymania (MT, Maintanance Teams). Można wyróżnić grupy robocze stworzone dla dziedzin programowania (IEC61131 [M26] oraz IEC61499 [M34]), inżynierii (IEC62453 [M51] oraz IEC61804 [M43], [M44]), jak również integracje poziome (IEC62541 [M52]). Komitet techniczny TC65 odpowiada za tworzenie standardów dla systemów i elementów używanych dla procesów przemysłowych i sterowania związanych z procesami ciągłymi i wsadowymi. (ang. International standards for systems and elements used for industrial process measurement and control concerning continuous and batch processes). W działaniach Komitetu uczestniczy 26 krajów, a 15 obserwuje proces standaryzacji w tym zakresie.

Tabela 3

Wykaz związanych tematycznie grup i prac standaryzacyjnych IEC (wybrane)

TC/SC	*WG/MT*	*Zakres działań*	*Powiąz. normy*
TC56	Niezawodność, rzetelność (ang. reliability, dependability)		
	WG1-4	Terminologia, techniki, zarządzanie i systemy (ang. terminology, techniques, management and systems.)	IEC 60300
JTC1	Technologie informacyjne (ang. Inforation Technology)		
	SWG5	Internet rzeczy (ang. Internet of Things)	w toku
	SWG7	Sieci sensorowe (ang. Sensor Networks)	w toku
TC65	Pomiary, sterowanie i automatyzacja w procesach przemysłowych (ang. Industrial-process measurement, control and automation)		
	WG10	Bezpieczeństwo dla pomiarów i sterowania przemysłowego – bezpieczeństwo sieci i systemów (ang. Security for industrial process measurement and control – Network and System Security)	IEC 62443
		Do grupy przynależą organizacje: EWICS, ODVA, EtherCAT TG, PI, ISA SP99, ISA SP100, PCSF, Modbus, PNET, Fieldbus Foundation.	
SC65A	Aspekty systemowe (ang. system aspects)		
	WG14	Wytyczne bezpieczeństwa funkcjonalnego (ang. Functional Safety Guide)	IEC 61508

cd. tabeli 3

TC/SC	*WG/MT*	*Zakres działań*	*Powiąz. normy*
SC65B	Urządzenia pomiarowe i sterujące (ang. Measurement and control devices)		
	WG7	Programowalne systemy sterowania (ang. Programmable Control Systems)	IEC 61131
	WG15	Bloki funkcyjne (ang. function block)	IEC 61499
SC65C	Sieci przemysłowe (ang. Industrial Networks)		
	WG12	Bezpieczeństwo funkcjonalne dla sieci polowych (ang. Functional Safety for Fieldbus)	IEC 61784-3
	WG13	Bezpieczeństwo cyfrowe (ang. Cyber Security)	tak jak TC65 WG10
	WG15	Sieci wysokiej niezawodności (Ang. High Availability Networks)	IEC 62439
	WG16	Technologie bezprzewodowe (ang. Wireless)	IEC 62591 IEC 62601
	WG17	Współistnienie rozwiązań bezprzewodowych (ang. Wireless Coexistence)	IEC 62657
	MT9	Sieci polowe (ang. Fieldbus)	IEC 61158 IEC 61784-1,2
	JWG10	Okablowanie przemysłowe (ang. Industrial Cabling)	IEC 61784-5
SC65E	Urządzenia i integracja w systemach przedsiębiorstw (ang. Devices and integration in enterprise systems)		
	WG4	Specyfikacja interfejsów narzędzi urządzeń polowych (ang. Field Device Tool (FDT) interface specification)	IEC 62453
	WG7	Bloki funkcyjne dla sterowania procesem i opisu urządzeń (ang. Function blocks for process control and EDDL)	IEC 61804
	WG8	OPC UA (ang. OPC Unified Architecture)	IEC 62541

Prace nad standaryzacją urządzeń PLC trwają w podkomitecie SC65B-WG 7. Działania nad normalizacją protokołów polowych i RTE odbywają się w podkomitecie SC65C-MT9. Wykaz grup, funkcji i związanych standardów przedstawiono w tabeli 3.

Normy ulegają ciągłym zmianom. W omawianej dziedzinie aktualizacje norm odbywają się co kilka lat.

Treści norm opisywane w książkach i opracowaniach mogą być nieaktualne w momencie ich czytania. Najlepszym źródłem dokumentów normalizacyjnych są one same.

2. MODELE ISP

Wykonywanie działań projektowych w oderwaniu od pojęć abstrakcyjnych i bazowaniu tylko na konkretnych rozwiązaniach, w domyśle: tych akurat znanych danemu inżynierowi, prowadzi nie do tworzenia, lecz do rzemieślniczego powielania rozwiązań, które niekoniecznie są odpowiednie dla danego przypadku, a już na pewno nie są rozwojowe. Innymi słowy, inżynier przystępujący do rozwiązania postawionego przed nim problemu powinien spoglądać przez pryzmat wiedzy i doświadczenia, które ma, a nie przez pryzmat produktu czy technologii. Postrzeganie rozwiązań na płaszczyźnie abstrakcji odróżnia twórców od rzemieślników.

W niniejszym rozdziale skupiono się na ISP, abstrahując od modeli szerszych, obejmujących zarządzanie produkcją i całym przedsiębiorstwem opisanych w 1.2. Przedstawiono ogólne paradygmaty[16] (ang. paradigm) i wzorce projektowe[17] (ang. design pattern) mające potencjalny związek z ISP, model warstwowy jako najbardziej odpowiedni dla ISP, zagadnienia rozproszenia w ISP, abstrakcje warstw i elementów systemu, a także podejścia do modelowania systemów stosowane w normach IEC61499 oraz IEC61131. Pojęcia te należy wyróżnić, zrozumieć i przydzielić im role, zanim rozpocznie się programowanie czegokolwiek. Istnieje wiele zależności pomiędzy opisywanymi elementami, dlatego czytelnik znajdzie niektóre definicje i opisy później niż ich wystąpienia. Jest to skomentowane stosownymi odniesieniami.

2.1. Paradygmaty i wzorce projektowe

W niniejszym podrozdziale zamieszczono systematyzację popularnych paradygmatów i wzorców projektowych, które wykorzystane być mogą do tworzenia architektury lub implementacji informatycznych systemów przemysłowych i ich elementów. Należy na wstępie

[16] Paradygmaty są to pojęcia bazowe dla danej dziedziny. Stanowią podstawę do budowania pojęć pochodnych. Paradygmaty pozostają tak długo w użyciu, póki zachodzi ich przydatność dla rozwoju dziedziny.

[17] Wzorzec projektowy jest to rozwiązanie jakiegoś powtarzalnego problemu projektowego w sposób należący do zdefiniowanej i identyfikowalnej klasy rozwiązań.

zwrócić uwagę, że w większości przypadków ISP odróżniają się od tzw. typowych[18] systemów informatycznych. Systemy typowe są zorientowane na człowieka i na wspieranie jego aktywności. ISP natomiast są zorientowane na proces przemysłowy, a interakcja z człowiekiem jest funkcjonalnie odseparowana i sprowadzona tylko do funkcji towarzyszących, czasem bardzo ograniczonych. Ponadto, wzorce i modele stosowane przy projektowaniu ISP nie stanowią wyrafinowanych informatycznie i skomplikowanych rozwiązań. Ich siła bazuje zwykle na prostocie i przejrzystości, a wszelkie niepotrzebne wyrafinowanie może prowadzić do wzrostu zawodności. Dlatego wykorzystanie złożonych wzorców czy też odwoływanie się do popularnych, aczkolwiek niekoniecznie pasujących paradygmatów nie jest przeważnie potrzebne, a nawet może niepotrzebnie zaciemnić i skomplikować dany przypadek rozwiązywania problemu projektowego. Wzorce projektowe w swym szerszym spektrum znajdują zastosowanie, gdy rozważa się oprogramowanie systemów nadrzędnych, narzędzi, środowisk uruchomieniowych (ang. run-time environment) lub zagadnień integracji (por. 3.2, 3.5).

Paradygmaty i wzorce projektowe są to swoiste archetypy myślowe, umożliwiające ujednoznacznienie pojęć, definicji i sposobów działań dotyczących procesu projektowania. Stanowią więc podstawy pojęciowe w dziedzinie. W przypadku ISP istnieją paradygmaty związane z pewnymi ideami konstrukcji takich systemów czy też z ich modelami. Można wyróżniać i dobierać paradygmaty zależnie od rozpatrywanej aktywności systemu. Istnieją paradygmaty definiujące na wysokim poziomie abstrakcji działanie systemu jako całości, jak również paradygmaty dotyczące działania elementów systemu, związane np. z komunikacją czy budową programów.

Wzorce projektowe stanowią opis postępowania przy rozwiązywaniu problemów projektowych danej klasy, nie stanowią natomiast rozwiązania tych problemów. Można je porównać do pojęcia algorytmu w odniesieniu do procesu projektowania. Istnieje wiele kategorii wzorców projektowych [131], [331], ale w dalszej części rozważania dotyczą głównie wzorców architektury i implementacji.

2.1.1. Wzorce architektury

Wzorce architektury opisują współprace i współzależności elementów systemu z użyciem pojęć abstrakcyjnych wykraczających poza abstrakcję tych elementów. Są to najbardziej ogólne wzorce budowane z użyciem wysokich abstrakcji i odnoszące się do szerokiego spektrum aspektów projektowych. W kontekście architektury systemu do przydatnych wzorców ISP należy zaliczyć:

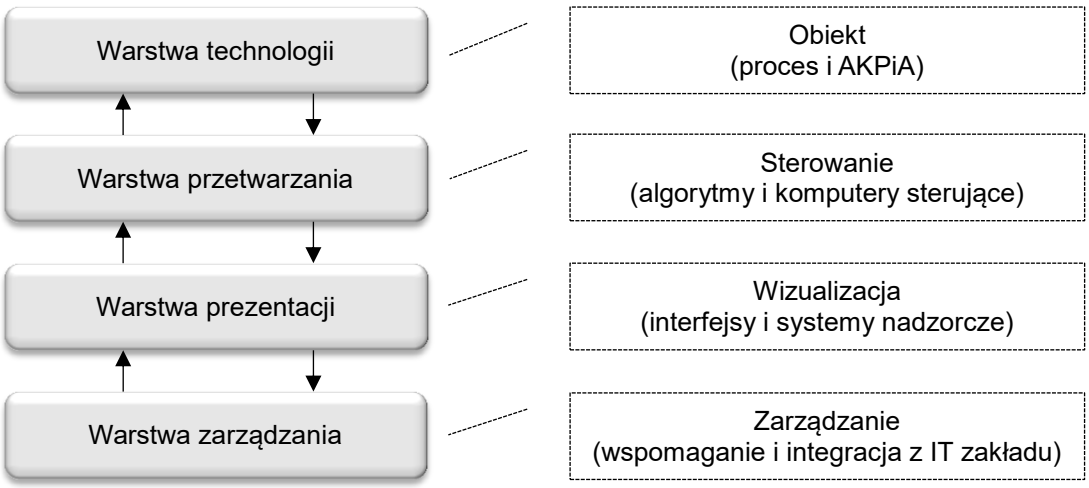
[18] W domyśle systemy biurowe, domowe, rozrywkowe (ang. home, office, entertainment) itp.

- wzorce budowy warstwowej,
- wzorce obiektowe i komponentowe,
- wzorce dziedzinowe,
- wzorce podziału struktury,
- inne wzorce w fazie adaptacji i rozwoju.

Poniżej zamieszczono opis i dyskusje wyżej wymienionych wzorców. Ponieważ wzorzec stanowi określenie sposobu realizacji, można o nim mówić jako o określonym podejściu do realizacji i dlatego pojęcia wzorzec i podejście są stosowane wymiennie.

❖ **Podejście warstwowe**

Architektura warstwowa jest architekturą najlepiej pasującą do tworzonych w praktyce ISP. Jest ona dość specyficzna. W typowych wzorcach wykorzystuje się bazę danych, logikę biznesową i prezentację, np. MVC, MVVM itp., natomiast warstwy ISP zawierają warstwę technologii, przetwarzania i prezentacji, co w prostym przypadku przekłada się na znane z automatyki funkcjonalności obiektu, sterowania i wizualizacji. Istotne jest tu rozdzielenie funkcjonalności i odpowiedzialności warstw. Przedstawiono to na rysunku 31. Można doszukiwać się większej lub mniejszej liczby warstw, zależnie od szczegółowości analizy lub złożoności systemu.



Rys. 31. Wzorzec warstwowy architektury ISP
Fig. 31. Layered pattern of ICS architecture

Warstwy są funkcjonalnie odseparowane i pracują tylko w zakresie swoich kompetencji. Pomiędzy warstwami następuje wymiana informacji. Rozwiniecie tego tematu znajduje się w kolejnych rozdziałach.

❖ **Podejście obiektowe**

Wspomniana wcześniej (zob. 1.2.3) tendencja do dostosowywania produkcji do bieżących potrzeb rynku powoduje, że cenne jest używanie rozwiązań, które dają możliwości reorganizacji systemu w sposób dynamiczny (elastyczny, ang. flexible), niegenerujących długich przestojów i wysokich kosztów. Dobrym, choć już nie najnowszym rozwiązaniem w tej mierze jest podejście zorientowane obiektowo i komponentowo (ang. Component Base Automation) [69], [71]. Podejście takie pozwala na minimalizację wysiłku przy przebudowie części zarówno sprzętowej, jak i programowej systemu. Wpływa również pozytywnie na takie cechy systemu, jak stopień rozproszenia, ogólną skalowalność, odporność na uszkodzenia, samoadaptacje, bezpieczeństwo i upraszcza utrzymanie systemu [199]. Typowe paradygmaty wzorców obiektowych to tworzenie elementów (komponentów) systemu z możliwością ich enkapsulacji, agregacji (zawierania) i łączenia.

Enkapsulacja, czyli tzw. hermetyzacja, ma na celu ukrywanie tych elementów, które są istotne dla działania danego układu, a nie są istotne i nie powinny być dostępne dla innych. W przypadku programowania obiektowego dotyczy to pól i metod danej klasy, ale w kontekście ISP enkapsulacja sprowadza się do traktowania pewnych elementów systemu jako „czarne skrzynki" z interfejsem. Użycie podejścia komponentowego umożliwia utworzenie „standardowych", jawnie zdefiniowanych interfejsów między elementami (komponentami) systemu i jego warstwami logicznymi. Tworzy się tym samym komponenty z odseparowanymi danymi i funkcjami z jednej strony oraz danymi i funkcjami przeznaczonymi do integracji z drugiej. Tworzone komponenty mogą być zagnieżdżane, czyli agregowane, jak również i łączone ze sobą celem uzyskania kolejnych komponentów o rozszerzonych funkcjonalnościach.

Rozpatrując jednak wzorce implementacji [17] w odniesieniu do podejścia obiektowego i pomijając implementację systemów wspomagających i interfejsowych[19], implementacje narzędzi deweloperskich czy implementację oprogramowania systemowego/pokładowego węzłów[20], to w ISP stosowanie wzorców obiektowych jest bardzo ograniczone. Norma IEC61131 oraz IEC61499 dopuszcza tworzenie obiektów programistycznych typu bloki funkcyjne (zob. 2.5, 2.6). Są to elementy, które wykazują cechę abstrakcji i hermetyzacji oraz umożliwiają agregację, jednak nie występuje polimorfizm i dziedziczenie, więc normalnie wzorce obiektowe są w tym przypadku przydatne tylko częściowo lub wcale.

W podejściu komponentowym również można się doszukać enkapsulacji i agregacji komponentów, jednak tak jak powyżej brak jest polimorfizmu i dziedziczenia. Podejście komponentowe jest zbliżone do przedstawionego w 2.5 użycia pojęcia bloków funkcyjnych,

[19] Na przykład SCADA, HMI (zob. 3.2).
[20] Tak zwany firmware, np. dla PLC.

lub nawet do pojęcia bloków funkcyjnych wg normy IEC61131 (2.6), tyle że pojęcie komponentu lokuje się wyżej w abstrakcjach opisujących działanie systemu.

❖ **Podejście dziedzinowe**

Dana dziedzina aplikacji z reguły wymusza tworzenie specyficznych wzorców związanych z tą dziedziną. Nie są one powszechnie znane, a ich zakres stosowalności ogranicza się do dziedziny lub konkretnej, wąskiej klasy rozwiązań. Przykładem mogą być ogólne modele przedstawione w 1.2.1, różnorakie wzorce współpracy procesów i wymiany danych stosowane w aplikacjach (zob. 3.2), czy też wzorce zapewniania bezpieczeństwa (ang. safety & security) (zob. 4).

Ponadto, można wyróżnić typowe wzorce dla komunikacji między aplikacjami (zob. 1.2.2, 2.4.4, 2.6.2). Najczęściej wykorzystywaną w praktyce architekturą dla aplikacji jest Klient-Serwer (ang. Client-Server) [94], [154], a dla przekazywania komunikatów między nimi Publikator-Subskrybent (ang. Publisher-Subscriber) [415].

❖ **Podejście z podziałem struktury**

Istnieje również szereg wzorców projektowych, które mogą mieć zastosowanie przy tworzeniu ISP, choć nie są dla tego typu systemów stworzone, a ich adaptacja może budzić dyskusje. Można do nich zaliczyć stosowanie podziału (rozdzielenia) struktury systemu wg pewnych funkcji i zadań, jakie wyodrębnione elementy pełnią w systemie. O ile rozdzielenie i odseparowanie funkcjonalności jest w ISP wskazane i wynika z omawianych powyżej podejść, o tyle standardowe wzorce podziału, jak np. popularna architektura **MVC** (ang. Model View Controller) nie przystaje ani do ISP, ani do jego elementów. Model ten został stworzony dla systemów z interfejsem graficznym, a podstawową jego zaletą jest oddzielenie warstwy prezentacji od przetwarzania. Dyskusyjność zastosowania wynika z faktu, że w węzłach ISP nie występuje ani warstwa widoku, ani modelu, może z wyjątkiem węzłów interfejsowych i specyficznego oprogramowania działającego w węzłach na bazie PC. Warto też wspomnieć, że klasyczny wzorzec projektowy MVC, mimo ogólnego przekonania, stanowi bardziej wzorzec implementacji niż architektury i powinien być rozważany na niskim poziomie abstrakcji. W przełożeniu na ISP jeszcze bardziej to dowodzi, że nie znajduje on w nim zastosowania. W żadnym z elementów programowych ISP uzależnionych od czasu nie występuje ani odseparowany model, ani widok.

Oczywiście można doszukiwać się podejścia MVC w kontekście tworzenia modelu, np. regulatora, kontrolera wykonującego zadania sterujące oraz widoku jako interfejsu. Można zaadaptować MVC jako model architektury systemu i doszukiwać się kontrolera w węzłach sterujących systemu (np. PLC), widoku w systemie SCADA i modelu w bazie danych. Można też wydzielić ekrany wizualizacyjne jako widok, przetwarzanie w SCADA i PLC oraz

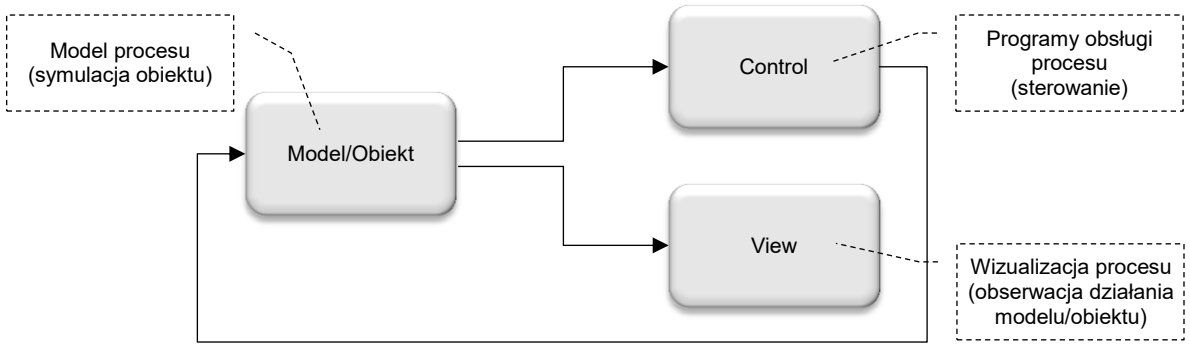
sterowniki komunikacji jako kontroler i struktury danych wraz z ich powiązaniami jako model. Jednak przy takim podejściu widać słabo rozdzieloną odpowiedzialność składowych modelu i bez wątpienia można to zaklasyfikować jako stosowanie koncepcji na siłę, co współcześnie jest częstym problemem projektantów podatnych na naciski swoistych mód lub znajdujących się pod presją marketingu.

Nie stosuj modeli na siłę, tylko dlatego że są znane lub modne.

Istnieją jednak przypadki, gdzie stosowanie podziału MVC jest przydatne i wskazane. Należą do nich zadania projektowe związane z oprogramowaniem, którego stworzenie jest uciążliwe lub nie istnieje wiedza opisująca działanie na wystarczająco sformalizowanym i szczegółowym poziomie. Są to np. zadania nietypowych regulatorów ciągłych. Gdy zadanie regulacji jest złożone i opisane tylko charakterystyką pracy układu, wówczas stworzenie modelu zapewniającego taką charakterystykę, widoku umożliwiającego dostrojenie i kontrolera, realizującego przetwarzania jest wygodne i relatywnie proste w użyciu. Potrzebne do tego są odpowiednie narzędzia udostępniające wsparcie dla podejścia MVC. Jednym z najpopularniejszych jest pakiet Matlab[21] wraz z narzędziami (ang. toolbox) pakietu Simulink[22] [47]. Efektem finalnym tak działającego narzędzia jest zwykle kod źródłowy możliwy do uruchomienia na danej platformie urządzenia komputerowego. Obecnie, stosowanie tego typu narzędzi nie sprawdzi się, jednak dla typowych zadań bądź zadań klasyfikowanych jako proste. Wynika to z bilansu nakładu pracy związanej z procesem modelowania i symulacji względem klasycznego kodowania, a także z bilansu ekonomicznego wynikającego z konieczności zakupu stosownych licencji. Jednakże dla wspomnianych zadań złożonych, tworzenie systemów wspomagane narzędziami pracującymi według wzorca MVC może stanowić korzystną alternatywę. Należy jednak wyraźnie podkreślić, że wykorzystanie MVC i innych modeli do tworzenia narzędzi oraz wszelkiego oprogramowania pokładowego węzłów niestanowiącego elementów tworzonej aplikacji ISP, ma się nijak do wykorzystania MVC do tworzenia tej aplikacji. Innymi słowy, co innego jest mieć i wykorzystywać narzędzia (np. SCADA, IDE itp.) zbudowane na podstawie danego modelu, a co innego wykorzystywać dany model do tworzenia systemu lub jego elementów z użyciem tych narzędzi.

[21] Matlab jest językiem wysokiego poziomu wraz ze środowiskiem interaktywnym umożliwiającym wygodne przygotowanie i wykonanie szerokiego zakresu zadań obliczeniowych, w tym zadań złożonych.
[22] Simulink jest interaktywnym środowiskiem dla symulacji i projektowania typu Model-Based przeznaczonym głownie dla systemów dynamicznych i wbudowanych, w tym różnorodnych systemów zależnych od czasu.

Podejście MVC w rozwinięciu Model-View-Control jest wspierane przez normę programowania systemów rozproszonych IEC61499 (por. 2.5). Według takiego podejścia dla procesu fizycznego jest tworzony model zawierający jego zależności dynamiczne, opis struktury i interfejsu wymiany danych. Dla zdefiniowanego modelu tworzony jest kod obsługujący zamodelowany proces (sterowanie) oraz wizualizacja jego działania. Zostało to zilustrowane na rysunku 32. Na etapie projektowania każdy przedstawiony blok jest opisany tzw. blokami funkcyjnymi (por. 2.5.2) umożliwiającymi realizację jego zdań. W efekcie, można dokonać dowolnych symulacji działania procesu i jego obsługi, a następnie model zastąpić rzeczywistym obiektem, łącząc rzeczywiste układy przez zdefiniowane wcześniej interfejsy. Finalnie uzyskuje się przetestowany system sterowania i wizualizacji, gotowy do pracy z obiektem ze ścisłością powiązań i interakcji wynikającą z dokładności modelu.



Rys. 32. Przykład podejścia MVC
Fig. 32. Example of MVC approach

Problemem tego podejścia jest brak narzędzi, umożliwiających zadowalające jego stosowanie w praktyce. Nad rozwiązaniami związanymi z automatyczną generacją kodu z modeli formalnych pracowały również grupy w Polsce, np. [212].

❖ **Rozwój i adaptacja wzorców**

W ostatnich latach nastąpił duży postęp w rozwoju i wykorzystaniu architektur: chmurowych (ang. cloud), wirtualizacji funkcjonalności i zasobów (ang. virtualization), internetu rzeczy (IoT – ang. Internet of Things), a także idei środowisk otaczających (ang. ambient environment).

Rozważanie wzorców typu chmura obecnie nie ma jeszcze większego sensu praktycznego w odniesieniu do ISP. Stosowanie przetwarzania i składowania w chmurze (ang. cloud computing & storage) zdarza się na poziomie wizualizacji i zarządzania lub powyżej, ale poniżej nie znajduje istotnego zastosowania. Prace w zakresie wirtualizacji zasobów i funkcjonalności systemowych oraz sieci, a także umieszczania zadań ISP w chmurach trwają i mogą stanowić atrakcyjną ścieżkę rozwoju, przynajmniej dla niektórych funkcjonalności ISP (zob. 3.2.3, 3.2.12). Dużym wsparciem dla aplikowania sterowników wirtualnych oraz

komunikacji RT w sieciach publicznych są sieci przemysłowe trzeciej generacji, w tym sieci komórkowe LTE oraz sieci IP6, umożliwiające uzyskanie gwarantowanej przepustowości w ramach usług QoS [8]. [93].

Podobnie wygląda sprawa wzorców architektur zorientowanych na usługi (ang. SOA – Service Oriented Architecture). Do konstrukcji systemów dynamicznych stosuje się też bardzo często paradygmat **SOA** [307], [18], [96], [43]. Jest to paradygmat stosowany w różnych gałęziach technologii informatycznych i nie tylko. Zgodnie z definicją [96] pojęcie SOA dotyczy modelu funkcjonowania nastwionego na zwiększenie efektywności, sprawności i produktywności przedsiębiorstwa przez potraktowanie usług jako podstawowego środka, przez który reprezentowana jest realizacja celów strategicznych systemu. Przez definiowanie i wykonywanie odpowiednich usług podejście to łączy zagadnienia biznesowe zakładu z jego informatycznymi zasobami sprzętowo-programowymi, na których te usługi są realizowane. W projektowaniu SOA abstrahuje się od fizyczności zasobów na rzecz ich logicznego opisu. Od strony użytkowej wszelkie zawiłości sprzętowo-programowe są ukryte za interfejsem usług. Stosowanie wysoko abstrakcyjnej komunikacji na poziomie usług pomiędzy węzłami systemu, nawet prostymi, jest wygodne od strony projektowej na każdym etapie życia systemu.

Dlatego wydaje się, ze idea wykorzystania SOA w ISP jest równie atrakcyjna jak w innych systemach IT. Koncepcja realizacji funkcjonalności systemu przez zestaw zdefiniowanych usług z określonymi interfejsami pozornie pasuje do opisywanych modeli. Ukrywanie implementacji jest wskazane ze względu na separacje warstw i funkcjonalności oprogramowania ISP. Jednak dla większości praktycznych aplikacji ISP nie występuje wyraźna potrzeba wydzielania i definiowania odseparowanych usług systemowych. Dla przykładu, system sterujący pracą maszyny czy prostej linii produkcyjnej z reguły składa się z jednego węzła klasy PLC, jakiejś formy lokalnego interfejsu użytkownika (HMI) i ewentualnie podłączenia do warstw wyższych typu SCADA/HMI. System bardziej zaawansowany będzie posiadał od kilku do kilkunastu węzłów różnego typu. W zakresie lokalnym pracy takiego systemu możliwe jest co najwyżej wydzielenie usług związanych z akwizycją i wymianą danych oraz synchronizacją aplikacji. Są to jednak raczej funkcje systemu w warstwie sterownia, a nie usługi w rozumieniu SOA. Usługi powinny być niezależne, co jest trudne do uzyskania na tym poziomie w ISP. Zastosowanie wzorców typu wstrzykiwanie zależności (ang. dependency incjection), service broker czy innych związanych z uniezależnianiem usług w SOA jest wysoce problematyczne. Bardziej sensowne jest rozpatrywanie tego typu architektur dla współpracy ISP z systemami nadrzędnymi. Dlatego współcześnie najlepszym zastosowaniem SOA są architektury systemów nadrzędnych i komunikacja pionowa, gdzie maleją wymaga-

nia względem ograniczeń czasowych, a rośnie rozmiar reprezentacji przesyłanej informacji [214].

Ponadto, problem z praktycznym wykorzystaniem dostępnych technologii SOA[23] wiąże się ze specyfiką ISP w dziedzinie czasu oraz rozmiaru danych. Usługi podlegają ograniczeniom czasowym. Narzut informacyjny niezbędny do funkcjonowania komunikacji jest irracjonalnie duży względem zwykle niewielkich paczek danych, jakie przekazywane są poziomo w ISP. Powoduje to problemy z ograniczeniami czasowymi zarówno po stronie komunikacji, jak i przetwarzania w węźle. Węzły ISP nie dysponują mocą obliczeniową porównywalną z komputerami klasycznymi (por. 3.1). Zatem, zadania obsługi SOA, jak choćby parsowanie XML, AutomationML [M97], [W13] czy serializacja danych, wymagają czasu na ich realizację, który wpływa negatywnie na inne zadania realizowane przez zasoby węzła, w tym zadania sterowania. Istnieją zaawansowane próby wykorzystania koncepcji SOA i są prowadzone badania w tym kierunku, np. [230], [313], [69], [70].

Bezsprzeczne jednak jest, że ujednolicona komunikacja, standardowe interfejsy usług, oddzielenie od producenta i technik składowych sprzyjają tworzeniu ISP i ich integracji. Dlatego na potrzeby ISP warto rozważać lepiej pasujące **wzorce komponentowe**[24] [48], [413]. Choć i tu praktyka pokazuje, że są to rozwiązania głównie dedykowane dla komunikacji pionowej, a zależności czasowe są określane na zasadzie oceny uzyskiwanej jakości, a nie gwarancji.

❖ Podsumowanie

Z globalnego punktu widzenia obejmującego wszystkie podsystemy informatyczne zakładu, współpracujące ze sobą na różnych poziomach abstrakcji i realizujących różne funkcje otrzymuje się zwykle system oparty na paradygmacie warstwowym opisanym na początku niniejszego rozdziału, z wyróżnionym obiektem, jego sterowaniem, wizualizacją, nadzorem, zarządzaniem itp. wraz z odpowiednimi interfejsami tychże warstw. Dopasowywanie innych wzorców jest tu sztuczne i bezcelowe, chyba że rozpatruje się konkretne elementy i podsystemy. Wskazane jest jednak mieć otwarty umysł na zagadnienia projektowania ISP i nie obawiać się zmieniać, tworzyć czy adaptować podejścia. Wywracanie utartych poglądów jest kreatywne, o ile przynosi nowe, lepsze efekty. Jak powiedział Dee Hock[25] “The problem is never how to get new, innovative thoughts into your mind, but how to get old ones out.”

[23] Na przykład SOAP, CORBA, Web Services, WCF, RPC itp.
[24] CBA, DCOM, OPC UP/DA/DX itp.
[25] Twórca i były dyrektor generalny Visa Inc.

2.1.2. Wzorce implementacji

Wzorce implementacji dotyczą ogólnej idei związanej ze sposobem kodowania. Dla dziedziny ISP istnieją wzorce implementacji typowo dziedzinowe. W ramach takich wzorców można wyróżnić wzorce typowe, przydatne dla programowania, komunikacji, przetwarzania w aplikacjach i prezentacji danych. Poniżej wymieniono kilka przykładów:

- dla programowania ISP
  - automat sekwencyjny i maszyna stanów

  Wzorzec dotyczy możliwości formułowania opisu działania procesu i wiązania z nim odpowiednich akcji obsługi. Opisany szerzej w 1.2.4 oraz w dalszej części rozdziału.
  - odpytywanie zasobów

  Określa interakcje ze źródłami danych w czasie rzeczywistym. Opisany szerzej w dalszej części rozdziału.
  - przetwarzanie z wzajemnym wykluczaniem

  Określa współdziałanie wielu zadań w węźle, jak np. działanie w cyklu lub szeregowanie. Opisany szerzej w dalszej części rozdziału.
  - zachowywanie stanu

  Określa zachowanie stanu aplikacji w sytuacji jej restartu. Opisany szerzej w dalszej części rozdziału.

- dla przekazywania danych
  - znacznik życia (bit, słowo itp.)

  Prosty i skuteczny mechanizm informowania o aktywności węzła. Najczęściej implementowany jako zmiana wartości zmiennej w określonym czasie. Przekazywanie takiej zmiennej wśród rozproszonych aplikacji i analizowanie zmian wartości umożliwia detekcję działania innej aplikacji.
  - polecenie – potwierdzenie

  Ogólny wzorzec przekazania informacji przez wysłanie sekwencji danych określających, co należy wykonać (polecenie, rozkaz, np. odczyt, zapis, synchronizacja itp.) od aplikacji źródłowej i sekwencji potwierdzającej wykonanie lub przyjęcie rozkazu do wykonania od aplikacji docelowej.
  - dane – rozkaz

  Stosowany przy dostarczaniu wszelakich danych stanowiących rodzaj parametrów działania (np. receptur). Polega na przesłaniu najpierw danych, a następnie polecenia, które określi, co i jak należy wykonać z ich użyciem.

- dla aplikacji nadzorczych
  - elementy GUI prezentujące stan procesu

  Implementacje typowych elementów graficznych wspomagających zrozumienie bieżącego stanu i przebiegu procesu (np. pola tekstowe, sygnalizatory, bargrafy, trendy, okna i inne) z lokalnymi atrybutami kontrolowanymi przez zmienne procesowe.
  - elementy GUI modyfikujące stan procesu

  Implementacje typowych elementów graficznych w formie zrozumiałej i pasującej do powiązanej funkcji technologicznej, umożliwiających użytkownikowi dokonywanie zmian stanu i wartości zmiennych lokalnych bądź systemowych (np. przyciski, suwaki, kontrolki, stacyjki, i inne). Interakcja ze zmiennymi dokonywana jest zwykle przez obsługę zdarzeń związanych z takim elementem lub bezpośrednią modyfikację powiązanych zmiennych.
  - lokalne przetwarzanie skryptowe

  Poza elementami GUI umożliwiającymi interakcje między użytkownikiem a innymi elementami systemu, typowym wzorcem implementacji jest zwykle mechanizm wykonywania skryptów (programów użytkownika) dokonujących lokalnego przetwarzania na zmiennych niezależnie od użytkownika. Ich wyzwalanie jest zwykle cykliczne lub zdarzeniowe.

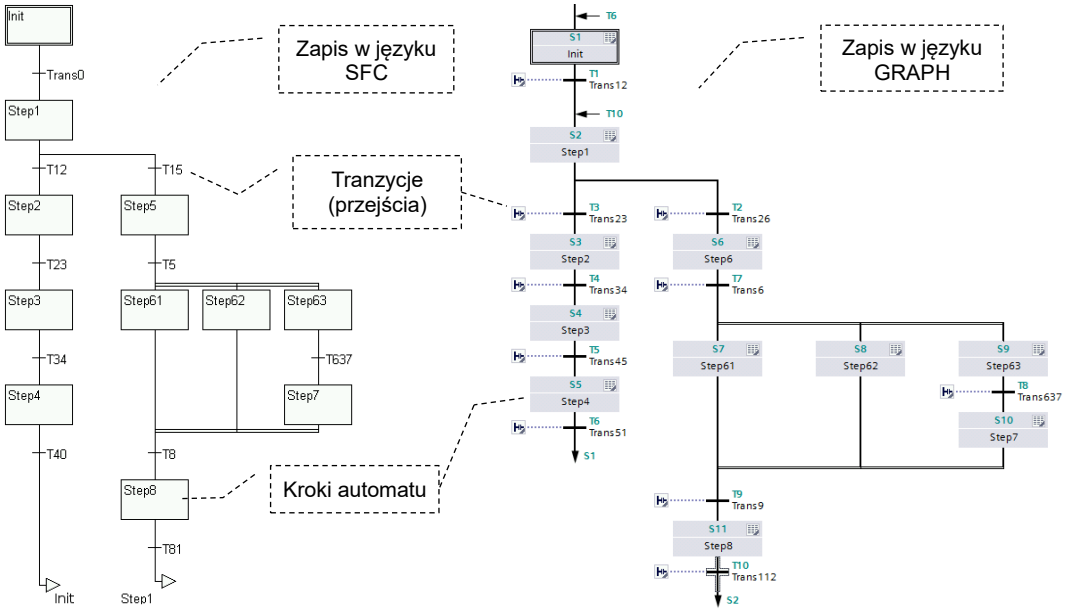
❖ **Automat sekwencyjny i maszyna stanów**

Jednym z typowych zadań programistycznych w ISP jest oprogramowanie powtarzających się sekwencji działań. Sekwencje takie mogą być zwykle opisane skierowanym grafem stanów. Zadanie projektowe sprowadza się zatem do zdekomponowania wymaganego zachowania układu do postaci niepodzielnych i niezależnych stanów oraz zdefiniowania przejść między nimi. Może to posłużyć do zbudowania modelu funkcjonowania oprogramowywanego układu w postaci abstrakcyjnej maszyny (por. 1.2.4).

Maszyna taka jako model teoretyczny działania układu stanowi dobry wzorzec implementacyjny dla ISP. Do stworzonego wzorca programista może dołożyć kod opisujący przetwarzanie i przejścia, uzyskując tym samym czytelny i łatwo utrzymywalny kod programu.

Istnieją specjalne języki programowania przeznaczone do wsparcia tworzenia modelu[26] sekwencji. Z założenia służą one do opisu automatu sekwencyjnego, a nie umożliwiają dokonania jakiegokolwiek przetwarzania. Pozwalają natomiast na dobudowywanie kodu przetwarzającego do elementów składowych takiego modelu. Na rysunku 33 przedstawiono przykładowe zapisy w dwóch językach. Zobrazowany automat jest zgodny z grafem przedstawionym na rysunku 30.

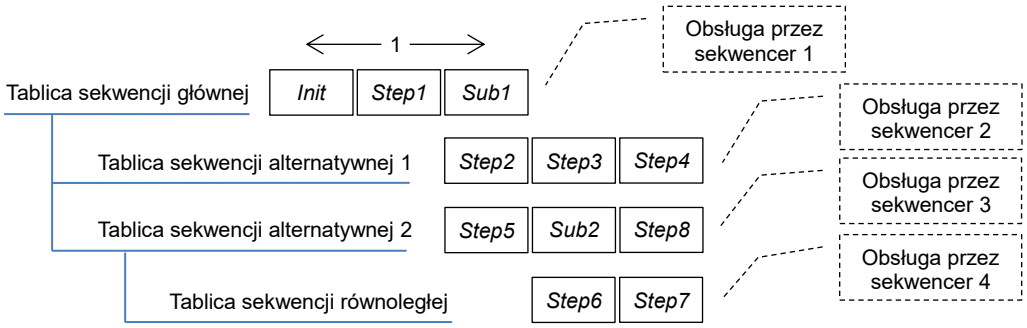
[26] Na przykład SFC, Grafcet, Graph, AS.



Rys. 33. Przykłady użycia języków opisu sekwencji
Fig. 33. Examples of using sequence charts

Opisania automatu sekwencyjnego można również dokonać bez użycia wsparcia dedykowanych języków. Jednym ze sposobów jest wykorzystanie wektorów opisujących stan automatu i obsługujących je tzw. sekwencerów. Są to elementy programowe umożliwiające opisanie maszyny w postaci jedno- lub wielowymiarowych tablic bitowych. Ich zawartość reprezentuje stan maszyny, a sterowanie pracą sekwencerów odpowiada za przejścia międzystanowe. Wartości przechowywane w tablicach mogą stanowić sekwencję bitów lub kody (liczby) identyfikujące konkretne stany maszyny. Najczęściej dany stan koduje się z użyciem położenia jednego aktywnego bitu w danej tablicy, a w skali całego opisu przez wektor indeksów takich położeń dla wszystkich tablic i ich wymiarów. Aktywny krok to ten, którego powiązany w tablicy element zawiera logiczną jedynkę lub inny określony kod. Przedstawiono to na rysunku 34. Przykładowa sekwencja z rysunku 33 jest obsługiwana przez cztery sekwencery. Sterowanie nimi dokonywane jest przez wyliczanie warunków tranzycji. Akcje w danym kroku wykonywane są w funkcji jego aktywności.

Świadoma rezygnacja z użycia niezależnego kodu opisującego automat sekwencyjny jest poważnym błędem projektowym. Oddzielony od przetwarzania opis stanowi szkielet programu, umożliwiając hermetyzację i agregację pozostałego kodu. Zarówno testowanie, jak i modyfikacja tak skonstruowanego oprogramowania będą dużo prostsze niż kodu zintegrowanego, bazującego na zwielokrotnionych konstrukcjach warunkowych.



Rys. 34. Idea stosowania sekwencerów do opisu automatu
Fig. 34. Idea of using sequencers to describe automata

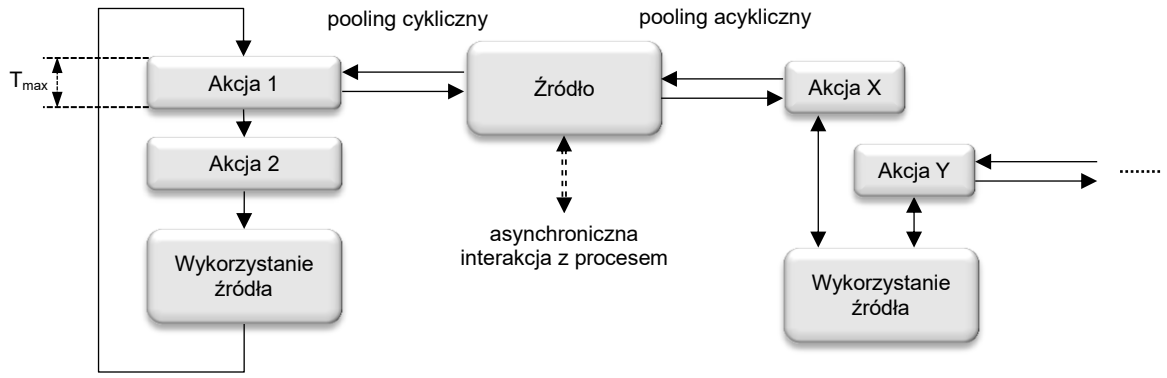
Powyższe wskazówki dotyczą oprogramowywania automatów sekwencyjnych, czyli grafów określających chronologie przejść, bez relacji zwrotnych. Do oprogramowania klasycznej maszyny stanów można z powodzeniem wykorzystać powyższą metodę opisu stanów z użyciem sekwencerów, języki uniwersalne lub dedykowane języki modelowania[27]. Jednak użycie wspomnianych wyżej języków opisu automatów sekwencyjnych do opisu maszyn stanów nie jest wskazane z racji braku możliwości tworzenia jawnie definiowanych relacji zwrotnych między stanami. Istnieje w nich możliwość opisu relacji zwrotnych przez użycie skoków, ale taki zapis traci na przejrzystości i jest podatny na pomyłki.

❖ Odpytywanie zasobów

W kwestii interakcji elementów systemu z otoczeniem w ISP głównie stosowany jest tzw. pooling, czyli odpytywanie o stan źródeł informacji. Źródła można ogólnie rozpatrywać jako pojedyncze rejestry lub ich tablice (bufory – grupy rejestrów) przechowujące wartości chwilowe lub serie danych za określony czas. Fizycznie, źródła są urzeczywistnione w zasobach sprzętowo-programowych systemu. Są niezależne od działań komputerów pozyskujących i przetwarzających dane, stanowią elementy będące w gotowości do dostarczenia żądanych danych, oraz stanowią elementy współdzielone między inne elementy systemu. Dzięki temu pozyskanie danych (akwizycja) jest możliwe z różnych elementów systemu, w momentach czasu kiedy są potrzebne.

Odpytywanie jest alternatywną metodą względem obsługi źródeł informacji na przerwaniach. Przy dużej liczbie niezależnych i asynchronicznych źródeł lub gdy źródła są związane z procesami szybkozmiennymi praca z przerwaniami może powodować zjawisko migotania przerwań. Może to skutkować nieracjonalnym zużyciem mocy obliczeniowej na ich obsługę. Ponadto odpytywanie rozwiązuje problem współdzielenia zasobów.

[27] Na przykład HiGraph, UML, ArchiMate, STD.



Rys. 35. Odpytywanie zasobów
Fig. 35. Resource pooling

Na rysunku 35 przedstawiono przykładowy schemat odpytywania źródła. W poolingu prowadzi się interakcję z otoczeniem w pewnych kontrolowanych względem czasu oknach czasowych. Są one tworzone przez zdefiniowanie porządku wystąpień, akcji w ramach aktywności oraz maksymalnych czasów ich otwarcia (T_{max}). W efekcie akcje wykonywane w ramach aktywności w danym oknie czasowym przygotowują lokalne obrazy danych mogące być wykorzystane przez dany element. Są to lokalne bufory przechowujące informacje o określonym czasie ważności. Przykładem może być cykl PLC i komunikacja CPU z układami IO, programatorem czy modułami „inteligentnymi", lub cykl sieci przemysłowej i związane z nim okna transakcyjne.

❖ **Wzajemne wykluczanie**

W praktyce, w węzłach ISP działają współpracujące procesy. Zachodzi zatem współbieżność wykonywanych zadań. Jest to wielozadaniowa praca kooperacyjna. Realizacja takich zadań w ISP jest obarczona pewnymi ograniczeniami:

- działania muszą podlegać ograniczeniom czasowym a zatem musi istnieć skuteczny mechanizm kontroli czasu ich wykonania;
- dane z otoczenia (procesu, rozproszonych alokacji, innych aplikacji) są pozyskiwane i wyprowadzane na drodze poolingu zasobów stanowiących ich źródła i odbiorców;
- dostęp do niektórych zasobów jednostki przetwarzającej (pamięć, interfejsy) jest współdzielony z racji jednoczesnego współdziałania wielu elementów operujących na tych samych danych;
- niektóre zadania są wykonywane współbieżnie (np. zadania podstawowe, obsługa zdarzeń, przerwań, błędów, diagnostyki, sieci komputerowych, koprocesorów itp.).

Niezbędny jest mechanizm, który byłby w stanie kontrolować wykonywanie zadań w czasie rzeczywistym oraz obsługiwać wzajemne wykluczanie ich dostępu do zasobów

stwarzających sekcje krytyczne. Potrzebny jest zatem mechanizm szeregujący. Mechanizm taki w węzłach ISP musi zapewniać gwarantowany czas reakcji na zdarzenie, punktualność i przewidywalność.

Rozwiązaniem jest implementacja powtarzalnych sekwencji działań obsługujących rozważane zadania. Działania w sekwencji są wykonywane w oknach czasowych kontrolowanych przez system operacyjny jednostki. Użycie zapętlonych sekwencji okien gwarantuje wzajemne wykluczanie dostępu, a kontrola ich rozmiaru zapewnia pracę w czasie rzeczywistym. Można tu doszukać się analogii do pracy wielowątkowej i mechanizmu, który zamiast obsługiwać sekcje krytyczne (ang. mutex) wykorzystuje nieblokującą synchronizację zadań w postaci kolejki FIFO. Zapętlenie takiej kolejki tworzy cykliczną sekwencję zadań nieblokujących (ang. lock-free). Każde okno jest kontrolowane pod kątem maksymalnego czasu otwarcia i ewentualnej synchronizacji bieżącego otwarcia do tego czasu (por. 1.1.4). Dokonuje tego specjalne zadanie systemu operacyjnego, a przekroczenie czasu jest traktowane jako błąd. Przykładem algorytmu może być CES (ang. Cyclic Executive Scheduling).

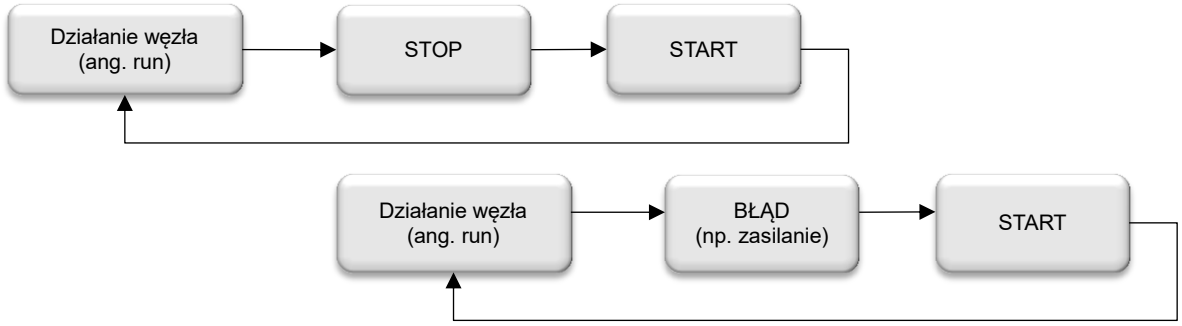
W węzłach ISP spotyka się również zadania, które nie mogą być szeregowane w cyklicznej kolejce. Są to zadania wykonujące obsługę zdarzeń wyzwalanych jakimś zdefiniowanym wyzwalaczem (ang. trigger, np. warunek, przerwanie, błąd, zdarzenie systemowe). Do obsługi przetwarzania zdarzeniowego (ang. Event-Driven programming) stosuje się kontrolę czasu wykonania obsługi zdarzenia oraz typowe mechanizmy obsługi sekcji krytycznych (np. semafory). Zadania wyzwalane zdarzeniowo mogą być szeregowane bez wywłaszczenia lub z wywłaszczeniem. Szeregowanie bez wywłaszczenia zwykle bazuje na kolejce zadań typu FIFO z ewentualnymi priorytetami zaburzającymi ideę FIFO dla zadań specjalnych, lub na wielopoziomowych kolejkach FIFO związanych z priorytetami zadań. Stosowane algorytmy szeregujące są typowe np. algorytm karuzelowy (ang. round-robin) stosowany dla obsługi zadań danego priorytetu lub RMS (ang. rate-monotonic scheduling), EDF (ang. Earlier Deadline First) [41], [9], [185], [72], [178] dla statycznej lub dynamicznej obsługi wielu priorytetów.

❖ **Stan i jego przechowywanie**

ISP i jego elementy tworzą stan opisu informacyjnego dla danej chwili czasu. Jest to swoista instancja systemu. Stan ten jest przechowywany w pamięci węzłów i na bieżąco aktualizowany przez wykonywane zadania. Niezbędne jest, aby w przypadku zatrzymania przetwarzania dla danego elementu lub jego wyłączenia stan był zachowany dla tych informacji, które tego wymagają. Typowe sytuacje wymagające zachowania stanu przedstawiono na rysunku 36.

Zatem niektóre obszary pamięci wymagają zapewnienia nieulotności danych (ang. data persistent) oraz ochrony przed domyślną reinicjalizacją. Wykorzystuje się do tego mecha-

nizm zachowania stanu (ang. retentiveness). Najczęściej działa on na bazie potrzymania bateryjnego lub zrzucania obrazu do pamięci nieulotnej (np. flash) oraz zadań systemowych, które obsługują jego funkcjonowanie.



Rys. 36. Sytuacje wymagające zachowania stanu
Fig. 36. Circumstances requiring state persistence

Do typowych stref pamięci wymagających zachowania w sytuacji restartu węzła należą:

- stan systemu operacyjnego, np.:
  - strefy systemowe,
  - obszary zarezerwowane dla działania systemu operacyjnego,
- program sterowania (ang. program logic),
  - program aplikacyjny węzła,
- informacje diagnostyczne, np.:
  - sumy kontrolne (programu, danych),
  - logi (tablice błędów, ang. fault tables),
- znaczniki (np. markery, bity tranzycji, wymuszeń),
- wektory opisu stanów wejść i wyjść (cały obraz lub konfigurowalne elementy),
- adresowalne obszary robocze pamięci (danych – cały obraz lub konfigurowalne elementy),
- obszary instancji symboli i obiektów (np. liczników, ang. counter, układów czasowych, ang. timer itp.).

Wiele urządzeń oferuje możliwość definiowania zachowywanych elementów. Można zatem przyjąć, że dla węzłów programowalnych mechanizm zachowania stanu podlega konfiguracji.

Dla zaniku zasilania (ang. power failure) problem jest podobny. Typowe obszary zachowywane są takie same jak powyżej. Spotykane wyjątki to referencje błędów typu run-time i niektóre referencje do stref systemowych, które wymagają inicjowania przy uruchamianiu węzła.

❖ **Podsumowanie**

Biorąc pod uwagę różne wzorce implementacji stosowane w ISP, należy zwrócić uwagę, że są one mocno związane z wymaganymi funkcjonalnościami i odnoszą się do architektury warstwowej systemu. Nie istnieją inne sensowne wzorce, które mogłyby wykraczać poza kodowanie specyficznych, separowanych funkcjonalności warstw. Oczywiście, im bardziej funkcjonalności te są oderwane od ograniczeń czasowych lub wręcz ze swoim działaniem wykraczają poza warstwy ISP uzależnione czasowo, tym bardziej poszerzają się możliwości stosowania różnych innych wzorców implementacji. Jednak podobnie jak dla wzorców architektury, programista powinien być kreatywny i dobierać wzorce do problemów, nawet jeśli spowoduje tym odejście tworzonego rozwiązania od szablonu. Warunkiem podstawowym jest, aby efekt był lepszy, bez utraty rozważanych dla danego przypadku aspektów produkcji oprogramowania dla danej implementacji wynikających z inżynierii systemów [166], [383].

2.1.3. Wzorce dynamiczne

Dużą nadzieję dla projektantów systemów stanowią wszelkie wzorce umożliwiające projektowanie i reorganizację systemu w sposób dynamiczny. Model obiektowy jako mocno ogólny i uniwersalny nie jest zbyt dobrze pasującym do architektur ISP. Przykładowym dedykowanym podejściem do konstrukcji systemów dynamicznych są systemy **EPS** (ang. Evolvable Production Systems) [302], [6], [269], stanowiące szersze potraktowanie idei EAS (ang. Evolvable Assembly Systems) [248], [43]. Podejście EPS stanowi stosunkowo nowy paradygmat projektowania, działania, utrzymania i rozwoju systemów przemysłowych. EPS umożliwia automatyczne określenie wytycznych i rozwiązań wspierających budowę, działanie, konserwację i rozwój kompletnych infrastruktur systemów przemysłowych. Umożliwia również automatyczną diagnostykę i reakcję systemu na zakłócenia produkcji [301]. Główne wytyczne EPS obejmują sterowanie rozproszone, budowę modułową, otwartą architekturę oraz kompleksowe i automatyczne wsparcie metodologiczne, które bazuje na architekturze referencyjnej i modelu wiedzy. Istnieją dwie podstawowe zasady przewodnie określone w [329].

- Przy automatycznym ewoluowaniu systemu najbardziej innowacyjny efekt można osiągnąć w przypadku braku ograniczeń doboru rozwiązań. Optymalne rozwiązania można osiągnąć stosując w pełni niezależną procedurę doboru w ramach dostępnej ontologii.
- Dynamiczność systemów nie może być tylko z nazwy, lecz system musi być „ewoluowalny”, tzn. musi mieć zdolność do ewoluowania, czyli możliwości podjęcia nowych lub zmiany istniejących zestawów wymagań.

Ontologia, czyli baza wiedzy dziedzinowej związanej z systemem, jest tutaj kluczowa. Musi ona dotyczyć wiedzy o produkcie, systemie, środowisku, przedsiębiorstwie, powiązaniach organizacyjnych, o zarządzaniu i o elementach umożliwiających uczenie.

2.1.4. Wzorce wytwarzania, testowania i certyfikacji

Mało które aplikacje systemów informatycznych wymagają tak wysokiego stopnia niezawodności i bezpieczeństwa jak ISP (zob. 4). Dotyczy to zarówno warstwy sprzętowej, jak i programowej. W niniejszej książce nie skupiono się jednak na aspektach projektów sprzętowych. W tym zakresie warto zapoznać się z inną literaturą [373], [414], [279]. Natomiast przy tworzeniu informatycznych, profesjonalnych rozwiązań softwarowych dla przemysłu, w celu zapewnienia niezawodności już od strony projektu, wysoce wskazane jest stosowanie wzorców dotyczących zarówno wytwarzania, testowania, jak i certyfikacji oprogramowania. Dla ISP nie istnieją jednak dedykowane wzorce ani rodzaje metodyk wytwarzania czy testów. Należy podkreślić, że mowa tu o wytwarzaniu systemu, a nie programu dla węzła [115], [97]. W praktyce, zwykle wykorzystuje się większość ze znanych typów testów i przeprowadza się je według wybranych wzorców ze zbioru ogólnie znanych i popularnych wzorców formalnych. Istnieją również normy, które dają wytyczne do produkcji oprogramowania dla systemów klasy ISP i jego certyfikacji, co opisane zostało w dalszej części podrozdziału.

Tak jak w innych przypadkach, w testowaniu ISP testy jedynie dowodzą istnienia błędów, a nie ich braku, i co za tym idzie testy wyczerpujące oprogramowania całego systemu jak i jego elementów są niemożliwe. Czasami aplikacje węzłów są relatywnie proste i mogą dawać błędne złudzenie, że przygotowane i wykonane testy wyczerpują wszystkie przypadki testowe działania węzła. Niestety, testowanie ISP jest niezwykle kontekstowe, a środowisko pracy aplikacji wprowadza często przypadki, których projektant nie przewiduje w warunkach laboratoryjnych. Twórcy często skupiają się na testach funkcjonalnych, pomijając lub marginalizując testy niefunkcjonalne. Praktyka pokazuje, że urządzenia i oprogramowanie doskonale działające na stanowisku testowym (projektowym, prototypowym, ang. test-bed), przechodzące wszystkie procedury i przypadki testowe zarówno manualne, jak i automatyczne, nie spełniają wymagań na rzeczywistym obiekcie. Dlatego praca testera ISP nie jest prosta. Wymaga doświadczenia, znajomości realiów funkcjonowania systemu dla danej klasy aplikacji lub dla konkretnego przypadku oraz technicznej wyobraźni wykraczającej daleko poza kod programów.

Specyfika zastosowań ISP powoduje, że istnieje pewna grupa akcji w systemie wykonywana niezwykle często względem innych. Wynika to z faktu, że podczas normalnej pracy wymagane jest użycie tylko niewielkiego zakresu funkcjonalnego ISP, zwykle zadań cy-

klicznych (zob. 1.1.4). Reszta stanowi obsługę zdarzeń acyklicznych, błędów, interfejsów, diagnostyki itp. Powoduje to, że mogą występować pewne obszary działania testowane intensywniej, w których finalnie znajduje się niewielka liczba defektów. Pozostałe obszary, z racji incydentalności użycia i wynikającego z niej braku dobrze przemyślanych testów, nie są równie dobrze przebadane. W efekcie niektóre błędy mogą objawiać się dopiero po znaczącym czasie użytkowania systemu. Zjawisko jest niekorzystne z punktu widzenia użytkownika, a twórcy może przynieść kłopoty wizerunkowe, jak również i prawne.

Oprogramowanie ISP nie wykazuje typowej cechy uodparniania na testy, jaka występuje w systemach współdziałających z człowiekiem. Lista przypadków testowych, o ile jest dobrze przemyślana, jest skończona i nie wymaga modyfikacji, oczywiście pod warunkiem niezmienności aplikacji i środowiska pracy. Wynika to ze statyczności środowiska (por. 1.1.3) i determinizmu działania węzłów. Poprawnie działający węzeł ISP, o ile nie ulegnie awarii, zachowuje się zawsze w taki sam sposób, zgodnie ze specyfikacją, czego nie można powiedzieć o człowieku. Dlatego przypadki testowe związane ze współdziałaniem węzłów z człowiekiem powinny być stale ulepszane i zmieniane, aby uniknąć miejscowego uodpornienia na testy.

Nigdy nie należy zakładać, że człowiek działa deterministycznie, a zatem akcje zależne od interfejsu z człowiekiem również. Podobnie nie należy zakładać, że urządzenie jest bezawaryjne. Jeśli w ISP może coś zawieść, to zapewne w czasie eksploatacji tak się stanie.

❖ Rodzaje testów

Testowanie produktów informatycznych rozwinęło się w ostatnich latach od nieokreślonych, często chaotycznych procedur prowadzonych przez deweloperów, do sformalizowanych i metodycznych działań prowadzonych przez wyspecjalizowane i nierzadko certyfikowane grupy testerów[28]. Zwykle przyjmuje się, że do podstawowych rodzajów testów należą: testy jednostkowe, funkcjonalne, integracyjne, systemowe, akceptacyjne. Jednak to, jakie testy, kiedy i w jaki sposób są używane, zależy od rodzaju testowanego produktu i kontekstu jego użycia. Specyfika ISP powoduje, że klasyczne podejścia nie zawsze się sprawdzają. Dla przykładu, testy jednostkowe powinny oprócz klasycznych wariantów uwzględniać weryfikacje działania względem czasu, a konkretnie ograniczeń czasowych narzuconych na testowaną jednostkę kodu.

Pierwszym rodzajem testów, jakie są wykonywane podczas tworzenia kodu, są testy deweloperskie. Są one ważne, ale oczywiste i mało interesujące z punktu widzenia specyfiki

[28] Na przykład certyfikacje ISTQB (ang. International Software Testing Qualifications Board) [W29].

ISP. Wiążą się z zapewnieniem poprawności składni i semantyki kodu oraz jego zgodności z przyjętym standardem i regułami. Zwykle związane jest to z wewnętrznymi wymaganiami wytwórcy systemu[29], a nie ze specyfikacją ISP.

Przy metodycznym prowadzeniu projektu, czyli przynajmniej zgodnie ze wzorcem z rysunku 1, dokonuje się testów strukturalnych oraz testów funkcjonalnych, w obu przypadkach bazujących na specyfikacji. Są to podstawowe testy sprawdzające poprawność realizacji koncepcji oraz poprawność działania i współdziałania wymaganego zbioru funkcjonalności systemu, reprezentacji danych (opisu informacyjnego) i jego elementów składowych.

Co do testów niefunkcjonalnych oprogramowania (operacyjnych), to warto poddać pod dyskusję zakres testowania w zależności od rodzaju testu. Dla tworzenia oprogramowania, stanowiącego finalnie procesy[30] uruchamiane w środowisku systemu operacyjnego węzła (por. 3.2), wskazane jest wykonywać pełne testy niefunkcjonalne. Programy tego typu są tworzone przez programistów z użyciem języków programowania i narzędzi. Zatem w postaci finalnej stanowią produkty, które pomijając użyte biblioteki i frameworki są tworzone od podstaw i w związku z tym wymagają sprawdzenia nie tylko poprawności działania względem wymagań funkcjonalnych związanych ze specyfikacją, ale i kontroli jakości względem aspektów pozafunkcjonalnych z nią niezwiązanych [114]. Zakres testów powinien obejmować ocenę działania programów od strony takich cech jakości, jak np. niezawodność (ang. reliability), praca pod obciążeniem (stresowanie), wydajność (ang. efficiency), responsywność (ang. responsively), używalność (ang. usability) itp. Więcej na tematy cech jakości znajduje się w dalszej części podrozdziału.

Dla aplikacji uruchamianych w dedykowanych funkcjonalnie środowiskach uruchomieniowych (por. 3.2) wskazane jest wykonywać testy niefunkcjonalne dla parametrów związanych z charakterystyką czasową pracy (np. responsywność) i używalnością interfejsów. Teoretycznie, dla tego przypadku można założyć brak potrzeby testów stresujących, choć w praktyce zdarza się, że niektóre środowiska generują podczas pracy aplikacji problemy z gospodarką zasobami. Zatem sugeruje się, aby nie zawierzać producentom i aplikacje osadzane w środowiskach uruchomieniowych również poddawać pełnym testom niefunkcjonalnym. Narzędzia i środowiska uruchomieniowe nie są wolne od błędów. Warto też zauważyć, że część testów niefunkcjonalnych można poprzedzić analizą teoretyczną, która może być wykonana przed kodowaniem. Dla przykładu charakterystyka czasowa danej aktywności systemu może być wyznaczona teoretycznie na bazie analizy najgorszego przypadku lub analiz statystycznych. Analiza responsywności czy wydajności jest możliwa na podstawie para-

[29] Wymagania bazują na formalnym dokumencie wewnętrznym przedsiębiorstwa lub oficjalnych dokumentach standardów kodowania.
[30] W tym wątki, zadania, moduły, POU itp. jednostki wykonywalne kodu zależnie od węzła i jego systemu operacyjnego.

metrów i kalkulacji obciążeń. Używalność można wstępnie przebadać na etapie projektowania interfejsu przez konsultacje z użytkownikiem itp.

Nie ufaj, że produkt, z którym pracujesz, działa bezbłędnie, czyli nie zawiera defektów konstrukcyjnych i implementacyjnych. Dotyczy to zarówno urządzeń, jak i oprogramowania. Testuj całość rozwiązania, również testami niefunkcjonalnymi.

Wszystkie poprawki wynikające z wykrytych nieprawidłowości powinny być potwierdzane testami konfirmacyjnymi (tzw. testy potwierdzające, retesty). Natomiast na każdym etapie testowania warto wykonywać regularne testy regresyjne (regresywne, ang. regression testing). Jeśli ze względu na bieżące ograniczenia środowiska testowego bądź problemów interakcji z obiektem jest to niemożliwe, to przynajmniej jeden całościowy test regresyjny powinien być wykonany jako finalny, przed przekazaniem systemu do eksploatacji. Niestety, w praktyce wszelkie modyfikacje wykonywane poza laboratorium i poza środowiskiem testowym sprzyjają testowaniu bez regresji. Jest to bardzo niebezpieczne, gdyż dokonywane wówczas zmiany mogą interferować z innymi fragmentami systemu zarówno w zakresie funkcjonalnym, jak i niefunkcjonalnym, powodując nieprawidłowości działania, które w dokumentacji pozostaną jako potwierdzone testami z wynikiem pozytywnym. W skrajnym przypadku takie zmiany powinny być poddane testom utrzymaniowym (ang. sustainability testing) [42], dokonywanym na obiekcie przez twórcę we współpracy z użytkownikiem. Są to jednak testy, które odbywają się w interakcji z rzeczywistym obiektem i w rzeczywistym czasie. Nie zawsze zatem mogą być wykonane w dowolnym momencie, i nie zawsze wykryją defekty, zanim wystąpią ich negatywne wpływy na obiekt. Nie oznacza to jednak, że testy utrzymaniowe są złe i należy ich unikać. Wszystko zależy od kontekstu. Jeśli system jest tworzony jako rozwiązanie statyczne, niepodlegające ciągłym zmianom i modyfikacjom, to wskazane jest, aby cały proces testowania został przeprowadzony poza obiektem, w ramach możliwości technicznych. Testy wdrożeniowe i utrzymaniowe obsłużą wówczas przypadki nierealizowalne w laboratorium, zmiany bieżące, spóźniony regres i defekty ukryte. Natomiast jeśli system z założenia jest dynamiczny i w czasie swojej eksploatacji musi być dostosowywany do nowych wymagań (ang. flexible), wówczas celowe jest zapewnienie stałych testów utrzymaniowych produktu.

Jeśli tylko możesz, testuj w laboratorium, a nie na obiekcie.

Jeśli nie musisz, nie modyfikuj kodu na obiekcie.

W teorii testowania występują jeszcze pojęcia testów tzw. białej i czarnej skrzynki. Oba rodzaje nadają się dla tworzenia ISP. Testy białoskrzynkowe lepiej sprawdzają się podczas

testowania funkcjonalności aplikacji fizycznych węzłów, natomiast testy czarnej skrzynki podczas testów ich współdziałania i integracji, czyli podczas testowania funkcjonalności systemowych i aplikacji wirtualnej (por. 2.4). Na takim etapie nie jest zwykle potrzebna znajomość reprezentacji danych, implementacji programów węzłów, usług itp., wykraczających poza definicje ich interfejsów i współdziałania. Wszelkie idee komponentowe oraz podejście IEC61499 (zob. 2.5) wspierają takie właśnie podejście do testów.

Do testowania ISP można podejść nie tylko od strony aspektów funkcjonalnych, ale również od strony architektury. Model warstwowy (zob. 2.1.1) wprowadza abstrakcje, na bazie których można przygotować plany testów i prowadzić testowanie. Jest to dobre podejście dla ISP, gdyż przeważnie warstwy są na tyle separowane, że naturalna staje się ich niezależna i zrównoleglona implementacja, a co za tym idzie i testowanie.

W ISP koszty błędów na poszczególnych etapach projektowania zdecydowanie rosną. Dla przykładu: źle dobrany rozmiar pamięci lub przestrzeni IO w specyfikacji doboru sprzętu może skutkować niemożnością zrealizowania projektu na tak dobranym sprzęcie. Dlatego testowanie należy rozpoczynać na jak najwcześniejszym etapie projektowania, nie czekając na fazę implementacji kodu.

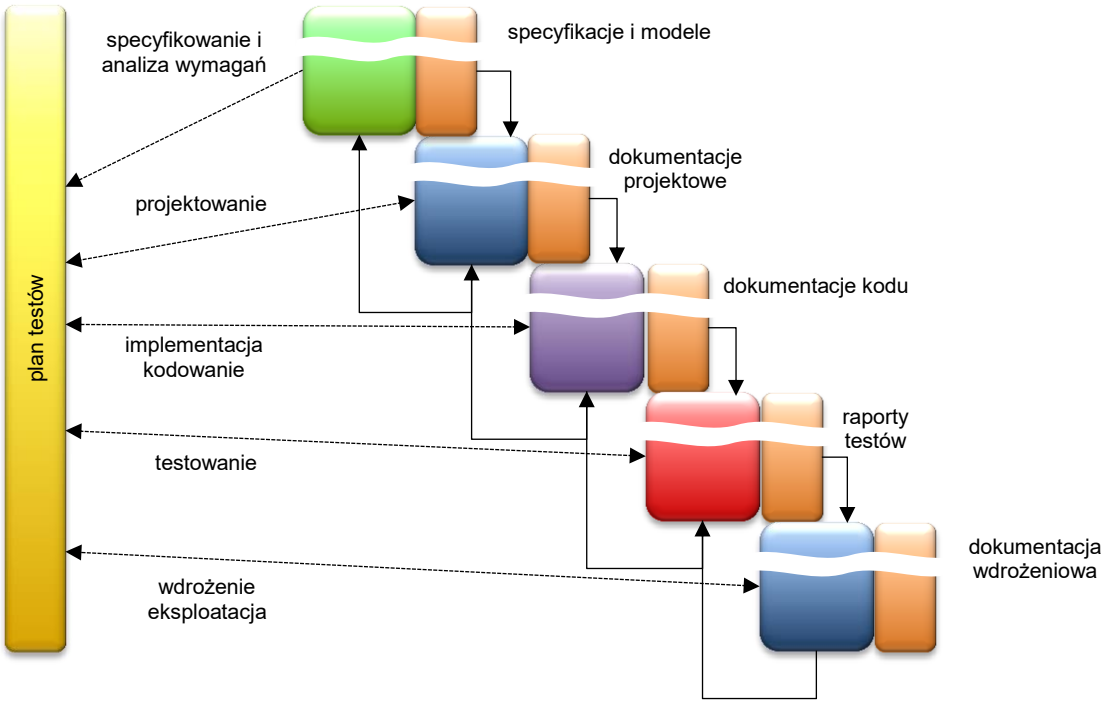
❖ **Modele uwzględniające testy**

Najczęściej przy testowaniu rozwiązań na etapie ich tworzenia stosuje się popularny model V lub model kaskadowy z lub bez planu testów. Na rysunku 1 znajduje się zobrazowanie ogólnej kolejności działań podczas tworzeniu ISP. Przedstawiono, że na podstawie dokumentacji projektowej dokonuje się realizacji projektu, a następnie testowania tejże realizacji. W praktyce sprawa wygląda bardziej skomplikowanie. Istnieją sprzężenia zwrotne pomiędzy aktualnie realizowanym etapem projektu a etapami wcześniejszymi. W ISP często zdarza się, że na etapie testowania znajdują się nie tylko defekty implementacji, ale również defekty projektu lub nawet specyfikacji. Ponadto, projekty ISP charakteryzują się dość znaczącą ewolucyjnością wymagań, po części wynikającą ze wspomnianych defektów, a po części z niskiej świadomości klientów, czego mogą i czego mogliby oczekiwać od systemów informatycznych. Dlatego właśnie, z punktu widzenia testowania, modele kaskadowe i typu V są dobrze przystające do procesu tworzenia ISP.

Model kaskadowy najlepiej się sprawdza w wersji zmodyfikowanej, z planem testów i iteracjami. Tworzenie ISP jest często procesem, gdzie żaden etap nie jest finalnie zamykany i podlega on ciągłym modyfikacjom. Wynika to ze wspominanej wielodziedzinowości rozwiązań (por. 1), w które zaangażowane są różne grupy zawodowe i różne dziedziny nauki oraz z konieczności modyfikacji wspomnianej wcześniej. W modelu takim założono istnienie

etapów z możliwością przejścia wprzód i wstecz, wraz ze zintegrowanymi planami testów tworzonymi na bieżąco względem danego etapu (rysunek 37).

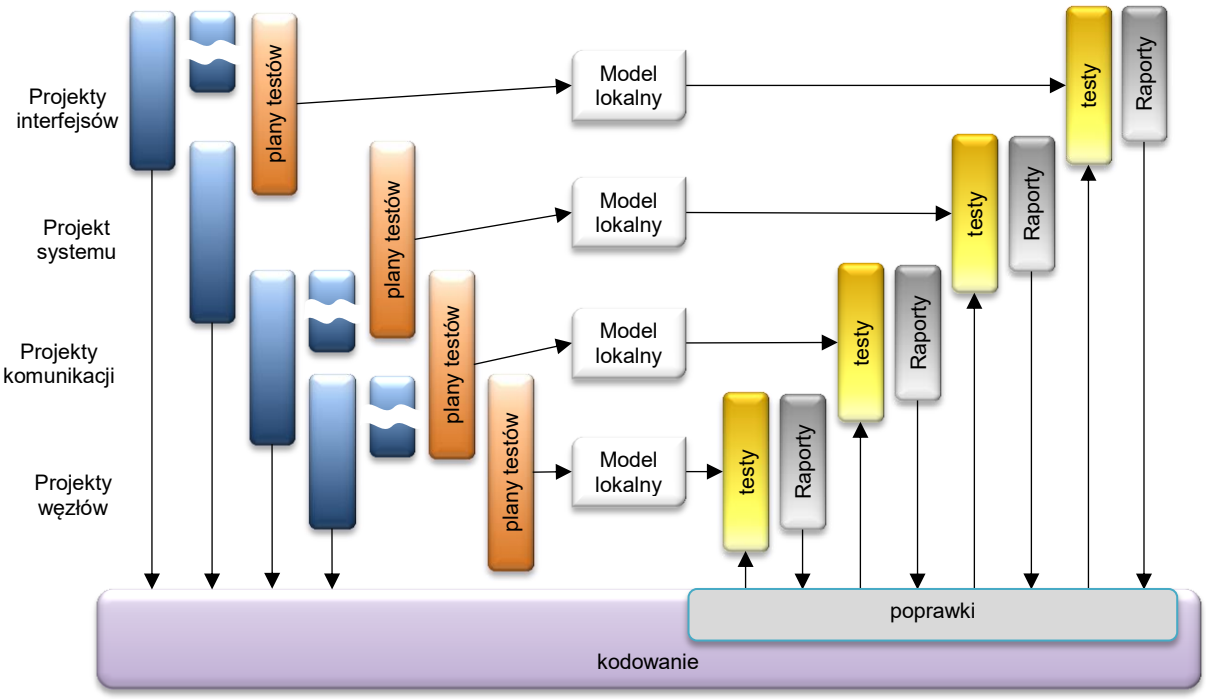
Jest to model relatywnie prosty i uniwersalny. Dla złożonych systemów daje dużą swobodę działań. Plan testów powinien być ustalany na każdym etapie realizacji, względem pojęć i abstrakcji, na których się operuje. Dla przykładu, specyfikacja stanowiąca wytyczne do algorytmizacji może wprowadzić ograniczenia parametrów, które wynikają z technologii, i które powinny być walidowane przez jakiś program, stanowiący element aplikacji węzła. Bezpieczniej jest to umieścić w planie testów na etapie tworzenia specyfikacji, niż przerzucić na etap projektu lub kodowania. Na etapie projektu może się okazać, że dany parametr podlega szerszym ograniczeniom, jak np. współzależności z innymi wymaganiami technologicznymi, ograniczenia czasowe, wieloetapowa i rozproszona walidacja wynikająca ze struktury danych itp. Na etapie kodowania, oprócz ograniczeń technologicznych i przepływu danych, wymagana kontrola może zostać rozbudowana np. o kontrolę reprezentacji danych. Dlatego zintegrowany plan testów stanowi element dokumentacji projektowej i może posłużyć do wszelkich działań projektowych, w tym weryfikacji poprawności etapów, a nie tylko do testowania oprogramowania.



Rys. 37. Modyfikowany kaskadowy model wytwarzania i testowania ISP
Fig. 37. Modified cascade model of ICS coding and testing

Warto zauważyć, że pewne etapy procesu tworzenia ISP mogą zostać rozpisane szczegółowiej, a względem procesu testowania mogą być rozpatrywane wielowymiarowo. Chodzi tu o dekompozycje przypadków testowych względem architektury ISP oraz podejścia do testo-

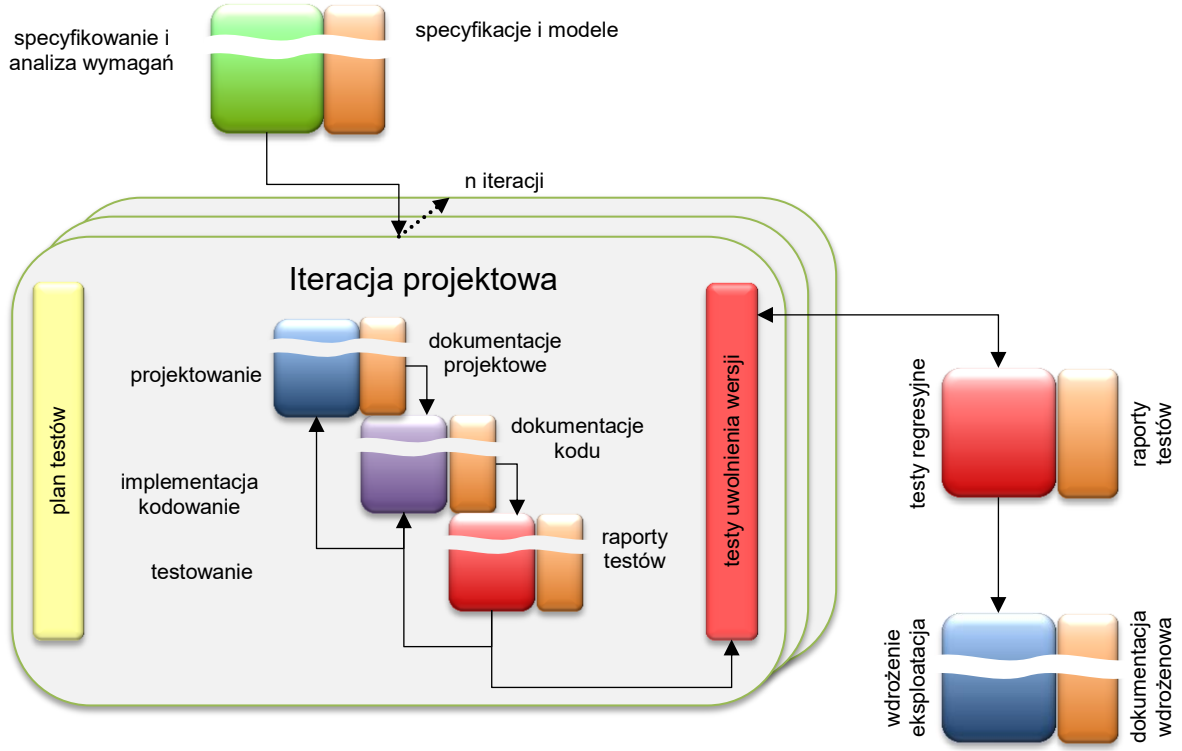
wania takich przypadków różnymi metodami. Zatem, oprócz modelu kaskadowego, można posłużyć się dowolnie innym modelem lokalnym, pasującym do danego podzadania. Dla przykładu, do testowania aplikacji węzła można posłużyć się modelem typu „V”, a do testów komunikacji modelem kaskadowym itd. Dla przedstawienia ogólnego podejścia do testów ISP jako całości można zaproponować zmodyfikowany model „W” (rysunek 38), gdzie modyfikacja polega na wprowadzeniu możliwości zrównoleglenia prac projektowych, implementacyjnych i testowych. Dobry będzie też klasyczny model spiralny [258], [244].



Rys. 38. Zmodyfikowany model „W” w przykładzie względem typowego ISP
Fig. 38. Modified ‘W’ model in relation to the typical ICS

Z punktu widzenia systemu, a nie jego elementów etap programowania powinien być dobrze przemyślany, aby optymalnie względem czasu wykorzystać zespół programistów i testerów. Podział zadań wynika z reguły z architektury warstwowej systemu, a ich zrównoleglenie z dostępności wykonawców. Przy odpowiednio dużych zasobach ludzkich zadania mogą zostać rozpoczęte w tym samym czasie na bazie dokumentacji projektowej, stanowiącej efekt etapu wcześniejszego. Niestety, testowanie może być zrównoleglone tylko częściowo. Testy deweloperskie oraz testy jednostkowe elementów systemu mogą być prowadzone równolegle. Testy całościowe aplikacji systemowej oraz wszelkich elementów, gdzie występuje współdziałanie warstw, mogą być wykonywane tylko w określonej kolejności. Zwykle kolejność ta wynika z przepływu danych i dostępności elementów uczestniczących w tym przepływie oraz z gotowości planów testów.

Rozpatrując typową architekturę warstwową ISP, implementację należy rozpocząć od aplikacji węzłów, następnie zapewnić im komunikację, zintegrować działanie w ramach systemu i zakończyć, budując integrację z innymi systemami. Każdy z tych etapów podlega testom jednostkowym w ramach modelu lokalnego, natomiast uruchomienie całościowe oprogramowania systemu wspomnianym testom według modelu „W" lub innego umożliwiającego ogarnięcie wszystkich projektów i programów składowych.



Rys. 39. Podejście „agile" do wytwarzania i testowania ISP
Fig. 39. 'Agile' approach to ICS producing and testing

Podczas wyboru modelów testowych warto rozważyć, czy specyfikacje wymagań i projekty związane z danym systemem mogą podlegać modyfikacjom na etapie realizacji. Jeśli tak, to testowanie będzie wymagało modeli ewolucyjnych, przyrostowych lub iteracyjnych. Jedną z najlepszych metodyk wytwarzania i testowania oprogramowania w takim przypadku będzie programowanie zwinne (ang. agile programming) [166], [257]. Jest to bardzo dobra metoda wytwarzania dla ISP z punktu widzenia praktyki wytwarzania takich systemów. Obecnie istnieje częsta potrzeba specyfikowania i tworzenia systemów „w locie", czyli bez precyzyjnie sformułowanych wymagań, a wręcz opartych na specyfikacjach nieformalnych. Wytwarzanie typu „agile" umożliwia dynamiczne dopasowywanie rozwiązania do potrzeb obiektu i klienta. Jest to podejście, gdzie dzięki wielokrotnym iteracjom związanym z modyfikacjami projektu i implementacji doregulowuje się rozwiązanie do bieżących potrzeb. Dzięki temu możliwe jest również jego zastosowanie po etapie wdrożenia, do utrzy-

mywania systemów dynamicznych (por. 2.1.3). Ponadto, testowanie zwinne sprzyja zapobieganiu powstawaniu błędów, a nie ich wykrywaniu, co w przypadku ISP jest bardzo pożądaną cechą. Ogólną ideę podejścia tego typu przedstawiono na rysunku 39.

Zwinna realizacja projektów softwarowych dla ISP wymaga niewielkich zespołów z dobrym zarządzaniem (wyznaczony dedykowany lider, kierownik, manager) i z dobrą komunikacją między członkami zespołu. Zespół „agile" musi być kreatywny i zainteresowany tym, co robi. Model ten nie sprawdzi się dla programistów nastawionych na wykonanie tylko swojej dokładnie sprecyzowanej części.

Projektowanie i walidacje złożonych systemów i ich aplikacji można również wspomóc formalnym procesem modelowania na podstawie modeli dziedzinowych (MDE – ang. Model-Driven Engineering) oraz znanych technik i narzędzi, jak np. znormalizowanych języków programowania PLC i reprezentacji danych, formatów AutomationML [M97], [W13], formatów CAEX (ang. Computer Aided Engineering eXchange [324]), języków MathML, UML, AOSD (ang. Aspect-Oriented Software Development) itp. [97], [376].

❖ **Implementacja testów**

Generalnie w ISP nie ma ograniczeń względem sposobów i wzorców implementacji testów. Podstawowe pytanie, jakie się pojawia, to czy testy będą prowadzone w laboratorium czy na obiekcie. Testy w laboratorium mogą być implementowane:

- z użyciem urządzeń obiektowych – proces jest symulowany na rzeczywistych urządzeniach AKPiA, ale urządzenia nie mają połączenia z żadnym procesem;
  Jest to najbardziej komfortowa praca związana z testowaniem. Z racji dostępności fizycznych urządzeń testy są prowadzone przez rzeczywiste interfejsy źródeł i odbiorców danych procesowych. Tester może wykonywać dowolne operacje bez obaw o jakąkolwiek interakcje z procesem, skupiając się na weryfikacji rozwiązań względem założeń. Wykorzystywanie rzeczywistych urządzeń ma sens dla urządzeń tzw. 'inteligentnych', gdzie obsługiwane są dane reprezentowane przez zmienne o typach złożonych lub występuje jakikolwiek rodzaj dialogu z urządzeniem (np. protokół, handshake itp. z urządzeniem typu komputer, licznik, przekształtnik itp.).

  Można stwierdzić, że używanie sprzętu AKP do symulacji wartości zmiennych prostych jest przerostem formy nad treścią. Można rozważyć podłączanie np. termopary do wejścia sterownika i testować poprawność jej działania, ale nie opłaca się wykorzystywać jej jako zadajnika wartości testowych. Znacznie wygodniej i szybciej można tego dokonać z poziomu degugera zarówno w testach manualnych, jak i automatycznych.

W testowaniu współdziałania aplikacji istotna może być konkretna implementacja sprzętowa lub programowa urządzeń. Dla przykładu, stworzenie programu obsługującego protokół Modbus RTU dla danych funkcji i przetestowanie go z abonentem wzorcowym nie zawsze może oznaczać poprawną współpracę z abonentem obiektowym. Aplikacja firm trzecich może mieć błędy i niezgodności ze standardem lub z dokumentacją. Zatem dla tego przypadku ułatwione są również testy systemowe (niefunkcjonalne).

- bez urządzeń obiektowych – proces i urządzenia AKPiA są symulowane.
  Testowanie odbywa się tylko z użyciem testowego zestawu węzłów systemu. Może to być zestaw pełen lub częściowy. Dane obiektowe można symulować z poziomu używanego debugera lub narzędzia modelowania. Dobrym rozwiązaniem jest stworzenie symulacji obiektu przez inne niesystemowe urządzenia. Dla przykładu, można wykorzystać PLC i stworzyć aplikację testową symulującą zachowanie obiektu. Sprzyja to tworzeniu testów automatycznych, ale wymaga weryfikacji aplikacji testowej. Działanie współpracujących węzłów również może się odbywać na zasadzie testowania współdziałania z rzeczywistymi węzłami i ich aplikacjami lub z aplikacją testową symulującą takie węzły, a uruchomioną na urządzeniach pozasystemowych.

Rys. 40. Przykłady zadajników sygnałów
Fig. 40. Examples of signal adjusters

Wygodne jest też stosowanie zadajników sygnałów dyskretnych i/lub ciągłych. Służą do tego specjalne moduły sprzętowe instalowane w zamian rzeczywistych modułów wejść lub można do rzeczywistych modułów podłączyć zadajniki zewnętrzne. Są one jednak przygotowane tylko do testów manualnych. Przykłady takich zadajników pokazano na rysunku 40.

Dla testów automatycznych przydają się urządzenia komputerowe w postaci „inteligentnych” analizatorów i zadajników. Przykłady zostały przedstawione na rysunku 41. Z lewej znajduje się analizator IO i oscyloskop cyfrowy, natomiast z prawej analizator sieci Profibus.

Rys. 41. Przykłady urządzeń testujących
Fig. 41. Examples of test devices

Testy na obiekcie nie są zalecane, ale jeśli staną się niezbędne, to zachodzą:

- dla urządzeń obiektowych bez udziału w procesie – urządzenia i obiekt są rzeczywiste, ale proces jest niepodłączony (odłączone fizyczne powiązania z procesem, np. rozsprzęglony napęd, brak medium itp.);
  Z punktu widzenia bezpieczeństwa funkcjonalnego testowanie jest bezpieczne, o ile nie doprowadzi do zaburzeń lokalnych pracy urządzeń. Możliwe są testy manualne lub automatyczne, o ile pozwalają na to podłączone urządzenia.
- dla urządzeń z procesem w trybie manualnym – rzeczywiste urządzenia pracujące z rzeczywistym procesem, ale część lub wszystkie algorytmy pracy automatycznej są wyłączone.
  Testowany program może oddziaływać na obiekt. Może to prowadzić do zaburzeń lokalnych i globalnych działania urządzeń lub procesu. Możliwości testów są ograniczone do akcji dopuszczalnych dla danego stanu urządzeń i procesu. Zalecane są testy manualne pod warunkiem świadomości poczynań.
- dla urządzeń z procesem w trybie automatycznym – rzeczywiste urządzenia pracujące z rzeczywistym procesem.
  Program pracuje we współdziałaniu z innymi elementami w reżimie automatycznym. Nie są zalecane żadne testy wymuszające. Sugerowane są testy utrzymaniowe, polegające głównie na obserwacji działania elementów i weryfikacji tego działania względem modeli, specyfikacji i wytycznych projektowych.

Testowanie manualne stanowi podstawową implementację testów w ISP. Wymaga ona dobrej znajomości założeń i wymagań projektowych oraz przemyślanego planu testów. Jak już wspomniano, ISP nie służą do przetwarzania dużych danych. Dlatego testowanie manualne wymaganych funkcjonalności w większości przypadków jest możliwe. Problemem jawi się nie testowanie wymaganych funkcji, ale testowanie przypadków obsługi sytuacji niewłaściwych. Dotyczy to zarówno walidacji danych, jak i stanów nieprawidłowych

w sekwencjach działań i maszynach stanów. Kontrola reprezentacji i semantyki danych dotyczy zmiennych pochodzących od użytkownika, od procesu oraz od innych elementów systemu.

Testowanie automatyczne jest pożądane ze względu na dokładniejsze oddanie warunków rzeczywistych pracy układu oraz na zwiększenie liczby możliwych do wykonania testów. Wymaga jednak przygotowania elementów pozasystemowych generujących przypadki testowe i elementów sprawdzających poprawność efektu ich obsługi. Elementami tymi mogą być dodatkowe programy, skrypty lub urządzenia, dostarczające danych i sekwencji zdarzeń sprawdzających działanie komponentów ISP zarówno na konkretnie dobrane przypadki, jak i na przypadkowo generowane. Dobór danych testowych powinien wynikać z technologii procesu oraz z możliwych sytuacji awaryjnych procesu i elementów systemu.

Tworzenie testów automatycznych jest pracochłonne i wymaga dużej pracy organizacyjnej i twórczej do stworzenia środowiska symulacyjnego. Weryfikacja środowiska testowego stanowi problem sam w sobie. Zajmuje ona czas, generuje koszty i jest trudna do jednoznacznej akceptacji. Aby zredukować złożoność kontroli takiego środowiska, warto używać mechanizmów wsparcia dla testów czarnoskrzynkowych, dostarczanych przez producentów, np. definicji interfejsów, charakterystyk pracy, reprezentacji i zakresów danych, i innych, opisujących elementy układu w podejściu komponentowym. Nigdy jednak nie ma gwarancji, czy procedury automatyzacji testów aplikacji dostarczone przez firmy trzecie są bezbłędne oraz czy pasują do aktualnie rozpatrywanego przypadku zastosowań.

Testy automatyczne sprawdzają się zwykle lepiej dla większych systemów. Można zaproponować testy manualne dla aplikacji węzłów i testy automatyczne dla testów systemu. Celowe jest również przygotowanie dla każdej aplikacji węzła funkcji symulującej jej działanie na potrzeby testów automatycznych.

Dla wspomnianej wcześniej metodyki „agile” testy jednostkowe wykonywane są przez programistów w czasie tworzenia kodu. Dotyczą one poprawności kodu i testów funkcjonalnych. W podejściu zwinnym zadania programistów i testerów nie są ostro rozdzielone. Można zaproponować przekazanie testerom produktu w wersji „beta” dla wymagań uzgodnionych dla danej iteracji projektu i do testów czarnoskrzynkowych. Może to być aplikacja węzła, całego systemu, komunikacja, interfejs lub inne elementy, o ile stanowią zamknięte rozwiązanie zidentyfikowane w projekcie. Więcej o testowaniu w kontekście ISP można znaleźć np. w [372], [211], [190].

❖ Certyfikacja

Funkcje realizowane przez ISP są z reguły dość odpowiedzialne, gdyż od ich poprawnego działania zależy zarówno wymiar biznesowy produkcji, jak i bezpieczeństwo zdrowia i życia pracowników. Wskazane jest zatem, aby procedury związane z ich tworzeniem i testowaniem

były w jakiś sposób nadzorowane, a użytkownik, aby wiedział, że otrzymuje produkt informatyczny godny zaufania, nad powstawaniem którego ktoś czuwał w sposób metodyczny. W gąszczu istniejących modeli testowych i różnorodności aplikacji ISP trudno narzucić twórcom jeden słuszny model testowania i ograniczać swobodę twórczą, na przykład przez stosowanie jedynych słusznych protokołów, czy też eliminację skoków wstecz lub pracy z przerwaniami. Prowadzi to do absurdów, gdzie projektant-programista musi natrudzić się nad implementacją prostych zadań w uciążliwym środowisku. Można jednak dać wytyczne, według których powinien on postępować. Pojawiają się zatem wzorce ogólne, które porządkują działania, wymagają pewnego ich zbioru minimum oraz umożliwiają weryfikację ich istnienia i poprawności. Wzorce te sprzyjają osiąganiu wymaganego bezpieczeństwa funkcjonalnego systemów, są związane z ogólnym bezpieczeństwem (por. 4) i nie ograniczają kreatywności rozwiązań.

Ważne jest, aby produkt softwarowy był określonej jakości, którą można zmierzyć lub odnieść do wzorca. Certyfikacja jest niczym innym jak badaniem zgodności danego rozwiązania z przyjętym wzorcem rozwiązań [223]. Sformalizowane uzgodnienia w ramach ustalenia takiego wzorca (ustalenia normatywne) tworzą normę. Obecnie istnieje bardzo wiele norm dotyczących aspektów tworzenia i kontroli produktów technicznych. Ogólne definicje związane z jakością można znaleźć w normie ISO 9000 (wcześniej ISO 8402) [M59]. Dla ISP przede wszystkim narzucają się normy dedykowane dla przeznaczonych dla nich urządzeń komputerowych (IEC61131), normy dla sieci przemysłowych (np. IEC61158, IEC61784) oraz dla systemów rozproszonych (IEC61499), a także dla zagadnień bezpieczeństwa (IEC61508 [M37], IEC61511 [M38], IEC62061 [M48] i inne). Zagadnienia związane z tymi normami są poruszone w dalszej części książki. Są jeszcze normy, które dotyczą samego procesu wytwarzania oprogramowania i jego certyfikacji [M69], [88]. Mają one na celu zapewnienie, że podejście techniczne i organizacyjne jest przemyślane, i tym samym wolne od rozwiązań ryzykownych, działań chaotycznych i realizacji ad hoc.

W normie IEC61508 [M37] znajdują się zalecenia dokładnego specyfikowania cyklu życia bezpieczeństwa oprogramowania. Jest to zestaw czynności, które należy wykonać na każdym etapie życia oprogramowania w celu zaimplementowania elementów związanych z bezpieczeństwem i nadzorem tego bezpieczeństwa. Cykl taki systematyzuje podejście do procesu wytwarzania, testowania, utrzymania i wycofania produktu oraz definiuje wymagania względem genealogii danego procesu (tzw. śladu, ang. tracebility). Wszelkie czynności wykonuje się względem niezmiennych elementów zwanych elementami konfiguracji, które należy jednoznacznie zidentyfikować definiując cykl. Zatem, przy znormalizowanym procesie wytwarzania oprogramowania ISP należy uwzględnić:

- definicje bezpieczeństwa
  Podstawowa czynność to określenie kryteriów i wymagań bezpieczeństwa oraz poziomu jego nienaruszalności w cyklu życia produktu. Nienaruszalność bezpieczeństwa jest miarą prawdopodobieństwa, określającą wykonywalność rozpatrywanych funkcjonalności zgodnie z założeniami. Zarządzanie bezpieczeństwem przy wytwarzaniu oprogramowania nie zależy od wymagań narzuconych przez normę, lecz od wymagań zdefiniowanych dla konkretnego produktu w ramach wytycznych normy. Specyfikacja wymagań bezpieczeństwa powinna zawierać wszelkie wymagania dotyczące funkcji bezpieczeństwa dla danego przypadku. W ramach takiego określania wymagane są definicje związane z pracą, testowaniem, integracją, procedurami zmian oraz mechanizmami oceny bezpieczeństwa dla każdego etapu procesu.

- aspekty wytwarzania kodu (aplikacji, programów, zadań, algorytmów itp.)
  Tworzenie znormalizowanego oprogramowania oznacza wykonanie go zgodnie z zaleceniami normy. Natomiast certyfikowanie takiego oprogramowania oznacza sprawdzenie, czy zostało ono wykonane zgodnie z zaleceniami tej normy. Kontrola na poziomie ogólnym dotyczy porównania etapów tworzenia z zalecanym wzorcem.

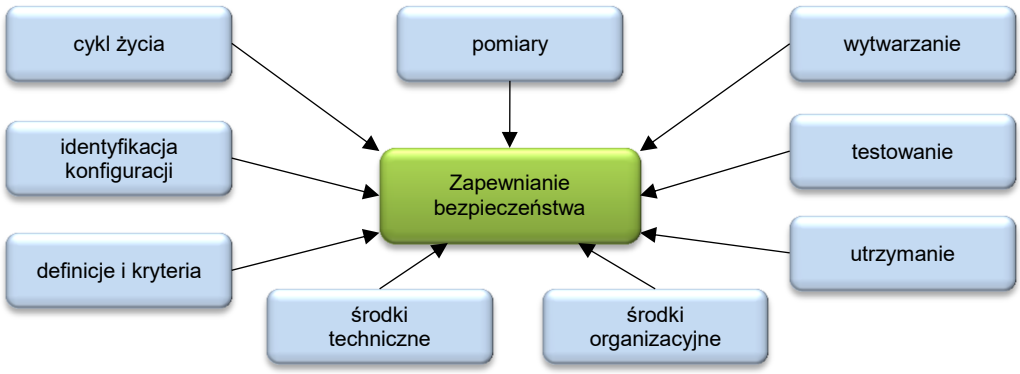
  Wymagany w danym przypadku ISP poziom nienaruszalności bezpieczeństwa wymusza dobór zapewniających go środków deweloperskich w postaci języków, narzędzi (kompilatory, interpretery, maszyny wirtualne itp.), konfiguratorów, symulatorów, środowisk testowych itp. Cechy, którymi muszą odznaczać się narzędzia, zależą od wymagań i mogą dotyczyć możliwości pracy bez zatrzymywania obsługiwanego układu, nieingerowania w charakterystykę czasową jego pracy, rodzaju i dziedzinowości języków, istnienia określonego wsparcia, użycia konkretnych stylów programowania itp.

- aspekty testowania
  Na etapie definiowania procesu testowania i jego późniejszego użycia wymagane jest określenie modelów testowania, głębi dekompozycji na czynności elementarne, zasad administracyjnych przepływu i nadzoru nad informacją, rodzajów i warunków technicznych testów, planów walidacji i weryfikacji itp.

- aspekty utrzymania
  W przypadku ISP cykl życia nie obejmuje tylko etapów kreacji systemu, ale również czas jego eksploatacji (utrzymanie, modyfikacje) i wycofania produktu z użycia. Praca systemu oraz jego modyfikacje muszą podlegać pomiarom i ocenie, a wszelkie działania być zgodne z przyjętą specyfikacją. Wytwórca produktu musi utworzyć i udostępnić procedury śledzenia i rozwiązywania niezgodności. Zarówno proces utrzymania, w tym

szczególnie modyfikacji, jak i wycofania systemu z eksploatacji lub zastąpienia przez inny może stanowić znaczące wyzwanie, porównywalne do procesu wdrożenia.

Na rysunku 42 przedstawiono zestawienie najważniejszych elementów, mających wpływ na zapewnienie bezpieczeństwa certyfikowanego oprogramowania.



Rys. 42. Elementy wpływające na zapewnianie bezpieczeństwa
Fig. 42. Elements which influence ensuring safety

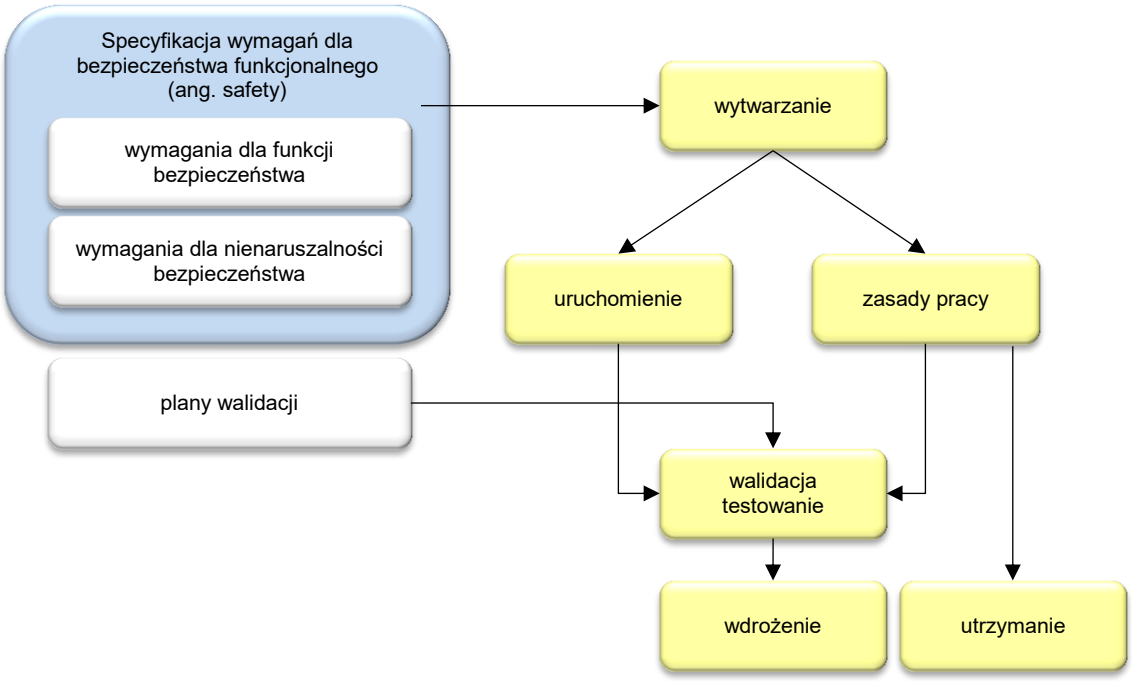
Według zaleceń normy IEC61508 tworzenie oprogramowania ISP[31] winno być wsparte systemem do zarządzania jego jakością [32], [354], [183]. System taki musi definiować:

- czynności zarządzania fazami cyklu życia oprogramowania,
- odpowiedzialność osób i organizacji za każdą fazę cyklu życia oprogramowania.

Dlatego kluczowa jest identyfikacja faz tworzenia ISP oraz jego komponentów. Proces certyfikacji, niezależnie od badanych aspektów technicznych, musi bazować na zidentyfikowanych i niezmiennych stadiach (fazach) życia oprogramowania, rozpatrywanych w kontekście konkretnych elementów systemu, względem których dokonuje się jakiś akcji tworzenia lub modyfikacji. Do kontroli, pomiarów i zarządzania takimi akcjami istnieją techniczne i organizacyjne mechanizmy nadzoru, których celem jest zapewnienie i wykazanie, że akcje i ich skutki nie naruszą przyjętych wymagań bezpieczeństwa.

Na rysunku 43 przedstawiono ogólny schemat cyklu życia bezpieczeństwa oprogramowania na bazie wytycznych normy IEC61508. Mowa tu o bezpieczeństwie w rozumieniu funkcjonalnym (ang. safety), choć dla innego rodzaju bezpieczeństwa (por. 4) omawiane mechanizmy również mogą się przydać. Plany walidacji bezpieczeństwa powstają jako efekt planowania na bazie specyfikacji. Proces walidacji i testowania produktu odbywa się na bazie modeli opisanych wcześniej, uwzględniając aspekty testów walidacyjnych bezpieczeństwa. Rozszerzenie tematyki certyfikacji oprogramowania w takim kontekście można znaleźć w dokumentach norm [M37], [M69] oraz w ich opracowaniach.

[31] W nomenklaturze normy występuje pojęcie PES (ang. Programmable Electronic System). Pojęcie to ogranicza jednak definicje systemu względem zakresu omawianego w książce.



Rys. 43. Cykl życia bezpieczeństwa oprogramowania
Fig. 43. Life cycle of software safety

Z określaniem ogólnej jakości oprogramowania wiąże się wiele miar struktury i złożoności oraz cechy jakości kodu. Powiązana norma to ISO9126 [M73]. Cechy jakości względem oprogramowania węzła ISP przedstawiono poniżej.

- Funkcjonalność (ang functionality)
  - zgodność z wymaganiami procesu (ang. Suitability),
  - poprawność algorytmizacji (ang. Accurateness),
  - zdolność do współpracy funkcjonalnej (ang. Interoperability),
  - zgodność ze specyfikacją (ang. Compliance),
  - zdolność do zachowania wymagań bezpieczeństwa (ang. Security).
- Rzetelność (ang. reliability)
  - dojrzałość rozwiązania, brak tymczasowości (ang. Maturity),
  - tolerowanie awarii (ang. Fault Tolerance),
  - odtwarzalność po awarii (ang. Recoverability).
- Wydajność (ang. efficiency)
  - zachowanie w czasie – charakterystyka czasowa (ang. Time Behavior),
  - obsługa zasobów (ang. Resource Behavior).
- Używalność (ang. usability)
  - zdolność rozumienia kodu (ang. Understandability),

  - przyswajalność obsługi (ang. Learnability),
  - zdolność do działania/operowania z oprogramowaniem (ang. Operability).
- Utrzymywalność (ang. maintainability)
  - analizowalność działania i problemów, diagnostyka (ang. Analyzability),
  - zdolność do modyfikacji (ang. Changeability),
  - stabilność (ang. Stability),
  - zdolność do testowania (ang. Testability).
- Przenośność (ang. portability)
  - zdolność do adaptacji względem zmian wymagań (ang. Adaptability),
  - zdolność do instalacji/reinstalacji (ang. Installability),
  - zgodność formalna, brak dedykowalności (ang. Conformance),
  - zdolność do zmian, wymian i zastąpień elementów systemu (ang. Replaceability).

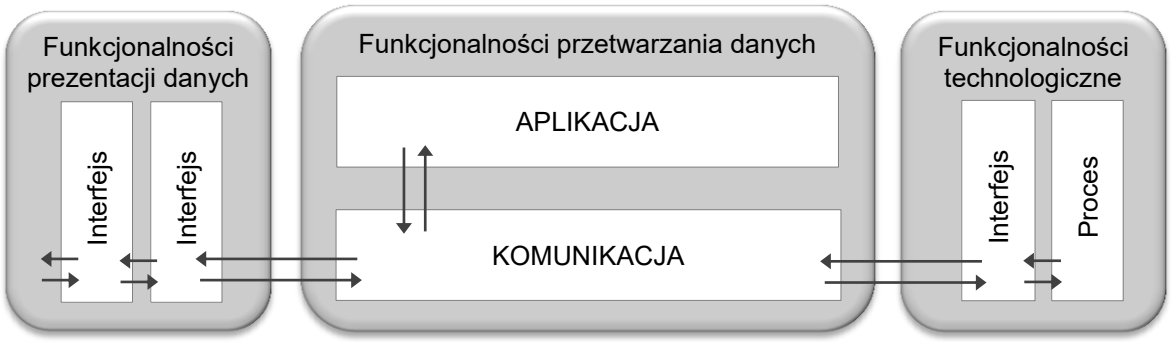
Do klasyfikacji oprogramowania użyteczne jest stosowanie określonych miar względem kodu. Pozwala to na stosowanie właściwych, dopasowanych metod i procedur certyfikacji. Przykładami miar struktur i złożoności kodu [194] względem węzła ISP mogą być:

- miara liczby linii kodu (LOC – ang.Lines of Code),
- miara Halsteada,
- miara złożoności cyklomatycznej McCabe'a,
- miara przepływu informacji TIP – ang. Tree Impurity.

Wymienione powyżej cechy i miary mogą posłużyć do zbudowania wytycznych certyfikacji oprogramowania ISP. Więcej można znaleźć np. w [405] lub w normie ISO9126 [M73].

2.2. Model przepływu danych

Podstawowym modelem odnoszącym się do przepływu informacji w ISP jest model zbudowany na bazie warstwowego wzorca architektury (por. 2.1). Dalszą systematyzację pojęć wprowadzono dla tego modelu. Model zawiera trzy odseparowane grupy ogólnych i niezbędnych funkcjonalności systemu oraz zachodzące między nimi przepływy informacji. Każda z grup ma swoje podstawowe warstwy, w których informacja posiada swoją reprezentację i lokalne zasady jej przekazywania. Zostało to przedstawione na rysunku 44.



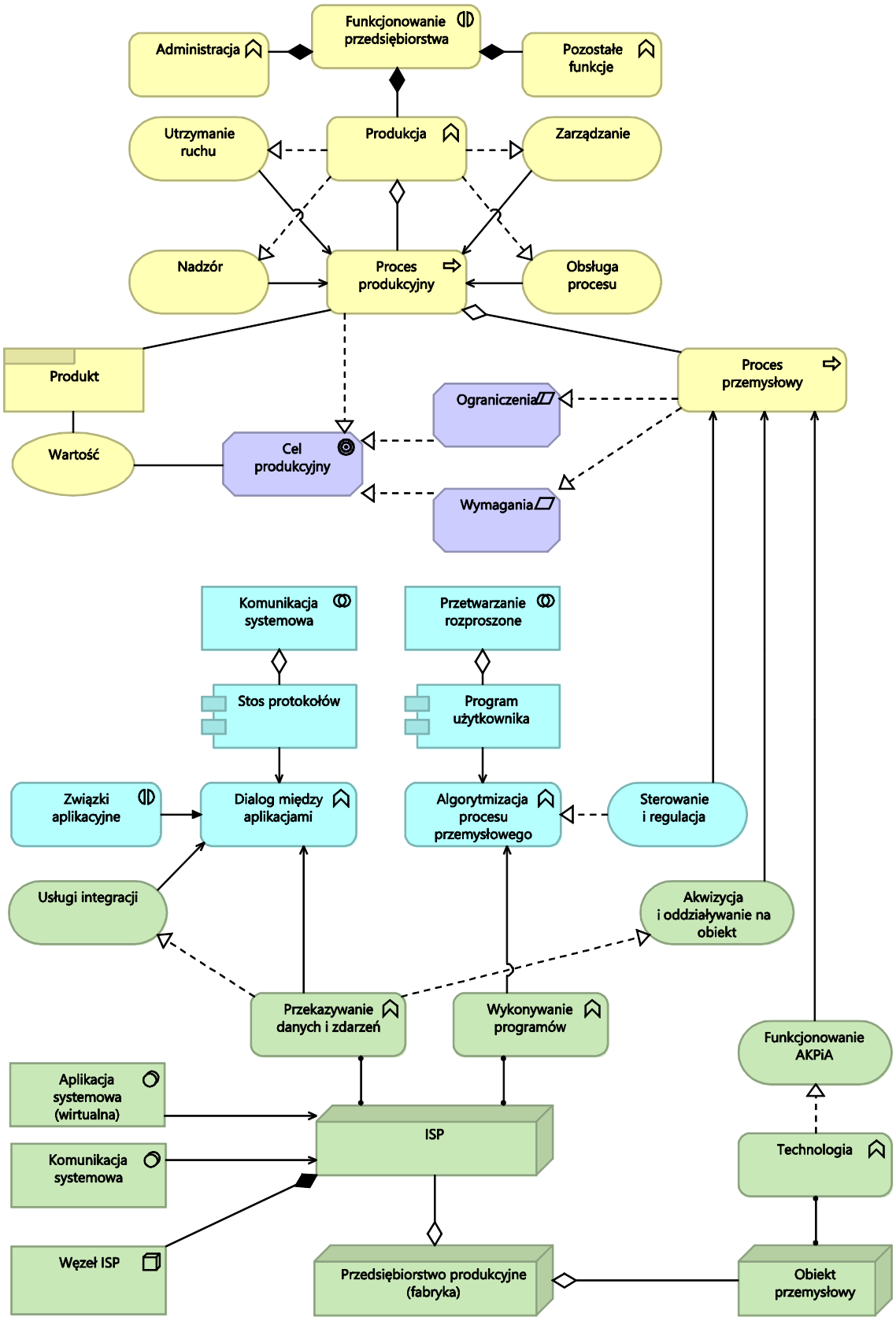
Rys. 44. Ogólny model ISP
Fig. 44. General model of ICS

Warstwa aplikacji zajmuje się realizacją zadań na rzecz procesu przemysłowego, warstwa komunikacyjna nadzoruje przepływ danych między elementami systemu. Interfejs technologii jest odpowiedzialny za dostarczanie informacji z procesu przemysłowego i wyprowadzanie informacji do procesu. Interfejsy prezentacji odpowiadają za skomunikowanie aplikacji z użytkownikiem oraz innymi systemami.

Powyższy model jest bardzo ogólny. Można go przedstawić nieco bardziej szczegółowo i bardziej formalnie, prezentując również połączenia z innymi abstrakcjami towarzyszącymi działaniu fabryki. Na rysunku 45 przedstawiono metamodel w notacji ArchiMate 2.0, uwzględniając powiązania z pojęciami technicznymi i biznesowymi opisanymi w 1.1 i 1.2 oraz sygnalizując pojęcia z podrozdziału 2.4. Jest to model systemu, a zatem abstrakcje dotyczą poziomu systemowego. Dla zwiększenia przejrzystości zawarto w nim także komponenty nawiązujące do modelu węzła ISP z rysunku 48. Przedstawiony model nie wyczerpuje opisu wszelkich zawiłości konstrukcyjnych systemów klasy ISP, ale oddaje ogólną jej ideę, uwypuklając najistotniejsze abstrakcje.

Metamodel przedstawia warstwy motywacji, biznesową, aplikacji i technologiczną (infrastruktury). Warstwę motywacji modelu stanowią ograniczenia i wymagania technologiczne oraz cel produkcyjny. Proces przemysłowy realizuje wymagania i ograniczenia, natomiast wymagania i ograniczenia, a także proces produkcyjny realizują cel. Produkt charakteryzowany jest przez jego wartość, która może być rozumiana szeroko, od kosztów do wskaźników jakości (por. 1.1.2). Wartość jest również powiązana z celem.

Przedstawiona warstwa biznesowa jest mocno uproszczona, ponieważ nie jest szczegółowo rozważana w niniejszej książce. Funkcjonowanie przedsiębiorstwa stanowi kompozycje podstawowych funkcji zapewniających zasoby biznesowe, w szczególności zasoby produkcji. Produkcja jest agregacją procesów produkcyjnych, a te z kolei procesów przemysłowych. Z produkcją związana jest realizacja podstawowych usług, jak: zarzadzanie, utrzymanie ruchu, nadzór i obsługa procesu. Usługi te są używane przez proces produkcyjny.



Rys. 45. Model ISP wraz z elementami przedsiębiorstwa
Fig. 45. ICS model together with enterprise elements

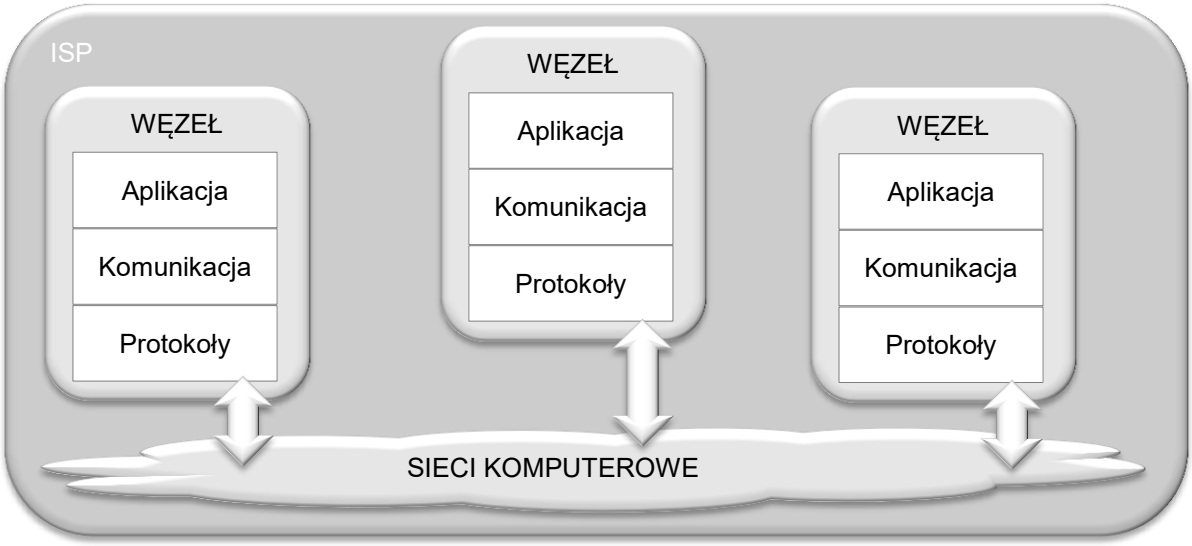
Obiekt przemysłowy stanowi informacyjną reprezentację (obraz, opis informacyjny) konkretnego procesu przemysłowego wykorzystywaną przez infrastrukturę ISP. Obiekt jest związany z konkretną funkcją technologiczną, obsługującą proces przez realizację usługi funkcjonowania elementów AKPiA. Obiekt jest agregowany przez fabrykę tak samo jak fabryka agreguje ISP. Rozdzielenie funkcji obiektu od funkcji ISP wynika z faktu, że na poziomie abstrakcji opisu fabryki istnieją dwie reprezentacje procesu technologicznego. Jedna to reprezentacja technologiczna związana z fizycznością procesu i zachodzących zjawisk, a druga to reprezentacja cyfrowa związana z wirtualnym jego obrazem i aplikacją jego obsługi w systemie informatycznym. Proces przemysłowy w warstwie biznesowej wykorzystuje usługę funkcjonowania AKPiA oraz usługę akwizycji i odziaływania na obiekt w warstwie infrastruktury. Zatem, na poziomie biznesowym rozdzielone usługi infrastruktury są widziane jako funkcjonalności oddziałujące z obiektem.

ISP stanowi abstrakcyjny zasób obliczeniowy przeznaczony do realizowania pewnych określonych funkcjonalności na rzecz obiektu. Skomponowany jest z fizycznych węzłów (por. 48). Na poziomie systemowym używa oprogramowania w postaci utworzonych przez węzły wirtualnej aplikacji i komunikacji systemowej. Głównymi funkcjami ISP są przekazywanie danych i zdarzeń oraz wykonywanie programów. Pierwsza z nich dostarcza szeroko pojętej usługi integrowania przepływów informacyjnych oraz akwizycji i oddziaływania z obiektem. Druga dotyczy realizacji systemowego przetwarzania danych i zdarzeń.

W warstwie aplikacji istnieją dwie systemowe funkcje. Jedna z nich zajmuje się obsługą procesu przemysłowego względem jego algorytmizacji. Funkcja ta używa komponentu programów użytkownika, które współpracują w ramach przetwarzania rozproszonego. Druga funkcja to obsługa dialogu pomiędzy aplikacjami. Używa ona stosu protokołów operujących w ramach komunikacji systemowej. Zasady wymian określa interakcja zdefiniowana na poziomie związków komunikacyjnych łączących fizyczne byty systemu. Algorytmizację realizuje usługa sterowania i regulacji, która połączona jest z warstwą biznesową przez proces przemysłowy.

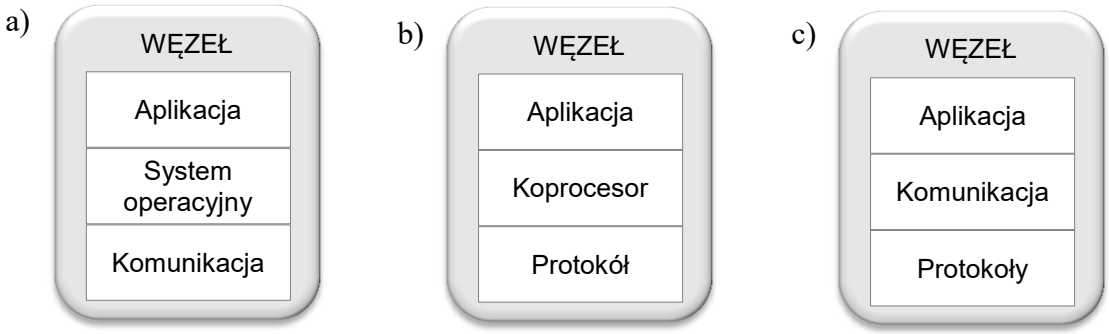
Jak wspomniano powyżej, ISP jest abstrakcyjnym zasobem obliczeniowym agregującym rozproszone węzły, które w rzeczywistości zwykle stanowią fizyczne urządzenia z fizycznymi zasobami, programami i interfejsami. Dla wszystkich interfejsów węzła zachodzą międzywarstwowe pionowe wymiany danych, a w warstwie komunikacji zachodzą dodatkowo wymiany poziome komunikujące elementy aplikacji. Warstwy te można odnieść do całego systemu jako do jego warstw wirtualnych. Można też przełożyć ją na fizyczne warstwy poszczególnych fizycznych elementów węzłów wchodzących w skład ISP. Dlatego model ISP można przedstawić jako system składający się z rozproszonych węzłów, komunikujących się siecią

komputerową (lub sieciami). Przedstawiono to na modelach ArchiMate z rysunków 45 wraz z 48 oraz w uproszczeniu na rysunku 46.



Rys. 46. Rozproszone węzły ISP
Fig. 46. Distributed nodes of ICS

W literaturze można spotkać trójwarstwowe modele węzła, składające się z warstwy aplikacji, systemu operacyjnego i komunikacji czy też z warstwy aplikacji, koprocesora i protokołu. Trzy warstwy są wystarczające do odzwierciedlenia przepływów pionowych i poziomych. Protokoły dostarczają języków i zasad składniowo-semantycznych dialogu, a sieci są środkiem do realizacji komunikacji. Modele takie przedstawiono na rysunku 47.



Rys. 47. Modele trójwarstwowe węzła
Fig. 47. Three-layers models of a node

W każdym z przedstawionych przypadków warstwa aplikacji odpowiada za przetwarzanie danych i realizację zadań przedstawionych w 1.2.4. W przypadku pierwszym (a) [292] warstwa komunikacji zawiera funkcje obsługi wejścia i wyjścia, włączając w to sieć. Warstwa systemu operacyjnego reprezentuje wszelkie zadania obsługi lokalnej węzła i nie ma bezpośredniego związku z modelowaniem jego konstrukcji względem jego roli w ISP.

W przypadku drugim (b) [220] abstrahuje się od systemu operacyjnego i traktuje go jako niezbędne tło działania urządzenia, tak jak zasoby sprzętowe. W zamian tego wprowadzono pojęcie warstwy koprocesora, która odpowiada za niezależną od aplikacji obsługę komunikacji sieciowej z innymi węzłami systemu. Warstwa protokołu modeluje język, jaki jest używany do realizacji tej komunikacji. Model ten, lepiej niż pierwszy, uwypukla zadania węzła nowoczesnych systemów. Pomija aspekty konstrukcyjne urządzenia, a uwypukla komunikację sieciową. Jednak we współczesnych urządzeniach pełniących funkcje węzłów ISP pojęcie koprocesora uległo dewaluacji. Zadania koprocesorów przejęły procesy systemu operacyjnego, a założenie pracy węzła w jednej sieci polowej stanowi ograniczenie konstrukcyjne. Obecnie przepływ danych w węźle odbywa się nie tylko między aplikacją i siecią, lecz również z układami IO, różnymi procesami lokalnymi, a przede wszystkim zachodzi w środowisku wielosieciowym. Dlatego w przypadku trzecim (c) wyodrębniono warstwę komunikacji jako warstwę „inteligentnego" przekazywania danych między elementami sprzętowo-programowymi urządzenia. Można by ją również nazwać warstwą kontroli przepływu. Warstwa protokołów natomiast odpowiada za wykonywanie zadań przetwarzania danych na rzecz działania sieci i wymiany danych z innymi węzłami. Warstwę tę można by też nazwać warstwą sieciową. Nasuwa się wniosek, że do zamodelowania węzła wystarczą dwie warstwy. Jedna odpowiedzialna za przetwarzania na rzecz procesu, czyli realizacje zadań funkcjonalnych, i druga, odpowiedzialna za wszelkie komunikacje, czyli realizację wymiany danych. Jednak aby jasno uwypuklić zagadnienie komunikacji zarówno wewnątrz, jak i na zewnętrz węzła, do dalszych rozważań przyjęto, że każdy z węzłów musi mieć przynajmniej trzy warstwy:

1. aplikacji – warstwa, w której pracuje program (programy) realizujące zadania funkcjonalne węzła i procesy je wspomagające.
2. komunikacji – warstwa ta odpowiedzialna jest za przekazywanie danych wewnątrz węzła, obsługując współdzielony dostęp do zasobów.
3. protokołów – warstwa, w której pracują programy realizujące dialog w sieciach z użyciem wymaganych protokołów.

Każdy węzeł ma w swoich warstwach zaimplementowane zadania związane z funkcjonowaniem takiej warstwy. W warstwie aplikacji węzła pracują programy użytkownika, stworzone i skojarzone funkcjonalnie z zadaniami węzła przez programistę. Działają w niej również programy obsługi modułów, programy diagnostyki węzła oraz jego konfiguracji, programy kontroli pracy i wszelkie inne procesy systemu operacyjnego (ang. OS – operation system) węzła, umożliwiające realizacje zadań na rzecz procesu przemysłowego. W przypadku PLC użytkownik-programista nie ma możliwości modyfikacji

wszystkich programów wchodzących w skład aplikacji węzła. Często może jednak parametryzować i konfigurować ich pracę.

Warstwa komunikacji obsługuje wymianę danych pomiędzy wszelkimi ich źródłami, a w szczególności pomiędzy aplikacjami węzła, sieciami i układami IO. Zadania w tej warstwie muszą zapewniać synchronizację dostępu wszelkich programów pracujących w węźle do współdzielonych zasobów. Na tym poziomie realizowane są usługi komunikacyjne, alokacje tymczasowe, buforowania, translacje reprezentacji, dostępy z poziomu programatora itp. Warstwa ta jest niezbędna, aby uniezależnić obsługę obiegów informacji, jakie występują w węźle i nimi zarządzać. Skojarzenie warstwy komunikacji z pojęciem koprocesora jest możliwe. Jednak we współczesnych węzłach klasy PLC przepływ danych może być na tyle złożony, że rozpatrywanie tylko pojedynczej sieci i koprocesora jest silnym uproszczeniem. Ponadto, fizyczne koprocesory są rzadko stosowane z racji prostszej i tańszej implementacji obsługi sieci w procesach systemu operacyjnego.

Warstwa protokołów zapewnia wykonanie zadań odpowiedzialnych za dialog w sieciach. Znajdują się tam programy obsługujące zdefiniowane w węźle protokoły sieci przemysłowych, sieci lokalnych i łączy szeregowych, protokoły serwisowe i dostępowe z poziomu programatora itp. Konstrukcja tej warstwy może się opierać na warstwowym modelu sieci wywodzącym się z IOS/OSI (np. trójwarstwowym modelu sieci polowych lub pięciowarstwowym modelu ogólnym [64]).

Na rysunku 48 przedstawiono metamodel węzła ISP w notacji ArchiMate 2.0. Dotyczy on podejścia (c) z rysunku 47. Przedstawiono powiązane w odpowiednich relacjach elementy infrastruktury i aplikacji. Podobnie jak wyżej omawiany model systemu, jest to model uproszczony uwzględniający tylko istotne elementy i pojęcia poruszane w treści książki. Budowanie opisu bardziej szczegółowego spowodowałoby utratę jego uniwersalności. Prezentowanie dalszych szczegółów miałoby sens tylko przy modelowaniu konkretnych rozwiązań sprzętowo-programowych.

Węzeł ISP stanowi kompozycję:

- zasobów przetwarzających, które zwykle należy postrzegać jako jednostki przetwarzające (komputery, zob. 3.1),
- oprogramowania realizującego aplikacje, komunikacje i protokoły w węźle, stanowiącego zwykle programy uruchomione w środowisku OS węzła (zob. 3.2 i 3.3),
- oraz interfejsów technologii i prezentacji, umożliwiających oddziaływania z obiektem i innymi systemami przy użyciu układów IO oraz sieci komputerowych węzła (zob. 1.2.4).

Interfejsy stanowią komponenty węzła i są używane przez oprogramowanie obsługi komunikacji. Do zapewnienia właściwej cyrkulacji danych oprogramowanie aplikacji używa

oprogramowania komunikacji, a komunikacja używa oprogramowania protokołów. Zgodnie z wcześniejszym opisem, protokoły realizują zewnętrzne przekazywanie danych, komunikacja węzła zapewnia wewnętrzne przekazywanie danych, a aplikacja dostarcza usług obsługujących zadania lokalne węzła, czyli obsługę zadań funkcjonalnych oraz obsługę zadań operacyjnych. Dialogi z innymi węzłami prowadzone są z użyciem sieci komputerowych i stosu protokołów na podstawie zdefiniowanych związków komunikacyjnych i z użyciem odpowiednich ścieżek komunikacyjnych (por. model z rys. 45). Przekazywanie wewnętrzne dotyczy funkcji obsługi obiegów informacyjnych w węźle i komunikacji z obiektem przemysłowym.

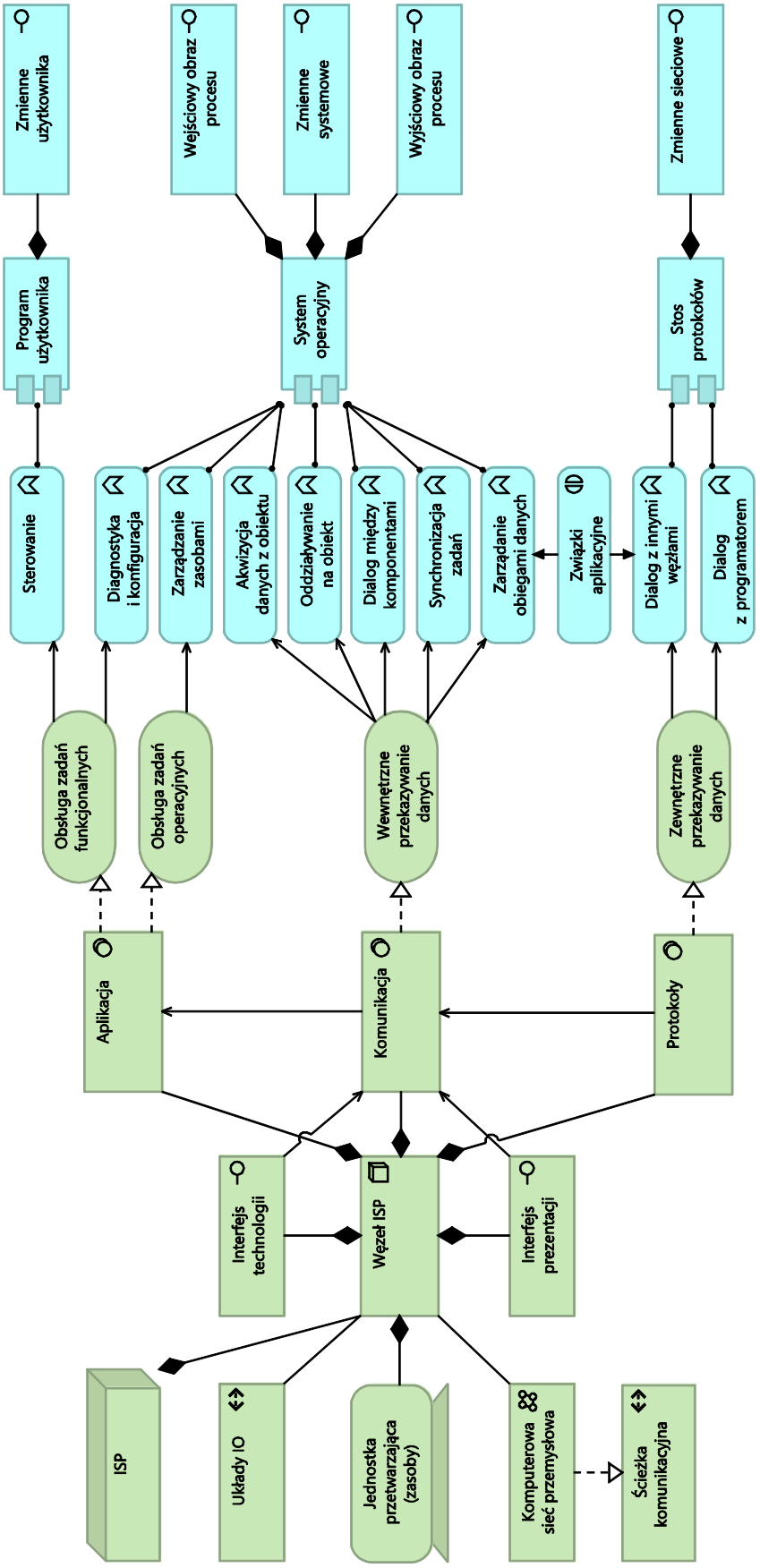
W warstwie biznesowej modelu istnieje wiele funkcji, które używają wymienionych powyżej usług infrastruktury węzła.

- Obsługa zadań funkcjonalnych jest wykorzystywana przez funkcję sterowania oraz przez funkcje diagnostyczne i konfiguracyjne.
- Obsługa zadań operacyjnych jest używana przez funkcje zarządzania zasobami.
- Usługa wewnętrznego przekazywania danych jest używana przez funkcje akwizycji danych, oddziaływania na obiekt, dialogu między komponentami węzła, synchronizacji zadań i zarządzanie obiegami danych w węźle.
- Usługa zewnętrznego przekazywania danych jest wykorzystywana przez funkcje dialogu z innymi węzłami oraz dialogu z narzędziami deweloperskimi.

Ponadto, funkcje dialogu z innymi węzłami oraz zarządzania obiegami w węźle są wyzwalane przez zdefiniowane związki aplikacyjne definiujące interakcje między elementami aplikacji.

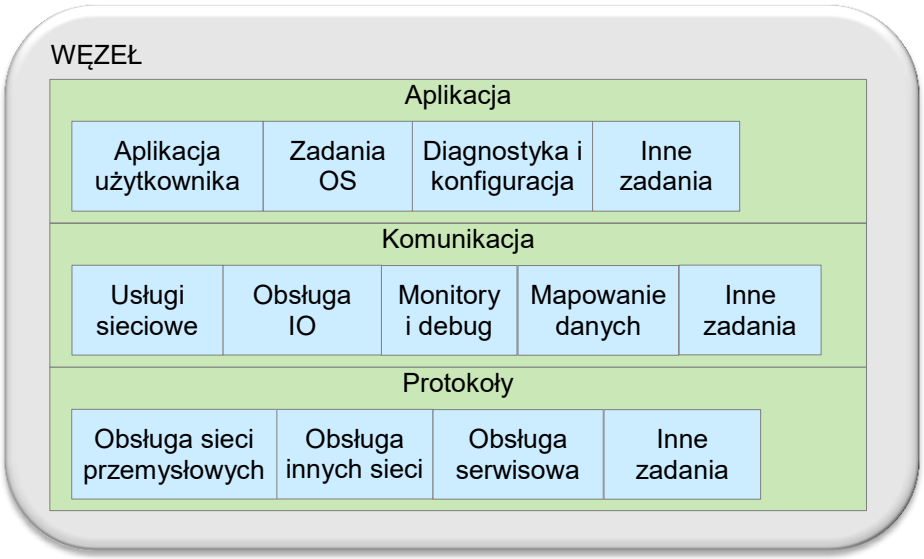
Funkcje aplikacji są powiązane z działaniem trzech komponentów aplikacji: programu użytkownika, systemu operacyjnego i stosu protokołów. Z założenia komponenty te są modułowe i wymienne, przynajmniej w pewnym zakresie. Program użytkownika jest przypisany do funkcji sterowania, a system operacyjny do funkcji operacyjnych węzła. Stos protokołów jest związany z funkcjami dialogu z innymi węzłami oraz z programatorem.

Na prezentowane powyżej komponenty składa się pewna liczba interfejsów aplikacyjnych fizycznie realizowanych w postaci zestawu zmiennych i ogólnie pojętych obrazów informacyjnych. Zmienne lokalne węzła podzielono na zmienne użytkownika i zmienne systemowe, a obraz procesu, służący do reprezentowania opisu informacyjnego obiektu, na obraz wejściowy i wyjściowy. Interfejsy udostępniają wewnętrzne usługi aplikacyjne węzła innym komponentom aplikacji.



Rys. 48. Model węzła ISP
Fig. 48. Model of ISP node

Przykładowe zadania oprogramowania infrastruktury węzła przedstawione zostały w innym ujęciu na rysunku 49. Funkcje węzła można więc mnożyć w zależności od stopnia uszczegółowienia modelu lub chęci uwypuklania określonych funkcji.



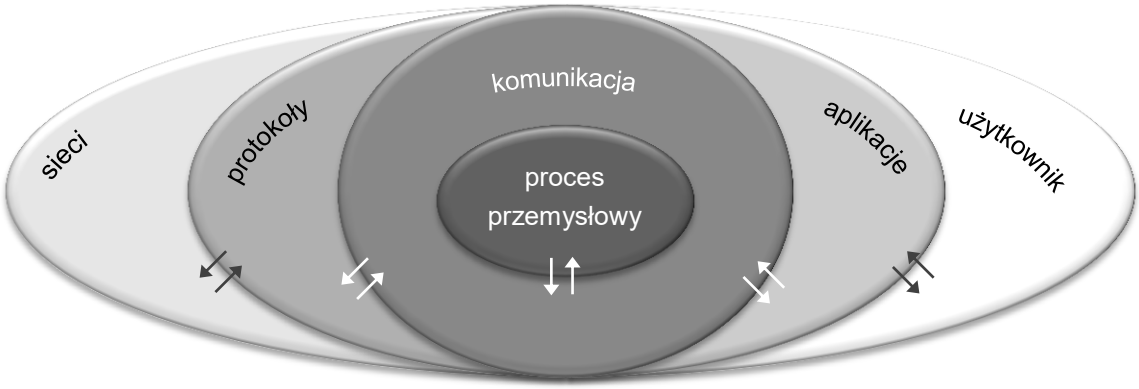
Rys. 49. Przykłady zadań w poszczególnych warstwach
Fig. 49. Example of tasks in each of the layers

Przedstawione modele są dość uniwersalne. W praktyce konstrukcje węzłów i systemów mogą od niego nieco odbiegać. Jednak do ogólnego opisania współczesnych ISP, działających na poziomie procesu (zob. rysunki 14, 16) i tworzonych na bazie urządzeń komputerowych klasy PLC, jest on wystarczający.

2.3. ISP jako system rozproszony

Przenosząc przedstawiony w poprzednim podrozdziale model węzła na środowisko systemowe przedstawione modelem z rysunku 44, można zobrazować ISP jako abstrakcyjne środowisko sprzętowo-programowe, składające się z rozproszonych grup funkcjonalności realizowanych przez rozproszone zasoby. Przedstawiono to na rysunku 50.

Zastosowanie komputerów w dziedzinie systemów automatyki można najprościej podzielić ze względu na liczbę urządzeń przetwarzających i sposób wykonywania zadań funkcjonalnych. ISP są systemami rozproszonymi, gdzie węzły są rozproszone terytorialnie, a zadania są rozproszone logicznie w węzłach. Zatem, najprościej rzecz ujmując, liczba komputerów determinuje rozproszenie systemu, a miejsce wykonywania kluczowych akcji determinuje decentralizację przetwarzania.



Rys. 50. Rozproszenie funkcjonalne ISP
Fig. 50. Functional distribution of ICS

2.3.1. Klasyfikacja rozproszenia

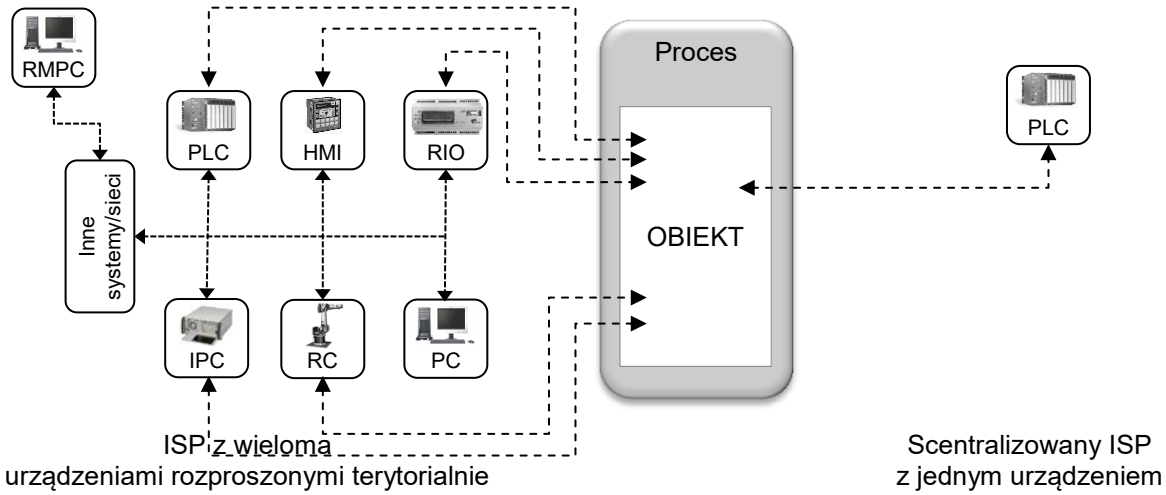
Informatyczny system przemysłowy może być postrzegany ze względu na liczbę i lokalizację fizycznych elementów zajmujących się przetwarzaniem informacji, czyli tzw. urządzeń „inteligentnych". Z tego punktu widzenia można wydzielić dwa przypadki.

W systemie może pracować tylko jeden „inteligent" wraz z układami wejścia/wyjścia, które służą mu do komunikowania się z obiektami przemysłowymi. Informacja z procesu przemysłowego będzie dostarczana przez interfejs w postaci analogowej i/lub cyfrowej, a następnie kodowana i gromadzona w pamięci jednostki, po czym przetwarzana i przekazywana pomiędzy zadaniami przez nią realizowanymi.

Może również pracować wiele „inteligentów" rozmieszczonych w różnych miejscach i pełniących różne funkcje. Wówczas informacja z obiektu jest dostarczana poprzez interfejsy, tak jak poprzednio w postaci analogowej i/lub cyfrowej, lecz nie do jednej jednostki, tylko do wielu jednostek rozproszonych na terenie działania procesu.

Należy zauważyć, że w systemie może być np. jeden komputer, ale z wieloma zasobami umożliwiającymi równoległe przetwarzanie lub z systemem operacyjnym umożliwiającym wykonywanie wielu procesów w wielu wątkach na jednym zasobie. Możliwości w takim postrzeganiu jest wiele. Dla uporządkowania dalszych rozważań należy wyróżnić pojęcie zadania x wykonywanego na zasobie r. Powszechnie przyjmuje się, że gdy liczba zasobów $N_r = 1$, to mamy do czynienia z systemem scentralizowanym, a gdy $N_r > 1$, to z rozproszonym. Niemniej jednak dla systemu, czyli dla elementów pozostających w zależnościach, należy również brać pod uwagę liczbę zadań N_x. Dla $N_x > 1$ można mówić o obliczeniach rozproszonych lub ogólnie o przetwarzaniu rozproszonym, niezależnie od liczby fizycznych zasobów N_r, na których to przetwarzanie zachodzi. Jednak na potrzeby informatycznych systemów przemysłowych wygodnie jest klasyfikować systemy, rozpatrując liczbę urządzeń fizycz-

nych. W praktyce, gdy komputerów jest wiele, to sprowadza się to do rozproszenia terytorialnego. Innymi słowy, rozległość terytorialna obiektów przemysłowych wymusza rozproszenie terytorialne systemu. Oczywiście teoretycznie można tworzyć ISP, umieszczając w jednej szafie wiele współdziałających komputerów, ale jeśli nie ma szczególnych powodów ku temu, to z praktycznego punktu widzenia nie ma to uzasadnienia. Często w nowych rozwiązaniach spotyka się przypadek, gdy N_r i Nx są nieznane, dynamicznie zmienne i stosunkowo duże. Wynika to z faktu, że często do systemów lokalnych dynamicznie dołącza się komputery pracujące zdalnie. Często są to komputery działające w dedykowanych sieciach telekomunikacyjnych lub intranecie zakładowym, np. stacje robocze realizujące zadania wizualizacji, raportowania, zarządzania, planowania itp. Równie często są one zlokalizowane w sieciach publicznych, np. w Internecie, w sieciach komórkowych itp. i pełnią podobne funkcje.

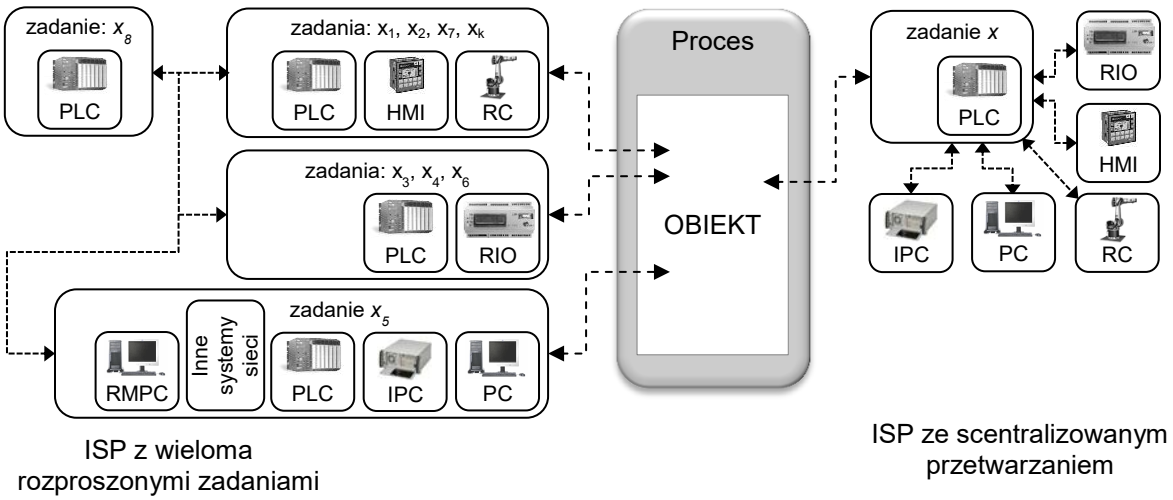


Rys. 51. Klasyfikacja ISP względem rozproszenia urządzeń
Fig. 51. ICS classification according to devices distribution

We współczesnych ISP, w przypadku rozległych intersieci liczba zdalnych abonentów i zadań dynamicznie podłączanych do systemu teoretycznie może być bardzo duża. W praktyce, liczba dynamicznych i zdalnych abonentów zawiera się przeważnie w przedziale od kilku do kilkunastu. Podział ISP wg liczby komputerów przedstawiono na rysunku 51. Typy węzłów są przykładowe, a ich oznaczenia są opisane w 3.1.

Oprócz klasyfikacji względem liczby urządzeń i zadań, system może być również postrzegany ze względu na sposób podejmowania istotnych decyzji. Dlatego drugim ważnym kryterium podziału systemów jest kryterium miejsca wykonywania zadań funkcjonalnych. Z jednej strony, może istnieć główny „mózg", który podejmuje decyzje na podstawie pełnego obrazu systemu dostarczanego przez elementy systemu. Abstrahując od organizacji kodu, takim „mózgiem" jest algorytm, który przetwarza informacje na podstawie pełnego wektora

informacyjnego, opisującego obiekt w danej chwili czasu oraz pełnego wektora stanu obiektu wypracowanego wcześniej. Z drugiej jednak strony, system może nie posiadać takiego „centrum dowodzenia", a może istnieć wiele współpracujących „mózgów", które zajmują się tylko swoim zakresem działań, komunikując się z wybranymi innymi elementami celem pozyskania niezbędnych do funkcjonowania informacji. W takim przypadku są to algorytmy obsługujące tylko wybrany zakres funkcjonalny systemu lub nawet wykonujące tylko fragment przetwarzania na potrzeby innego „mózgu". Bazują one na wektorze informacji opisującym tymczasowy lokalny stan fragmentu systemu oraz na swoim stanie lokalnym. Pierwszy opisany przypadek to systemy ze scentralizowanym przetwarzaniem, a drugi to systemy ze zdecentralizowanym przetwarzaniem. Można zauważyć, że zarówno jedne, jak i drugie występują w naturze, jak również były tworzone przez człowieka od zarania dziejów. Podział ten został zobrazowany na rysunku 52. Typy węzłów są przykładowe, a ich oznaczenia są opisane w 2.4.1.



Rys. 52. Klasyfikacja rozproszenia ISP względem zadań
Fig. 52. ICS classification according to tasks

Systemy scentralizowane są pozornie prostsze w realizacji. Jednak w praktyce tworzenie, testowanie i modyfikacja algorytmu obsługującego cały proces jest złożone i błędotwórcze. Ponadto rozwiązanie takie jest słabo skalowalne. Przy ewentualnej rozbudowie grozi sytuacja, że braknie zasobów, np. pamięci lub mocy obliczeniowej. W efekcie zmiany będą niemożliwe do implementacji lub charakterystyka czasowa pracy całości ulegnie niekorzystnej zmianie. W przypadku systemów zdecentralizowanych poszczególne algorytmy są prostsze i nie odwołują się do całego stanu systemu. Testowanie i modyfikacja odnosi się tylko do części pozostających w zależnościach. Skalowalność jest duża ze względu na możliwość rozbudowy systemu o kolejne zasoby i kolejne zadania na nich pracujące. Wadą jednak takiego rozwiązania jest istnienie wąskiego gardła, jakim jest kanał komunikacyjny, służący do wy-

miany informacji między zadaniami. Kanał spowalnia działanie systemu oraz w skrajnym przypadku może uniemożliwić rozbudowę.

Zarówno we wspomnianej historii, jak i w praktyce inżynierskiej oba rozwiązania mają wady i zalety. Trudno jednak jednoznacznie określić, że któreś podejście jest lepsze lub jedynie słuszne. Wszystko zależy od obiektu i wymagań, jakie są stawiane przed systemem. Bez wątpienia system, który obsługuje proces zlokalizowany na niewielkim obszarze, nie wymaga rozproszenia. Decyzja o budowaniu systemu rozproszonego, oprócz aspektów technicznych, powinna być podyktowana względami ekonomicznymi.

2.3.2. Właściwości systemu

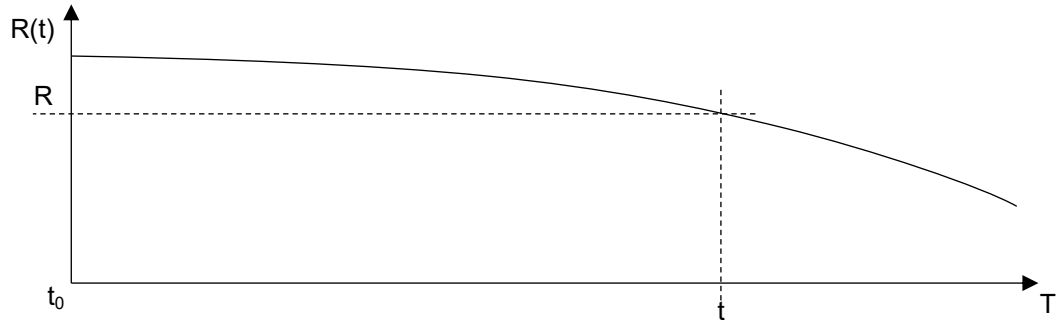
Na stronie 120 przedstawiono cechy jakości oprogramowania związanego z ISP powiązane z normą ISO9126 [M73]. Przemysłowe systemy informatyczne czasu rzeczywistego odznaczają się pewnymi charakterystycznymi atrybutami jakości związanymi z wymaganiami funkcjonalnymi. Jest to spojrzenie szersze, dotyczące własności systemu, takich jak:

- Niezawodność – (ang. reliability)
  Jest to ogólna właściwość współczesnych ISP i dotyczy zdolności systemu do poprawnej realizacji powierzonych mu zadań. Związana jest ona z pojęciem bezpieczeństwa, na które składa się pewność działania (ang. safety) oraz integralność systemu (ang. security), a także pojęciem dostępności systemu (ang. availability) w okolicznościach zaburzających jego normalną pracę. Pewność działania, czyli tzw. bezpieczeństwo funkcjonalne, zależy od prawidłowego zadziałania ISP w odpowiedzi na informacje wejściową i zgodnie z ograniczeniami czasowymi, a zapewnienie integralności systemu wiąże się z klasycznym pojęciem bezpieczeństwa zasobów, dostępu i danych [M37]. Dostępność dotyczy zdolności systemu do wykonania danego zadania w zaistniałych warunkach w wymaganym czasie, przy założeniu że zewnętrzne zasoby wymagane do jego realizacji są dostępne. Więcej na ten temat znajduje się w rozdziale o bezpieczeństwie (3.4) oraz np. w [208].
- Rzetelność – (ang. dependability)
  Rzetelność określana jest wskaźnikiem *R* i jest definiowana jako prawdopodobieństwo, że system będzie pracował przez czas większy od czasu bezawaryjnej pracy systemu.

  $$R(t) = P(t > T)$$

  gdzie *T* to czas bezawaryjnej pracy systemu, a *t* to czas wymagany.

  Z upływem czasu rzetelność systemu maleje ze względu na zużycie elementów. Zostało to przedstawione na rysunku 53. Czas T_0 oddaje początek pracy systemu po wdrożeniu, czyli po wyeliminowaniu błędów początkowych związanych z procesem uruchamiania.



Rys. 53. Ilustracja utraty rzetelności i wyznaczenia wymaganego współczynnika *R*
Fig. 53. The graph of dependability loss and estimation of the required *R* factor

- Spójność informacyjna – (ang. data & time consistency)
  Cecha określająca przestrzenną i czasową spójność informacji. Zdolność do zachowania wartości danych przy współdzieleniu informacji i współistnieniu jej reprezentacji w różnych elementach systemu [234].

  Ze względu na rozproszenie terytorialne w systemie występują fizycznie odseparowane źródła informacji. Istotne jest zapewnienie, aby wartości danej u źródła i w dowolnym innym miejscu systemu nie różniły się przez czas dłuższy niż pewny gwarantowany czas graniczny. Dla przykładu, wartość pomiaru ciśnienia V w węźle i: V_i dokonanego w momencie czasu t: V_i^t powinna być zaktualizowana we wszystkich innych elementach systemu, zainteresowanych tą wartością, w czasie mniejszym niż T_G. Innymi słowy, dopuszcza się istnienie różnic w lokalnej wartości danej tylko przez określony czas. Powyżej tego czasu wartość V_i^t musi być zaktualizowana we wszystkich innych komputerach.

$$V_i^t(t_p) = V_j^t(t_q)\ gdy\ \left(\forall i,j\right)(i \neq j)\left(\forall q > p\right)\left(t_q - t_p \geq T_G\right)$$

$$\left(\exists i,j\right)(i \neq j)\ \left(\exists q > p\right)\left(t_q - t_p < T_G\right), że\ V_i^t(t_p) \neq V_j^t(t_q)$$

  gdzie:

  i, j – elementy systemu wymieniające informacje,

  p, q – momenty w czasie działania systemu.

  Prawdziwość nierówności wynika z faktu, że transmisja danych między dwoma elementami systemu nie może zachodzić w czasie zerowym.

  Ponieważ występuje tu problem dystrybucji wartości w środowisku terytorialnie rozproszonym oraz problem zapewnienia zgodności tych wartości w czasie, więc o spójności informacyjnej mówi się często, jako o cesze spójności przestrzennej i spójności czasowej.

- Skalowalność – (ang. scalability)
  Właściwość zachowania parametrów pracy systemu przy zmianie skali systemu, np. rozbudowie, modyfikacji. System charakteryzuje się pewnymi cechami związanymi z przetwarzaniem, komunikacją, możliwością składowania informacji itp. Przy zmianie skali systemu wartości opisujące te cechy powinny pozostać niezmienione.

  Dla przykładu, zakładając, że system realizuje n zadań, każde z cyklem t_{c1}, to przy dołożeniu m nowych zadań z cyklem t_{c2} system powinien realizować n zadań z niezmienionym cyklem t_{c1} oraz m zadań z cyklem t_{c2}.

  W praktyce, przy zwiększaniu skali, najtrudniej jest dotrzymać wartości czasowych, opisujących komunikację sieciową. Wynika to z faktu, że sieć komputerowa ma ograniczoną przepustowość maksymalną. Zwiększanie liczby transmitowanych danych lub skracanie cykli transakcji jest możliwe do pewnej granicy, powyżej której zachodzi wpływ na istniejącą charakterystykę czasową pracy sieci. Rozwiązaniem problemu może być tylko zwiększenie zasobów, czyli w tym przypadku zwiększenie przepustowości kanału komunikacyjnego przez zmianę sieci/protokołu lub dołożenie dodatkowej sieci. Można też stosować mechanizmy opisane w [23] lub podobne triki.

- Przewidywalność – (determinizm, ang. determinability)
  Własność określająca deterministyczne zachowanie systemu w każdej, szczególnie pesymistycznej (ang. worst-case), sytuacji. Jest to podstawowa cecha systemów informatycznych. Wszystkie klasyczne systemy, działające za pomocą komputerów technicznych w logice boolowskiej w danych warunkach i stanie, muszą wykazywać się zawsze taką samą reakcją. W przeciwnym wypadku system byłby niestabilny i nieprzewidywalny.

  Jest to cecha, którą w praktyce łatwo zapewnić, gdyż każdy klasyczny komputer, o ile jest sprawny, oraz każdy system zbudowany z takich komputerów wykazuje cechę determinizmu działania. Brak determinizmu jest zawsze spowodowany awarią lub błędami konstrukcyjnymi. Dlatego zmniejszenie podatności na utratę deterministycznego działania wymaga zapewnienia zwiększonej odporności na awarie (por. cecha tolerancji awarii).

  Czasami w ISP stosuje się urządzenia, których działanie oparte jest na logice rozmytej. Nie oznacza to utraty cechy przewidywalności. Urządzenia i podsystemy są w takich przypadkach nadal deterministyczne. Jedynie algorytmy, według których realizują swoje zadania, oparte są na wartościowaniu rozmytym [193]. Dla niektórych zastosowań, gdzie charakterystyka obiektu nie jest znana lub trudno ją wyznaczyć, rozwiązania takie sprawdzają się lepiej niż podejście klasyczne.

Pojęcie przewidywalności pracy systemu występuje również w wymiarze biznesowym procesu produkcyjnego. Ma ono jednak zupełnie inne znaczenie i dotyczy przewidywalności kosztów wytwarzania związanego z powtarzalnością działania danego procesu technologicznego oraz określonością czasową danego procesu wytwarzania.

- Punktualność – (zdeterminowanie czasowe, ang. time-constrainig)
  Właściwość określająca sposób wykonywania działań względem czasu. Można przyjmować klasyfikację systemów pochodzącą z teorii systemów czasu rzeczywistego (por. 1.1.4). Jeśli system pracuje z ograniczeniami czasowymi to każdy jego element uczestniczący w przetwarzaniu informacji musi podlegać tym ograniczeniom. W praktyce większość systemów przemysłowych działa z ograniczeniami czasowymi, przeważnie w kategorii FRTS, rzadziej w HRTS, a bardzo rzadko w SRTS.

  Praca pod rygorem ograniczeń czasowych wiąże się z wykonywaniem zadań w czasie określonym przez parametr czasu granicznego. Zagadnienie te są opisane w rozdziale 1.1.4.

- Tolerancja awarii – (ang. fault-tolerant)
  Właściwość określająca zdolność do działania mimo awarii. Zakłada się, że wystąpienie awarii elementu systemu lub wystąpienie błędu działania może zostać obsłużone przez inny element systemu, jego element zapasowy (ang. redundancy) lub specjalne funkcje oprogramowania. W praktyce systemy tolerujące awarie są droższe, ale w przypadku aplikacji krytycznych, od których zależy życie, zdrowie lub znaczące mienie, ich wykorzystanie jest konieczne [273].

- Przezroczystość – (ang. transparency)
  Własność związana z postrzeganiem systemu jako jednolitego rozwiązania bez wyróżnionych elementów składowych. System ma zbiór funkcjonalności, które są realizowane przez system jako całość, a z punktu widzenia użytkownika (klienta) nie są widoczne elementy struktury systemu, przetwarzania danych czy ich przepływów. Brak przeźroczystości jest w konflikcie z wcześniejszą definicją systemu (por. 1.1.5, 2.2).

  Przezroczystość nie jest cechą specyficzną ISP. Jest ona związana z każdym rozproszonym systemem informatycznym. W praktyce i w kontekście ISP, system przeźroczysty zapewnia łatwiejszą, względem nieprzeźroczystego, modyfikację, umożliwiając spojrzenie na nią od strony funkcjonalnej, a nie sprzętowej.

- Współbieżność – (ang. concurrency)
  Właściwość związana z możliwością równoległej pracy wielu zadań na wielu zasobach. Jest to również cecha ogólna systemów rozproszonych. Z racji rozproszenia dość naturalne jest uzyskanie zrównoleglenia przetwarzania. Należy jednak pamiętać

o konieczności synchronizowania pracy aplikacji wynikającej z interakcji z procesem przemysłowym lub wymiany informacji między elementami systemu.

W praktyce wykorzystuje się wiele urządzeń komputerowych, w tym PLC, które pracują równolegle. W wybranych momentach działania ich aplikacje dokonują niezbędnych synchronizacji swoich lokalnych zadań. Ewentualne synchronizowanie działania wszystkich zadań prowadzi do utraty mocy obliczeniowej poszczególnych jednostek przetwarzających, gdyż po zakończeniu działania zadania muszą być wstrzymywane. Sekwencyjne synchronizowanie zadań prowadzi do zaniku systemowej współbieżności i pracy wsadowej (ang. batch processing)[32].

- Trwałość – (utrzymywalność, ang. sustainability)
  Cecha określająca zdolność systemu do pracy bez przerw i utrzymania procesu w ciągłym ruchu. Dotyczy to nieplanowanych postojów wynikających z utrzymania i pielęgnacji systemu, jak i procesu oraz awarii. Cecha wiąże się z tolerancją awarii, jednak opisuje system szerzej, gdyż oprócz awarii dotyczy takich działań, jak administracja, modyfikacja aplikacji i konfiguracji, zmiana reguł komunikacji, diagnostyka itp.

  Jest to cecha, która szczególnie jest istotna w systemach ISP. W wielu przypadkach zatrzymanie procesu nie jest możliwe, jest kosztowne lub wymaga wiele czasu. Dlatego ISP obsługujący taki proces musi wykazywać cechę trwałości działania, aby nie doprowadzić do utraty pewności działania. Jednym z przykładów użytecznej funkcji systemu zwiększającej trwałość jest możliwość aktualizacji kodu PLC bez zatrzymywania aplikacji. W praktyce znane są przypadki oczekiwania kilkunastu dni na zezwolenie zatrzymania fragmentu systemu w celu dokonania drobnych zmian w kodzie.

  Ponadto cecha ta dotyczy zapewnienia możliwości modernizacji systemu bez uraty funkcjonalności i bez konieczności jego wymiany. Projektanci tworzący system odznaczający się tą cechą muszą brać pod uwagę rozwój technologii zarówno od strony realizacyjnej, jak i od strony produkcyjnej, aby nie doprowadzić do sytuacji, gdy utrzymanie systemu stanie się niemożliwe z powodu np. niedostępności komponentów czy zarzucenia rozwoju danej technologii IT, lub nie będzie możliwości skalowania systemu do nowych, nieznanych wcześniej, potrzeb. Czas życia ISP jest relatywnie długi (por. 1.2.3), dlatego proces konserwacji zarówno oprogramowania, jak i sprzętu musi trwać również dłużej niż w systemach domowo-biurowych. Podobnie z procesem ewolucji i to nie tylko oprogramowania, ale i elementów sprzętowych, a nawet systemowych funkcjonalności.

[32] W procesach sterowania istnieją procesy wsadowe ([351], [337]), które nie stanowią głównego nurtu w tej książce.

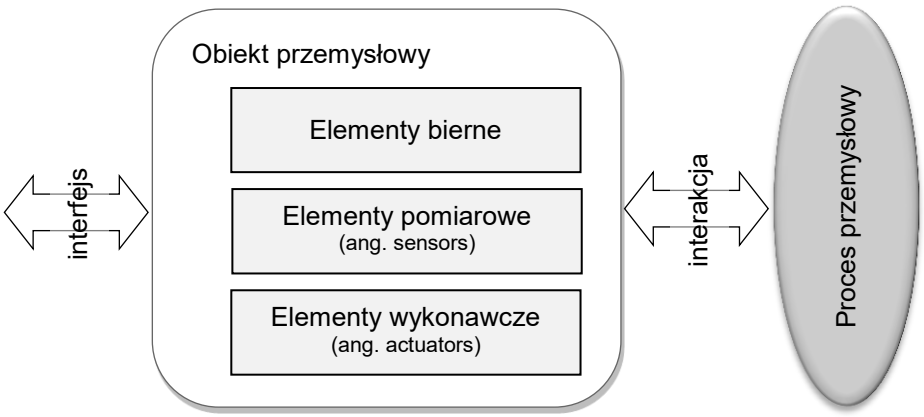
Powyższa lista nie wyczerpuje listy cech. Więcej cech jakości systemów i oprogramowania można znaleźć we wspomnianej normie IEC 9126 [M73] oraz normie IEC 25010 [M70]. Więcej informacji o systemach rozproszonych można znaleźć w literaturze, np. w [66].

2.4. Abstrakcje w ISP

W przemysłowych systemach rozproszonych z decentralizowanym przetwarzaniem, które stanowią tło niniejszej książki i w dalszej jej części są nadal oznaczane jako ISP, można wyróżnić kilka elementów abstrakcyjnych niezależnych od konkretnych rozwiązań. W kolejnych podrozdziałach są one pokrótce opisane. Każdy ISP dobrze jest na wstępie jego tworzenia postrzegać w takich właśnie kategoriach abstrakcyjnych, aby uniknąć wpływów (czytaj ograniczeń), wynikających z konkretnych technik, technologii i gotowych rozwiązań.

2.4.1. Obiekt przemysłowy

Obiekt przemysłowy jest na potrzeby dalszych rozważań definiowany jako abstrakcyjne urządzenie automatyki, wchodzące w interakcję z procesem przemysłowym oraz posiadające interfejs wymiany danych z ISP. Model takiego obiektu przedstawiono na rysunku 54.

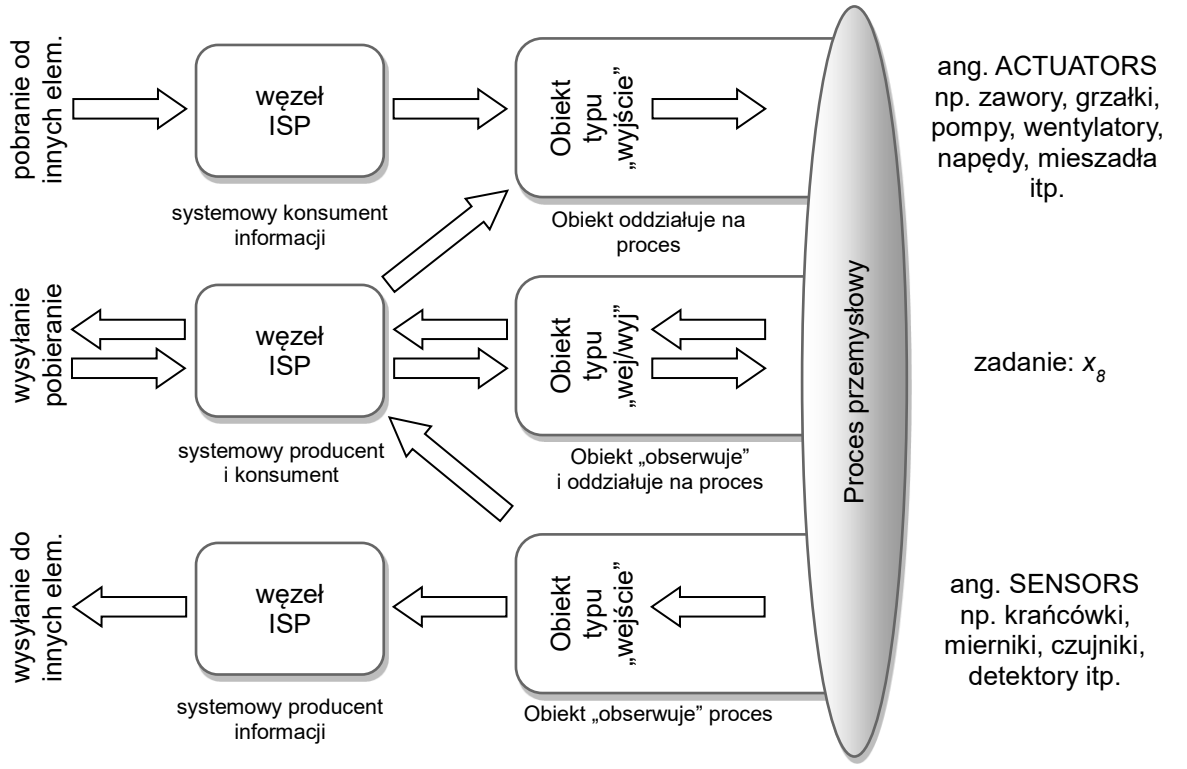


Rys. 54. Model obiektu przemysłowego
Fig. 54. The model of an industrial object

W skład obiektu wchodzą elementy bierne, które nie wpływają na działanie systemu oraz uczestniczące w przepływie informacji elementy czynne, w postaci układów pomiarowych (sensorowych) i wykonawczych. Na obsługę procesu przemysłowego może składać się wiele obiektów jak również na dany obiekt może się składać wiele obiektów podrzędnych. Dla przykładu, dla procesu destylacji ropy naftowej obiektem może być cała kolumna destylacyjna, jak również obiektami mogą być podzespoły tej kolumny, a nawet poszczególne zawory, czujniki, grzałki i inne elementy automatyki wchodzące w skład tych podzespołów.

W skrajnym przypadku cały zbiór elementów automatyki obsługujących dany proces stanowić będzie obiekt przemysłowy, jak to przedstawiono na rysunku 54 oraz rysunku 20.

Dany obiekt przemysłowy ma **opis informacyjny** reprezentowalny i podlegający obsłudze w ISP. Obiekt dostarcza zbioru informacji do systemu komputerowego lub pobiera zbiór informacji z systemu komputerowego przy użyciu jakiegoś interfejsu lub grupy interfejsów. Rodzaj takiego interfejsu może być analogowy lub cyfrowy, a informacja dotyczy wielkości fizycznych procesu podlegających kontroli. Wielkościami tymi mogą być ciśnienie, temperatura, przepływ, kwasowość lub inne związane z procesem, w tym również wielkości dyskretne określające stan obiektów typu otwarty, zamknięty, załączony, wyłączony, przekroczony poziom itp.



Rys. 55. Przekazywanie informacji między elementami systemu
Fig. 55. Information transfer between system elements

Informacja taka z punktu widzenia informatycznego jest **produkowana** (wytwarzana, wysyłana) i/lub **konsumowana** (pobierana, odczytywana) przez obiekt przemysłowy. Informacja obsługiwana przez ISP jest również pobierana i wysyłana z punktu widzenia wymiany informacji między jego komponentami. Elementy obiektu przemysłowego mogą być fizycznie zintegrowane z komponentami systemu informatycznego lub połączone zdalnie przy użyciu jakiegoś interfejsu. Informacja ze zbioru związanego z obiektem może być zatem reprezentowana przez zbiór zmiennych w komponencie systemowym, a jej wymiana podlegać systemowym regułom komunikacji. Dlatego opis informacyjny obiektu, oprócz definicji zna-

czenia i struktury danych procesowych, musi zawierać jeszcze definicje ich powiązań wraz z zależnościami semantycznymi oraz czasowymi wynikającymi z technologii obsługiwanego procesu. Prowadzi to do możliwości stworzenia odpowiednich algorytmów przetwarzania, a w konsekwencji programów realizujących takie algorytmy, i aplikacji wypełniających zadania funkcjonalne danego ISP.

Elementami składowymi obiektów przemysłowych może być różnego typu aparatura kontrolno-pomiarowa (AKP), w tym sensory i urządzenia wykonawcze, jak również urządzenia przetwarzające typu sterownik (ang. controller) czy komputer. Przyjęło się, że elementy stanowiące różnego typu urządzenia, które mogą uczestniczyć w przetwarzaniu i wymianie informacji, są nazywane komponentami „inteligentnymi". Komponenty pobierające informację o stanie procesu realizują funkcję „wejścia" informacji do systemu, natomiast obiekty modyfikujące stan procesu realizują „wyjście" z systemu.

Prawie zawsze komponenty systemowe mają charakter układów mieszanych stanowiących zarówno wejście, jak i wyjście. Przedstawione to zostało na rysunku 55.

2.4.2. Węzeł systemu

Rzeczywiste przetwarzanie informacji odbywa się w węźle systemu. Jest to najczęściej komputer z interfejsami dostosowanymi dla obiektu przemysłowego oraz z systemem operacyjnym umożliwiającym realizacje zadań w czasie rzeczywistym. Niektóre komputery tego typu mają również interfejsy typowe dla człowieka typu klawiatura, monitor itp. Należy jednak postrzegać węzeł ISP jako komputer dedykowany dla obsługi obiektów przemysłowych. Podstawowymi zadaniami węzła z punktu widzenia informatycznego jest przetwarzanie i przesyłanie informacji. Z punktu widzenia automatyki węzeł może aczkolwiek nie musi stanowić źródło sterowania (regulacji) dla związanego z nim obiektu. Węzły można postrzegać, biorąc pod uwagę ich typ, budowę oraz działanie.

❖ **Typy**

W ISP można wyróżnić dwa typy węzłów:

- procesowe – węzły wykonujące przetwarzanie na poziomie procesu (ang. field level, device level) i realizujące wymiany sieciowe poziome.

  Są to komputery, które zajmują się realizacją lokalnych zadań funkcjonalnych systemu. PLC i inne komputery służące do sterowania znajdują się najczęściej na tym poziomie abstrakcji. Wymiana pozioma informacji oraz działanie węzła muszą mieć charakter czasowy, zgodny z wymaganymi ograniczeniami czasowymi względem danego systemu sterowania.

- interfejsowe – węzły zapewniające interfejs systemu zrozumiały dla użytkownika systemu, jakim może być człowiek lub inne systemy (M2M, ang. Men to Machine, Machine to Machine). Węzeł uczestniczy w wymianach poziomych, ale realizuje również wymiany sieciowe pionowe.

  Interfejsy dla człowieka budowano od dawna. Są niezbędne, aby człowiek mógł zrozumieć, co się dzieje w systemie oraz mógł w intuicyjny sposób wpływać na to działanie. Pierwsze stanowiły układy synoptyczne. Obecnie najczęściej interfejs jest budowany na bazie ekranów graficznych implementowanych z użyciem komputerów klasy PC lub panelowych, choć zdarzają się inne (węzły interfejsu użytkownika). Typowymi rozwiązaniami interfejsowymi dla człowieka są wizualizacyjne stacje robocze z interfejsem multimedialnym (obraz, animacja, dźwięk). Spotyka się również rozwiązania tekstowe, np. w prostych urządzeniach mobilnych, przy korzystaniu z usług pocztowych czy ogólnie przy niewielkich zasobach węzła. Popularne są też nadal rozwiązania interfejsów synoptycznych, które są szczególnie przydatne blisko obiektu.

  Węzły interfejsowe (integracyjne, dostępowe) dla innych systemów i urządzeń automatyki oraz do nadrzędnych systemów informatycznych typu MES, ERP itp. (zob. 3.2.9, 3.2.10) występują coraz częściej w nowoczesnych ISP. Wynika to ze zwiększającej się potrzeby integracji systemów lokalnych oraz z chęci zapewniania dostępu zdalnego. Aby przekazywać informację z systemu lokalnego do systemu zewnętrznego, informacja musi być przekazana poza lokalny stos protokołów zaangażowany w działanie systemu lokalnego. Dlatego wymiany tego rodzaju nazywane są pionowymi.

Węzły muszą pracować w rygorze czasu rzeczywistego, o ile jest wymagany, i wszystkie muszą podlegać żądanym ograniczeniom. Jednak, gdy użytkownikiem jest człowiek, wymóg pracy w trybie HRT nie jest wymagany. Człowiek z natury nie działa w trybie HRT i na potrzeby realizacji podsystemu HMI wystarczy tryb FRT lub nawet SRT, gdy ograniczenia czasowe w systemie nie są ostre.

❖ Budowa

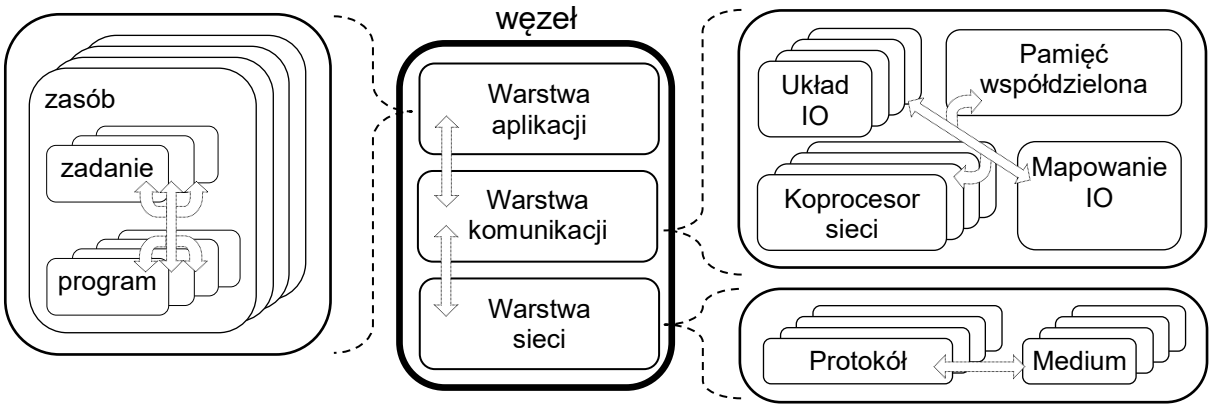
Interfejsy komunikacyjne węzła służące do wprowadzania i wyprowadzania informacji należy postrzegać w dwojaki sposób. Należy oddzielić interfejsy pośredniczące z obiektem przemysłowym od interfejsów pośredniczących pomiędzy jednostkami komputerowymi. Przy pozyskiwaniu informacji z obiektu przemysłowego istnieje potrzeba konwersji wielkości fizycznej pochodzącej z procesu do postaci cyfrowej niezbędnej po stronie komputera. Konwersja taka może być wykonywana przez układy elektroniczne obiektu przemysłowego, o ile jest w takowe wyposażony, lub przez układy samego węzła. Wymiana informacji pomiędzy węzłami zachodzi zwykle przy użyciu sieci komputerowych.

W celu realizacji zadań lokalnych oraz zadań komunikacyjnych każdy węzeł ISP musi składać się z co najmniej trzech abstrakcyjnych warstw współpracujących ze sobą. Są one urzeczywistniane zarówno w sprzęcie, jak i w oprogramowaniu.

- Warstwa aplikacji
  Część sprzętu i oprogramowania zajmująca się realizacją lokalnych zadań funkcjonalnych węzła. W aplikacji może być jeden lub wiele zasobów w postaci układów będących w stanie dokonać przetwarzania. Na zasób składają się procesory, pamięci, stos, sterta i inne elementy umożliwiające wykonanie programu. W takim rozumieniu zasób jest jednostką przetwarzającą (por. 2.6). Z zasobem mogą być zdefiniowane zadania, które są realizowane przez wiele programów. W skrajnym przypadku zadanie może być jedno i polegać na cyklicznym lub zdarzeniowym uruchamianiu programu.

- Warstwa komunikacji
  Część sprzętu i oprogramowania zajmująca się aktualizacją wektora stanu wejścia i wyjścia węzła. Następuje tu obsługa układów dostarczających dane oraz zapis wartości do i odczyt wartości z odpowiednich lokacji pamięci współdzielonej z zasobami warstwy aplikacji. W warstwie komunikacyjnej może pracować jeden lub wiele układów IO i koprocesorów sieci.

- Warstwa sieci
  Część sprzętu i oprogramowania zajmująca się realizacją wymian sieciowych między węzłami. Jest to warstwa odpowiedzialna za dialog z innymi węzłami.



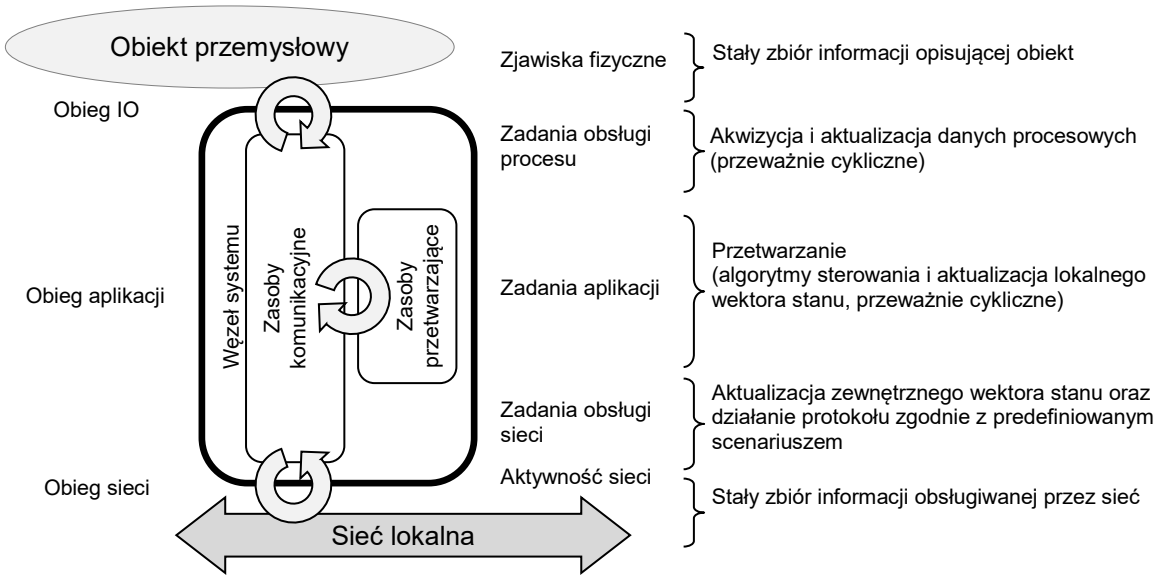
Rys. 56. Ogólny model węzła ISP
Fig. 56. General model of ICS node

Funkcje układów przetwarzających, IO i koprocesorów sieci mogą też pełnić specjalne zadania programowe systemu operacyjnego, o ile jego konstrukcja i moc obliczeniowa węzła pozwalają na zachowanie cechy punktualności. Ze względów niezawodnościowych wskazane jest, aby zapewnić niezależność funkcyjną warstw. Celem jest uniknięcie sytuacji awarii ca-

łego węzła przy awarii w którejś z warstw. Aby to osiągnąć, układy realizujące poszczególne funkcje w warstwach muszą być fizycznie odrębne (np. procesor i koprocesor) lub system operacyjny zarządzający taką realizacją w węźle musi zapewniać zdeterminowane wywłaszczanie zadań systemowych. Najczęściej są to systemy operacyjne czasu rzeczywistego. Może też być użyty dedykowany firmware, o ile węzeł nie wymaga systemu operacyjnego. Ogólny model węzła przedstawiono na rysunku 56.

❖ **Działanie**

Ze względu na konieczność pracy węzła w czasie rzeczywistym istnieje potrzeba stałego nadążania za zmianami występującymi w świecie zewnętrznym. Istnieją przynajmniej dwa sposoby na to, a mianowicie wykonywanie działań w przerwaniach lub w sposób cykliczny. Liczba zdarzeń wymaganych do obsługi w czasie rzeczywistym jest stosunkowo duża. Są to zdarzenia od aktywności poszczególnych układów w warstwach węzła. Przy dużej zmienności występują trudności w zapewnieniu limitów czasowych ich obsługi przy pracy w przerwaniach. Dlatego częściej spotyka się rozwiązanie z obsługą cykliczną. Polega ono na wykonywaniu kluczowych akcji, takich jak akwizycja danych, wykonanie programów, oddziaływania zewnętrzne itp. z określonym cyklem. Jest to rodzaj odpytywania czy omiatania elementów węzła (ang. pooling, sweep).



Rys. 57. Podstawowe obiegi informacyjne węzła
Fig. 57. Fundamental information circulations of the node

Generalnie, akcje wykonywane w ramach danej warstwy są niezależne względem wykonywania akcji w innych warstwach. Działania w warstwach mogą być jednak zsynchronizowane. Rodzaj współpracy zależy od wymagań czasowych względem węzła, a konkretnie rozproszonych aplikacji względem siebie. W efekcie uzyskuje się działanie, w którym istnieją

przynajmniej trzy obiegi informacji. Jeden wynikający z działania jednostki przetwarzającej, drugi wynikający z działania układów IO obsługujących obiekt, a trzeci z obsługi sieci. W ISP obiegi te mają najczęściej charakter cykliczny, choć nie jest to zasada. Obraz świata zewnętrznego, dostępny dla węzła, zawsze oddaje tylko rzeczywisty obraz w pewnych momentach czasu. Poza tymi momentami obiekt i system jest nieobserwowany. Na rysunku 57 [125] przedstawiono obiegi związane z węzłem oraz podstawowe zadania z nimi związane. W przypadku PLC obiegi IO i aplikacji są często zsynchronizowane w cyklu sterownika (por. 3.1.3), a obieg sieciowy jest niezależny. Zależy to jednak od działania danego modelu PLC.

W przypadku niezależnego obiegu sieciowego warstwa komunikacji pozyskuje dane ze świata zewnętrznego, zgodnie z predefiniowanym scenariuszem, według jakiego została skonfigurowana. Dla przykładu koprocesor sieciowy, lub specjalne zadanie systemu operacyjnego węzła, realizuje transakcje według scenariusza wymian, a dane użyteczne uaktualnia w pamięci współdzielonej. Równolegle jednostka przetwarzająca wykonuje programy, operując na swoich pamięciach oraz na pamięci współdzielonej. W efekcie mogą się pojawić zjawiska:

- nadpisywania danych
  Gdy obieg informacji dostarczający dane od innych węzłów działa szybciej niż obieg wynikający z aplikacji, dojdzie do sytuacji, że dane zapisane do pamięci współdzielonej nie zdążą zostać odczytane. Problem występuje też w drugą stronę, jeśli aplikacja zaktualizuje dane, a komunikacja ze względu na ograniczenia protokołu, prędkości transmisji itp. nie zdoła wytransmitować tej danej przed jej kolejną aktualizacją. Jeśli to zjawisko wynika z cykliczności obiegów informacyjnych, to rozwiązaniem jest ich synchronizacja, jeśli zjawisko zachodzi przy akcjach acyklicznych, to rozwiązaniem jest buforowanie. Zastosowanie buforowania do akcji cyklicznej będzie się zawsze wiązało z przepełnieniem bufora, natomiast synchronizacja dla akcji acyklicznych zmniejszy wydajność węzła. W praktyce w PLC bardzo rzadko stosuje się synchronizację aplikacji z cyklem sieci, a bardzo często synchronizację z układami IO.

- przeterminowania danych
  Zjawisko występuje, gdy na poziomie warstwy aplikacji istnieje ograniczenie czasowe na wartość danej. Ograniczenie takie często nazywa się czasem ważności, czasem życia lub świeżością zmiennej. Zakładając opisaną wcześniej niezależność układów, wysyłanie zewnętrznego wektora stanu węzła jest realizowane zgodnie ze scenariuszem komunikacji, natomiast aplikacja aktualizuje wektor zgodnie ze swoim obiegiem przetwarzania. W sytuacji, gdy z daną w wektorze związany jest czas ważności, może zaistnieć sytuacja, że aplikacja nie zdąży wytworzyć nowej wartości w czasie

wymaganym przez ten parametr. Wówczas niezależna warstwa komunikacji obsługuje daną z wartością przeterminowaną. Istotne jest zauważenie, że z punktu widzenia komunikacji nie występują błędy i działanie warstwy jest poprawne. Błąd jest na poziomie warstwy aplikacji. W praktyce takie zjawisko można zaobserwować, gdy czas wykonania programu PLC jest zbyt długi lub gdy program ulegnie awarii, np. wpadnięcie w nieskończoną iterację (popularne tzw. zawieszenie). PLC mają mechanizmy zabezpieczające przed taką ewentualnością (np. watchdog, por. 3.1.3), ale muszą one zostać użyte i odpowiednio skonfigurowane. Przykładowo, czas graniczny wartownika dla zadania powinien mieć wartość poniżej najkrótszego okresu ważności zmiennych obsługiwanych przez to zadanie.

- obsługi danych nieważnych
  Zbliżone do powyższego zjawisko zachodzi dla danych, które nigdy nie zostały zainicjowane przez warstwę aplikacji. Takie dane mogą zostać obsłużone przez warstwę komunikacji, gdyż z punktu widzenia komunikacji działanie jest możliwe i konieczne. Wstrzymanie działania komunikacji mogłoby zostać odebrane przez inne układy jako awaria. Przed pierwszą aktualizacją zewnętrzny wektor stanu węzła będzie zawierał wartości domyślne (np. zera) lub nieustalone, zależnie od sprzętu i systemu operacyjnego. Dla innych węzłów lub obiektów ich wartość nie ma związku z rzeczywistym stanem i jest nieprzydatna.

Zbudowanie zabezpieczeń przed powyższymi problemami może się okazać niemożliwe na konkretnej platformie sprzętowo-programowej. Możliwe jest z reguły stworzenie mechanizmów statusowych informujących o wystąpieniu powyższych zjawisk. Najczęściej mówi się tu o statusach jakości danych użytecznych, dotyczących świeżości i ważności danych (ang. refreshment status, validity status itp.).

W przypadku pracy synchronicznej opisane powyżej zjawiska nie występują. Wówczas jednak urządzenie musi mieć wystarczającą moc obliczeniową, aby nadążać w czasie rzeczywistym za wszystkimi zdarzeniami pojawiającymi się w systemie, a także aby praca danego układu nie wstrzymywała pracy innych układów węzła.

W praktyce, w większości przypadków PLC zadania aplikacji i IO są niezależne względem zadań sieci. Więcej na temat przepływu danych znajduje się w 1.2.2, a więcej na temat typowego węzła systemu, jakim jest PLC, w [196].

2.4.3. System

Pojęcie systemu informatycznego zostało zdefiniowane w 1.1.5, model ISP w 2.2 i 2.3, a model jego węzła w 2.4.2. W normie IEC61131 (2.6) pojęcie systemu występuje w dwóch kontekstach. Jeden to system automatyki, który jest pojęciem szerokim i dotyczy różnych

urządzeń, w tym PLC, wraz z ich programami aplikacyjnymi. Drugi to tzw. z ang. PLC-system, który jest rozumiany jako zestaw urządzeń dobrany przez użytkownika (projektanta systemu), a składający się ze sterownika i jego układów peryferyjnych. Układ taki przeznaczony jest do pracy w systemie automatyki. Jego elementy łączone są kablami i złączami na stałe do pracy w stacjonarnym lub mobilnym systemie automatyki.

Ewentualne problemy pojęciowe związane z abstrakcją systemu mogą wystąpić z postrzeganiem przepływu informacji pomiędzy obiektem przemysłowym i węzłami oraz z określeniem zasięgu działania systemu.

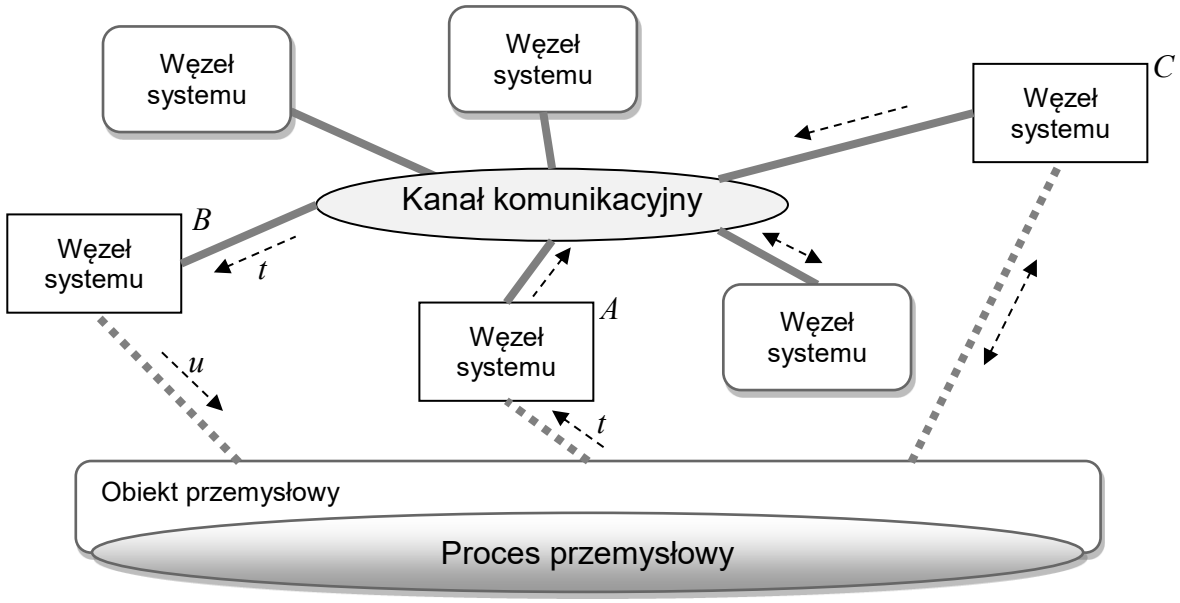
❖ **Przepływ**

Głównym celem istnienia systemów informatycznych jest przetwarzanie informacji na rzecz użytkownika. Główne zadania ISP, patrząc ogólnie od strony informatycznej, polegają na:

- tworzeniu cyfrowych reprezentacji informacji, dotyczących stanu procesu i systemu,
- przetwarzaniu tej informacji, zgodnie z postawionymi wymaganiami,
- przesyłaniu wymaganej informacji pomiędzy węzłami systemu,
- składowaniu wymaganej informacji w węzłach,
- interakcji z użytkownikiem.

Określenie użytkownika dla ISP jest umowne, ale najprościej można wskazać na człowieka. Można też powiedzieć, że użytkownikiem ISP jest obiekt przemysłowy, gdyż to z i do procesu krąży większość informacji, na podstawie której dokonuje się przetwarzanie. Człowiek pełni tu tylko role sprawczą i nadzorczą. Dostarcza on informacji parametryzującej przebieg procesu, a pobiera informację umożliwiającą jego śledzenie. Informacja musi być w sposób zrozumiały dla użytych technologii informatycznych pozyskana, zakodowana i dostarczona do jednostek przetwarzających. Dlatego węzły systemu z jednej strony wchodzą w interakcję z obiektem przemysłowym, ale z drugiej, aby przetwarzanie rozproszone mogło mieć miejsce, muszą wymieniać informację między sobą. Zatem, z punktu widzenia informatyki, model przedstawionego wcześniej układu regulacji (por. 1.2.4, rys. 19) można przedstawić trochę inaczej, uwypuklając zależności między węzłami, a nie interakcję z procesem. Przepływ informacji inaczej wygląda od strony technologicznej, a inaczej od strony systemu informatycznego. Jest to zilustrowane jako rozproszone zadanie regulacji na rysunku 58. Węzły, które mają kontakt z obiektem przemysłowym (*A*, *B*, *C*), pozyskują i wyprowadzają informację z/do obiektu, a ponadto wymieniają informacje z innymi węzłami. Kierunek przekazywania informacji (przerywane strzałki) zależy od punktu widzenia i jest częstym powodem nieporozumień w ustaleniach założeń działania systemu.

Informacja o temperaturze *t* w zbiorniku jest dostarczana z procesu do urządzenia pomiarowego w węźle *A*, a dalej w funkcji wartości tej temperatury węzeł *B* systemu oddziałuje przez u na pracę grzałki na obiekcie. Zmienna reprezentująca chwilową wartość temperatury musi zostać przekazana z węzła A do węzła *B*. Od strony procesu informacja w węźle *A* jest konsumowana, a w węźle *B* produkowana, natomiast od strony systemu w węźle *A* jest produkowana, a w *B* konsumowana.



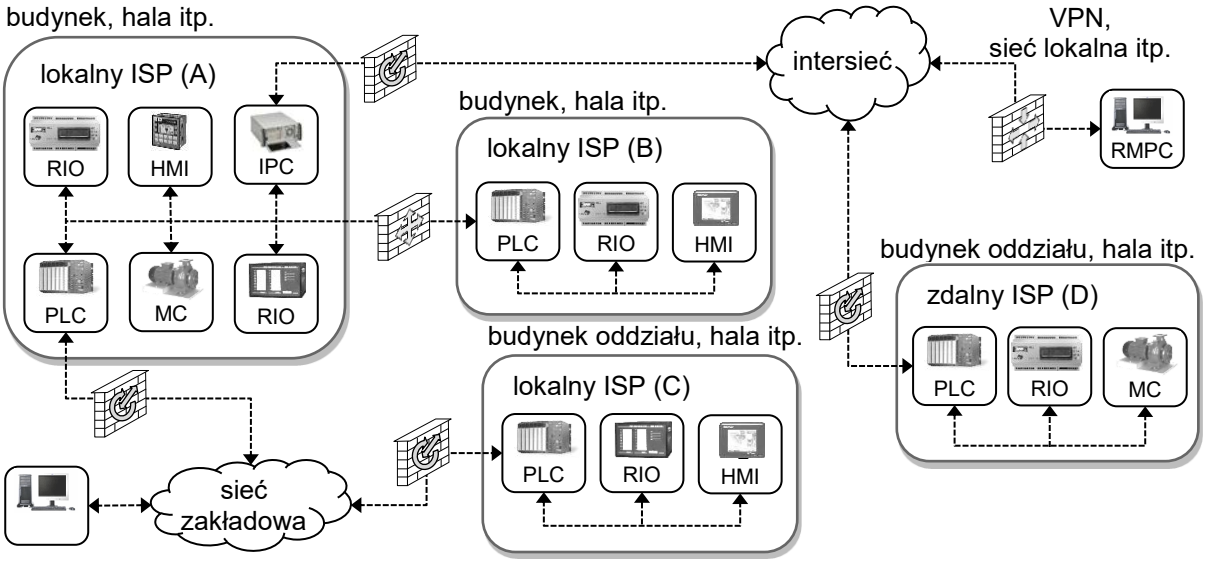
Rys. 58. Ogólny schemat przepływu danych między węzłami ISP
Fig. 58. General schema of data flow between ICS nodes

W rozważaniach dotyczących informatycznych systemów przemysłowych należy przyjąć punkt widzenia systemu i zachodzącej w nim komunikacji między węzłami. Z punktu widzenia przepływu informacji w systemie można abstrahować od zadań funkcjonalnych obiektu przemysłowego i potraktować go jako urządzenie zewnętrzne, stanowiące źródło lub odbiorcę informacji, podłączone do układów wejścia/wyjścia komputera. Przepływy i zasady oddziaływania informacji z procesem powinny pozostać domeną technologów i automatyków po stronie obiektu przemysłowego.

❖ Zasięg

System lokalny ma ograniczony zasięg działania. Zasięg ten wynika z terytorialnej rozległości obiektu oraz z możliwości funkcjonowania sieci lokalnych. Zilustrowano to na rysunku 59. Przedstawione typy węzłów są przykładowe. W przypadku integracji oddalonych systemów lokalnych tworzony jest tzw. system wyspowy (por. 3.5). System taki może integrować systemy lokalne zlokalizowane w dowolnych miejscach, np. osobnych budynkach, oddziałach, miastach, ale również w odległościach niewielkich, np. kilka wysp w jednej hali

produkcyjnej. Technologia użyta do integracji zależy od odległości i dostępnej infrastruktury. Najczęściej są to sieci komputerowe rozległe lub łącza teletransmisyjne. Dla przykładu, z rysunku A i B tworzą dwie wyspy lokalne integrowane lokalną siecią przemysłową, system C może być niezależnym ISP lub wyspą dołączoną zakładową siecią lokalną (tzw. ang. intranet), natomiast system D jest zdalnym ISP wymieniającym dane przez rozległą intersieć (ang. internet), np. przez dobrze znany Internet. W przypadku tymczasowego dołączania urządzeń spoza systemu lokalnego system musi zapewnić interfejs umożliwiający zdalny dostęp. W obu przypadkach wskazana jest kontrola przepływu danych w celu utrzymania istniejącego poziomu niezawodności systemu lokalnego [158], [247]. Do infrastruktury kontrolującej wymianę danych pomiędzy systemami zalicza się standardowe urządzenia, wynikające z przyjętych technik komunikacyjnych. Ich opis znajduje się w 3.3 oraz w wielu pozycjach literatury, np. [128], [321]. W przykładzie z rysunku 59 przedstawiono router dla intersieci, gateway dla sieci przemysłowej i dla wszelkich integracji z ISP. Wszystkie urządzenia mają funkcjonalności ściany ogniowej, choć ich specyfika jest zależna od lokalizacji.



Rys. 59. Zasięg działania i współdziałania ISP
Fig. 59. The ICS operational and cooperation scope

2.4.4. Sieć

Jak wspomniano w poprzednich podrozdziałach, sieć komputerowa stanowi podstawowy środek stosowany do wymiany informacji w systemach informatyki przemysłowej. Rozpatrywanie innych możliwości, np. kanałów analogowych, nie ma większego uzasadnienia praktycznego. Przez sieć komputerową rozumie się ogół środków i metod potrzebnych do zrealizowania komunikacji pomiędzy komputerami (abonentami). Rozpatrywanie sieci dla liczby abonentów mniejszej niż trzy jest trywialne i w dalszych rozważaniach nie jest to bra-

ne pod uwagę. Mowa tu oczywiście o liczbie abonentów w sieci, a nie o sposobie podłączenia (topologii) dwóch abonentów typu point to point.

Infrastruktura sieciowa zakładów i jej eksploatacja na potrzeby różnego rodzaju systemów informatycznych jest przeważnie dość złożona (por. 1.2.2). Jednak w odniesieniu do ISP wykorzystuje się tylko wydzielone sieci, specjalnie przeznaczone dla najniższego poziomu wymiany informacji, gdzie pracują węzły procesowe [1, 8 iem]. Nie jest to „klasyczna" sieć lokalna, niejednego rodzaju i niekoniecznie jedna. Ze względu na wymagania obiektu przemysłowego wymiana danych w sieci często musi być zdeterminowana w czasie, a ze względu na środowisko pracy jej działanie musi się cechować zwiększoną niezawodnością względem sieci biurowych. Dlatego niezbędne jest stosowanie specjalnych protokołów [220], [119], [385] odpowiednich łączy (warstwy fizyczna i łącza ISO/OSI, ang. link, physical layer, MAC – Media Access Control) i infrastruktury [384] zapewniających taki charakter pracy sieci. Dla tego typu sieci komputerowych powszechnie używa się pojęcia „sieci przemysłowe" (ang. industrial networks), choć bez kontekstu „komputerowe" pojęcie to może być niepoprawnie rozumiane przez inżynierów innych dziedzin (np. sieci energetyczne, gazowe, wodociągowe itp.). Do sieci przemysłowych często zalicza się również rozwiązania, które znalazły zastosowanie w systemach motoryzacyjnych (ang. automotive). Nie są to rozwiązania popularne w ISP, ale ze względu na istniejące mechanizmy obsługi ograniczeń czasowych nadają się one do zastosowań w systemach przemysłowych i często z takich systemów się wywodzą [6823162]. Z rozwiązań popularnych w obu dziedzinach zastosowań najszerzej wykorzystywane jest rozwiązanie z użyciem sieci CAN w warstwach niższych oraz protokołu CANopen w warstwie aplikacji [74]. Ponadto, istnieją jeszcze rozwiązania sieciowe wspierające obsługę pakietów w czasie rzeczywistym, które są dedykowane do obsługi multimediów. Rozwiązania te nie znajdują jednak miejsca w ISP.

Niniejsza książka nie zawiera treści szczegółowo opisujących komputerowe sieci przemysłowe. Są to zagadnienia bardzo szerokie i dość złożone. Na potrzeby opisu wykorzystania PLC w systemach przemysłowych wystarczy poznać cechy abstrakcyjnej warstwy wymiany informacji i zrozumieć jej funkcjonowanie. Jest to niezbędne do poprawnego projektowania systemu oraz programowania PLC. Mowa tu o zapewnieniu właściwej współpracy aplikacji pomiędzy węzłami. Więcej na ten temat znajduje się w rozdziale 2.4.5. Z punktu widzenia programowania węzłów nie jest potrzebna szczegółowa wiedza o danej technologii sieciowej. Przydaje się wiedza „kursowa" dotycząca konfiguracji i ogólnej idei działania. Dobrym podejściem jest traktowanie komunikacji jako abstrakcji z usługami sieciowymi, które można zapewnić przy użyciu różnych technik. Technologie sieci przemysłowych są dobrze rozwinięte oraz całkiem nieźle przebadane i udokumentowane [385], [415], [253], [126], [121]. Ponadto, komunikacja dla tego typu zastosowań objęta jest w większości przypadków standa-

ryzacją [M27], [M39]. Klasyczne rozwiązania stanowią sieci polowe (miejscowe, systemowe, ang. fieldbus) oraz bardziej uniwersalne sieci klasy RTE (Ethernet czasu rzeczywistego, przemysłowy, ang. Real-Time Ethernet).

❖ **Generacje**

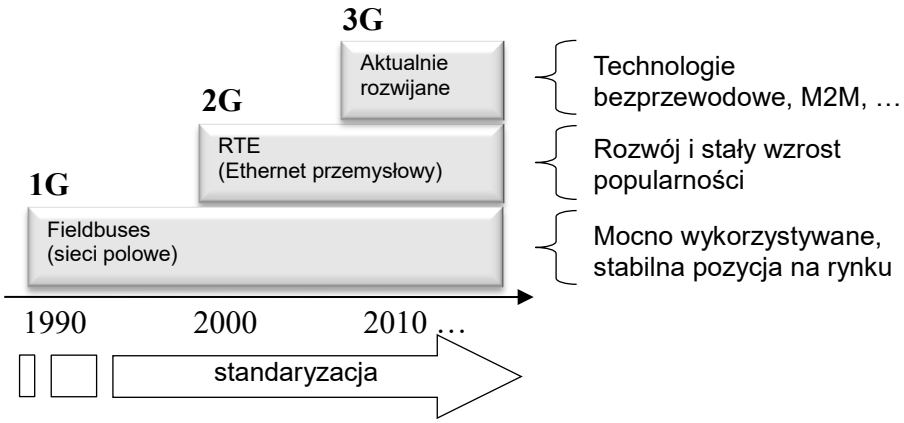
Sieci polowe stanowią tzw. pierwszą generację sieci przemysłowych (1G). Poza specjalnymi zastosowaniami [396], [325], [137], [236], [191], [150] nie są one aktualnie rozwijane. Są one natomiast bardzo popularne i szeroko stosowane. Ponadto, pomimo głosów o nieuchronnym zmierzchu technik polowych wynikających prawdopodobnie z fascynacji nowymi technologiami, będą one jeszcze długo wykorzystywane. Bierze się to z faktu, że następna generacja została wprowadzona nie po to, by zastąpić 1G, tylko aby umożliwić realizacje funkcjonalności niedostępnych dla sieci polowych, a wymaganych w nowoczesnych systemach [125]. Istnieje cały szereg obiektów, dla których ze względów ekonomicznych i niewielkich wymagań procesu opłaca się nadal instalować sieci polowe. Najczęściej używaną topologią w sieciach polowych jest magistrala, choć spotyka się również pierścienie, łańcuchy i inne.

Sieci RTE są drugą generacją sieci przemysłowych (2G) i stanowią nowoczesną platformę dla komunikacji w ISP. Ze względu na ciągły rozwój, jawią się jako obiecująca technologia na przyszłość. Intensywne prace nad technologiami 2G rozpoczęto w połowie pierwszej dekady XX wieku i prace te ciągle trwają. Sieci RTE dzięki parametrom łącza oraz specjalnym protokołom determinującym w czasie dostęp do medium umożliwiają uzyskanie dużych przepustowości z zachowaniem ograniczeń czasowych. Zależnie od technologii ograniczenia mogą mieć charakter ostry lub łagodny (por. 1.1.4). Możliwe jest również realizowanie usług, które w sieciach 1G nie były możliwe, jak np. transmisje multimedialne czy klasyczny ruch TCP/IP. Topologia używana w sieciach RTE wynika ze standardu Ethernet, a zatem nie wprowadza dużych ograniczeń.

Istnieją jeszcze sieci trzeciej generacji (3G), które są w fazie projektów, badań i dedykowanych rozwiązań. Są to przemysłowe sieci bezprzewodowe o bliskim zasięgu, sieci komórkowe czy też techniki multisieciowe i sieci wirtualne (por. 3.3.5) [125], [320], [67], [360]. Sieci takie są wykorzystywane zarówno do komunikacji w lokalnych ISP, jak i do realizacji dostępu zdalnego, komunikacji typu M2M, komunikacji serwisowej, czy też w nowoczesnych systemach samochodowych (ang. vehicles communication). Standaryzacja w tym zakresie jest na etapie tworzenia (np. normy dla rozwiązań bezprzewodowych IEC62657 [M18], [M13], IEC62734 [M53]).

Podział generacji przedstawiono na rysunku 60 [125], datowanie jest orientacyjne. W tabeli 4 zamieszczono ogólną historię sieci przemysłowych, z uwzględnieniem sieci dla systemów samochodowych.

Dobór rozwiązań sieciowych jest kluczowy z punktu widzenia projektowania ISP. Generację sieci oraz konkretną technikę w ramach generacji należy dobrać kierując się wskazówkami przedstawionymi np. w [118], [79]. Zasada podstawowa to wybór takiego rozwiązania, które spełni wymagania aplikacyjne ISP, nie będzie ograniczać dostępnej bazy sprzętowo-programowej i rozwoju danego systemu. Nie zawsze sieci najbardziej zaawansowane technologicznie są najlepszym wyborem. Oprócz wymagań aplikacyjnych należy zawsze brać pod uwagę aspekty ekonomiczne, utrzymania i rozwoju oraz preferencje użytkownika, przy czym, dla ogólnego rozwoju dziedziny i postępu w sferze aplikacji, wskazane jest przedkładać rozwiązania nowoczesne nad rozwiązania przestarzałe i sztampowo powielane.



Rys. 60. Ilustracja generacji sieci przemysłowych
Fig. 60. Illustration of the generation of industrial networks

Tabela 4

Historia sieci przemysłowych w ogólnym zarysie

Lata	*Prowadzone prace*	*Przykładowe rozwiązania*
1970 - 1985	Pierwsze prace nad komunikacją sieciową dla przemysłu. Rozwiązania bazujące na standardowych łączach szeregowych (np. RS485, RS422).	Różne rozwiązania firmowe
1986 - 1990	Wyróżniono różne standardy firmowe i rozwiązania wspierane w najbardziej rozwiniętych w dziedzinie sieci przemysłowych krajach. Sformułowano pierwsze założenia działania sieci polowych. Pierwsze rozwiązania dla systemów motoryzacyjnych.	Profibus FIP CAN
1990 - 1994	Tak zwana wojna technologii sieci polowych, głównie niemiecko-francuska. Próby utworzenia specyfikacji ogólnych bazujących na WorldFIP i Interoperable System Project. Początki niekomercyjnych organizacji wspierających dane technologie. Początki standaryzacji.	Profibus WorldFIP TTP LIN
1995 - 1998	Utworzenie amerykańskiego standardu Foundation Fieldbus oraz europejskiego standardu pod szyldem CENELEC grupującego różne rozwiązania. Międzynarodowa standaryzacja utyka jednak w martwym punkcie.	Jw. plus: FF, Interbus, P-Net, ASI

cd. tabeli 4

Lata	*Prowadzone prace*	*Przykładowe rozwiązania*
1999 - 2000	Ustanowiono kompromis w standaryzacji przez wybranie ośmiu specyfikacji technologii sieci przemysłowych i utworzeniu norm je uwzględniających.	Jw. plus: ControlNet, DeviceNet, SDS, CANOpen, …
2000 - 2002	Rozwój i poprawki rozwiązań standardowych. Rozwiązania ustandaryzowane są umacniane przez nowe typy i profile określone w normie IEC 61784.	Jw.
2000 - teraz	Poszukiwania nowych rozwiązań dla motoryzacji i rozwój starych.	TTCAN, TCN, VAN, FlexRay, …
2000 - teraz	Rozwój sieci klasy RTE.	Profinet, EtherCAT, EPL, …
2008 - teraz	Rozwój sieci trzeciej generacji.	–

❖ **Wymagania**

W rozproszonych systemach lokalnych z ograniczeniami czasowymi przepływ informacji musi być zdeterminowany czasowo, a działanie musi wykazywać się cechą podwyższonej niezawodności, trwałości i spójności informacyjnej. Dotyczy to zarówno przetwarzania w węzłach, jak i w komunikacji między węzłami. Dlatego do transmisji informacji wykorzystuje się tylko te sieci komputerowe, które umożliwiają:

- tworzenie systemów rozproszonych terytorialnie
  Zasięg rozproszenia terytorialnego determinowany jest przez dane media i łącze. Typowy zasięg działania sieci polowych waha się od kilkudziesięciu do 1000 metrów segmentu sieci. Należy jednak brać pod uwagę wiele ograniczeń fizycznych rozwiązań magistralowych opartych na kablach miedzianych. Typowe są: limit rozgałęzień (podłączeń abonentów), minimalne odległości między abonentami, zapewnienie właściwej impedancji oraz lokalizacja terminatorów itp.

  Długość segmentu RTE w technologii 100Base-TX 5e STP wynosi 100 m. Stosowanie techniki przełączania (ang. swiching) umożliwia wydłużenie segmentu, jednak nie jest możliwa rozbudowa sieci o dowolną liczbę przełączników (ang. switch) (zob. 3.3.3).

- wykorzystanie protokołów determinujących w czasie dostęp do medium
  Determinizm czasowy działania sieci wynika z protokołu warstwy łącza lub aplikacji ISO/OSI. Musi być to protokół kontrolujący dostęp do medium, a dodatkowo umożliwiający zdefiniowanie zależności komunikacyjnych pomiędzy aplikacjami. Najczęściej zależności takie buduje się na podstawie predefiniowanego globalnego

scenariusza wymian (ang. time schema, scenario). Scenariusz definiowany jest na poziomie warstwy aplikacji. Dany węzeł może dysponować wieloma interfejsami sieciowymi, ale dla sieci przemysłowych na jednym segmencie może pracować tylko jeden protokół lub musi istnieć nadrzędny mechanizm, który zapewni kontrolę dostępu do współdzielonego medium. W przeciwnym razie protokoły utracą cechę przewidywalności działania (por. 2.3.2).

Istnieje kilka modeli sieci umożliwiających implementacje protokołów kontrolujących dostęp do medium (np. Master-Slave, Token Passing, PDC, TDMA, SCNM, DOMA i inne) i kilka umożliwiających zestawienie połączeń na poziomie aplikacji (ang. Client-Server, Producer-Consumer, Publisher-Subscriber itp.) [122], [102], [80].

- zapewnienie pewności działania (ang. safety)
  Sieci przemysłowe muszą mieć zabezpieczenia fizyczne i logiczne umożliwiające pracę w środowisku przemysłowym (por. 1.1.3). Są to:
  - odpowiednie kable (uziemiane ekrany, wzmocnione konstrukcje mechaniczne, parametry elektryczne itp.),
  - odpowiednie łączówki (uziemiane obudowy, wyższa klasa IP niż dla rozwiązań biurowych, zabezpieczenia przed odpięciem, wypadnięciem, zabezpieczenia przed błędnym podpięciem, zabezpieczenia wolnych gniazd itp.),
  - odpowiednia detekcja i korekcja ramek (preambuły, postabmuły, sumy kontrolne, transmisje cykliczne i/lub retransmisje, kodowanie i transmisja odporna na zakłócenia EMC itp.),
  - identyfikacja i obsługa przekroczeń czasów granicznych na warstwie łącza i aplikacji (ang. timeouts, mechanizmy statusów jakości – por. 2.4.2 pkt „Działanie”),
  - kontrola ruchu pochodzącego spoza systemu (infrastruktura z priorytetami i/lub klasyfikacją ruchu, np. odpowiednie przełączniki, firewalle w węzłach interfejsowych),
  - działanie przełączników musi być zgodne ze stosowanymi protokołami. Często sieci RTE wymagają priorytetyzacji ruchu (np. sieci VLAN IEC802.1Q) lub konkretnych metod przekazywania danych na portach (np. Store and Forward, Cut Through, Time Triggered),
  - działanie programów antywirusowych wymaga cyklicznej aktualizacji i wprowadza opóźnienia w przetwarzaniu danych. Należy zapewnić możliwość aktualizacji takiego oprogramowania oraz upewnić się, że złożone analizy (np. heurystyczne) nie zaburzają pracy węzła systemu. W przeciwnym wypadku należy zrezygnować z jego działania i skanować system okazjonalnie urządzeniami zewnętrznymi,

  - działanie ścian ogniowych wprowadza opóźnienia w ruchu sieciowym z danym węzłem. Może to mieć wpływ na działanie tego węzła oraz, z racji zwiększenia czasów odpowiedzi (ang. timeout), na działanie całego systemu.

- zapewnienie odporności na zagrożenia bezpieczeństwa (ang. security).
  Sieci przemysłowe muszą mieć zabezpieczenia przed niepowołanym dostępem fizycznym oraz logicznym do podłączonych zasobów i do samej sieci. Najczęściej bezpieczeństwo zapewniają:
  - Utrudniony fizyczny dostęp do switchów, routerów, access-pointów, hubów, kabli i oraz innych aktywnych i pasywnych elementów infrastruktury. Elementy te powinny być zlokalizowane w miejscach umożliwiających dostęp tylko osobom uprawnionym. Z racji pracy na terenie zakładów przemysłowych, jest to łatwe do zapewnienia względem osób spoza zakładu. Jednak zawsze pracowników należy również traktować jako potencjalne źródło zagrożenia.
  - Działanie mechanizmów autoryzacji w węzłach interfejsowych. Dla człowieka i innych urządzeń, uniemożliwienie nieuprawnionego dostępu do węzłów, uniemożliwienie wprowadzenia niepożądanego ruchu.
  - Pokrycie zasięgiem sieci bezprzewodowych powinno być w zakresie minimalnym niezbędnym.
  - Sieci bezprzewodowe powinny mieć załączone zabezpieczenia w standardzie WPA2 z szyfrowaniem AES/TKIP[33] lub innym dającym porównywalny lub wyższy poziom bezpieczeństwa transmisji jak również powinno się wykorzystywać złożone, niesłownikowe hasła dostępu.
  - Montaż uniemożliwiający uzyskanie dostępu do medium i łączówek.
  - Obsługa transmisji sieciowych tylko z określonych adresów.
  - Detekcja i odrzucanie ruchu obcego przez wykorzystanie ścian ogniowych lub podobnych rozwiązań.

- zapewnienie współpracy w działaniu (ang. interoperability).
  Współczesne sieci przemysłowe wraz ze swoją infrastrukturą muszą dostarczać środków do zapewnienia przepływu danych w środowisku międzysystemowym i heterogenicznym [117]. Dla uzyskania systemu elastycznego, otwartego na różne technologie i urządzenia, przepływ ten nie może być ograniczany przez dostępne środki, natomiast powinien być kontrolowany względem czasu oraz bezpieczeństwa. Jest to problem bezpośrednio związany z integracją węzłów, sieci, aplikacji i systemów. Kluczowe jest jednak zapewnienie dopasowania sieci. Można tego dokonać integrując sieci na poziomie

[33] Wymóg minimum na sierpień 2015.

infrastruktury łączącej segmenty, bezpośrednio przy węźle lub wykorzystując różne techniki do tego przeznaczone, a działające na wyższym poziomie abstrakcji, np. OPC (zob. 3.2.4) [339], [176], FDI (zob. 6.2) [W19], [M8], [M7], EDDL [W15], [M44], EPS (2.1.3) [302], [6], [269], SOA (zob. 2.1.1) [96], VAN [406], czy CBA[34] [359], [25].

Ponadto, przy wyborze konkretnego standardu sieci należy rozważyć zagadnienia:

- zbioru wymaganych usług
  W ISP przeważnie wymaga się zapewnienia transakcji cyklicznych oraz acyklicznych. Wszystkie sieci przemysłowe dostarczają takich usług. Różnica może polegać na sposobie zestawiania połączeń aplikacyjnych i uzyskiwanej charakterystyce czasowej takich transakcji [123].

- charakterystyk czasowych obsługi informacji
  Dobrze jest określić potrzeby przez określenie zbioru obsługiwanej informacji i wymagań względem czasu. Do obiektywnej analizy można wykorzystać pojęcie sprawności użytecznej [121], [126] lub inne analizy [179], [102], [80].

- topologii sieci
  Topologia jest często narzucana przez standard, choć w wielu przypadkach projektant może decydować o doborze topologii. Wybór topologii pociąga za sobą złożoność okablowania, jak i podatność na upadek systemu w przypadku awarii medium.

- techniki łączenia
  Wybór właściwego medium dla środowiska i występujących w nim zaburzeń, a także zagadnienia podłączeń (punkty przyłączeniowe, minimalne i maksymalne odległości, terminatory, techniki przełącznikowe, rozdzielające, wzmacniające itp.) jest kluczowy do zapewnienia poprawnego działania sieci.

Sieci w kontekście elementów ISP przedstawiono w 3.3.1.

2.4.5. Aplikacja

Węzeł realizuje lokalne zadania funkcjonalne w warstwie aplikacji. Aplikacje w poszczególnych węzłach są lokalne i mają charakter fizyczny z racji działania programów na konkretnych zasobach. W systemie niezbędna jest sieciowa współpraca aplikacji lokalnych między sobą. Działanie zadań lokalnych wraz z komunikacją pozwala realizować globalne zadania funkcjonalne postawione przed całym systemem. Współpracujące aplikacje lokalne tworzą w skali systemu wirtualną aplikację globalną, czyli aplikację systemową. Projektując system, należy zwrócić uwagę, że z punktu widzenia obsługi procesu istotne jest

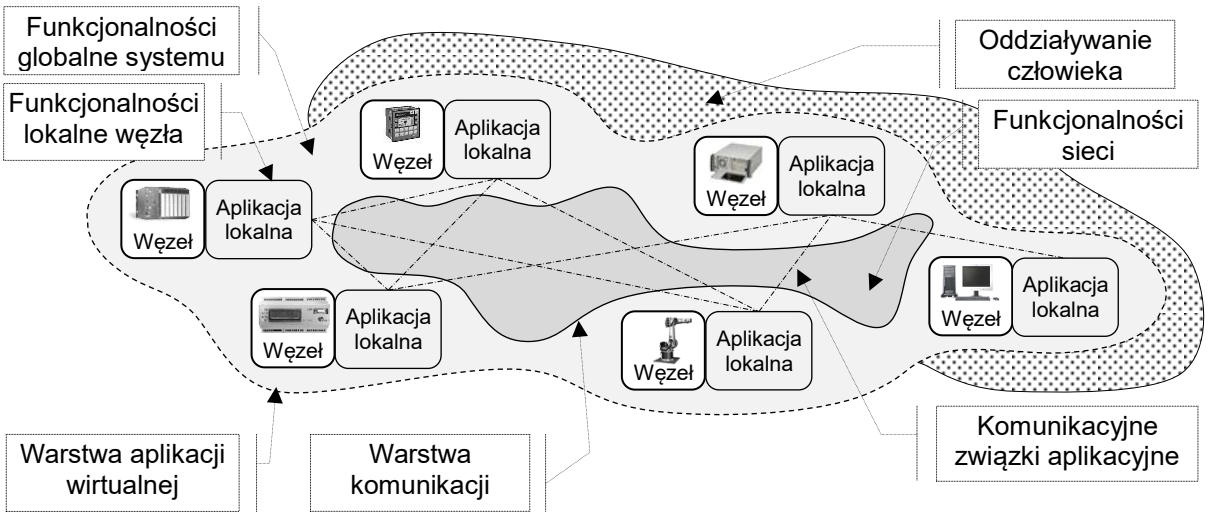
[34] Profinet CBA jest techniką obecnie nierozwijaną.

zapewnienie funkcjonalności właśnie na poziomie tej wirtualnej aplikacji. Aplikacje lokalne wraz z zasobami mogą ulegać dowolnej modyfikacji na dowolnym etapie życia systemu. Współpraca między aplikacjami jest realizowana przez wymianę informacji, na którą składają się:

- wymiana danych
  Wymiana danych dotyczy aktualizacji ich wartości pomiędzy fizyczną aplikacją źródłową a fizycznymi aplikacjami docelowymi. Reprezentacja danych użytecznych na każdym etapie wymiany może być inna.

- synchronizacja
  Synchronizacja jest przekazaniem informacji o wystąpieniu zdefiniowanego wcześniej zdarzenia. Może nie przenosić danych użytecznych.

Aplikację wirtualną systemu przedstawiono na rysunku 61.



Rys. 61. Fizyczne aplikacje lokalne i wirtualna aplikacja globalna
Fig. 61. Physical, local applications and virtual, global application

Jak wyjaśniono w 2.4.4, do realizacji komunikacji pomiędzy aplikacjami lokalnymi używa się sieci przemysłowych. W węźle istnieje współpraca pomiędzy aplikacją a stosem protokołów tegoż węzła. Dlatego w dalszych rozważaniach należy rozgraniczyć aplikację w węźle od warstwy aplikacji wykorzystywanego stosu. Warstwa aplikacji węzła zajmuje się realizacją zadań funkcjonalnych węzła, natomiast warstwa aplikacji stosu protokołów odpowiedzialna jest za zadania komunikacyjne, takie jak realizacja scenariusza wymian, obsługa powiązań aplikacyjnych, usługi aplikacyjne (por. 2.4.7), [126], [121], [123], [252] i inne.

2.4.6. Zmienne

W ISP występuje zbiór rozproszonych węzłów zajmujących się obsługą informatyczną procesu technologicznego. Technologicznie rzecz ujmując, informacja w postaci pomiarów i stanów dyskretnych pochodząca z procesu jest dostarczana do danego węzła, w którym podlega lokalnej obróbce. Systemowo natomiast każdy węzeł dysponuje wektorami informacyjnymi, reprezentującymi stan kluczowych elementów układu uczestniczących w przetwarzaniu. Wartość wektorów jest zależna od czasu.

❖ **Stan**

Do podstawowych wektorów informacyjnych opisujących stan należą:

- wejściowy wektor stanu obiektu
  Zbiór informacji opisujący bieżący stan obserwowany obiektu. Jest to wektor utworzony w pamięci węzła, na który składa się:
  - obraz obiektu pozyskiwany przez układy wejść, a przechowywany w specjalnej pamięci danych aktualizowanej zgodnie z lokalnymi zasadami akwizycji oraz
  - obraz obiektu pozyskiwany od innych węzłów, a przechowywany w pamięci danych aktualizowanej zgodnie z cyklem sieci.

  Zatem, wektor budowany jest ze wszelkich dostępnych w węźle danych opisujących rzeczywisty obiekt przemysłowy w określonym momencie czasu, i stanowi zawsze odzwierciedlenie chwilowego stanu tego obiektu. W węźle ISP aktualizacja wektora wejściowego odbywa się w sposób zsynchronizowany z pracą aplikacji i pracą sieci. Zwykle, program aplikacyjny i obsługa sieci są wykonywane niezależnie przez zadania uruchamiane w oknach czasowych, będących elementami systemowych działań cyklicznych [89]. Dokładność odzwierciedlenia rzeczywistego stanu procesu przez jego obraz wynika z rozmiaru takich okien i okresu cykli (por. 1.1.4). Wektor ten stanowi obraz procesu (ang. proces input image) tworzony z danych wejściowych. Aktualizacja obrazu zależy od cyklu akwizycji, i nie może być rzadsza niż wymagany czas ważności przechowywanej w nim informacji.

- wyjściowy wektor stanu obiektu
  Zbiór informacji opisujący bieżący stan oddziaływania na proces. Tak jak powyżej, tylko kierunek przekazu informacji jest odwrotny, tzn. wektor z pamięci węzła oddziałuje na proces. Wektor jest budowany na podstawie pozostałych wektorów informacyjnych oraz działania aplikacji lokalnej i systemowej. Stanowi on obraz wyjściowy procesu (ang. proces output image).

- lokalny wektor stanu
  Zbiór informacji opisujący bieżące wartości zmiennych lokalnych węzła wynikające z przetwarzania lokalnego, czyli działania aplikacji lokalnej węzła. Jest to zawartość pamięci danych powiązanej z wykonywanymi programami w węźle. Zmienia się ona w czasie z cyklem wynikającym z uruchamiania programów. Stanowi on prywatny (lokalny, wewnętrzny, ang. private, internal) obraz funkcjonowania węzła.

- zewnętrzny wektor stanu
  Zbiór informacji wejściowo-wyjściowej, który jest produkowany (wytwarzany, wysyłany, dostarczany) na potrzeby innych węzłów systemu. Jest to publiczny (zewnętrzny, ang. public, external) obraz funkcjonowania węzła.

- wektory stanu innych węzłów
  Zbiory informacji konsumowanych (pobieranych, odczytywanych) od innych węzłów produkujących informację w systemie. Są to publiczne obrazy innych węzłów.

Dla ISP bardzo ważne jest **zachowywanie stanu** (por. 2.1.2). W praktyce nie można pozwolić na uruchamianie programów zawsze ze stanem domyślnym, niezależnie czy jest to zwykła praca, uruchamianie po zatrzymaniu awaryjnym, serwisowym, czy wynikającym z braku zasilania. Wynika to z faktu, że węzeł podłączony do fizycznego obiektu i systemu nie przestaje być do nich podłączony w momencie zatrzymania jego jakichkolwiek zadań. Uruchomienie węzła z wartościami początkowymi (domyślnymi, inicjującymi) wektorów stanu mogłoby być tragiczne w skutkach, poza przypadkiem hipotetycznego „pierwszego uruchomienia". Zachowanie stanu wiąże się z zapewnieniem cechy niezawodności systemu. Przy programowaniu wielu komputerowych urządzeń sterujących stosuje się pojęcie stanu procesu (ang. proces state) lub obrazu procesu (ang. proces image) reprezentującego jakiś wycinek całego opisu informacyjnego podlegającego danemu ISP.

❖ **Reprezentacja**

Informacja stanu jest składowana w pamięci węzła, często specjalnie wydzielonej, w postaci zakodowanych ciągów bitowych (liczby, znaki, bity) i reprezentowana przez abstrakcje **zmiennych** o określonych typach. Zmienne występujące w informatycznych systemach przemysłowych wygodnie jest grupować ze względu na ich:

- przeznaczenie, czyli rodzaj informacji, jaką reprezentują, np. pomiary, stany, parametry, nastawy, zmienne tymczasowe, IO itp.,
- pochodzenie, czyli rodzaj przynależności do elementów logicznych lub fizycznych systemu, jak np. zmienne globalne węzłów, lokalne węzłów, globalne systemu, zadań, programów, modułów, przynależne do warstw stosów sieciowych itp.,

- reprezentację, czyli sposób odzwierciedlenia w pamięci, np. bity (zmienne dyskretne, binarne, bitowe, boolowskie, ang. discrete, binary, boolean), rejestry (liczbowe, rejestrowe, słowowe, analogowe, ang. register, word, analog/analogue), znaki (znakowe, bajtowe, ang. characters), strefy (obszary kontrolne, parametryzujące, ang. control-block, zone) itp.,
- typ, czyli sposób kodowania informacji w pamięci, np. ogólnie znane typy bool, integer, real, lub mniej oczywiste typy czasowe (np. time, duration), typy kodów (np. BCD) itp. Wskazane jest, aby typy były zgodne z typami znormalizowanymi, np. ze standardami IEEE/ISO/IEC [M74], [M72], [M67]. Ułatwia to integrację, gdy istnieje potrzeba przekazania informacji pomiędzy zupełnie różnymi urządzeniami. W praktyce typy stosowane w PLC są znormalizowane, choć można znaleźć przypadki różnych reprezentacji alokowanych zmiennych.

Typy liczbowe mogą występować w modyfikacjach wynikających z kodowania, dla przykładu liczby ze znakiem lub bez, precyzja pojedyncza lub wielokrotna.

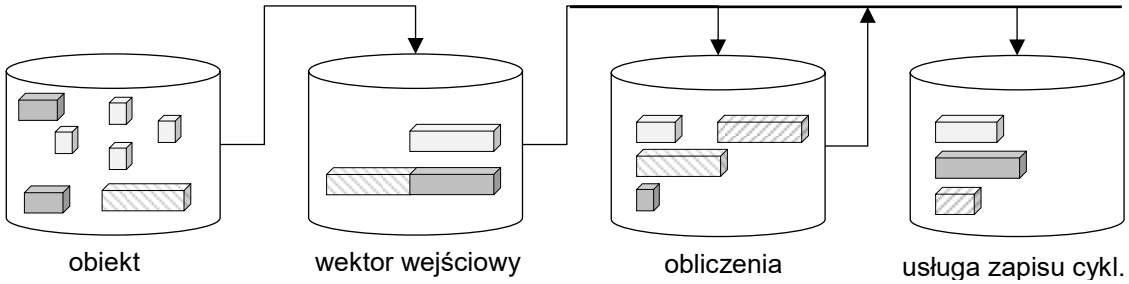
Podział zmiennych w procesie projektowania, poza określeniem typów, nie jest niezbędny, ale jak zostanie pokazane później wysoce wskazany. Ułatwia zrozumienie procesu oraz porządkuje opis informacji, czyli ułatwia dokumentowanie, a co za tym idzie zmniejsza prawdopodobieństwo powstania błędów w założeniach i upraszcza późniejsze modyfikacje. Notacje deklaracji zmiennych stosowane w językach programowania PLC wspomagają grupowanie zmiennych, szczególnie tych opartych na normie IEC 61131 (2.6, [M26]).

Zmienne o reprezentacji i typach złożonych są tworzone ze zmiennych o typach prostych i w PLC mają najczęściej postać tablicy lub struktury danych. Alokacje o typach złożonych często są nazwane blokami danych (ang. data block), blokami kontrolnymi (ang. control block), strefami (ang. zone), obszarami (ang. area), zonami itp. Część zmiennych lub innych obiektów programistycznych może w PLC być predefiniowana i prealokowana. Oznacza to, że będą one miały utworzone w pamięci instancje niezależnie od kodu i poczynań programisty. Liczba takich instancji jest wówczas ograniczona predefinicją. Dotyczy to również wybranych obszarów pamięci, które są często przypisane do informacji o konkretnym przeznaczeniu (np. wejścia dyskretne związane z pamięcią obrazu wejść itp.).

Zmienne mogą być alokowane jawnie bądź niejawnie zależnie od idei, jaka przyświeca sposobowi programowania sterownika. Alokacje niejawne są tworzone automatycznie podczas kompilacji kodu, a alokacje jawne są definiowane manualnie przez programistę. Ponieważ liczba zmiennych opisujących środowisko, w jakim pracuje system, ma charakter statyczny, zatem i alokacje w pamięci urządzeń mają charakter statyczny (por. 1.2.3). Praktycznie w PLC nie spotyka się specjalnych mechanizmów wspomagających dynamiczne alokacje pamięci.

❖ **Filtrowanie i kondensacja**

Liczba zmiennych w różnych warstwach modelu może być inna, niezależnie czy dostępna jest ta sama informacja, czy też ilość informacji w tych warstwach jest różna. Wynika to z faktu, że w danej warstwie niekoniecznie jest potrzebna pełna informacja stanu. Potrzebne są te zmienne, które są przetwarzane. Przekazywanie kompletnego zbioru informacji stanowej danej warstwy do innej kosztuje czas, który można zaoszczędzić filtrując zbiory danych do niezbędnych podzbiorów potrzebnych do przekazania. Dotyczy to nie tylko ogólnych warstw modelu, ale i każdego elementu węzła od protokołów do elementów języka programowania.



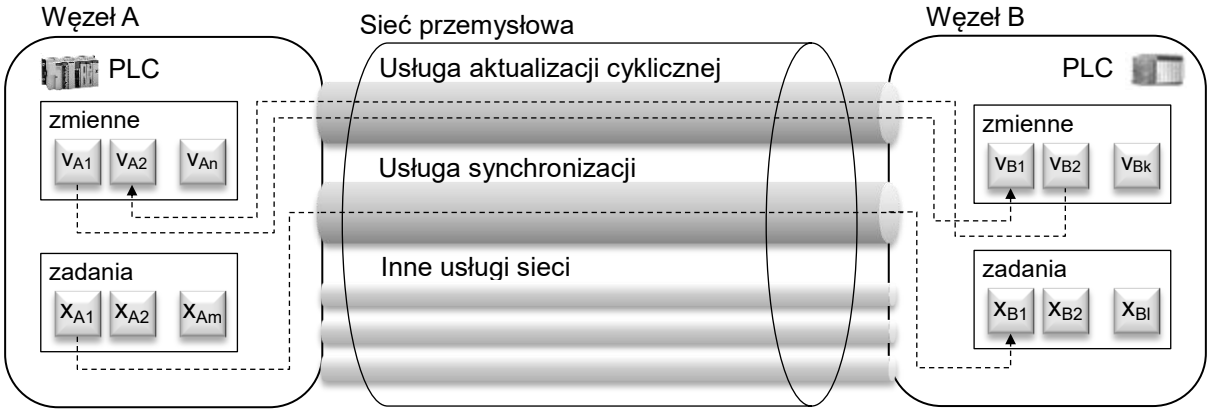
Rys. 62. Kondensacja zmiennych
Fig. 62. Variable packing

Naturalne też jest, że ze względu na różnorodność sposobów reprezentowania informacji pewne zmienne reprezentowane w dany sposób są alokowane inaczej w innym miejscu. Zatem w każdej warstwie modelu węzła, lub wręcz w każdym jego elemencie, zmienne mogą reprezentować informację w inny sposób. Dla przykładu, boolowskie zmienne reprezentujące dyskretny stan wejść mogą być alokowane w pamięci wejść sterownika jako słowo ciąg bitów, w pamięci modułu przetwarzania jako ciąg słów, a w pamięci usługi sieciowej jako ciąg bajtów. Jeżeli dana operacja nie wymaga pełnego zbioru informacji i/lub koszt jej obsługi rośnie z alokowanym rozmiarem, to wskazane jest, aby dokonać kondensacji zmiennych przez określenie zbioru minimum z dostępnego zbioru informacji i użyciu typów gwarantujących minimalną niezbędną przestrzeń alokacji. Graficzny przykład przedstawiono na rysunku 62. Prostopadłościany reprezentują alokacje zmiennych, wypełnienie – rodzaj informacji.

Urządzenia sterownikowe charakteryzują się przeważnie relatywnie niewielkimi rozmiarami pamięci przeznaczonej na alokacje zmiennych. Dlatego dobra identyfikacja i alokacja zmiennych jest kluczowa przy tworzeniu aplikacji systemowej. Można tu odwołać się ponownie do procesu twórczego, tym razem do rzeźbienia. Cała informacja opisująca obiekt jest materiałem, wydzielenie zmiennych i struktur jest wizją twórcy, dotyczącą tego co i gdzie należy obrabiać, stworzenie algorytmów jest doborem narzędzi, a kodowanie jest procesem twórczym. Może stąd często się słyszy o „rzeźbieniu” czy „dłubaniu” kodu.

2.4.7. Powiązania

Na rysunku 61 oraz 17 można zaobserwować zależności pomiędzy węzłami, które muszą być obsłużone przez komunikację sieciową. Zależności zachodzące wewnątrz wirtualnej aplikacji systemu wymagają kanałów komunikacyjnych o charakterze logicznym, natomiast zależności zachodzące między fizycznymi urządzeniami wymagają kanałów fizycznych. Kanały logiczne są definiowane na poziomie aplikacji, natomiast fizyczne na poziomie komunikacji (sieci). Dlatego w systemie można wydzielić powiązania (związki, zależności) aplikacyjne (programowe) i komunikacyjne (sieciowe). Związki aplikacyjne łączą elementy aplikacji, jak zadania i zmienne, natomiast związki komunikacyjne łączą urządzenia przez konkretną sieć. Można dla przykładu powiązać synchronizację zadań x_{A1} i xB1 przez zdefiniowanie powiązania aplikacyjnego pomiędzy zdarzeniem e_{A1} w jednym węźle a wyzwalaczem zadania x_{B1} w drugim. Ponadto, np. można zdefiniować aktualizację cykliczną zmiennej v_{B1} wartościom zmiennej v_{A1} i zmiennej v_{A2} zmienną v_{B2} przez zdefiniowanie połączeń aplikacyjnych między nimi. Natomiast realizację tego powierzyć odpowiednim usługom sieci przez zdefiniowanie odpowiednich powiązań komunikacyjnych. Jest to zilustrowane na rysunku 63.



Rys. 63. Przykład powiązań
Fig. 63. Example of relations

Teoretycznie połączenia aplikacyjne mogą być definiowane w relacjach jeden do wielu i z wykorzystaniem różnych usług. W praktyce, zależy to od możliwości oferowanych przez konkretne techniki komunikacyjne i wykorzystujące je urządzenia.

2.5. Podejście IEC61499

Współczesne wyzwania dla twórców ISP głównie dotyczą uzyskania dużej dynamiki w tworzeniu i modyfikacjach systemów produkcyjnych. Wynika to z oczekiwań rynku, który

wymaga produkowania krótkich serii i wprowadzania częstych zmian konstrukcyjnych produktów. Dokonywanie tego bez narzędzi wspomagających projektowanie systemów i bez możliwości analizy i symulacji działania przed implementacją na hali produkcyjnej jest czasochłonne i kosztowne. Języki programowania węzłów sterujących w ISP są ustandaryzowane normą IEC61131 (zob. 2.6). Norma ta oraz inne języki nią nieobjęte nie dają jednak wsparcia do projektowania, programowania czy też testowania systemu jako całości. Dlatego w roku 2005[35] powstał standard IEC61499 [136]. Został on stworzony celem ujednolicenia podejścia do projektowania, analizy i testowania systemów rozproszonych. Idea bazuje na blokach funkcyjnych z normy IEC61131-3 oraz koncepcji bloków i opisu urządzeń z normy IEC61804 [M44]. Zaproponowana norma jest bardzo ogólna i nie należy jej wiązać z konkretną klasą zastosowań. Może być wykorzystana w szerokim spektrum aplikacji od budowy algorytmów do tworzenia systemów klasy ISP [174], [369], [164], [266]. Można też doszukać się podobieństw z obiektowym paradygmatem projektowania (ang. Object-Oriented Paradigm) i zintegrować proces projektowania z innymi technikami, jak np. wykorzystanie UML [175], [277], tworzenie systemów wbudowanych [198] czy inne [87].

Norma składa się z czterech części, z czego część trzecia jest obecnie wycofana. Część pierwsza dotyczy definicji stosowanych w normie pojęć, zasad i modeli oraz zakresu normalizacji. Część druga to wymagania dotyczące narzędzi programowych względem specyfikacji elementów opisu systemu, funkcjonalności elementów modeli, wytycznych do analiz i zarządzania systemami rozproszonymi oraz wymiany informacji w środowisku narzędziowym. W części trzeciej prezentowane są zasady tworzenia rozwiązań, umożliwiające zachowanie wymagań normy przy współpracy urządzeń i oprogramowania pochodzących od różnych producentów. W roku 2013 części pierwsza i druga zostały zaktualizowane do wersji 2.0.

Zagadnienia, których dotyczy norma IEC61499, są istotne dla rozwoju współczesnych narzędzi deweloperskich. Dlatego, tak jak zauważono w 1.2.6, standard ten oraz związane z nim normy i narzędzia są stale rozwijane. Wszelkie cytowania norm i opracowania standardu spotykane w literaturze dotyczą jedynie konkretnych wersji, a jedyną poprawną wykładnią standardu jest sama norma.

Informacje zawarte w niniejszym podrozdziale nie zastępują i nie wyczerpują zakresu IEC61499. Dają tylko ogólny wgląd w ideę oraz dostępne środki. Ponadto, jako norma bardzo ogólna istnieje wiele interpretacji skupiających się na różnych aspektach projektowych. Niektóre dotyczą analizy formalnej, inne analizy wydajnościowej, a jeszcze inne wygodnego i intuicyjnego tworzenia modeli [370]. Swoboda interpretacyjna

[35] Pierwsze prace były prowadzone wcześniej, a pierwsze, dość ogólne, publikacje powstały w 2000 roku.

doprowadziła do niskiej popularności pierwszej edycji i konieczności wprowadzenia uściśleń w edycji drugiej. Zachęca się czytelnika do zapoznania się z oryginałami [M34], [M35], [M36] i bieżącym wsparciem online, np. [W24], [W2].

2.5.1. Modele

Podstawowym celem normy IEC61499 jest określenie modeli sprzyjających projektowaniu systemów rozproszonych, w tym systemów klasy ISP. Norma ta nie dotyczy metodologii programowania urządzeń. Modele umożliwiają opisanie funkcjonowania systemów w sposób standardowy i formalny a tym samym jednoznaczny. Takie podejście pozwala na operowanie na wyższym poziomie abstrakcji niż urządzenia i komunikacja między nimi.

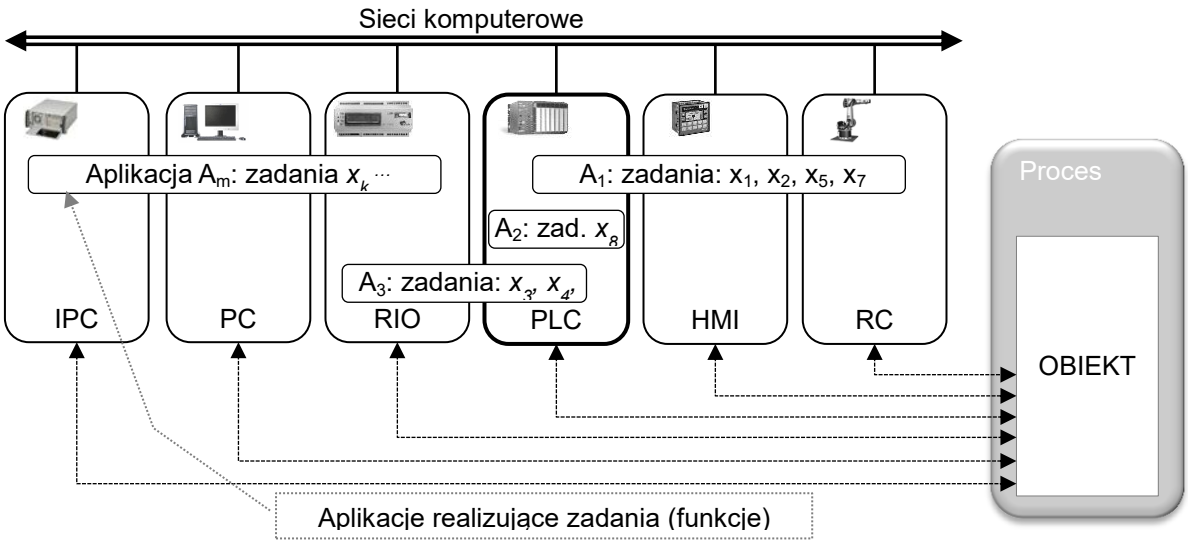
Do opisu systemu można podejść od strony urządzeń fizycznych lub od strony zadaniowej. Przedstawione we wcześniejszych podrozdziałach pojęcia służą do opisu systemu od strony urządzeń, elementów infrastruktury i związanych z nimi abstrakcji. Norma IEC61499 daje wsparcie dla tworzenia diagramów opisu systemu od strony przetwarzania i przepływu informacji. Dzięki abstrakcji, jaką jest blok funkcyjny, umożliwia opisanie ISP uwzględniające zadania przetwarzania danych, sterowanie tymi zadaniami przez przekazywanie zdarzeń oraz przepływ danych między nimi. Opis taki abstrahuje od sprzętu.

Zamodelowany system może zostać formalnie zrozumiany i zweryfikowany z wymaganiami niezależnie od warstwy sprzętowo-programowej. Implementacja funkcjonalności systemu dla konkretnych urządzeń jest tylko pochodną tego procesu. Związana jest ona z fizyczną dystrybucją zadań, generacją kodu dla konkretnych architektur i zapewnieniem komunikacji między zadaniami.

❖ **Model systemu**

Zdefiniowany w normie model systemu oddziela proces przemysłowy od urządzeń systemowych i komunikacji. Zadania (funkcje) systemowe są realizowane przez aplikacje umiejscowione w pojedynczym urządzeniu lub rozproszone pomiędzy wieloma urządzeniami. Model ten przedstawiony został w przykładzie na rysunku 64. Łatwo zauważyć, że jest on podobny do modelu rozproszonego z rysunku 50.

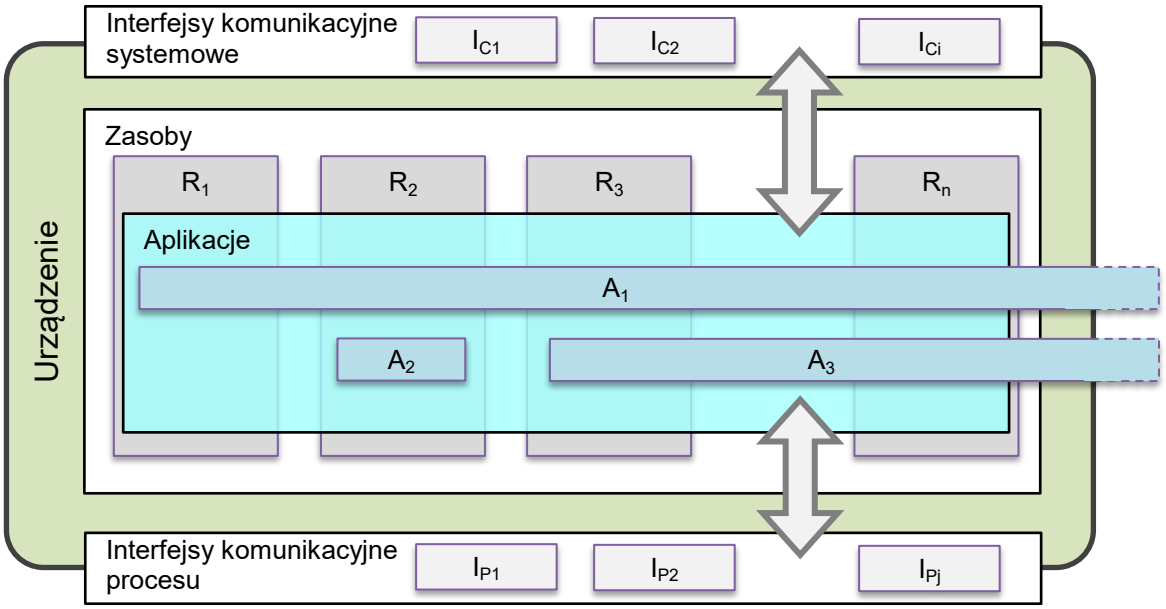
Sieci komputerowe zostały przedstawione schematycznie i dotyczą wszelkich środków komunikacyjnych wykorzystywanych na rzecz działania modelowanego systemu. Urządzenia mogą być podłączone do różnych sieci lub ich segmentów (ang. segment), jak i różne sieci mogą być integrowane. Na poziomie modelu rozróżnienie sieci nie ma znaczenia. Przedstawione na rysunku przykładowe grupy zadań stanowią aplikacje, których implementacja jest rozproszona między urządzeniami i ich zasobami. Można zatem doszukać się analogii do modelu przedstawionego na rysunku 61.



Rys. 64. Model systemu wg IEC61499 na prostym przykładzie
Fig. 64. IEC61499 system model based on a simple example

❖ Model urządzenia

Model urządzenia (węzła) określa interfejsy komunikacyjne i zasoby przetwarzające. Zadania funkcjonalne systemu są realizowane przez aplikacje. Programy z nimi związane mogą być wykonywane na jednym lub na kilku zasobach w danym urządzeniu, jak i na wielu urządzeniach. Na rysunku 65 pokazano urządzenie z *n* zasobami przetwarzającymi i interfejsami komunikacyjnymi oraz *j* interfejsami procesowymi.

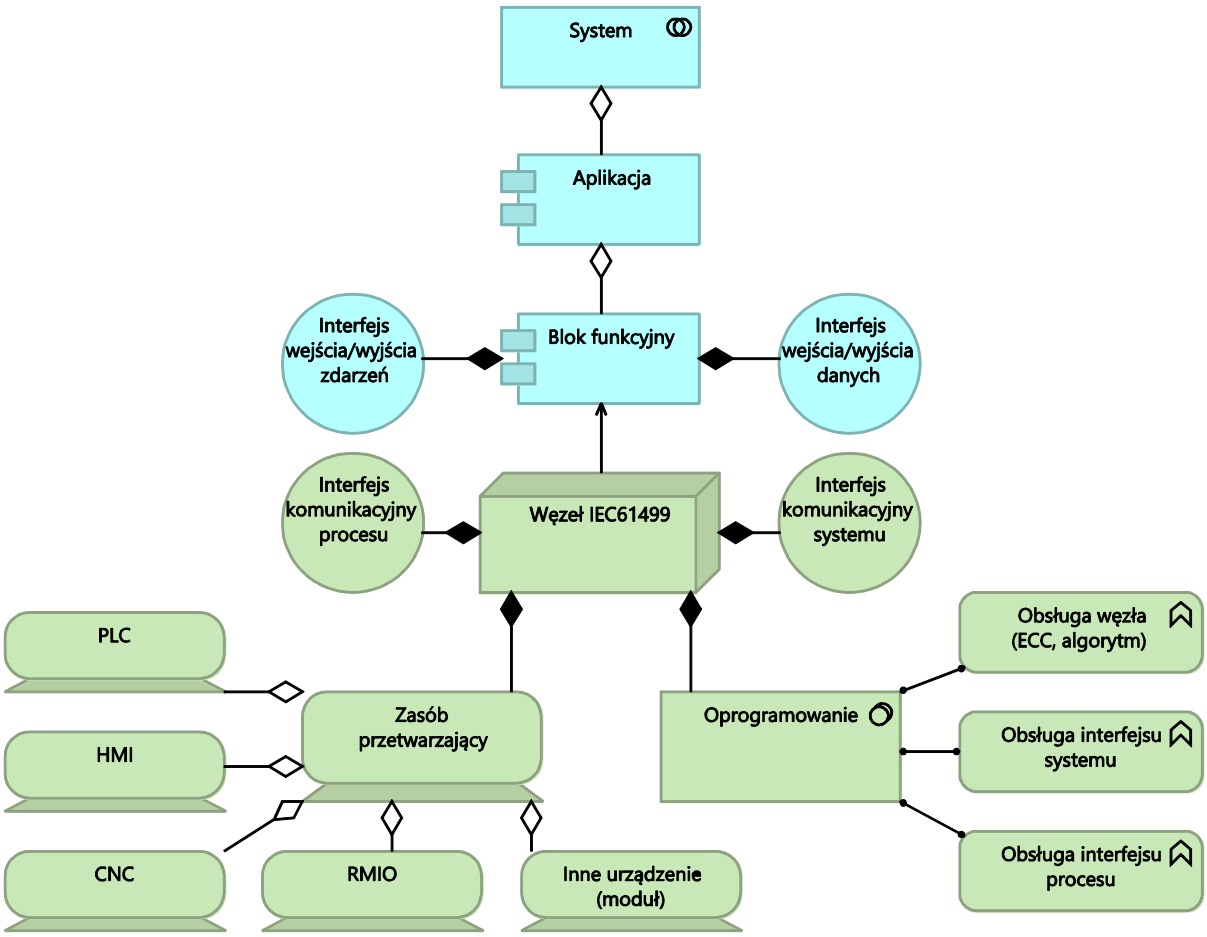


Rys. 65. Model urządzenia wg IEC61499
Fig. 65. Device model according to IEC61499

Jako przykład wykorzystano urządzenie PLC z rysunku 64 i umieszczono trzy aplikacje rozmieszczając ich wykonywanie w zasobach:

- A_1 na R_1, R_2, R_3 i R_4,
- A_2 na R_2,
- A_3 na R_3 i R_n.

Na rysunku 66 przedstawiono model węzła wg IEC61131 z użyciem formalnego metamodelu w języku ArchiMate 2.0.



Rys. 66. Metamodel węzła wg IEC61499 w ArchiMate 2.0
Fig. 66. Metamodel of IEC61499 node based on ArchiMate 2.0

W warstwie technologicznej węzeł jako element infrastruktury stanowi kompozycję współpracujących zasobów przetwarzających, oprogramowania oraz fizycznych interfejsów. W warstwie aplikacji przedstawiono komponent bloku funkcyjnego agregowany przez aplikacje oraz komponent aplikacji grupowany przez system. Każdy blok funkcyjny ma przynajmniej jeden interfejs wejścia i wyjścia dla zdarzeń i danych. Blok wykonywany jest w węźle. Istnieją trzy główne funkcje oprogramowania: obsługa węzła, obsługa interfejsu systemu i obsługa interfejsu procesu. Przetwarzanie w węźle realizowane jest w zasobie

przetwarzającym, który agreguje przynajmniej jedno urządzenie komputerowe. W zestaw takich urządzeń mogą wchodzić wszelkiego rodzaju jednostki przetwarzające zorientowane na pracę w czasie rzeczywistym i w środowisku przemysłowym, w tym urządzenia opisane w 3.1 oraz inne uniwersalne lub specjalizowane moduły. Zaprezentowane elementy metamodelu opisane są w kolejnych podrozdziałach.

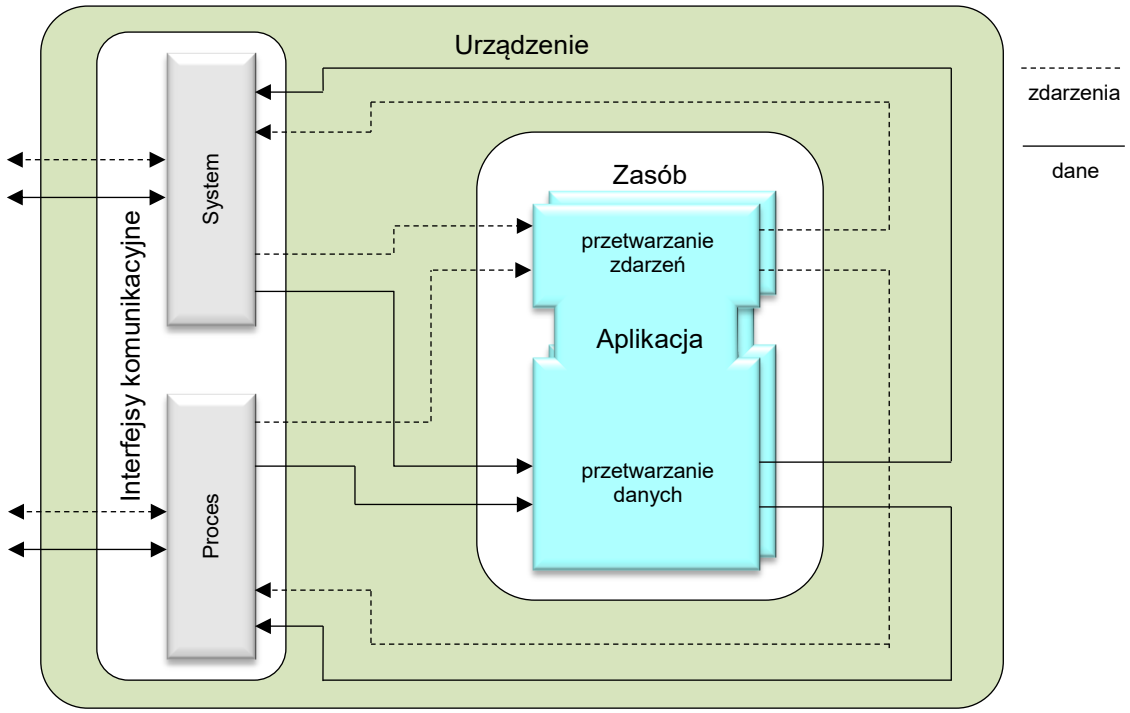
❖ **Model zasobu**

Pojedynczy zasób składa się z funkcji interfejsowych oraz przynajmniej jednej lokalnej aplikacji przetwarzania danych. Funkcje interfejsowe odpowiadają z jednej strony za przekazywanie danych z i do innych aplikacji oraz z drugiej za komunikację z procesem. Każda z aplikacji w zasobie stanowi zamkniętą całość lub część aplikacji rozproszonej. Jej działanie jest sterowane zdarzeniami. Zasób jest urzeczywistniony w urządzeniu i podlega niezależnemu sterowaniu ze strony OS. Oznacza to, że tworzy instancję określonego typu zasobu. Po uruchomieniu stanowi jednostkę funkcjonalną (ang. functional unit), czyli jako całość pełni jakąś konkretną rolę w systemie lub wykonuje zadanie w aplikacji rozproszonej pomiędzy wiele zasobów. Z poziomu OS zasób może podlegać działaniom administracyjnym (np. uruchomienie, zatrzymanie, zmiana konfiguracji, usunięcie itp.), ale działania te są niezależne od innych zasobów w urządzeniu i przez to nie wpływają na ich działanie. Nie dotyczy to tych aspektów pracy, gdzie zasoby muszą współpracować, np. synchronizując się, wymieniać dane itp.

Zadaniem zasobu jest przyjąć dane i zdarzenia z interfejsów, dokonać przetwarzania, a następnie zwrócić dane i zdarzenia do interfejsów. Interfejsy mogą być współdzielone pomiędzy zasobami. Zostało to przedstawione na rysunku 67.

Obsługa wejścia i wyjścia do zasobu odbywa się z wykorzystaniem bloków funkcyjnych usług interfejsowych (ang. service interface function block), osobnych dla komunikacji systemowej i osobnych dla procesu. Bloki te dokonują mapowania zdarzeń i danych pomiędzy interfejsami a aplikacją lokalną. Zgodnie z modelem systemu, niektóre zasoby mogą być dedykowane do realizacji tylko usług systemowych, np. komunikacyjnych, gdzie przetwarzanie nie jest związane z aplikacją ISP, lecz z obsługą urządzenia i jego komponentów, np. interfejsów sieci i stosu protokołów.

Przetwarzania w ramach aplikacji lokalnej dokonuje się przy użyciu bloku funkcyjnego (ang. function block), czyli specjalnej jednostki organizacyjnej aplikacji zawierającej algorytmy przetwarzania. Opis modelu bloku i jego własności znajduje się w 2.5.2. Blok taki wraz z blokami interfejsowymi może stanowić obsługę całej aplikacji lokalnej lub być lokalną częścią aplikacji rozproszonej.



Rys. 67. Przepływ zdarzeń i danych w zasobie
Fig. 67. Event and data flow within a resource

Kolejność wykonania bloków funkcyjnych i sterowanie przepływem danych realizowane jest przez mechanizm szeregujący (ang. scheduler). Działa on zgodnie z sekwencją i zależnościami czasowymi określonymi przez:

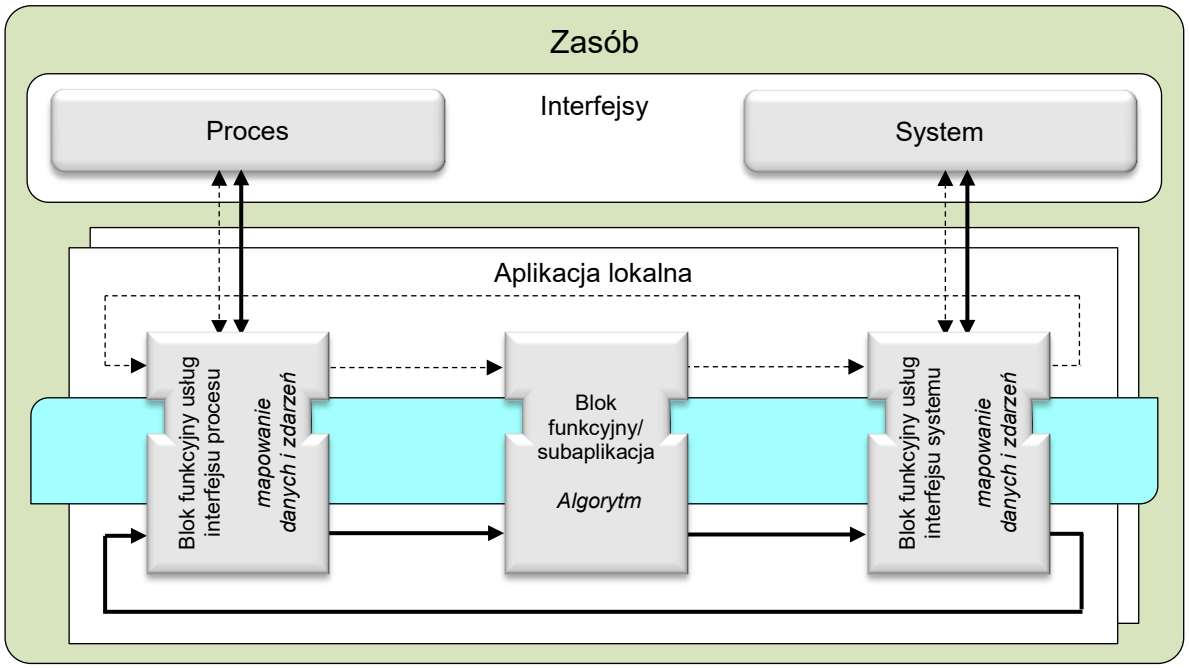
- kolejność wystąpienia zdarzeń,
- semantykę połączeń między wewnętrznymi blokami funkcyjnymi aplikacji,
- zdefiniowane okresy i priorytety szeregowania.

Ogólny model zasobu pokazano na rysunku 68. W aplikacji znajduje się sieć przetwarzająca w postaci bloków funkcyjnych. Węzłami sieci mogą być subaplikacje, bloki funkcyjne proste lub bloki funkcyjne złożone, czyli składające się z innych subaplikacji, bloków prostych i złożonych (zob. 2.5.2). Połączeniami w sieci są linie reprezentujące przepływ danych i zdarzeń. Dla lepszego zrozumienia, na rysunku zaznaczono kierunki przepływu, choć nie jest to niezbędne, gdyż przepływy są zawsze realizowane od wyjść jednego elementu do wejść drugiego.

Zasoby reagują na zdarzenia z interfejsów, sterując działaniem sieci przez:

- modyfikacje zmiennych,
- określanie porządku uruchamiania algorytmów (szeregowanie),
- generacje dodatkowych zdarzeń,
- interakcje z interfejsami.

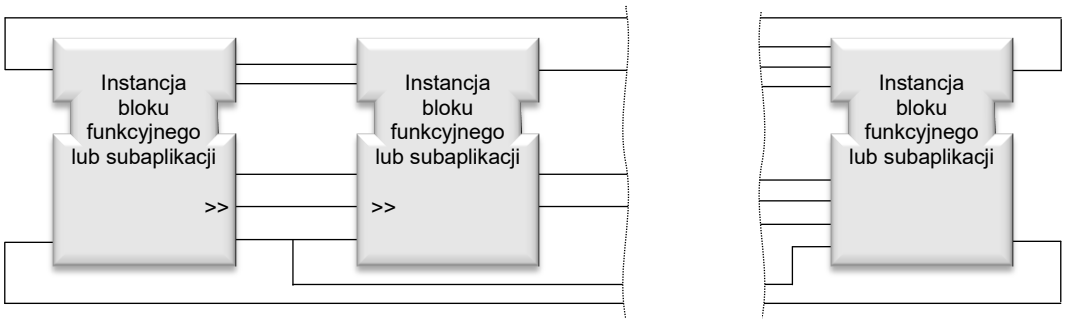
Odpowiednie reakcje wynikają ze zdefiniowanych powiązań zdarzeń i danych w aplikacji i ich chwilowego stanu.



Rys. 68. Model zasobu przetwarzającego urządzenia wg IEC61499
Fig. 68. Model of a device processing resource according to IEC61499

❖ Model aplikacji

Aplikacje tak jak i subaplikacje składają się z sieci bloków funkcyjnych, i tak jak bloki funkcyjne tworzą instancje określonego typu. Sieć bloków funkcyjnych może się składać z dowolnych bloków funkcyjnych i subaplikacji połączonych ze sobą liniami reprezentującymi przepływ zdarzeń i danych. Zostało to pokazane na rysunku 69. Zaprezentowane połączenia sieci są przykładowe.



Rys. 69. Model aplikacji wg IEC61499
Fig. 69. Application model according to IEC61499

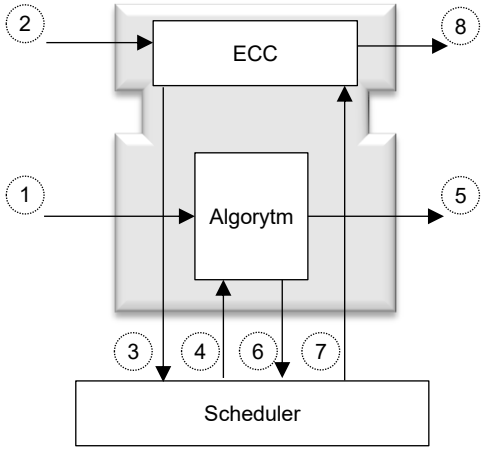
Dla uproszczenia formy sieci do łączenia mogą zostać wykorzystane również tzw. adaptery w postaci wtyków i gniazd (ang. plug, socket), które zawierają zgrupowany zestaw zda-

rzeń i danych. W ramach kompatybilnych par połączenie jest reprezentowane pojedynczą linią łączącą adaptery (oznaczane na rysunku przez „>>"). Łączone są zawsze wyjścia z wejściami oraz wtyk z gniazdem. Zdarzenia są łączone zawsze jeden do jeden. Rozdzielanie i łączenie zdarzeń jest możliwe przez specjalne bloki obsługi zdarzeń, które mogą być wbudowywane w inne bloki. Wejścia danych mogą być podłączane do co najwyżej jednego wyjścia danych. Natomiast wyjścia mogą być łączone do dowolnej liczby wejść, w tym również mogą pozostać niepodłączone.

Reguły łączeń są spójne dla bloków i aplikacji, a szczegóły można znaleźć w normie. Aplikacja może być wykonywana w jednym zasobie jednego urządzenia lub może być rozproszona pomiędzy wiele zasobów jednego lub wielu urządzeń. Przepływ zdarzeń w sieci określa, kiedy dany zasób ma uruchomić algorytm związany z danym blokiem funkcyjnym.

❖ Model wykonania bloku

Blok funkcyjny wg IEC61499 nie jest zwykłym rozkazem tak jak wg IEC61131. Kod jest uruchamiany w funkcji zdarzeń wejściowych, ECC oraz działania mechanizmu szeregującego zasobu, w którym jest osadzony. Jego wykonanie wiąże się z kilkoma istotnymi zdarzeniami, względem których można rozpatrywać czasy charakteryzujące takie wykonanie. Zilustrowano to na rysunku 70.



1 – zmienna wejściowa uzyskuje ważność
2 – występuje zdarzenie na wejściu bloku
3 – maszyna ECC zezwala na uruchomienie zadania i zleca to do mechanizmu szeregującego
4 – mechanizm szeregujący uruchamia zadanie
5 – zostaje wypracowany wynik
6 – scheduler jest informowany o zakończeniu zadania
7 – maszyna ECC jest informowana o zakończeniu zadania
8 – ECC generuje zdarzenie potwierdzenia wykonania zadania

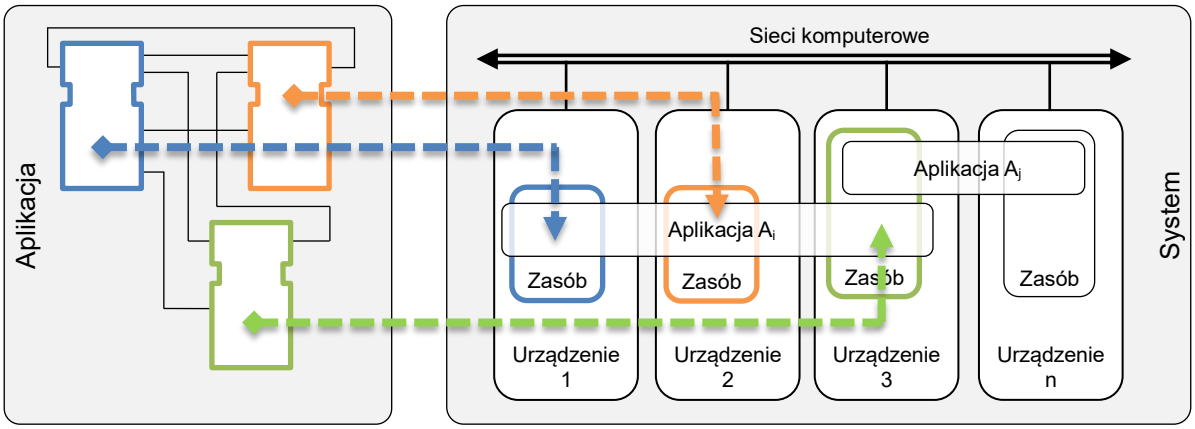
Czas przygotowania do pracy to różnica między czasem 2 a 1.
Czas działania zadania to różnica między 6 a 4.
Czas uruchamiania zadania to różnica między 4 a 2
i analogicznie czas sygnalizacji końca to różnica między 8 a 6.

Rys. 70. Ilustracja działań podczas wykonania bloku
Fig. 70. Illustration of actions during block execution

Zarówno czas uruchamiania, jak i czas sygnalizacji zakończenia pracy zależą od działania ECC i od działania funkcji szeregujących. Czasy te nie są pomijalne. Działania kontrolujące wykonanie bloku muszą uwzględniać zagadnienia spójności i ważności danych (por. 2.3.2). Mechanizm szeregujący może pracować w trybie wielozadaniowym z wywłaszczaniem i priorytetami.

❖ Model dystrybucji

Jak było wspomniane w opisie modelu systemu, aplikacja i subaplikacja mogą być rozpraszane pomiędzy różne zasoby i urządzenia. Jest to kluczowa właściwość konstruowania systemów wg IEC61499. Rozproszenia nie można jednak dokonać z blokami funkcyjnymi. Ich implementacja jest ukryta dla poziomu abstrakcji, na którym są wykorzystywane. Dlatego powinny być niepodzielne (atomowe) względem dystrybucji między zasoby i urządzenia. Blok funkcyjny musi przynależeć do konkretnego zasobu. Zostało to pokazane na rysunku 71.



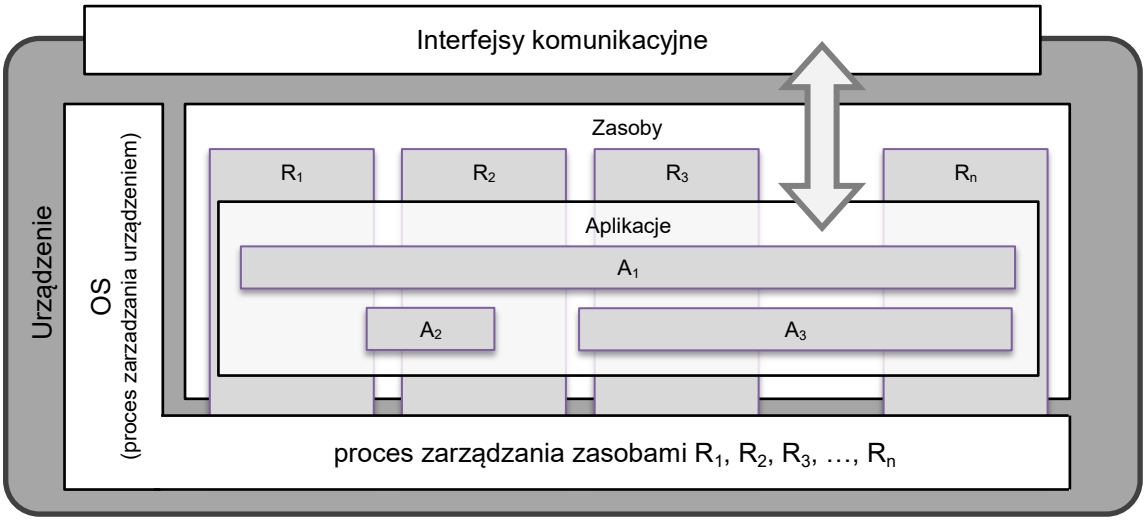
Rys. 71. Dystrybucja aplikacji i bloków funkcyjnych
Fig. 71. Distribution of applications and function blocks

W idealnym rozwiązaniu rozproszenie nie powinno mieć istotnego wpływu na charakterystykę czasową przekazywania danych i zdarzeń między rozproszonymi elementami aplikacji. Jest tak, gdy aplikacja jest dystrybułowana do jednego zasobu czy nawet między różne zasoby tego samego urządzenia. Jednak przy rozproszeniu pomiędzy różne urządzenia należy pamiętać o opóźnieniach wnoszonych przez warstwę komunikacji. Warto o tym pomyśleć przy planowaniu dystrybucji oraz przy doborze środków komunikacji. Zbędne rozproszenie może spowodować negatywny wpływ na właściwości czasowe rozwiązania.

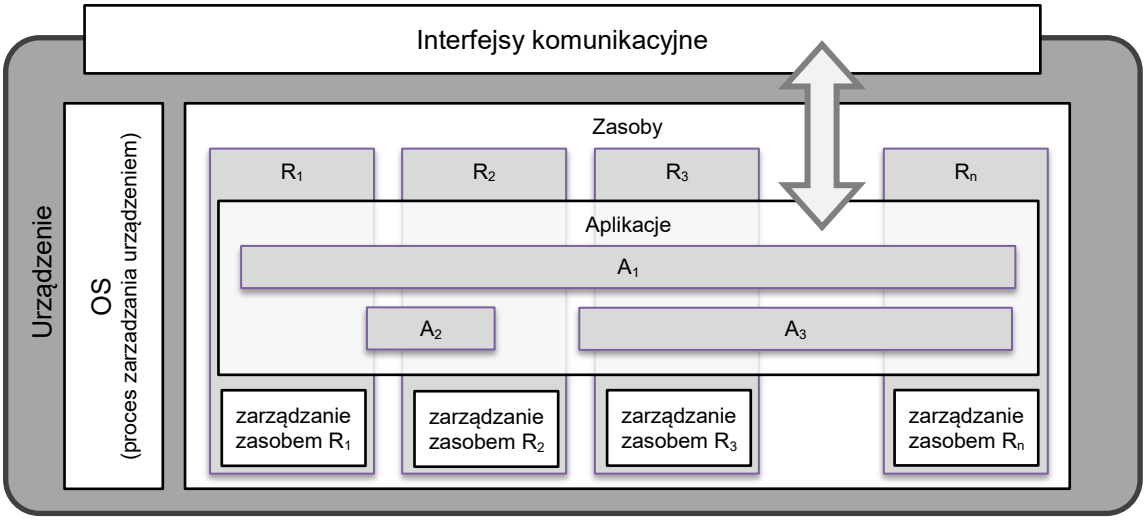
❖ Model zarządzania

Zasoby i urządzenia muszą być zarządzane, a zatem musi istnieć proces lub procesy w OS urządzenia, które będą takie zadania realizować. W normie przewidziano dwa podejścia do konstrukcji takiego procesu: współdzielony i rozproszony.

W podejściu współdzielonym urządzenie i jego zasoby są zarządzane z jednego procesu obsługującego wszystko w sposób centralny. Zakłada się istnienie specjalnego zadania systemowego, które realizuje takie nadrzędne zarządzanie. W podejściu rozproszonym każdy zasób posiada swoje osobne zadanie zarządzające. OS zajmuje się zarządzaniem tych zadań, ale nie zajmuje się zarządzaniem zasobami. Zostało to zobrazowane na rysunkach 72 i 73.



Rys. 72. Centralne zarządzanie zasobami
Fig. 72. Centralized management of resources



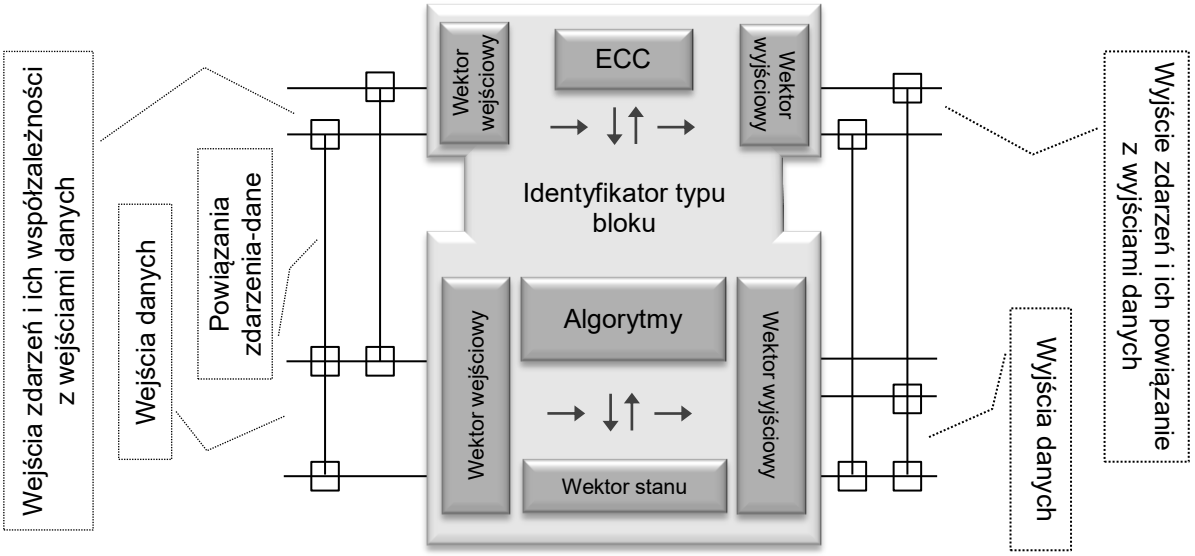
Rys. 73. Rozproszone zarządzanie zasobami
Fig. 73. Distributed management of resources

2.5.2. Blok funkcyjny

Działanie aplikacji systemowej jest opisane kodem zapisywanym w postaci graficznego diagramu. Blok funkcyjny jest jednostką takiego kodu związaną z jednostką funkcjonalną aplikacji [165]. Kod jest wykonywany w zasobach urządzeń określonych dla danej aplikacji. Urządzenia, zasoby i inne elementy systemu muszą być zgodne z IEC61499.

Z blokiem funkcyjnym wiąże się wejściowy i wyjściowy wektor danych, wejściowy i wyjściowy wektor zdarzeń oraz wektor stanu. Wektory wejścia i wyjścia stanowią interfejs do komunikacji z innymi blokami. Tak jak dla innych elementów graficznych spotykanych w graficznych językach programowania, tak i tutaj wejścia znajdują się z lewej strony sym-

bolu bloku funkcyjnego, a wyjścia z prawej. Symbolem bloku jest prostokąt, często podzielony na dwie części, z czego górna wiązana jest z obsługą zdarzeń, a dolna z obsługą danych. Graficzna reprezentacja bloku funkcyjnego została przedstawiona na rysunku 74. Zaprezentowane wejścia i wyjścia oraz ich powiązania są przykładowe i służą tylko ilustracji opisywanych własności.



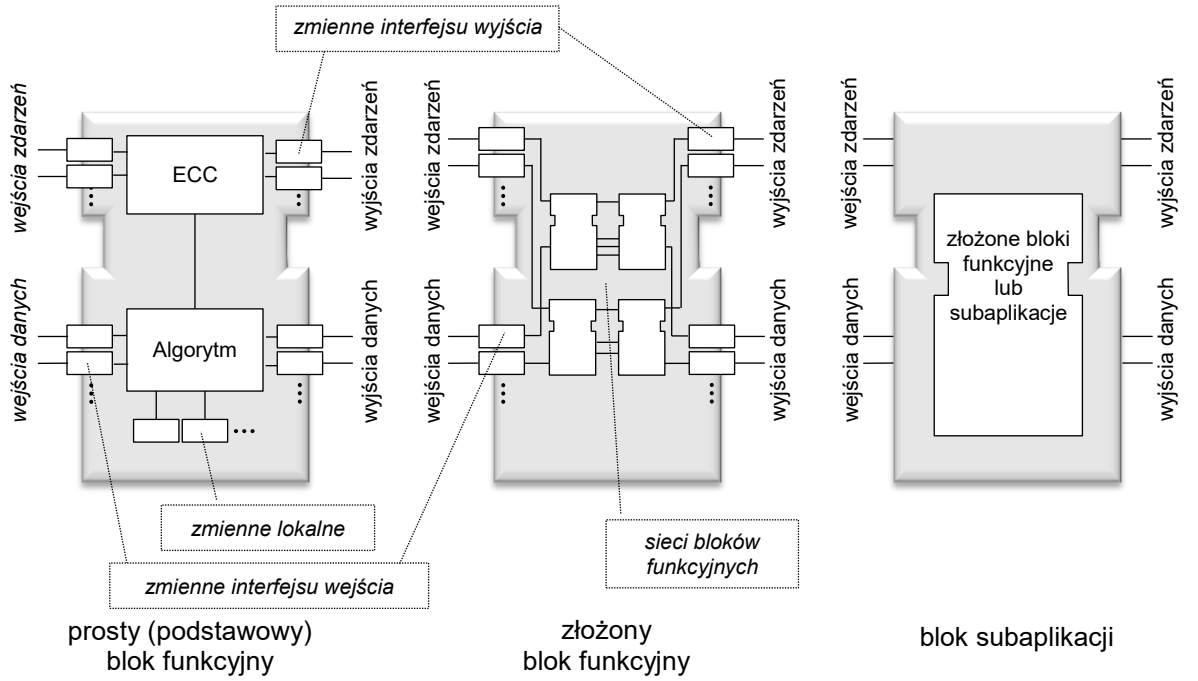
Rys. 74. Przykład reprezentacji bloku funkcyjnego podstawowego
Fig. 74. Example of basic function block representation

Blok może być typu:

- podstawowego (ang. basic),
- złożonego (ang. composite),
- usług interfejsowych (ang. service interface).

Bloki usług interfejsowych służą do uzyskania dostępu do usług systemowych środowiska IEC61499, takich jak np. usługi sieci komputerowych. Blok złożony składa się z bloków podstawowych i/lub z innych bloków. Natomiast blok podstawowy w swym składzie nie ma już innych bloków funkcyjnych, lecz algorytmy (zadania, akcje), według których dokonywane jest przetwarzanie danych w bloku. Stanowi on swoisty element grupujący, umożliwiając zagnieżdżanie bloków funkcyjnych celem uzyskania złożonych funkcjonalności na bazie innych, z reguły mniej złożonych.

Blok podstawowy zawiera schemat kontroli wykonania (ECC, ang. Execution Control Chart), służący do sterowania uruchamianiem kodu. Wykonanie algorytmu skutkuje wytworzeniem nowego wektora stanu i nowych wektorów wyjściowych bloku. Wartości wektorów wyjściowych są przekazywane do innych bloków diagramu przy użyciu linii łączących. Definiują one jednoznacznie przepływ danych i zdarzeń w rozumieniu połączeń i kierunku przepływu pomiędzy blokami. Nie definiują natomiast zależności czasowych.

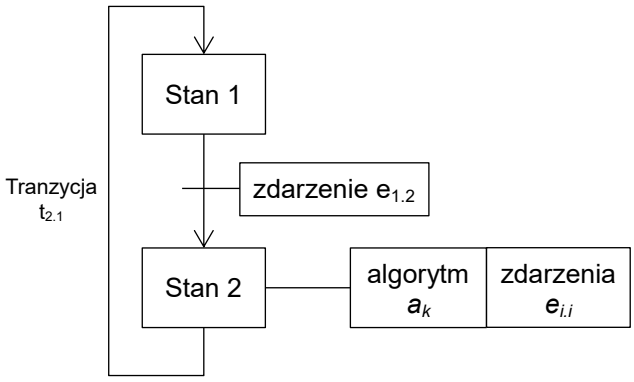


Rys. 75. Elementy aplikacji
Fig. 75. Application elements

Na rysunku 75 przedstawiono elementy aplikacji. Każdy z nich określany jest przez swój typ i urzeczywistniany przez swoją instancję w zasobie. Blok funkcyjny prosty ma interfejs zmiennych wejścia i wyjścia, niezależny dla zdarzeń i danych i podobny do interfejsów IO POU z normy IEC61131 (por. 2.6), oraz maszynę ECC i algorytm, czyli program wywoływany w określonych warunkach. Złożony blok funkcyjny, podobnie jak prosty, ma interfejs zmiennych wejścia i wyjścia oraz sieć bloków funkcyjnych, które mogą być zarówno proste, jak i złożone. Sieci może być więcej niż jedna. Kontrola wykonania sieci zależy od podłączeń zdarzeń i danych. Subaplikacja jest podobna do bloku złożonego, z tym że nie ma zmiennych interfejsowych. Zamiast tego zakłada się, że każde zdarzenie wejściowe subaplikacji jest połączone z dokładnie jednym wyjściem jednego bloku wewnętrznego lub wyjściem i każde zdarzenie wyjściowe subaplikacji jest połączone z dokładnie jednym wyjściem jednego bloku wewnętrznego. Dodatkowo subaplikacja ma specjalną własność umożliwiającą rozproszenie sieci bloków funkcyjnych pomiędzy różne zasoby. Subaplikacje mogą być zagnieżdżane.

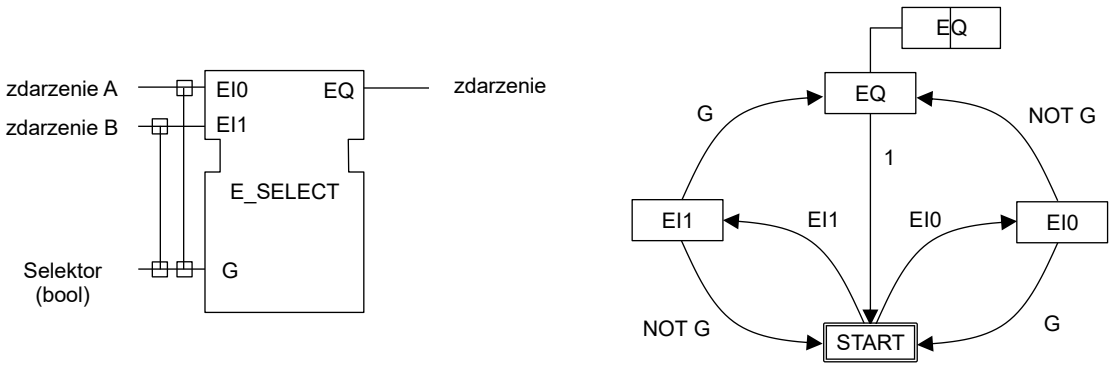
Pomiędzy zdarzeniami i danymi z wejść oraz z wyjść bloku mogą być tworzone powiązania (ang. event-data association). Reprezentowane są one liniami pionowymi pomiędzy odpowiednimi liniami wejść lub wyjść z wyraźnie zaznaczonym punktem łączenia w postaci kwadratu lub kropki umieszczanej na przecięciu linii. Powiązania nazywane są asocjacjami (ang. „with” association) wejścia lub wyjścia. Określają one, które dane są odczytywane na wejściu bloku wraz z nadejściem powiązanego zdarzenia, a które wyjścia danych są aktualizowane przy zajściu zdarzenia na wyjściu. Asocjacje dotyczą zawsze tylko danego bloku

funkcyjnego, przy którym są zdefiniowane. Mogą zatem wystąpić dwie definicje asocjacji przy przepływie danych pomiędzy dwoma blokami, jedna na początku (wyjściu bloku) i druga na końcu (wejściu bloku) danego przepływu.



Rys. 76. Maszyna stanów na bazie ECC
Fig. 76. State machine based on ECC

Asocjacje wraz z wyzwalaniem zdarzeniowym kodu (ang. event-driven code execution) umożliwia tworzenie ECC w formie maszyny stanów, i tym samym stworzenie zależności od czasu. Definiowane są stany, w których wykonywany jest kod. Zmiana stanu możliwa jest pod warunkiem wystąpienia określonego zdarzenia, warunku lub stałej. Zakończenie wykonywania kodu skutkuje wypracowaniem nowych wartości danych i stanów zdarzeń. Asocjacje sterują przypisywaniem wartości danych pomiędzy ich wewnętrzną i zewnętrzną reprezentacją w danym bloku. Na rysunku 76 zilustrowano działanie ECC. Zmiana stanu ze *Stan1* na *Stan2* nastąpi, gdy wystąpi zdarzenie $e_{1.2}$. W aktywnym stanie drugim zostanie wykonany algorytm a_k i ustawione zdarzenia $e_{i.j}$. Powrót do stanu *Stan1* zależy od tranzycji $t_{2.1}$.



Rys. 77. Przykład bloku selekcji zdarzeń
Fig. 77. Example of an event selection block

Przykład bloku z powiązaniami przedstawiono na rysunku 77. Jest to blok selekcji zdarzenia. Ma dwa wejścia zdarzeń i jedno wejście danych stanowiące selektor zdarzeń. Jeżeli na wejściach wystąpi zdarzenie, to na wyjście przekazywane jest wystąpienie zdarzenia A lub

B w zależności od wartości selektora. Na rysunku zamieszczono wygląd interfejsu takiego bloku oraz jego ECC. Algorytmem jest tu operacja przekazania zdarzenia.

Formalnie ECC można opisać przez strukturę danych (krotkę, ang. tuple) o dwóch stałych zbiorach:

$$ecc = \langle Q, T \rangle,$$

gdzie Q jest zbiorem stanów maszyny, a *T* jest zbiorem tranzycji.

2.5.3. Aplikacja

Aplikacja systemowa jest tworzona jako abstrakcja opisująca funkcjonalność systemu, przy czym zadania realizujące te funkcjonalności są rozproszone w blokach funkcyjnych i sterowane przepływem informacji. Każde wystąpienie bloku funkcyjnego w kodzie jest związane z określeniem referencji do jego instancji, najczęściej przez nazwę. Zatem użycie bloków wiąże się z koniecznością ich urzeczywistnienia przez alokacje ich instancji. Aplikacja nie zawiera powiązań z fizycznymi urządzeniami, a jedynie opisuje działanie. Nie określa, gdzie będzie wykonany dany blok. Przepływy nie są powiązane z fizycznymi sieciami, a jedynie opisują źródło i element docelowy przekazywania danych i zdarzeń. Aplikację można formalnie określić jako pięcioelementową krotkę:

$$a = \{T, I, S, V_I, V_O\}$$

gdzie:

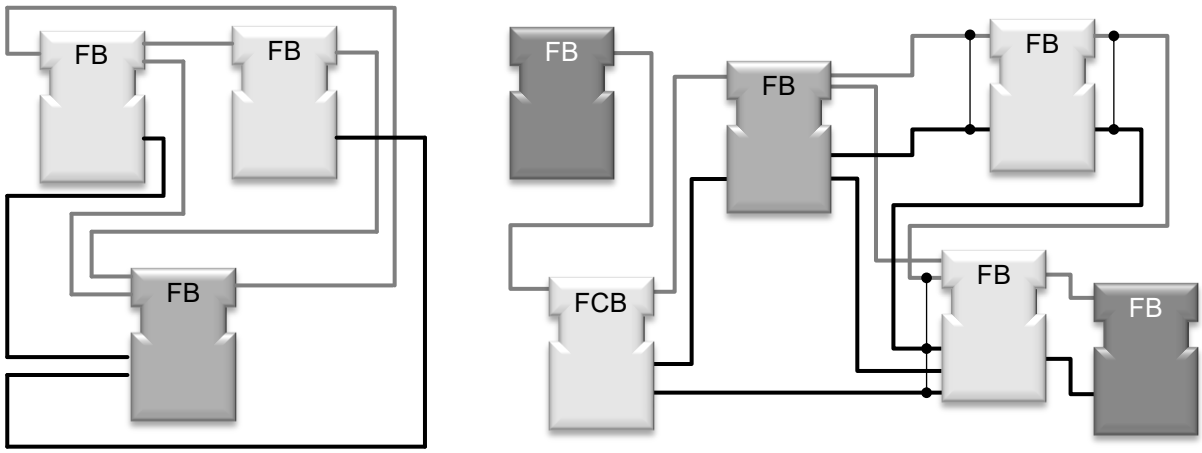
- T – zbiór dostępnych typów bloków funkcyjnych,
- I – zbiór instancji bloków w programie,
- S – przestrzeń stanów aplikacji, reprezentująca wszystkie możliwe stany wszystkich bloków aplikacji,
- V_I – wektor wejść aplikacji,
- V_O – wektor wyjść aplikacji.

Formalny opis i analizę bloków funkcyjnych i aplikacji można znaleźć w [51].

Aplikacja jest wykonywana przez środowisko uruchomieniowe zgodne z IEC61499. Może ono definiować urządzenia i zasoby zgodnie z modelami przedstawionymi wcześniej. Wykorzystywane są one do wykonywania kodu. Dzięki temu w środowisku można zmapować abstrakcyjny opis aplikacji do konkretnej konfiguracji sprzętowej. Kluczowym elementem sterującym wykonaniem bloków funkcyjnych jest mechanizm szeregowania, czyli sterowania wykonaniem bloków (ang. scheduler). Zadanie to zapewnia deterministyczny porządek wykonywania bloków, zgodnie z przepływem informacji w diagramie. Niestety, standard IEC61499 nie wprowadza formalnej semantyki opisu systemu, co skutkuje brakiem kompatybilnych platform narzędziowych i problemami z przenoszeniem aplikacji. Norma opisuje działanie bloków funkcyjnych, lecz nie definiuje zasad wykonywania diagramu bloków, czyli

zasad działania schedulera. Istnieją prace, które opisują i rozwiązują po części ten problem [51], [50], [86].

Na rysunku 78 przedstawiono przykładowe kody aplikacji składające się z kilku bloków funkcyjnych. Wypełnieniem oznaczono rodzaje bloków: ciemny szary – blok obsługi serwisowej, szary – blok złożony, jasny – bok podstawowy.



Rys. 78. Przykład przepływu informacji opisanego z użyciem bloków funkcyjnych
Fig. 78. Example of information flow described by function blocks

Należy zauważyć, że o ile deweloper dysponuje narzędziami zapewniającymi przemapowanie wirtualnych definicji urządzeń w tworzonych aplikacjach na urządzenia fizyczne, dostępne na danym obiekcie, to możliwe jest tworzenie rozwiązań całkowicie niezależnych od sprzętu. Urzeczywistnienie takiego rozwiązania następuje dopiero w momencie jego wdrażania na konkretnym obiekcie przemysłowym. Cały proces projektowania, implementacji kodu i testów może być prowadzony niezależnie od rzeczywistych urządzeń. Sprzyja to współczesnym wymogom tworzenia systemów elastycznych (zob. 1.2.3), np. [163].

Dlatego w pracach nad wdrożeniami narzędzi do projektowania systemów według koncepcji IEC61499 wykorzystuje się np. język UML. Dla przykładu w koncepcji CORFU-FBDK [355] lub [16] oprócz środków do opisu systemu wprowadzono pojęcie Systemu Wspomagania Inżynierskiego czy też środowiska inżynierskiego (ESS – ang. Engineering Support System, Engineering Environment) służącego jako narzędzie programowe wspomagające tworzenie systemów rozproszonych zgodnych z normą, umieszczono wytyczne modelu i funkcji takiego narzędzia z użyciem języka UML. Do podstawowych zadań ESS zaliczono dostarczanie wsparcia dla:

- specyfikowania bloków funkcyjnych,
- specyfikowania funkcjonalności zasobów i urządzeń,
- specyfikowania, analizy i walidacji ISP,
- konfiguracji i implementacji ISP,

- działania i utrzymania ISP,
- wymiany informacji pomiędzy narzędziami.

Słaby rozwój narzędzi inżynierskich to obecnie główny problem tego podejścia. Poniżej zamieszczono kilka przykładów projektów związanych z tworzeniem i rozwojem takich produktów.

- CORFU – Rozwijane przez Uniwersytet w Patras w Grecji. Dostarcza mechanizmów przejścia pomiędzy UML a blokami funkcyjnymi IEC 61499 [W9].
- Fbench – Projekt narzędzie graficznego typu Open Source wspierany przez Uniwersytet w Auckland w Nowej Zelandii. Bazuje na OOONEIDA Workbench [M4], [W3].
- ISaGRAF – Pierwsze narzędzie komercyjne klasy ESS. Wspierane przez firmę ICS Triplex z Kanady stanowiącą własność Rockwell Automation [W28].
- 4DIAC – Framework for Distributed Industrial Automation and Control. Niezależne narzędzie zaimplementowane jako plug-in dla środowiska Eclipse. Rozwijane przez Politechnikę Wiedeńską [W20].
- FBDK – Function Block Developement Kit – popularne narzędzie deweloperskie tworzone przez Holobloc Inc. z Ohio, USA [W22].

2.6. Podejście IEC61131

Modele i mechanizmy opisane w 2.5 dotyczą programowania systemu w ujęciu całościowym. W normie IEC61131 (por. 1.2.6) [M26] zawarto rozważania, dotyczące modeli odnoszących się jedynie do węzła systemu. Aspekty systemowe ograniczają się tu tylko do komunikacji między węzłami. Węzeł jest postrzegany jako komputerowe urządzenie sterujące, a używanym pojęciem jest „sterownik programowalny" (ang. PC – Programmable Controller). Modele zawarte w normie nie dotyczą innych typów węzłów (zobacz 2.7).

W normie wprowadzono cztery modele obrazujące różne spojrzenia na sterownik. Warto je poznać dla zrozumienia spójności idei budowy i działania takich urządzeń oraz uniwersalnego podejścia do programowania. Zrozumienie modeli pozwoli czytelnikowi na relatywnie swobodne poruszanie wśród dowolnych urządzeń zgodnych z normą zarówno w zakresie ich konstrukcji, konfiguracji, jak i wykorzystania języków programowania. Są to modele: sprzętu, komunikacji, funkcjonalny i programowania, a ich poniższy opis jest tylko naświetleniem podstawowych kwestii powiązanych z ISP wraz z komentarzem i interpretacją autora książki. Zagadnienia te są w pełni rozwinięte w dokumencie normy.

2.6.1. Model sprzętu

Na rysunku 79 przedstawiono model węzła w ujęciu sprzętowym. Jest on zilustrowany na bazie definicji zawartych w częściach pierwszej i piątej normy IEC61131. Ponadto, pojęcia związane z modelem ogólnym ISP, a także z modelem węzła ISP przedstawionym w 2.2 oraz w 2.4.2 są zbieżne z prezentowanym modelem.

Rys. 79. Ogólny model sprzętu dla urządzeń klasy PLC
Fig. 79. General hardware model for devices of PLC type

Model przedstawia podstawowe wymagane moduły (podsystemy) węzła i możliwości ich zwielokrotniania. Układ lokalny sterownika (MPU – ang. Main Processing Unit) składa się z co najmniej jednego zestawu zasobów przetwarzających, układów IO, układów komunikacyjnych i układów specyficznych dla danej konstrukcji oraz z układów zasilania. Moduły lokalne sterownika mogą występować pojedynczo lub w większej liczbie. Z jednostką lokalną mogą ponadto współpracować zdalne układy interfejsowe, jak np. zdalne moduły IO oraz inne urządzenia peryferyjne. Istotne w tym modelu jest to, że uwzględniono w nim budowę modułową z możliwością wykorzystania wielu jednostek przetwarzających oraz wielu modułów interfejsowych. Daje to możliwości konstruowania sterowników w „trzech wymiarach”:

- lokalny zestaw minimum,
- lokalna rozbudowa,
- rozbudowa zdalna.

Tym samym urządzenia bazujące na takim modelu dają potencjalnie duże możliwości dla konstruktorów, zachowując jednocześnie spójność idei. Przekłada się to w produkcie finalnym na dość swobodny dobór odpowiedniej w danym przypadku konfiguracji.

W praktyce sterowniki zgodne z tym modelem, w wersji minimalnej, mają jednostkę centralną (CPU) z jednym zasobem przetwarzającym (procesor i pamięć), zintegrowanym zasi-

laczem, układami IO i interfejsem komunikacyjnym umożliwiającym podłączenie układów zdalnych. W wersjach bardziej rozbudowanych możliwe jest stosowanie zwielokrotniania modułów w jednej konfiguracji. Skutkuje to możliwością użycia redundantnych zasilaczy, wielu jednostek centralnych, dodatkowych modułów przetwarzających, czy też wielu różnych modułów komunikacyjnych i IO.

Producenci zwykle wprowadzają do swoich rozwiązań wiele unikalnych funkcjonalności, aplikują techniki i wykorzystują technologie, które nie są objęte normalizacją. Dlatego nie sposób jest precyzyjnie kategoryzować dostępne na rynku produkty. Wielu dba jednak o to, aby w wymaganym zakresie ich produkty były zgodne z wymaganiami normy[36].

Warto zwrócić uwagę, że model sprzętu wg ww. normy limituje liczbę niezbędnych modułów do jednego, zatem urządzenie zgodne z normą niekoniecznie będzie miało wiele różnych i zaawansowanych funkcji rozbudowy. W procesie doboru inżynier powinien zawsze zapoznać się z możliwościami rozbudowy danej konstrukcji i jej zakresem zgodności z normą.

2.6.2. Model komunikacji

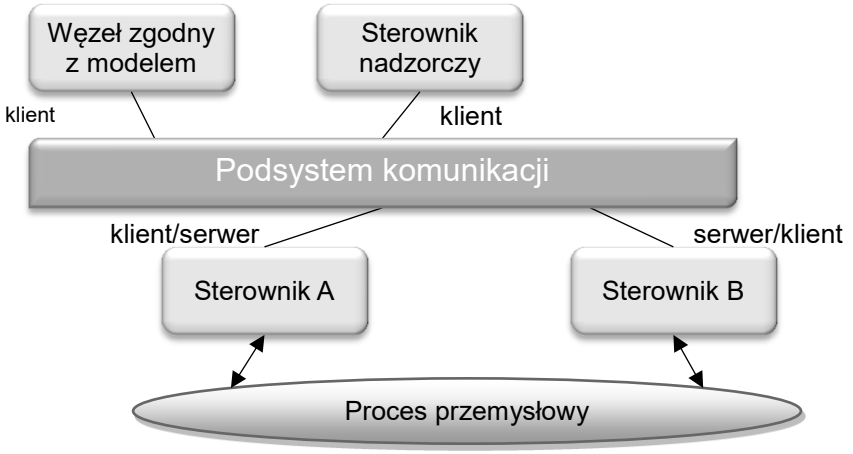
Prezentowany model komunikacji jest zgodny z abstrakcjami przedstawionymi w 2.4.4 do 2.4.7. Bazuje on na wytycznych zawartych w piątej części normy IEC61131. Model dotyczy ogólnej abstrakcji przekazywania danych i abstrahuje od protokołów, stosów i innych pojęć związanych z sieciami komputerowymi.

W ISP PLC musi dostarczać pewnych funkcjonalności komunikacyjnych umożliwiających przekazanie danych do innych urządzeń. Funkcjonalności te są realizowane przez tzw. podsystem komunikacyjny. Na tym pojęciu bazuje model. Podsystem realizuje zadania i raportuje wynik ich wykonania, a w szczególności błędy, do sterownika. W modelu przewiduje się trzy typy urządzeń:

- sterownik programowalny (procesowy),
- sterownik nadzorczy,
- inny węzeł systemu komunikujący się ze sterownikami.

Urządzenia te są powiązane w relacjach aplikacyjnych typu Klient-Serwer. Każdy typ urządzenia może działać jako klient lub jako serwer. Sterownik zgodny z tym modelem może komunikować się z dowolnym innym zgodnym urządzeniem przy użyciu swoich funkcjonalności klienta, włączając w to węzły niestanowiące sterowników. Przedstawiono to na rysunku 80.

[36] Przykładem może być sterownik XC100 serii xSystem firmy Eaton (Moeller) lub sterownik CX9010 firmy Backhoff. Sterowniki bardziej rozbudowane będą dysponowały możliwością zestawienia MPU z modułów wraz z możliwością ich zwielokrotnienia. Przykładem mogą być sterowniki Modicon M340 firmy Schneider czy Simatic S7-300 firmy Siemens.



Rys. 80. Ogólny model komunikacji dla urządzeń klasy PLC
Fig. 80. General communication model for devices of PLC type

Zakres działania podsystemu komunikacji dotyczy zarówno dostarczania informacji o jakości funkcjonowania elementów (podsystemów) sterownika, jak i definiuje podstawowe usługi aplikacyjne. Nie definiuje natomiast konkretnych rozwiązań komunikacyjnych.
Informacja statusowa opisuje układy według następujących koncepcji:

- zdrowie (ang. health)
  Oznacza ogólny stan funkcjonowania danego elementu. Każdy element jest opisany takim statusem. Określenie jakości może przyjmować tylko jeden określony stan z trzech:
  - good – element jest sprawny i funkcjonuje poprawnie; bez zagrożenia działania związanych z nim zadań,
  - warning – element funkcjonuje, ale jego poprawne działanie jest zagrożone; oznacza to, że realizowanie zadań jest nadal możliwe, jednak występują pewne ograniczenia,
  - bad – element jest niesprawny; nie jest możliwe realizowanie powierzonych mu funkcji.

- stan (ang. state)
  Określenie stanu działania elementu może zawierać listę atrybutów, których wartości są różnych typów i różne. Mogą to być informacje o uruchomieniu, wymuszeniach, dostępności, zabezpieczeniach, trybach pracy, rodzaju błędów itp.

Informacja statusowa jest dostępna w strefie systemowej (%S) pamięci sterownika i jest dostępna przez konkretny adres ze strefy (zależny od implementacji):

- %SC – status podsystemu,
- %SU – typ podsystemu,
- %SN – nazwa podsystemu,

- %SS – stan podsystemu,
- %SI – status specyficzny zależny od implementacji,

oraz przez predefiniowane zmienne P_PCSTATE oraz P_PCSTATUS. Sygnalizacja jakości funkcjonowania elementów sterownika dotyczy:

- sterownika jako całości – informacja opisuje status urządzenia, włączając ogólny status jego modułów uwzględnionych w modelu sprzętu,
- układów IO,
- jednostki przetwarzającej,
- układów zasilania,
- układów pamięci,
- podsystemu komunikacji,
- systemów specyficznych dla konkretnej implementacji sterownika.

Usługi aplikacyjne definiują cały szereg funkcji związanych z zachowaniem sterownika względem innych urządzeń. Model nie definiuje jednak zagadnień przekierowywania danych poprzez sterownik. W zakres usług wchodzi:

- weryfikacja zdolności urządzenia do działania – zdalny odczyt statusów innych węzłów,
- akwizycja danych – usługi pozyskiwania danych z innych urządzeń; wyróżnia się trzy metody:
  - odpytywanie (ang. pooling) – odczyt danych determinowany przez aplikację klienta,
  - zaplanowanie (ang. programmed) – aktualizacja danych determinowana przez aplikacje serwera,
  - konfigurowanie (ang. configured) – komunikacja wymuszona przez klienta na skutek zdalnego skonfigurowania interfejsu serwera do przesyłu danych do klienta,
- kontrola (ang. control) – wpływanie na działanie innego węzła; działa w dwóch trybach:
  - parametrycznym (ang. parametric) – gdy kontrola odbywa się przez wpisanie odpowiednich wartości do zmiennych lokalnych węzła, wpływając tym samym na działanie jego aplikacji,
  - blokowanym (ang. interlocked) – gdy klient żąda od serwera wykonania konkretnej operacji i dostarczenia potwierdzenia efektu jej realizacji. W tym przypadku konieczne jest synchronizowanie aplikacji kilenta i serwera,

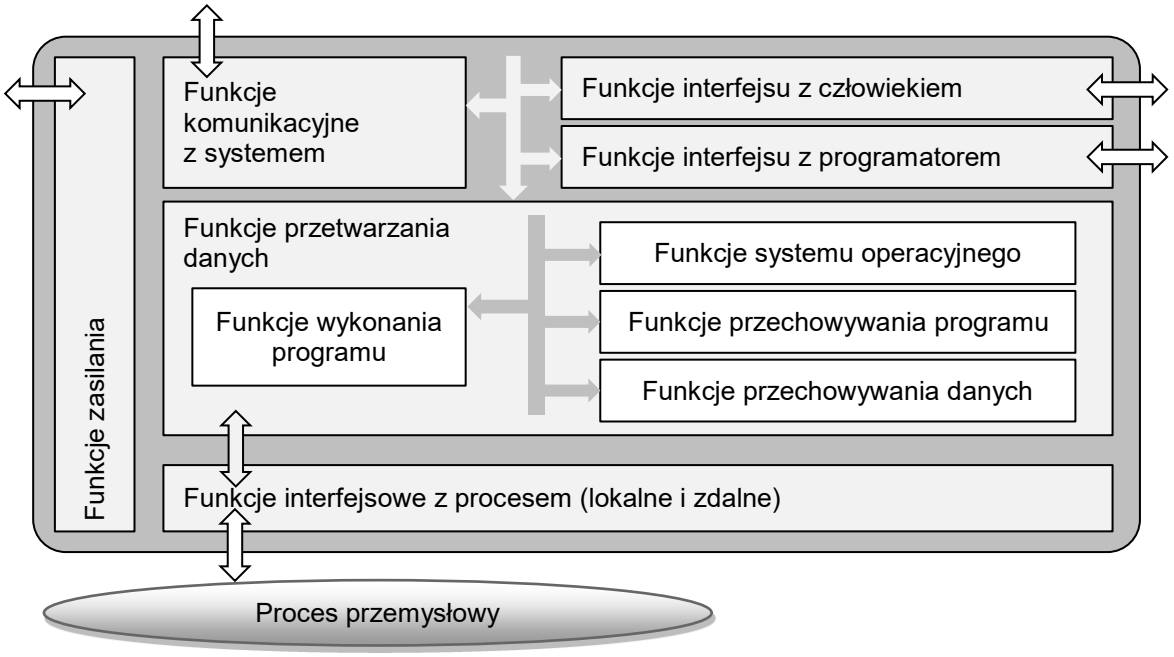
- synchronizacja – mechanizm synchronizowania w czasie działań aplikacji, dotyczy wstrzymywania i uaktywniania wykonania operacji w funkcji działania innej aplikacji,
- informowanie o alarmach – usługi nieżądanej i niezaplanowanej sygnalizacji o wystąpieniu predefiniowanych zdarzeń,
- wykonanie programów – usługa kontrolująca działanie programów aplikacyjnych oraz trybu dostępu do danych IO. Program aplikacyjny może być uruchomiany lub zatrzymany. Uruchomienie może zostać dokonane ze stanu początkowego uwzględniającego wartości inicjujące oraz z zachowaniem stanu (zob. 2.4.6),
- kontrola układów IO – stan IO jest klasyfikowany jako:
  - kontrolowalny (ang. controlled) – informacja reprezentowana przez układ IO zależy od działania programu,
  - wstrzymane wyjścia (ang. hold outputs) – stan wyjść nie jest zależny od wykonania kodu programu, jest natomiast zapamiętany i niezmienny. Stan wejść jest związany i odstępny dla programu,
  - wstrzymanie stanu (ang. hold current state) – stan wejść nie jest przekazywany do programu, tak jak i wyjścia nie są aktualizowane danymi wypracowanymi przez program,
  - dla poniższych stanów, ich dostępność jest tylko podczas zatrzymania wykonania programu, a stan wejść nie jest określony. Są to stany:
  - producenta (ang. implementer state) – stan wyjść jest determinowany przez specyfikę urządzenia,
  - wyzerowany (ang. zero outputs) – wyjścia są wyzerowane,
  - użytkownika (ang. user specific) – stan wyjść zależy od ustawień użytkownika (programistę),
- transfer kodu i danych programów – usługa dotyczy odczytu i zapisu (ang. upload, download) danych i kodu programu lub ich części do lub z pamięci, w której są one składowane, a w szczególności programatora. Inicjacja usługi, niezależnie od kierunku transferu, leży po stronie aplikacji programatora. Usługa ta dotyczy:
  - odczytu w celu weryfikacji danych, programu lub konfiguracji ze wzorcem (np. projektem źródłowym),
  - odczytu do archiwizacji,
  - zapisu w celu przywrócenia poprzednio działającego systemu,
  - zapisu programu, danych i konfiguracji przygotowanych w trybie off-line,

- zarządzanie połączeniem – obsługa połączeń z innymi węzłami.

We wszystkich przypadkach kontrola dostępu leży po stronie, z której informacja jest pobierana lub modyfikowana.

2.6.3. Model funkcjonalny

Na rysunku 81 przedstawiono zbiór minimum funkcji sterownika wymaganego w normie. W swej idei model ten jest zgodny z modelem węzła przedstawionym w 2.4.2.



Rys. 81. Ogólny model funkcjonalny dla urządzeń klasy PLC
Fig. 81. General functional model for devices of PLC type

Grupy funkcji dotyczą przetwarzania danych na rzecz zadań funkcjonalnych aplikacji węzła oraz komunikacji z procesem, systemem i człowiekiem. Wszystkie funkcje komunikacyjne wymagają użycia protokołów. Natomiast nie wszystkie protokoły muszą pracować z ograniczeniami czasowymi. Protokoły dla funkcji kontaktu z użytkownikiem-człowiekiem mogą być dowolne zapewniające wymagany interfejs. Protokoły programatora muszą współpracować z węzłem ze znanym i określonym wpływem na jego działanie, ale same nie muszą gwarantować zdeterminowanej w czasie komunikacji.

Zakłada się istnienie przynajmniej jednej funkcji komunikacyjnej. Węzeł ISP bez takiej funkcji jest nieprzydatny do zastosowań praktycznych. Wymagane jest, aby funkcje interfejsowe z procesem posiadały atrybuty umożliwiające określenie stanu oddziaływania z obiektem. Wyróżnia się dwa stany oddziaływania, niezależnie dla wejść oraz dla wyjść:

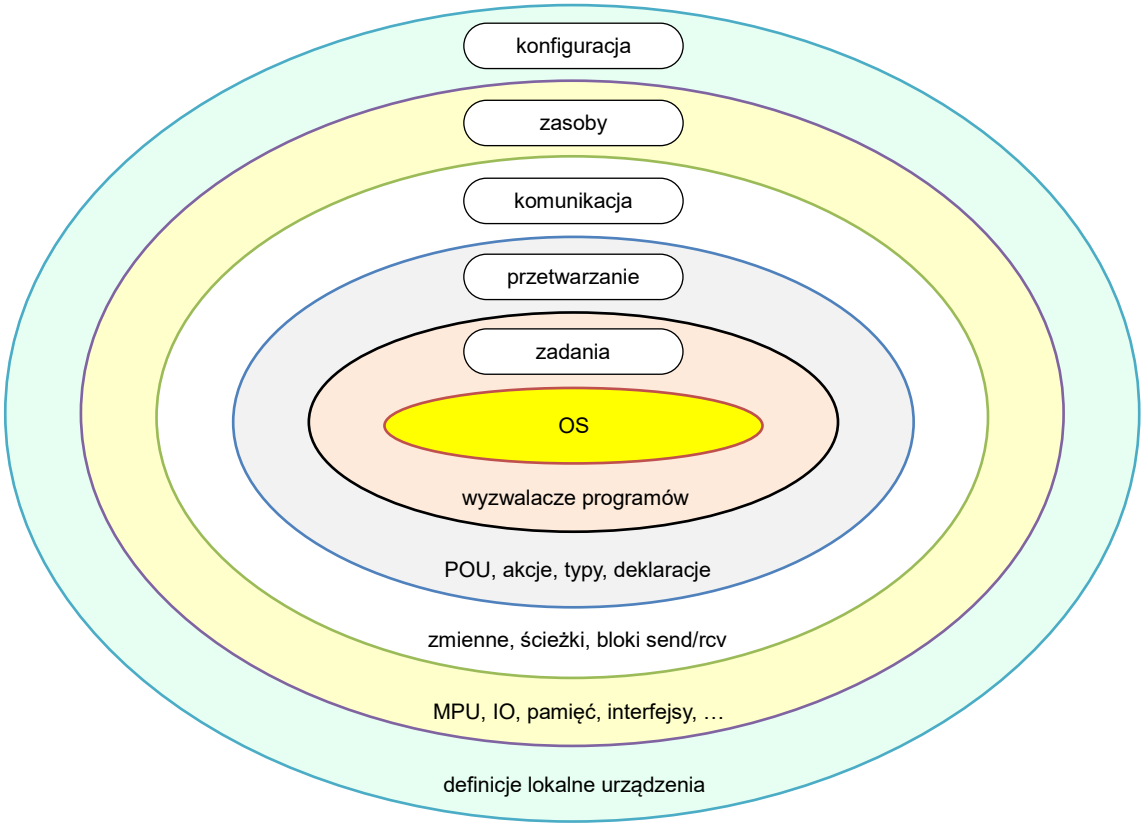
- Wstrzymana aktualizacja obrazu wejść/wyjść – bieżący wektor IO nie jest aktualizowany bieżącymi wartościami z wejść fizycznych, a wyjścia fizyczne nie są aktualizowane obrazem z pamięci.
- Aktywna aktualizacja stanu wejść/wyjść – obraz wejść jest aktualizowany zgodnie ze stanem sygnałów z obiektu, a stan obiektu jest aktualizowany obrazem z pamięci.

2.6.4. Model programowy

Aby stworzyć, zainstalować i uruchomić aplikację w urządzeniu komputerowym, w standardzie IEC61131-3 [M21] zdefiniowane są pewne elementy wspólne. Są one wspierane natywnie przez OS urządzenia i tym samym wspomagają programistę w wykonaniu tego zadania. Do elementów tych należą:

- Jednostki kodu (POU – ang. Program Organization Unit)
  Są to elementy organizacyjne odnoszące się zarówno do definicji, jak i instancji bloku kodu stanowiącego odrębną całość i mogącego zostać wykonanym. POU mogą być wykonywane rekurencyjnie. Istnieją trzy rodzaje POU:
  - programy (PRG, ang. program),
  - funkcje (FC, ang. function),
  - bloki funkcyjne (FB, ang function block).
- Elementy konfiguracji
  Są to elementy organizacyjne odnoszące się do definicji fizycznego sprzętu oraz abstrakcyjnych elementów definiujących zarządzanie jednostkami kodu i komunikacją między nimi. Należą do nich:
  - konfiguracja (ang. configuration),
  - zasoby (ang. resource),
  - zadania (ang. task),
  - komunikacja
    - zmienne globalne (ang. global variable),
    - zmienne bezpośrednie (ang. direct variable),
    - ścieżki dostępu (ang. access path),
    - mechanizm przekazywania danych (bloki funkcyjne SEND/RCV).
- Definicje inicjujące instancji (ang. instance-specific initialization).

Dodatkowo do elementów wspólnych należą deklaracje elementów programowych, rodzaje zmiennych, typy oraz sposoby prezentacji danych. Ogólny model warstw programowych został przedstawiony na rysunku 82.



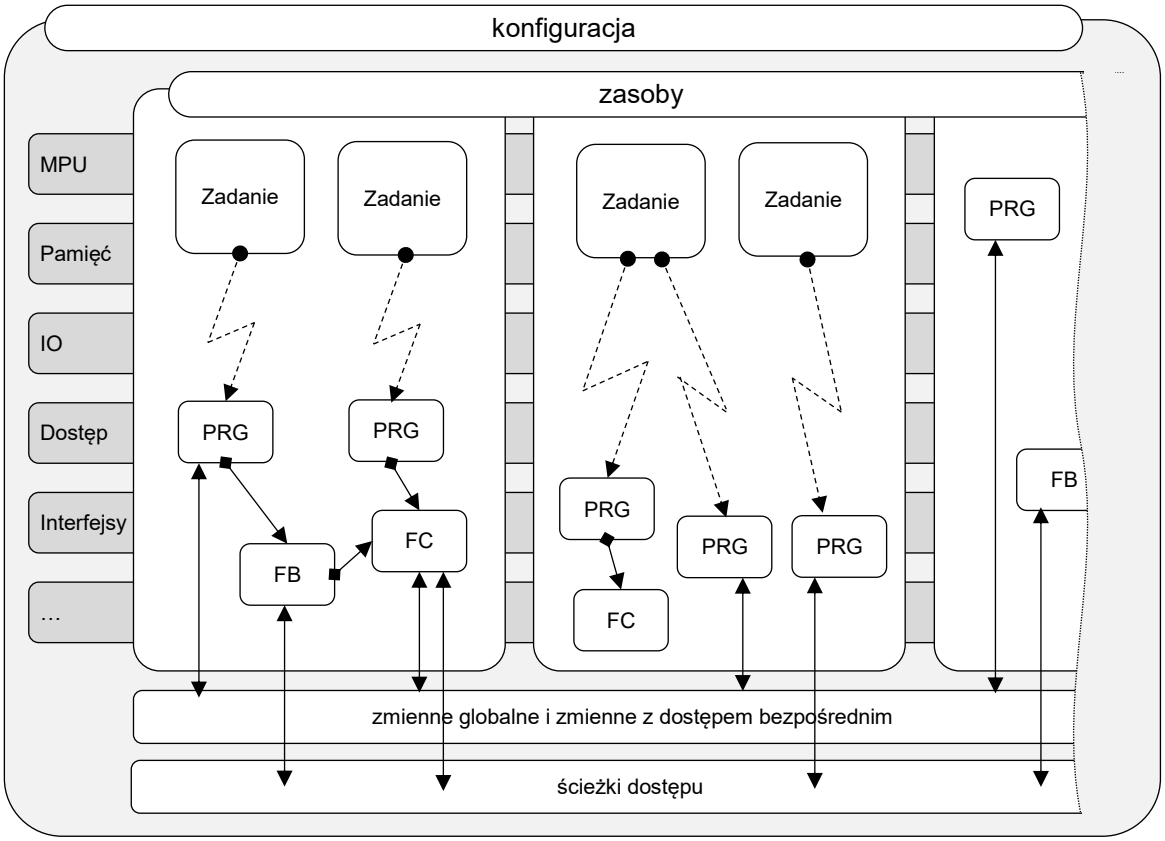
Rys. 82. Ogólny model warstw programowych
Fig. 82. General model of program layers

W modelu programowym istnieją pewne poziomy abstrakcji, dla których tworzy się definicje związane z konkretną aplikacją oraz przez które następuje współpraca w ramach działania aplikacji lokalnej. Można go interpretować jako warstwy zależności. OS steruje zadaniami, które uruchamiają programy współpracujące ze sobą przez mechanizmy komunikacji, działając w określonych zasobach zdefiniowanych w ramach konkretnej konfiguracji.

Konfiguracja jest zestawem definicji określających, jakie zasoby fizyczne są wykorzystywane do realizacji danego zadania w urządzeniu i jak mają być obsługiwane przez OS. Wchodzą w to wszelkie zasoby sprzętowe i programowe urządzenia, w tym konfiguracja MPU, przestrzeni adresowej i kanałów IO, interfejsów komunikacyjnych, zagadnienia bezpieczeństwa itp. elementy definiujące działanie urządzenia. Przez urządzenie rozumie się kompletny, dla danego przypadku, komputer (podsystem lokalny), czyli jednostki centralne, zasilacze oraz wszelkie występujące moduły.

Zasoby to zestaw wszystkich niezbędnych elementów, umożliwiających przetwarzanie danych, czyli wykonanie programu. W skład konfiguracji danego urządzenia może wchodzić wiele zasobów przetwarzających. Do zasobów można zaliczyć procesor, pamięć, układy wewnętrzne i inne mogące działać niezależnie względem wykonania programów.

Zadanie to zdefiniowane w ramach zasobów działanie, związane z zarządzaniem wykonywaniem programów. Jest to rodzaj konfigurowanego wyzwalacza, który może uruchomić program w określonych warunkach, np. cyklicznie lub od predefiniowanego zdarzenia. Zadania mogą, ale nie muszą być definiowane w konfiguracji. Gdy brak jest przypisania programu do zadania, program główny jest wykonywany cyklicznie.



Rys. 83. Ogólny model programowy z przykładowymi powiązaniami
Fig. 83. General program model with sample connections

Na rysunku 83 zamieszczono ogólny model programowy (ang. software model), na którym przykładowo pokazano powiązania pomiędzy programami, blokami funkcyjnymi, funkcjami, zadaniami oraz mechanizmami komunikacji międzyelementowej.

Definicja programu i funkcji nawiązuje do klasycznych definicji znanych z inżynierii programowania. Programy są to jednostki kodu, które zawierają logicznie powiązane elementy języka programowania. Umożliwiają przetwarzanie danych na rzecz obsługi procesu przemysłowego dokonywanego przez urządzenie komputerowe, na którym zostały uruchomione. Program jest wołany przez system operacyjny. Programy mogą mieć swoje zmienne lokalne oraz mogą wołać inne POU.

Funkcje są to POU, które mogą mieć parametry, zwracają dokładnie jedną wartość określonego typu i są wołane przez inne POU. Najważniejszą jednak cechą funkcji jest brak sta-

nu. Blok funkcji nie przechowuje stanu i w działaniu przypomina automat kombinacyjny. Wywołanie funkcji nie wymaga tworzenia jej instancji. Dane zwracane zależą tylko od parametrów funkcji.

Inaczej działa blok funkcyjny. Może on przyjmować wiele parametrów oraz zwracać wiele wyników. Posiada i przechowuje stan wewnętrzny, czyli przypomina działanie automatu sekwencyjnego. Może być wołany z innego POU. Wywołanie, z racji przechowywania stanu, wymaga utworzenia instancji.

Oprócz powyższych trzech rodzajów POU, w IEC61131-3 spotyka się jeszcze jeden typ organizacji kodu. Jeżeli z aplikacją związana jest jakaś struktura opisująca wykonanie programów (np. z językiem SFC), to z nią wiążą się często tzw. akcje. Akcje nie stanowią POU. Akcja może być zmienną boolowską lub sekwencją kodu aktywowaną w określonych momentach czasu i na określony czas. Aktywacji dokonuje OS w funkcji stanu programu i definicji w postaci tzw. kwalifikatora.

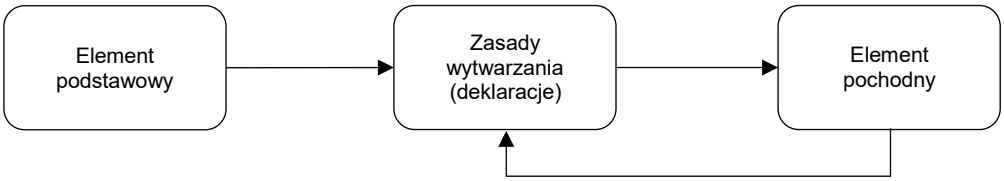
Pomiędzy elementami konfiguracji występuje komunikacja. Jest ona dokonywana przez zmienne globalne, zmienne o dostępie bezpośrednim, przez ścieżki dostępu lub przy użyciu specjalnych rozkazów Send/Receive. Dostęp bezpośredni dotyczy odwołania do zmiennej przez adres (ang. direct). Ścieżka dostępu (ang. access path) jest to konwencja odwołań do elementów konfiguracji w postaci powiązanych ze sobą zdefiniowanych nazw symbolicznych i służy do uzyskania dostępu do danego elementu znajdującego się poza aktualnym obszarem widoczności. Dodatkowo bloki mogą wymieniać dane przez interfejsy zmiennych wejścia i wyjścia. Są to zmienne lokalne bloku widziane na zewnątrz jako wejście, wyjście lub jedno i drugie. Dzięki połączeniom na poziomie wołającym można przekazywać dane, oczywiście o zgodnym typie. Komunikacja nie ma nic wspólnego z sieciami i modułami komunikacyjnymi. Jest to pojęcie o jedną klasą abstrakcji wyżej. Niemniej jednak jest to mechanizm umożliwiający komunikowanie elementów zarówno w zakresie danej konfiguracji, jak i między różnymi konfiguracjami. Natomiast sieci, moduły czy wspólna pamięć mogą stanowić środki do jej realizacji. Mechanizmy komunikacji międzykonfiguracyjnej bazują na definicjach z IEC61131-5 [M22] i modelu przedstawionym w 2.6.2.

2.6.5. Model programowania

Model programowania dotyczy sposobów wytwarzania kodu na bazie elementów programowych zdefiniowanych w normie. Podstawowymi elementami są:

- typy danych,
- funkcje,
- bloki funkcyjne,
- programy i zasoby.

Wytworzenie nowego elementu programowego, tzw. elementu pochodnego, dokonuje się przez użycie elementów podstawowych i ewentualnie istniejących już elementów pochodnych wraz z ustalonymi w normie deklaracjami. Dzięki temu programista może utworzyć własne typy, funkcje, bloki funkcyjne, programy lub całe konfiguracje, przy czym deklaracje elementów pochodnych muszą mieć charakter globalny, dzięki czemu są one zawsze dostępne do dalszej produkcji elementów pochodnych. Zostało to przedstawione na rysunku 84.



Rys. 84. Ogólna ilustracja modelu programowania
Fig. 84. General view of the programming model

Nie należy traktować takiego modelu jako czegoś innowacyjnego. Jest to bardzo typowy mechanizm w inżynierii programowania. Podobnie sprawa wygląda z wieloma innymi wymaganiami normy, jak chociażby istnienie zmiennych, stałych, symboli, rekurencji, strukturalności i możliwości dekompozycji kodu, zakresów widoczności itp. Należy spojrzeć na to zupełnie z drugiej strony. Istotne jest to, że standard wymusza istnienie takich mechanizmów. Przystępując do programowania urządzenia zgodnego z normą, programista może oczekiwać istnienia pewnych znanych mu mechanizmów, których dostępność w kontekście programowania specyficznych komputerów do specjalistycznych zadań może nie być oczywista. Poza-elementarnymi wymaganiami względem programowania, w standardzie pojawia się cały szereg charakterystycznych dla dziedziny mechanizmów. Najbardziej oczywiste z nich to specjalistyczne języki wspomniane wcześniej oraz istnienie swoistych deklaracji rodzajów zmiennych, jak np. deklaracje dla zmiennych zachowywanych, interfejsowych wejścia i wyjścia, wykrywania zboczy, czy deklaracje wiązania symboli z adresami i inicjalizacji wartości. Do tworzenia kodu w normie wprowadzono pięć języków podzielonych na trzy grupy względem formy i zastosowań:

- języki tekstowe ogólnego przeznaczenia
  - ST (tekstowy język strukturalny, ang. Structured Text)

  Język uniwersalny i zwięzły w formie. Dedykowany do programowania na wysokich abstrakcjach (wysokiego poziomu, jak np. Pascal, C).
  - IL (sekwencja instrukcji, ang. Instruction List)

  Język uniwersalny i zwięzły w formie. Dedykowany do programowania na niskich abstrakcjach (niskiego poziomu, jak np. asembler).

- języki graficzne ogólnego przeznaczenia
  - LD (diagram drabinkowy, ang. Ladder Diagram)

  Język uniwersalny, choć głównie dedykowany do tworzenia obwodów (sieci) logicznych i operacji na danych dyskretnych. Z racji formy graficznej nie jest tak w niej zwięzły jak języki tekstowe.
  - FBD (diagram bloków funkcyjnych, ang. Function Block Diagram)

  Język uniwersalny, ale rozwlekły w formie. Sprawdza się przy tworzeniu funkcji logicznych.
- język graficzny opisu działania sekwencyjnego
  - SFC (sekwencyjny schemat funkcyjny, ang. Sequential Function Chart)

  W przeciwieństwie do powyższych, język ten nie jest językiem służącym do przetwarzania danych. Jest to język opisu działania abstrakcyjnego automatu sekwencyjnego (por. 1.2.4). Do takiego opisu można wiązać kod przetwarzania w innych językach[37].

Więcej o językach można znaleźć w wielu publikacjach, np. [192], [187].

2.7. Podejście CPS

Na koniec omawiania modeli warto wspomnieć o stosunkowo nowym podejściu, które nie stanowi stricte modelu, ale oddaje ideę konstrukcyjną systemów i staje się w ostatnich latach coraz bardziej popularne. CPS to skrót od ang. Cyber-Physical Systems, czyli termin można tłumaczyć jako systemy cyberfizyczne. Dziedzina stosowalności takich systemów jest szeroka, ale głównie dotyczy komputeryzacji procesów produkcyjnych i dlatego plasuje się w klasie systemów informatycznych przedsiębiorstw, w tym również ISP. Rozważając CPS, należy jednak odnieść się do ogólnych modeli systemów przedsiębiorstwa przedstawionych w 1.2.1, a nie tylko do niskopoziomowych ISP. CSP zostały pomyślane jako systemy jednolite i trudno je odnosić do jakiejś konkretnej warstwy bądź funkcjonalności. W tym kontekście CPS stanowi element filozofii Industry 4.0 [182], [171], [347], [85].

Idea systemu jest dość przejrzysta, a genezą sięga do propozycji z początku lat dziewięćdziesiątych ubiegłego wieku. Proponowano wówczas osadzanie niewielkich komputerów komunikacyjnych przy każdym, nawet najprostszym urządzeniu obiektowym i rozpraszanie elementów i funkcjonalności systemu na poziom pojedynczych sensorów i układów wykonawczych oraz ich aplikacji. Idea nie była zła. Niestety, na przeszkodzie realnemu rozwojowi stanęły ograniczenia techniczne sieci, brak solidnego podejścia informatycznego oraz słaba promocja idei. W efekcie pojawiła się idea inteligentnych sensorów i układów wykonaw-

[37] Jednym z pierwszych języków tego typu był Grafcet. Obecnie spotyka się języki SFC, Graf, Hi-Graf, Graftec.

czych oraz popularny do dziś model warstwowy implementowany w postaci tzw. NCS (zob. 1.1.5). Poza tym, wówczas trwała tzw. „wojna fieldbusowa" (ang. fieldbus war) [106], [287] i niewiele firm było w stanie wznieść się ponad swój partykularny interes i rozwinąć technologie integracyjne. Obecnie rozwój i popularność systemów wbudowanych oraz pojawienie się sieci 2G/3G (por. 2.4.4) spowodowało, że idea wraca pod postacią CPS.

Idea CPS polega na komputeryzacji fizycznych elementów obsługujących proces przemysłowy, czyli na wbudowywaniu w nie „inteligencji" w postaci komputerów i funkcjonalności komunikacyjnych. Często CPS są właśnie określane jako systemy z wbudowaną inteligencją (ang. embedded intelligence). Dzięki nowoczesnym i wydajnym interfejsom komunikacyjnym, z założenia opartym na sieciach 2G/3G, możliwe jest budowanie silnie rozproszonych systemów informatycznych bezpośrednio zintegrowanych z procesem. W efekcie powstają „inteligentne" rozległe struktury fizycznych urządzeń procesowych zdolnych do jednoczesnego rozproszonego oddziaływania na proces i rozproszonego przetwarzania informacji, w tym do wykonywania procesów decyzyjnych dotyczących prowadzenia i planowania produkcji. To właśnie stanowi różnicę między klasycznym systemem wbudowanym a CPS. Elementy systemu CPS nie tylko są zintegrowane z urządzeniem i przetwarzają informację, ale są zawsze usieciowione, rozproszone i stanowią element większej całości. Podstawą działania CPS jest wydajna i jednolita wymiana danych oraz zaawansowane przetwarzanie rozproszone w aplikacji wirtualnej (por. 2.3, 2.4.5). Ani fizyczne, ani logiczne rozproszenie aplikacji nie powinno być widoczne dla funkcjonalności systemowych (ang. seamless data transfer). W klasycznym podejściu producenci informacji (por. 1.1.1) zarówno w postaci układów sensorowych, jak i urządzeń, maszyn i całych procesów dostarczają różnego typu informacji w postaci danych aktualizowanych z określonym cyklem. Jak już wspomniano, we współczesnych systemach produkcyjnych, z racji wzrostu złożoności informacyjnej, wymagań integracyjnych oraz wymagań rekonfiguracyjnych, liczba i typy przetwarzanych danych stale się zwiększają. Nie chodzi tu o rozmiar reprezentacji. Pomiary np. 12-bitowe zawsze będą 12-bitowe, ale ze względu na bardziej zaawansowane technologie i bardziej dokładny opis procesu ich liczba może być większa, jak również strumień danych może być gęstszy. Dla przykładu 12 bitów to niewiele, ale 12 bitów co 0.1 ms daje już 120 tysięcy bitów na sekundę i to z jednej tylko zmiennej. Ponadto, w opisie informacyjnym nietrywialnego ISP obsługiwane są nie tylko pomiary. Rodzaje kodowanej informacji, sposoby ich reprezentacji, przynależność do obiektu i poziom opisywanych abstrakcji mocno różnicują obsługiwane dane. Oprócz danych procesowych, dochodzą dane z ich przetwarzania, analiz i szerokiej gamy procesów biznesowych. Dane te trzeba pozyskać, przesyłać, składować, wyszukiwać, ale przede wszystkim przetwarzać na różne sposoby zależnie od kontekstu użycia. Można stwierdzić, że obecnie w ISP pojawiają się zagadnienia przetwarzania znane jako

ang. Big Data [411]. CPS to systemy, które służą obsłudze Big Data w kontekście prowadzenia procesów produkcyjnych w sposób automatyczny, dynamiczny i adaptacyjny. Rozproszenie „inteligencji", czyli przeniesienie realizacji zadań do elementów, których te zadania dotyczą, powoduje, że CPS stają się rozwiązaniami cechującymi się pseudosamoświadomością, a słowem, które określa atrybuty działania systemu, jest często „samo" (ang. self). I tak zamiast analizować stan urządzeń pod kątem ich stanu, urządzenia mogą same zająć się swoimi zadaniami, wymieniając tylko niezbędne informacje z innymi elementami systemu. Mówi się o cechach ang. self-aware i self-predict, którymi wykazują się węzły systemu w zakresie analizy przebiegu pracy i diagnostyki, oraz ang. self-configure/organize/maintain w zakresie prowadzenia procesu produkcyjnego.

Ogólnie, działanie CPS na rzecz obsługi produkcji ma dwa obszary działania:

- ang. Physical Space (przestrzeń fizyczna)

  Odpowiada za dwukierunkową łączność (ang. connectivity), czyli przekazywanie informacji w czasie rzeczywistym pomiędzy światem fizycznym a systemową cyberprzestrzenią.

- ang. Cyber Space (cyberprzestrzeń)

  Odpowiada za mapowanie i zarządzanie danymi, ich przetwarzanie i analizę. W cyberprzestrzeni tworzony jest obraz tzw. bliźniaka (ang. cyber-twin), który tworzy parę z obiektem fizycznym i stanowi jego cyfrową reprezentację (instancję opisu informacyjnego).

W efekcie uzyskuje się obraz działania rzeczywistych układów oraz możliwość symulacji i predykcji parametrów produkcyjnych na bazie tego obrazu i wbudowanej w system wiedzy ekspertowej. Zakres oddziaływania CPS odnosi się do komponentów fizycznych (węzłów), ich układów funkcjonalnych (maszyn, linii produkcyjnych, procesów technicznych) i całych procesów jednostek produkcyjnych, w tym procesów biznesowych. Architekturę CPS charakteryzuje pięciowarstwowy model zwany 5C, składający się z pięciu grup funkcji:

- ang. Smart Connection Level (warstwa połączeń)

  Odpowiada za przekazywanie danych w czasie rzeczywistym pomiędzy elementami świata fizycznego i warstwami wyższymi. Do świata fizycznego zalicza się zarówno węzły proste (np. sensorowe, wykonawcze), węzły złożone (np. sterowniki, komputery), węzły interfejsowe (stacje SCADA, MES, ERP), jak i wszelkie inne węzły mające kontakt ze światem fizycznym. Dane są przechowywane w węzłach i w serwerach lokalnych. Implementacja tej warstwy w urządzeniach prowadzi do uzyskania atrybutów działania takich, jak samodołączalność (ang. self-connectivity), samowykrywalność (ang. self-sensing) i brak oddziaływania na aplikacje (ang. seamless transfer management), co

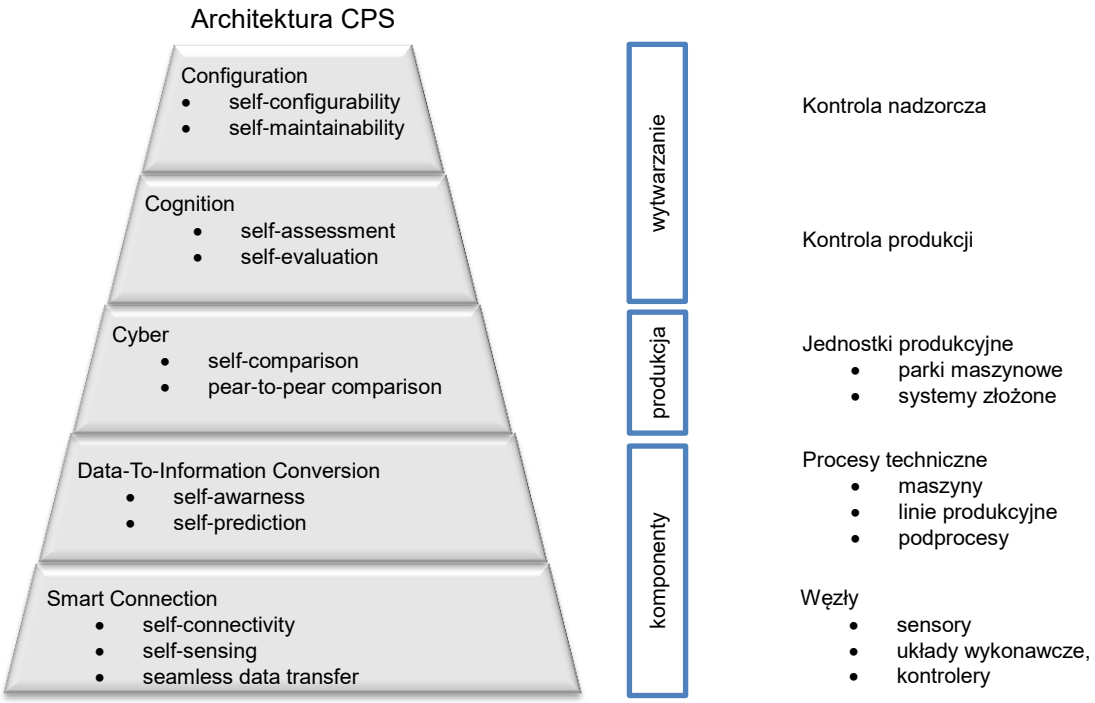
jest w innych dziedzinach bardziej znane jako działanie typu plug&play oraz komunikacja bez ograniczeń (ang. tether-free communication) [265]. Komunikacja może być czasu rzeczywistego, ale rodzaj zależy od obszaru działania i lokalnych ograniczeń [195].

- ang. Data-To-Information Conversion Level (warstwa samoświadomości)
  Warstwa odpowiada za analizę wybranych danych (np. krytycznych, jakościowych, diagnostycznych) i ich korelację względem poprawności funkcjonowania ich źródeł, tzw. zdrowia elementów oraz predykcję ich zachowań. Atrybuty działania warstwy to samoświadomość węzła (ang. self-awareness) i samoprzewidywalność działania (ang. self-prediction). Ogólnie, w tej warstwie dokonuje się wszelkiego wnioskowania, czyli wydobywania informacji z dostarczonych danych, związanego z „samoświadomym" działaniem węzła.

- ang. Cyber Level (warstwa opisu)
  Odpowiada za konstruowanie obrazu procesu, czyli przekształcanie otrzymywanych danych na opis informacyjny i jego analizę. Tworzy się wspomnianego wirtualnego bliźniaka obiektu w zasobach systemu. Bieżący opis oraz obrazy wsteczne (historyczne, ang. snapshot) są zwykle przechowywane na serwerze systemowym lub w chmurze. W warstwie dokonuje się wyekstrahowania informacji opisujących dany aspekt pracy i przeprowadza analizy, np. porównawcze z innymi procesami lub obrazami wstecznymi. Wykonuje się analizy dotyczące całej jednostki produkcyjnej lub zestawu jednostek. Wykorzystuje się algorytmy nastawione na konkretne analizy lub mechanizmy eksploracji danych (ang. data mining). Atrybuty działania to samoporównywalność (ang. self-comparison) i porównywalność równorzędna (ang. pear-to-pear comparison).

- ang. Cognition Level (warstwa poznawcza)
  Warstwa dedykowana do prezentacji danych dla celów decyzyjnych. Umożliwia podejmowanie strategicznych decyzji względem prowadzenia procesu wytwarzania. Atrybuty działania to samoocena (ang. self-assessment) oraz samoszacowalność (ang. self-evaluation).

- ang. Configuration Level (warstwa konfiguracji)
  Jest to warstwa kontroli nadrzędnej. Umożliwia wykonania akcji rekonfigurujących produkcję i dokonywania działań korygujących dla akcji warstwy poznawczej. Atrybuty to samokonfugurowalność (ang. self-configurability) oraz samoutrzymywalność (ang. self-maintainability).



Rys. 85. Architektura CPS i jej powiązania
Fig. 85. CPS architecture and its interconnections

Model architektury przedstawiany jest jako model piramidowy z warstwą połączeń na dole i warstwą konfiguracji na szczycie (rysunek 85).

Więcej na temat CPS można znaleźć np. w [235];

3. ELEMENTY ISP

Aby w ISP zrealizować wymagane zadania (zob. 1.2.5, 2.4.3), niezbędne są, wyszczególnione poniżej, trzy grupy elementów sprzętowo-programowych często spotykanych w konstrukcjach systemów. W poniższych grupach przedstawiono przykłady typowych urządzeń i programów. Przykłady te nie wyczerpują listy wszystkich możliwych elementów, a mają jedynie na celu naświetlenie, z czym projektant czy programista może się zetknąć. Dobre zestawienie technik, technologii i elementów związanych z ISP znajduje się w [386].

3.1. Urządzenia przetwarzające

Do tej grupy zalicza się wszelkie urządzenia elektroniczne umożliwiające przetwarzanie informacji, czyli ogólnie mówiąc urządzenia komputerowe. Przez takie urządzenia rozumie się komputery techniczne, budowane za pomocą klasycznej architektury harwardzkiej, von Neumanna lub mieszanej, zależnie od konkretnego rozwiązania. Urządzenia PLC, podobnie jak mikrokontrolery, są przeważnie budowane za pomocą architektury harwardzkiej z oddzieloną pamięcią danych i programu.

Ogólnie komputerowe urządzenia przetwarzające ISP można podzielić na:

- kontrolery (sterowniki):
  - komputerowe urządzenia sterujące – kontrolery/sterowniki procesowe, pracujące z mocnymi lub twardymi uwarunkowaniami czasowymi,
  - komputerowe urządzenia bezpieczeństwa – kontrolery/sterowniki bezpieczeństwa, pracujące z twardymi uwarunkowaniami czasowymi,
- urządzenia interfejsu danych:
  - komputerowe stacje robocze użytkownika i serwery, pracujące z miękkimi lub bez uwarunkowań czasowych,
  - komputerowe układy IO, pracujące co najmniej z mocnymi uwarunkowaniami czasowymi.

Z tych dwóch klas urządzeń w ISP najczęściej wyróżnia się kilka typów urządzeń opisanych w poniższych podrozdziałach. Niektóre z urządzeń są związane mocno z określoną klasą

zastosowań lub wręcz stanowią element sterujący większych urządzeń dedykowanych do realizacji konkretnych zadań w konkretnej klasie układów automatyki. Teoria sterowania takimi układami stanowi zagadnienie szerokie, dość złożone i daleko wykraczające poza tematykę niniejszej książki [219]. Stąd zawarty opis dotyczy tylko ogólnej charakterystyki dziedziny zastosowań oraz zadań, roli i miejsca takich urządzeń komputerowych w ISP. Przy niektórych urządzeniach skupiono się tylko na wybranych ich rodzajach, najpopularniejszych w kontekście zastosowań.

3.1.1. Komputery osobiste

(PC – ang. Personal Computer)

Popularny i powszechnie używany rodzaj komputera. W swym założeniu są to komputery dedykowane do obsługi przez człowieka. Ogólnie przyjmuje się je określać jako osobiste, czyli klasy PC, MAC w różnych wersjach wykonania obudowy, jak kasetowe, desktopowe, wieżowe (ang. tower) itp., czy w wersjach przenośnych, jak laptop, notebook itp.

Rys. 86. Ilustracja zastosowań komputerów PC na obiektach[38]
Fig. 86. View of PC utilization on industrial objects

Kilka przykładów pokazano na zdjęciach z rysunku 86. Widać tam zastosowania narzędziowe oraz jako stacje robocze w biurach i nastawniach.

[38] Fotografie dzięki uprzejmości Proloc Sp z o. o. i Coal-Control GmbH.

❖ **Dziedzina zastosowań**

Zastosowanie tego typu urządzeń z założenia jest przeznaczone do pracy domowej i biurowej (ang. home & office). Ich zastosowanie w innym kontekście nie jest poprawne i może prowadzić do dużej awaryjności systemu. Jednak węzeł ISP oparty na komputerze klasy PC ma wiele zalet, z których do najważniejszych można zaliczyć:

- popularność i ogólną znajomość obsługi,
- dużą[39] moc obliczeniową,
- dużą przestrzeń pamięci zarówno operacyjnej, jak i dyskowej,
- standardowe interfejsy użytkownika,
- standardowe interfejsy komunikacyjne i prostota ich rozbudowy,
- wygodę użycia i relatywną prostotę integracji,
- dostępność popularnych systemów operacyjnych,
- dostępność aplikacji i względną prostotę ich wytworzenia.

Dlatego nie rezygnuje się z wykorzystania komputerów klasy PC do pracy w roli węzłów ISP. Ich dziedzina zastosowań jest zawężona do tych zadań, które nie wywołują negatywnego wpływu na działanie systemu. Zatem komputery klasy PC stosuje się w ISP jako węzły dostępowe, integrujące funkcje z poziomu procesu z funkcjami biurowymi. Stosowane są one do realizacji lokalnych i zdalnych (RMPC – ang. Remote PC) stacji roboczych, pracujących w nastawniach, na stanowiskach nadzorczych, w biurach utrzymania ruchu, kierowania produkcją itp. Najczęściej stanowią one stacje kontrolno-wizualizacyjne (ang. SCADA, zob. 3.2.7), wspomagające produkcję i zarządzanie (ang. MES, ERP, zob. 3.2.9, 3.2.10), monitorujące, raportujące i wszelkie inne do budowania zrozumiałego człowiekowi interfejsu z procesem i systemem (ang. HMI zob. 3.2.8).

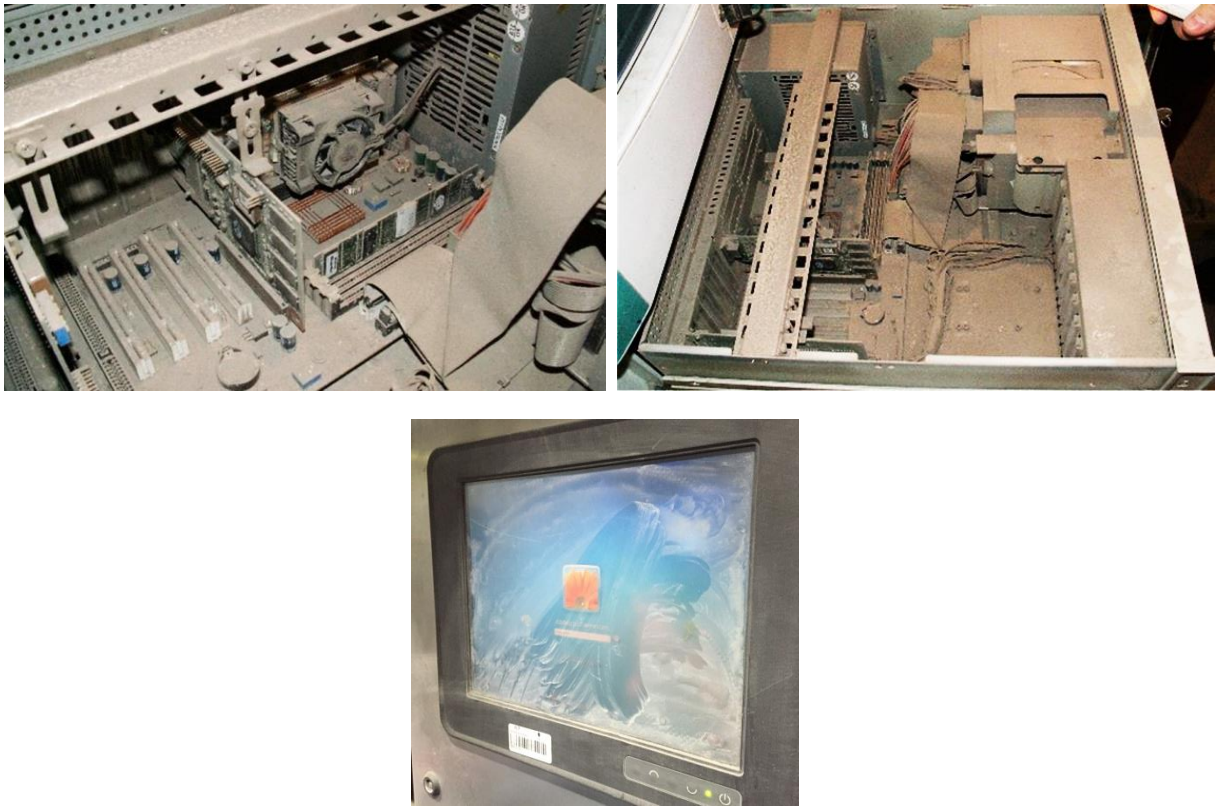
Z drugiej strony, komputery klasy PC nie są przystosowane do pracy jako węzły przetwarzające ISP. Typowymi błędami zastosowań komputerów osobistych są:

- przydzielanie zadań wymagających fizycznej obecności takiego urządzenia w środowisku procesu. Komputer osobisty klasy biurowej nie ma konstrukcji odpornej na zagrożenia środowiskowe (zob. 1.1.2). Jego zastosowanie powoduje szybkie zużycie elementów wirujących oraz awarie spowodowane negatywnymi wpływami takiego środowiska;
- przydzielanie zadań wymagających cech trwałości, niezawodności oraz przede wszystkim punktualności działania. Komputer klasy PC ma konstrukcję otwartą i mechanicznie nietrwałą. Staje się tym samym podatny na modyfikacje konfiguracyjne i uszkodzenia. Po-

[39] Względem innych komputerów stosowanych w ISP.

nadto, do pracy w czasie rzeczywistym musi być wyposażony w system operacyjny czasu rzeczywistego [350], [280], co w praktyce, ze względu na koszty i konieczność szkoleń, zdarza się dość rzadko.

Na rysunku 87 przedstawiono dość wymowny obraz komputerów po kilku miesiącach pracy w środowisku przemysłowym.



Rys. 87. Przykład zabrudzeń komputera wynikających ze środowiska pracy[40]
Fig. 87. Example of computer contamination resulting from the environment

Dodatkowo, istnieje zagrożenie zastosowań, dość oczywiste, choć często niedostrzegane lub ignorowane. Jest nim czynnik ludzki i fizyczna dostępność interfejsów komputera PC. Pracownicy, szczególnie ci niekoniecznie świadomi zagadnień informatycznych i zagrożeń opisanych wcześniej, bardzo często będą ingerować w oprogramowanie komputera, starając się uatrakcyjnić sobie pracę lub zmienić czynniki wpływające na ich osobistą ocenę. Innymi słowy, będą instalowali gry i modyfikowali oprogramowanie nadzorcze. Powoduje to niepożądane efekty w postaci zachwiania integralności systemu oraz przekłamania w danych. Nieszczęśliwym trafem zakładowe działy IT przeważnie nie interesują się komputerami PC, które są wykorzystane w strukturach ISP. Efektem są permanentne infekcje różnego rodzaju, osłabienie wydajności komputerów oraz upadki systemu.

[40] Fotografie A. Jestratjew i P. Gaj.

Systemy operacyjne stosowane z PC zależą od konkretnej architektury komputera. Są to jednak zwykle systemy biurowe typu Windows czy Linux. Pociąga to za sobą konsekwencje względem zadań realizowanych z ograniczeniami czasowymi. Jeśli takowe ograniczenia występują, należy zastosować RTOS[41] lub wyraźnie określić granice stosowalności, bądź realizowalności danych zadań.

❖ **Zadania**

Komputery PC powinny być stosowane tylko do realizacji wzmiankowanych wcześniej interfejsów z użytkownikiem, z zachowaniem zabezpieczeń wynikających z cechy niezawodności systemu (por. 2.3). Aktualnie do podstawowych zabezpieczeń należą:

- system operacyjny z logowaniem użytkowników (autoryzacja użytkownika komputera i autoryzacja użytkownika oprogramowania, a przy dobrej[42] administracji można to zintegrować),
- oprogramowanie typu firewall (blokowanie niepożądanego ruchu sieciowego),
- oprogramowanie antywirusowe (blokowanie ataków i odnajdywanie złośliwego oprogramowania),
- przydzielenie administratorów do zarządzania i pielęgnacji komputera (instalowanie aktualizacji, utrzymanie dysków, czyszczenie logiczne i fizyczne, diagnostyka, przeglądy i naprawy bieżące itp.).

Wygodnie jest, aby komputer mógł być zarządzany zdalnie, ale wówczas należy zatroszczyć się o odpowiednie zabezpieczenie dostępu. Zabezpieczenia należy zawsze dopasowywać do bieżącego stanu zagrożeń, nie ufając, że stare, sprawdzone sposoby będą zawsze wystarczające.

Oprogramowanie, jakie jest uruchamiane w aplikacjach dla tego typu węzłów to głównie programy kontrolno-nadzorcze (zob. 3.2.7), programy wspomagające prowadzenie procesu (zob. 3.2.9) oraz wspomagające zarządzanie produkcją (zob. 3.2.10). Jednak z racji lokalizacji biurowej lub pseudobiurowej takich komputerów można im przydzielać zadania integrujące dane procesowe z wszelką działalnością biurową powiązaną z danym ISP. Można tego dokonać przez dedykowane oprogramowanie pośredniczące (ang. middleware) lub przez oprogramowanie uniwersalne. W przypadku tego drugiego nie musi też być dostępny żaden dedykowany i złożony system oprogramowania. Integracje można prowadzić na poziomie typowych narzędzi biurowych dostarczających obsługi baz danych, arkuszy kalkulacyjnych

[41] Do popularnych systemów operacyjnych RT na platformę PC należą QNX, RT-Linux, Windows CE/Embedded w wersjach i konfiguracji jądra umożliwiającej pracę z ograniczeniami czasowymi [350].

[42] Pisanie o „dobrej” administracji wykracza poza temat tej książki, ale założono, że czytelnik jest świadom, że np. hasła typu „1234” nie są dobre.

czy procesorów dokumentów[43]. Na tej zasadzie można dokonywać prostej integracji międzysystemowej oraz integracji z zakładowymi serwerami, np. baz danych. Więcej o integracji systemów znajduje się w 3.5.3.

❖ **Miejsce i rola w ISP**

Komputer klasy PC może stanowić węzeł ISP. Jest to jednak węzeł „końcowy", czyli dostępowy. Może pełnić rolę dostępu dla człowieka, jak i dla innych systemów ISP czy zakładowych systemów IT. Mimo że nie uczestniczy on bezpośrednio w sterowaniu i regulacji procesem, rola takiego węzła jest kluczowa z punktu widzenia takiego dostępu. Stanowi on okno, przez które najczęściej następuje komunikacja ISP ze światem zewnętrznym. Obsługuje on zatem tzw. pionowy przepływ informacji, a projektując jego użycie w ISP, należy unikać zadań, które wymagałyby przepływów poziomych (por. 1.2.2). Komputer PC występuje również często w roli stacji deweloperskiej, ale wówczas pełni w systemie rolę tymczasową i niezwiązaną z normalną pracą ISP, a jedynie z jego wdrożeniem.

I na koniec jeszcze jedna oczywistość świata przemysłowych zastosowań PC, która dotyczy bezpieczeństwa użycia, a w praktyce jest niedostrzegana. Dobrze, aby konto administratora było wykorzystywane tylko do celów administracyjnych, tylko przez administratorów i zgodnie z wiedzą informatyczną. Praktyczne obserwacje autora bardzo niepokoją. W większości przypadków hasła są trywialne, nazwy użytkowników również, na koncie administratora pracuje niemal każdy, uprawnienia są dawane z tzw. „zapasem na wszelki wypadek, jakby coś nie chciało działać", czyli wszystkie, jakie tylko można. Dotyczy to każdego elementu systemu i stanowi ogromną oraz potencjalnie bardzo niebezpieczną dziurę w bezpieczeństwie systemu, umożliwiając hackerom prowadzenie szpiegostwa gospodarczego, zakłóceń i zatrzymań procesu, uszkodzeń produktów itp. Dlatego komputer klasy PC należy otoczyć szczególną troską i przemyśleniem, o ile występuje on w danym ISP.

3.1.2. Komputery przemysłowe

(IPC – ang. Industrial PC)

Klasa komputerów IPC jest podzbiorem klasy PC dedykowanym dla środowiska przemysłowego (zob. 1.1.3). Komputery tego typu są oparte na klasycznym rozwiązaniu PC, z tą jednak różnicą, że ich konstrukcja mechaniczna jest bardziej odporna na trudne warunki pracy niż klasycznych biurowych rozwiązań.

[43] Na przykład MS Office, Open Office czy inne tej klasy.

❖ **Dziedzina zastosowań**

Konstrukcja mechaniczno-elektroniczna IPC przystosowana jest do pracy w umiarkowanie agresywnym środowisku przemysłowym przy obiekcie. Dotyczy to zarówno wzmocnionej obudowy, używania filtrów powietrza, nieużywania elementów wirujących, takich jak wentylatory i dyski, używania dysków półprzewodnikowych (SSD – ang. Solid State Drive), jak i również używania dodatkowych i nietypowych dla urządzeń biurowych łączy magistralowych na karty rozszerzeń i dodatkowych portów komunikacyjnych. Konstrukcje IPC mają lepszą odporność na czynniki środowiskowe i w wielu przypadkach są w stanie zapewnić odpowiednią trwałość. Miarą odporności w zakresie szczelności obudowy są klasy IP [M80], a w zakresie kompatybilności EMC zgodność z normami dla środowisk przemysłowych, a nie biurowych (np. normy ogólne odporności i emisji [M55], [M56], [M57], [M58]). Jednak pozostałe słabe punkty stosowania PC nadal stanowią zagrożenie dla systemu (por. 3.1.1).

Rys. 88. Przykład komputera IPC
Fig. 88. Example of IPC computer

IPC jest nadal dedykowany dla człowieka, dlatego zastosowanie komputera tego typu sprawdzi się, gdy istnieje potrzeba przeniesienia funkcjonalności komputera PC bliżej procesu, np. na halę produkcyjną. Natomiast nie jest dobrym pomysłem, aby na IPC opierać konstrukcję węzłów procesowych (por. 2.4.2).

Stosowany w IPC OS wynika z funkcji, jaką pełni komputer, ale można ogólnie przyjąć, że będą to takie systemy jak przy komputerach klasy PC.

❖ **Zadania**

Zadania IPC również wynikają z zadań PC. IPC głownie są stosowane do budowy węzłów nadzorczych (SCADA), kontrolnych (HMI) oraz obsługi produkcyjnych baz danych, przy czym idea systemów nadzorczych kłóci się z umieszczaniem ich bezpośrednio przy procesie. Dlatego jeśli na IPC jest budowany interfejs dla użytkownika, to z reguły ma on charakter kontroli lokalnej, czyli interfejsu z maszyną (HMI), a nie nadzorczy nad procesem. Zatem IPC pracują tam, gdzie pracownicy obsługujący produkcję (operatorzy, utrzymanie

ruchu itp.). Zadania polegają na akwizycji informacji z innych węzłów ISP, najczęściej z użyciem przemysłowej sieci komputerowej, prezentacji tych danych wraz z lokalnych przetwarzaniem na potrzeby tej prezentacji oraz zapewnienia możliwości parametryzacji i konfiguracji związanej z działaniem stanowiska, maszyny czy danego procesu. Dodatkowo, IPC umożliwiają wygodne, lokalne składowanie danych i ich redystrybucje do systemów nadrzędnych.

Ze względu na ograniczenia konstrukcyjne oraz stosowane OS, dostępne spektrum zadań jest ograniczone do tych, klasyfikowanych jako Soft-RT. Węzeł na bazie IPC nie powinien zajmować się sterowaniem ani żadnym zadaniem, którego poprawne wykonanie jest zależne od czasu, a jego funkcja zysku *z(t)* (por. 1.1.4) dla $t \geq t_t$ zwraca zero lub $-\infty$.

❖ **Miejsce i rola w ISP**

IPC sprawdzają się głównie jako węzły interfejsowe użytkownika ISP lub jako platformy serwerowe (zob. 3.1.14, 3.2.3) działające blisko procesu. Przy stosowaniu IPC należy zwrócić uwagę na następujące problemy.

- Typ systemu operacyjnego takiego komputera powinien być zgodny z typami zadań przez niego realizowanymi. Przydzielanie zadań wymagających przetwarzania w czasie rzeczywistym potrzebuje systemu operacyjnego czasu rzeczywistego.
- Ze względu na aspekty bezpieczeństwa (zob. 4.2.1), wykorzystanie IPC na hali produkcyjnej powinno się wiązać z ograniczaniem fizycznego dostępu do takiego komputera. Najczęściej zamykane są one w szafach sterowniczych lub szafach przeznaczonych na infrastrukturę IT. Dostęp jest wówczas możliwy tylko z poziomu interfejsów udostępnionych użytkownikowi: najczęściej klawiatury, myszy czy też jakiegoś specjalnego urządzenia, np. skanera kodów itp.
- Odpowiednie odprowadzanie ciepła. Pasywne chłodzenie wymaga swobodnego przepływu powietrza. Przegrzanie komputera z pewnością doprowadzi do jego wyłączenia, a nawet awarii.
- Zabezpieczenie przed agresywnymi wpływami środowiskowymi. Istnieje niewiele rozwiązań architektury IPC odpornej na wibracje, wstrząsy, wilgoć czy pył węglowy. Dotyczy to również silnych zaburzeń EMC zarówno indukowanych, jak i przewodzonych (np. względem normy ogólnej IEC/PN-EN 61000-4-6 [M54]).

W sytuacji gdy istnieje potrzeba wykorzystania architektury PC blisko procesu, lepszym podejściem będzie koncepcja sterowników na bazie PC (por. 3.1.5).

3.1.3. Sterowniki klasy PLC

(sterownik swobodnie programowalny, programowalny sterownik logiczny, sterownik logiczny, kontroler przemysłowy, ang. Programmable Logic Controller, Programmable Controller)

Sterowniki klasy PLC od strony informatycznej są to klasyczne komputery zwykle o architekturze harwardzkiej, posiadające interfejsy wejścia i wyjścia oraz przynajmniej jeden układ przetwarzania (ang. MPU, por. 2.6.1). Konstrukcja mechaniczna, konstrukcja elektroniczna, system operacyjny, języki i listy rozkazów oraz interfejsy są dostosowane do potrzeb współpracy z układami automatyki, czyli docelowo z procesem przemysłowym.

Rys. 89. Przykłady PLC
Fig. 89. Example of PLC

Sterowniki montuje się w szafach sterowniczych. Zabezpiecza to częściowo urządzenia przed negatywnym wpływem warunków środowiskowych oraz przed niepowołanym dostępem osób nieuprawnionych. Ponadto, umożliwia wygodne i bezpieczne połączenie obwodów zasilających i IO z elementami elektrycznymi. Przykładowe sterowniki pokazano na zdjęciach 89, a sterowniki w szafach na zdjęciach 90.

Rys. 90. Przykłady PLC zamontowanych w szafach sterowniczych[44]
Fig. 90. Example of PLC mounted in cabinets

[44] Fotografie J. Stój.

Istnieje wiele literatury na temat PLC [28], [62], [299], [36], [218], [196], [210], [192], [364], jak również materiałów online, np. [W33], [W32], [W34], [W7].

❖ **Dziedzina zastosowań**

PLC są przeznaczone do realizacji zadań sterowania i regulacji w skomputeryzowanych układach automatyki [138]. Zakres zastosowań dotyczy wszelkich procesów, w tym ciągłych, dyskretnych, wsadowych itp., choć zależy to od konkretnych możliwości danego urządzenia. Sterowniki te są swobodnie programowalne, zatem program (programy) i konfiguracja urządzenia są wymienne i mogą dostosowywać działanie urządzenia do danej aplikacji. PLC mogą również wspierać wykorzystanie regulatorów rozmytych. Oznacza to w konsekwencji bardzo dużą uniwersalność zastosowań. Ponadto, z racji dużej popularności na rynku można stwierdzić, że współcześnie sterowniki klasy PLC stanowią podstawowy element przetwarzający informacje w skomputeryzowanych układach automatyki, czyli węzeł ISP.

Standaryzacja w zakresie komputerowych urządzeń sterujących bazuje na normach IEC61131. Sterowniki klasy PLC zawierają się w zbiorze komputerowych urządzeń sterujących, zatem wspomniana norma dotyczy tych urządzeń w pełnym zakresie (zob. 2.6). Większość współczesnych konstrukcji PLC spełnia w całości lub w przynajmniej w części wytyczne tej normy.

Ze względu na możliwości integracji PLC dzielą się na urządzenia typu otwartego i zamkniętego. Urządzenia zamknięte nie są interesujące z punktu widzenia ISP, gdyż charakteryzują się brakiem odpowiednich środków sprzętowo-programowych, służących do wbudowania ich w system. Urządzenia typu otwartego umożliwiają współpracę z innymi węzłami ISP, jak również innymi systemami, najczęściej przez interfejsy sieci komputerowych i oprogramowanie pośredniczące.

Inny podział urządzeń klasy ISP odnosi się do rozmiaru adresowanej dyskretnej przestrzeni IO i sprowadza się do klasyfikacji na tej podstawie urządzeń na bardzo małe, małe, średnie i duże. Urządzenia bardzo małe (tzw. nano) i małe (tzw. mikro) często są typu zamkniętego. Pozostałe z reguły doskonale nadają się do pracy jako węzły ISP. Urządzenia małe adresują do kilkunastu wejść/wyjść dyskretnych, urządzenia średnie do ok. tysiąca, a duże powyżej. Kolejny podział to rozróżnienie względem budowy lokalnej sterownika. Istnieją urządzenia kompaktowe (ang. compact) oraz modułowe (ang. modular). Różnica polega na możliwości ich rozbudowy lokalnej i dostosowania do specyficznych potrzeb danej aplikacji (danego węzła). Sterownikami kompaktowymi są przeważnie konstrukcje nano i mikro.

Dużym wyzwaniem projektowym jest dobór właściwego sterownika dla konkretnej aplikacji. Dostępnych jest na ten temat wiele literatury: [36], [218] i materiałów online: [W32], [W34]. Jest też sporo dobrych rad i metod przekazywanych nieformalnie. Jedną z nich jest

dobieranie PLC na zasadzie zachowania kompatybilności technik komunikacyjnych z innym sprzętem pracującym w zakładzie. Inną jest dobór przez zgodność producentów tegoż sprzętu. Niezależnie od wybranej metody doboru zawsze warto przewidzieć pewien zapas (rezerwę) na zasobu sterownika. Dotyczy to głownie przestrzeni IO oraz pamięci. Można przyjąć, że rezerwa rzędu 10-30% jest rozsądna. Precyzja specyfikowania poniżej 10% jest w praktyce niemal niemożliwa, a założenie powyżej 30% oznacza brak wiedzy o obiekcie.

Sterownik musi nadążać za zjawiskami w obsługiwanym procesie, a zatem musi pracować w reżimie czasu rzeczywistego. Systemy operacyjne używane w PLC to systemy RTOS (zob. 1.1.4). W klasycznych sterownikach działają one w trybie HRTS, gdzie przekroczenie czasu wykonania programu czy realizacji zadania jest traktowane jako błąd krytyczny. OS niektórych sterowników klasyfikowanych jako „soft" (zob. 3.2.3) lub „PC based " (zob. 3.1.5) pracują w trybie FRT. Generalnie, sterowniki powinny zapewnić działanie programów z takimi ograniczeniami, jakich wymaga aplikacja obsługująca dany obiekt. Wśród dostępnych rozwiązań można spotkać stosowanie znanych OS[45], jak i dedykowanego oprogramowania typu firmware.

Z tematem ograniczeń czasowych wiąże się pośrednio temat dostępnej mocy obliczeniowej. Ograniczenie mocy w PLC wynika z niemożności stosowania elementów elektronicznych chłodzonych aktywnie. Nie pozwala na to środowisko pracy. Problem dotyczy głównie procesorów i wentylatorów. Dlatego moce obliczeniowe sterowników nie są imponujące względem współczesnych komputerów biurowych, a sterownik klasy PLC nie powinien być postrzegany jako tzw. „szybki" komputer.

Wiele przykładów zastosowań PLC pokazano w [36].

❖ Zadania

Działanie PLC charakteryzuje się pracą w czasie rzeczywistym. Zapewnia to system operacyjny czasu rzeczywistego (RTOS zob. 3.2.1) na poziomie funkcjonowania węzła (por. 2.4.2) oraz systemowe mechanizmy kontroli wykonywania programów na poziomie aplikacji użytkownika. Zapewnienie działania dowolnego programu użytkownika z dowolnymi ograniczeniami czasowymi jest niemożliwe z racji dowolności kodu, jaki może zostać uruchomiony. Dla przykładu, można założyć, że istnieje zadanie x z ograniczeniem na czas jego wykonania T_{Tx}. W programie obsługi zadania znajduje się przynajmniej jedna iteracja z instrukcją warunkową i zmienna sterującą, której wartość nie jest stałą. W efekcie jeśli:

$$\left(\exists x_i\right)\left(\exists p > 0\right)\sum_{j=0}^{p} T_{Ij} > T_{Tx}$$

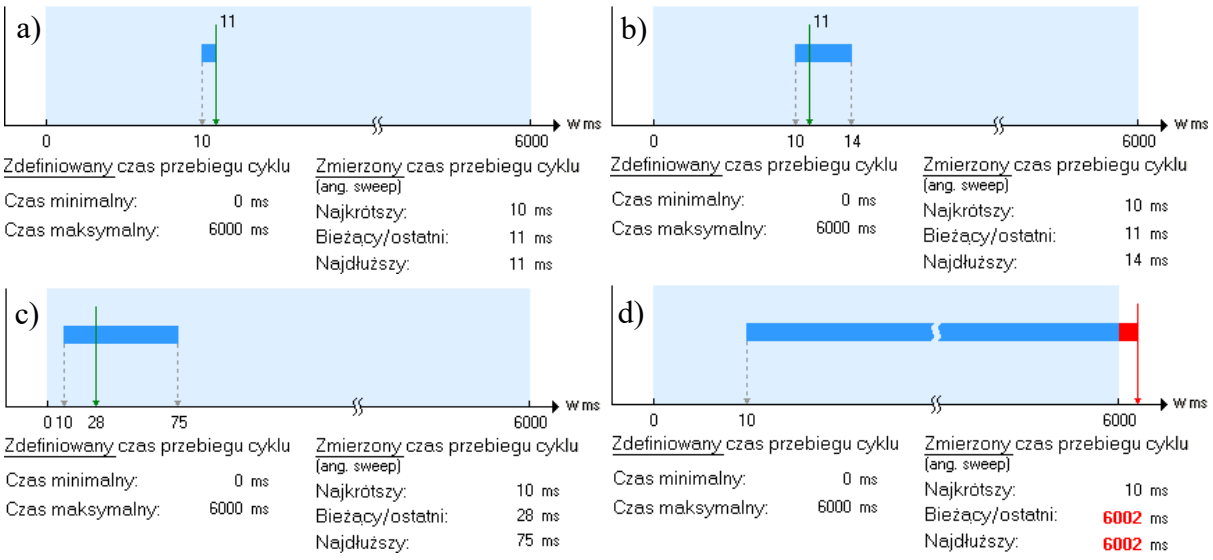
[45] Na przykład VxWorks, QNX, Windows Embedded, CE i inne.

gdzie:

p jest liczbą wykonywanych iteracji w i-tym wykonaniu zadania x,

T_{Ij} jest czasem wykonania j-tej iteracji,

to dotrzymanie warunków czasowych dla wykonania danego zadania nie jest możliwe. Dlatego w PLC stosuje się metody kontrolujące czas wykonania zadań (programów) oraz mechanizmy wspomagające zarządzanie ich wykonywaniem. Do najpopularniejszych mechanizmów kontroli należy tzw. watchdog, czyli mechanizm wartownika, mierzący czas wykonania programu i zgłaszający przerwanie w momencie, kiedy $T_{Sxi} > T_{Tx}$ (por. 1.1.4). Obsługa przerwania zależy od danej konstrukcji i/lub programisty, lecz zwykle sprowadza się do zatrzymania sterownika z błędem krytycznym lub uruchomienia programu obsługi błędów.



Rys. 91. Ilustracja kontroli zakresu czasu cyklu PLC[46]
Fig. 91. View of the range control of PLC cycle time

Typowymi mechanizmami wspomagającymi zarządzanie wykonaniem programów są tzw. cykl sterownika oraz mechanizmy szeregujące. Mechanizmy te oraz wiele pozostałych zagadnień związanych z PLC zostały opisane np. w [196], [9]. Zrozumienie działania cyklu sterownika jest kluczową sprawą niezbędną do skutecznego tworzenia obsługi zadań. Działanie cyklu sprowadza się do cyklicznego, sekwencyjnego wykonania pewnych zadań systemowych, w szczególności do cyklicznego wykonywania programu aplikacyjnego danego PLC. Dzięki cyklicznemu uruchamianiu programu możliwa jest wspomniana powyżej kontrola czasu jego wykonania. Program zawsze ma swój bezwzględny początek i koniec w czasie, a zatem mechanizm watchdog może określić, ile taki program się wykonywał

[46] Ilustracja zaczerpnięta z oprogramowania Step7 dla platformy Simatic.

w danym cyklu oraz prowadzić statystyki związane z określoną aktywnością sterownika. Ilustracja kontroli czasu cyklu została przedstawiona na rysunku 91.

Program aplikacyjny węzła może mieć zmienny czas wykonania, co pokazano po rysunku. Na przykładach z rysunku 91 a), b) i c) pokazano normalną pracę cyklu przy różnej złożoności czasowej kodu względem warunków jego wołania. W przykładzie a) pokazano program, w którym czas wykonania operacji jest zbliżony, niezależnie od warunków. W przykładzie c) rozpiętość czasu wykonania jest duża z powodu użycia kodu o różnej złożoności czasowej względem warunków jego uruchomienia. Zależy on od instrukcji warunkowych i różnorodności przetwarzania dla danych warunków. W przykładzie d) pokazano przekroczenie czasu zdefiniowanego jako maksymalny.

Pozostałe zadania uruchamianie przez OS w cyklu PLC oraz ich sekwencja zależą od konkretnego modelu lub rodziny sterownika. Istnieje jednak kilka zadań, które mogą być postrzegane jako podstawowe i występują niemal zawsze. Należą do nich:

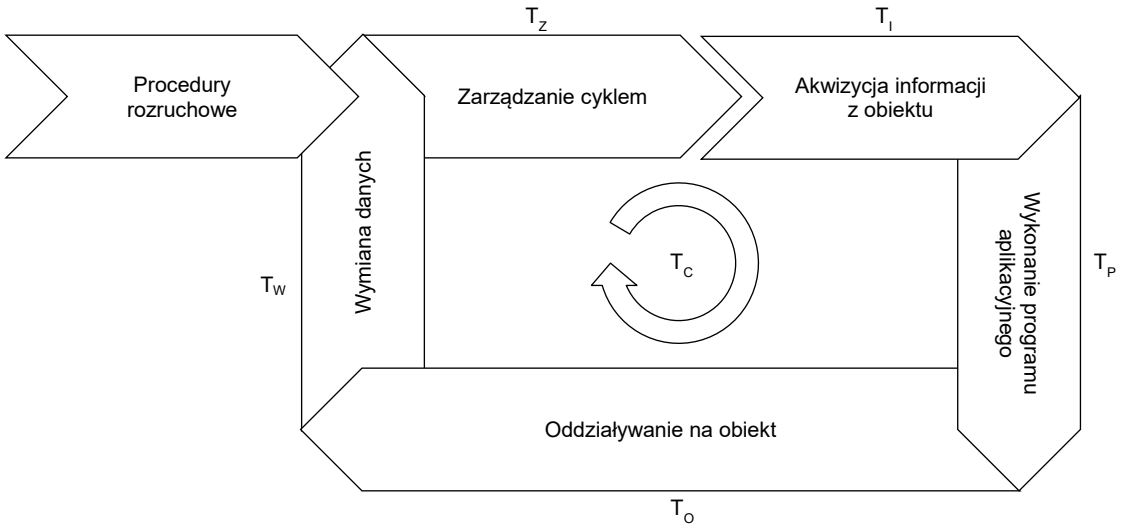
- zarządzanie wykonaniem cyklu
  Wykonanie wszelkich czynności, które warunkują działanie cyklu oraz konkretnego jego przebiegu.
  - określanie trybu działania: czynności określające moment rozpoczęcia kolejnego przebiegu cyklu (ang. sweep), w tym start, stop i wstrzymanie itp.,
  - inicjalizacje: obsługa wbudowanych generatorów, inicjujących bloków programowych, zegarów, flag błędów itp.,
  - autodiagnostyka: kontrola poprawności konfiguracji sterownika, zapisu w pamięci itp.

- akwizycja danych z obiektu
  Działanie specjalnego zadania OS (maper, skaner IO) zapisującego reprezentacje cyfrowe danych wejściowych, uzyskanych z przetworników ADC modułów wejścia, w odpowiednich miejscach pamięci (tzw. obrazie wejść) wynikających z konfiguracji lokalnej sterownika.

- wykonanie głównego programu aplikacyjnego
  Program główny sterownika jest wołany przez OS. Inne bloki kodu mogą być wołane z programu głównego lub z wyzwalaczy (ang. triger) związanych z przerwaniem lub ogólnie ze zdarzeniem. Podczas wykonywania programu następuje modyfikacja pamięci danych.

- oddziaływanie na obiekt
  Obsługa modułów wyjściowych przez przekazanie zawartości pamięci wyjść (tzw. obraz wyjść) do przetworników DAC. Podobnie jak dla wejść obsługę wykonuje zadanie mapera (skanera) IO zgodnie z konfiguracją lokalną.

- wymiana danych
  - komunikacja z programatorem: okno czasowe przeznaczone na wymianę danych z zewnętrznym urządzeniem programującym. W ramach aktywności w oknie działa specjalny, dedykowany protokół do wymiany danych (zapis, odczyt, debuging itp.),
  - komunikacja z elementami konfiguracji lokalnej: okno czasowe dla wymiany danych z innymi modułami „inteligentnymi" znajdującymi się w konfiguracji lokalnej sterownika. Wymiana odbywa się w strukturach wewnętrznych urządzenia,
  - komunikacja z innymi elementami systemu: okno czasowe na wymianę danych z innymi węzłami systemu. Jest ono realizowane najczęściej przez niezależny, asynchroniczny względem cyklu, obieg komunikacji sieciowej oraz wewnętrzne, zsynchronizowane z cyklem przekazywanie danych między komponentami lokalnymi urządzenia.



Rys. 92. Ilustracja działania cyklu PLC
Fig. 92. View of the execution of PLC cycle

Cykl PLC został zilustrowany na rysunku 92. Procedury rozruchowe dotyczą operacji wykonywanych przed uruchomieniem operacji cyklicznych. Należą do nich procedury testów sprzętowych oraz procedury aktualizacji i rozruchu sterownika z nośników wymiennych[47]. Przejście do cyklicznego działania możliwe jest, gdy procedury testowe nie zgłoszą błędów

[47] PLC często są wyposażane w interfejsy kart pamięci flash, np. MC, CF, CFast, MMC, SD.

oraz jest dostępny program, który można uruchomić. Czas cyklu PLC T_C jest zatem sumą czasów zadań T_{Zi} wykonywanych w każdym przebiegu, gdzie i określa zadanie w zakresie od 1 do aktualnie wykonywanej ich liczby. Zastosowane w poniższym wzorze oznaczenia odnoszą się do rysunku 92.

$$T_C = \sum T_{Zi} = T_Z + T_I + T_P + T_O + T_W$$

Niektóre elementy cyklu mogą być pomijane ze względu na aktualny tryb pracy. Dla przykładu, jeśli sterownik ma zablokowaną obsługę IO, to zadania ich obsługi (T_I, T_O) nie są załączane, a pamięć IO pozostaje niezmieniona. Podobnie z komunikacją. Jeśli w konfiguracji nie ma odpowiednich modułów lub programator nie jest podłączony, to stosowne okna czasowe (T_W) nie są otwierane. Stąd wniosek, że czas trwania pojedynczego przebiegu cyklu zależy od konfiguracji lokalnej, rodzaju obsługiwanych IO oraz czasu wykonania programu, przy czym dla danej konfiguracji składową zmienną jest tylko czas wykonania programu (T_P).

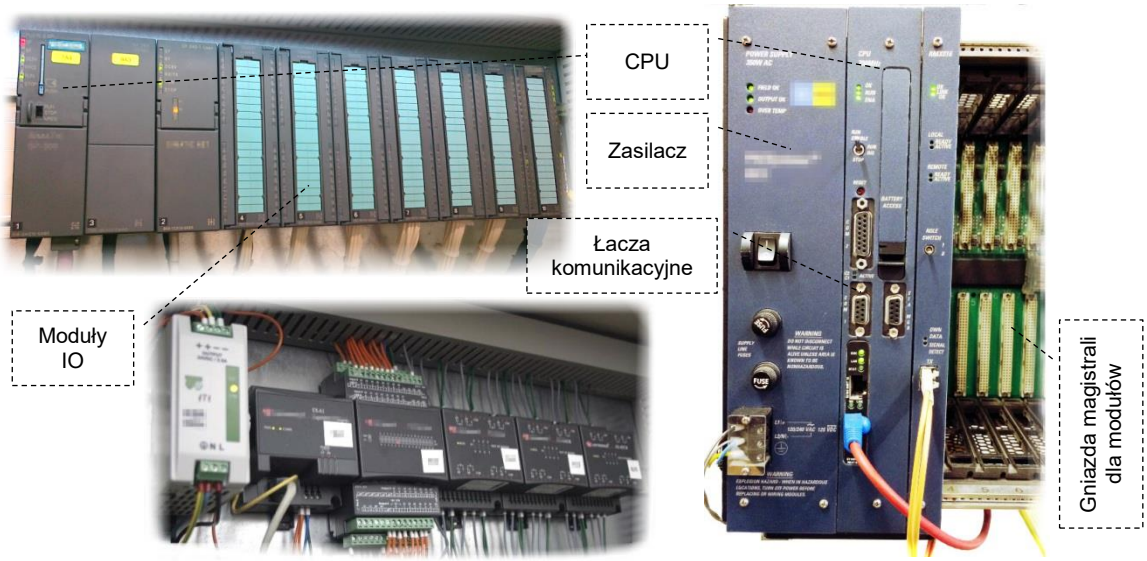
W niektórych sterownikach dopuszczono ustawienie minimalnego czasu trwania cyklu, przez co możliwe jest, aby program i inne zadania cykliczne były uruchamiane dokładnie co określony czas. Ma to sens, jeśli rzeczywisty, maksymalny czas wykonania cyklu

$$T_{Cmax} < T_{DEFmin},$$

gdzie T_{DEFmin} jest dolnym czasem ograniczenia przebiegu cyklu, wówczas wykonanie przebiegów nie nakłada się na siebie i możliwe jest zachowanie $T_C = const$. Ilustracja zmienności czasu wykonania programów jest dobrze widoczna, gdy porówna się czasy wykonania operacji (por. rys. 8) lub wyobrazi się program, w którym w zależności od warunku wykonuje proste i złożone operacje (por. rys. 9). Zostało to też opisane w [380] oraz [382].

Uruchamiane programy mogą realizować bardzo szerokie spektrum zadań – od akwizycji i przekazywania danych, przez sterowania dyskretne aż po regulacje ciągłe. Ze względu na cykl sterownika jego oddziaływanie z obiektem zawsze sprowadza się jednak do pewnego skwantowania czasu wynikającego z tegoż cyklu. Zatem zarówno pozyskiwanie danych z obiektu, jak i oddziaływanie na obiekt jest limitowane działaniem w określonym oknie czasowym. Jakiekolwiek oddziaływania poza tymi oknami są uzależnione od dostępności i użycia mechanizmów oddziałujących na obiekt bezpośrednio, z pominięciem sekwencji działania cyklu. Dla większości aplikacji czas cyklu wynosi od kilku do kilkudziesięciu milisekund, co jest wystarczające do zapewnienia pseudociągłej interakcji z obiektem. Gdy wymagania są większe, należy szukać szybszych jednostek centralnych lub korzystać z dodatkowych mechanizmów wspomagających takie interakcje. Z reguły PLC posiadają rozkazy umożliwiające oddziaływanie z układami IO w trakcie wykonywania programu oraz

umożliwiają uruchamianie zadań na przerwaniach. Dostępne są też moduły koprocesorów „inteligentnych”[48], oddziałujących z procesem asynchronicznie względem cyklu jednostki centralnej i wymieniających informacje z jednostką centralną synchronicznie względem cyklu. Niemniej jednak o skwantowaniu czasu oddziaływania z obiektem nie należy zapominać.



Rys. 93. Elementy konfiguracji sterowników PLC
Fig. 93. Configuration elements of PLC

❖ **Miejsce i rola w ISP**

Rola PLC w ISP jest kluczowa. PLC jest najczęściej spotykanym rodzajem węzła ISP. Trudno znaleźć bardziej typowy komputer przeznaczony do pracy w środowisku przemysłowym (por. 1.1.3). Może pracować jako komputer centralny jak i jako węzeł systemu rozproszonego (por. 2.3). Sterowniki zajmują się pozyskiwaniem, przetwarzaniem i wytwarzaniem informacji. Realizują algorytmy sterowania i regulacji oraz pośredniczą w wymianie danych.

3.1.4. Stacje procesowe systemów DCS

(ang. Distributed Computer System, Process Station, DCS Controller)

Stacje procesowe DCS to podobnie jak PLC komputery realizujące funkcje sterowania i regulacji. Mogą składać się z jednego lub więcej kontrolerów, pełniących określone role w przetwarzaniu informacji na rzecz obsługi obiektu i procesu przemysłowego. Popularność tych rozwiązań jest niższa niż PLC, aczkolwiek można je traktować jako typowy komputer dla tej klasy zastosowań.

[48] Na przykład moduły szybkich liczników, programowalnych IO, obsługi serwonapędów, koprocesorów sieciowych itp.

❖ **Dziedzina zastosowań**

Historycznie rzecz ujmując, gdy zaczęto konstruować komputery do zastosowań w systemach automatyki, stacje DCS były przeznaczone do obsługi sterowań ciągłych, a PLC do sterowań dyskretnych. Ogólny postęp technologii mikroprocesorowych spowodował, że zarówno współczesne PLC, jak i kontrolery DCS są pełnowartościowymi komputerami i mogą z powodzeniem wykonywać sterowania i regulacje zarówno dyskretne, jak i ciągłe [327], [337]. Warto zauważyć, że obecnie różnica między kontrolerami PLC a DCS sprowadza się do budowy tych mechanizmów systemów operacyjnych, które są odpowiedzialne za uruchamianie programów aplikacyjnych oraz do idei funkcjonowania narzędzi deweloperskich. Bardzo często zarówno jedne, jak i drugie urządzenia pracują na dokładnie tej samej bazie sprzętowej. Często jednak stacje procesowe DCS są w praktyce postrzegane jako rozwiązanie dedykowane stricte dla procesów ciągłych. Dlatego systemy DCS znajdują zastosowanie głównie w tych gałęziach przemysłu, gdzie takie procesy[49] występują. Można spotkać opinie, że różnice między PLC a DCS wynikają z dostępności lub jej braku pewnego zbioru funkcjonalności [336] związanych ze strategiami sterowania i syntezą układów regulacji, jak np. gotowe elementy układów typu dynamiczny kompensator całkująco-różniczkujący, automatyczne strojenie regulatorów, obsługa sygnałów analogowych, procesy statystyczne, logika rozmyta czy optymizacja na bazie modeli. Współcześnie jest to jednak kwestia dostępności bibliotek i narzędzi, a nie rodzaju komputera. Biorąc pod uwagę aspekty techniczne, decyzja o wyborze[50] powinna być podyktowana parametrami czasowymi związanymi z nadążnością, dostępnym zakresem przetwarzania danych oraz możliwościami integracji, a nie dość sztuczną klasyfikacją rodem z lat 70. ubiegłego wieku.

Rys. 94. Przykład stacji DCS
Fig. 94. Example of DCS station

[49] Głównie energetyka, przemysł chemiczny i rafineryjny, przemysł wodny i wydobywczy.
[50] Aktualnie istnieje na rynku kilku znaczących producentów systemów DCS, np. GE (Cimplicity Open Process), Siemens (PCS7), Emerson (DeltaV, Ovation), Yokogawa (Centum, Stardom), Invensys (I/A), Honeywell (Experion), Metso (DNA), ABB (Advant OCS, Symphony, Freelance), General Electric (seria Mark) i inne.

❖ **Zadania**

Założeniem wykorzystania stacji DCS jest praca w systemie rozproszonym. Zadania są obsługiwane zdarzeniowo, a nie cyklicznie jak w PLC. Sprowadza się to do uruchamiania zadań na przerwaniach lub zdarzeniach systemowych działających podobnie jak przerwania, tyle że generowanych przez OS od zdefiniowanych warunków. Z racji niemożności obsługi wszystkich definiowalnych zdarzeń na przerwaniach należy się spodziewać, że część zadań musi być obsługiwana na zasadzie odpytywania stanu (ang. pooling) działającej podobnie jak cykl PLC (por. 2.1.2). Konsekwencją takiej czy innej obsługi jest zawsze jakaś zwłoka lub okno czasowe wynikające z konieczności wykonania obsługi programowej i limitujące dolne ograniczenia względem czasu reakcji systemu RT. Dlatego, istotne jest poznanie minimalnej szczeliny, z którą dany system komputerowy jest w stanie reagować na zmiany swojego otoczenia.

Innowacyjność tworzenia aplikacji polega na programowaniu wirtualnej warstwy aplikacji w oderwaniu od fizycznego rozproszonego sprzętu. Standaryzacja w tym zakresie odwołuje się do normy IEC61499 (zob. 2.5) i budowania kodu aplikacji wirtualnej na podstawie tzw. bloków funkcyjnych z odseparowanymi zależnościami danych i zdarzeń. Znormalizowane podejście programowania systemów rozproszonych nie zdobyło dużego uznania wśród programistów, choć niesłusznie. Idea opisu systemu na wielu poziomach abstrakcji, gdzie każdy komponent systemu stanowi podsystem, z uwzględnieniem przepływów zdarzeń i przepływu danych jest interesująca i umożliwiająca oderwanie prac programistycznych od fizyczności urządzeń (por. 3.2.13). Na takich wysokich abstrakcjach można dokonać wirtualnego uruchomienia i testowania systemu. Warunkiem powodzenia jest użycie poprawnych modeli czasowych. Dopiero końcowym etapem jest osadzenie kodu i funkcji komunikacyjnych w rzeczywistym sprzęcie. Idea ta mocno zyskuje, gdy istnieje możliwość pracy DCS w heterogenicznym środowisku kontrolerów, sieci i urządzeń AKP, co w praktyce nie zawsze ma miejsce.

Budując DCS, należy mieć na uwadze dwa zestawy norm. W kwestii opisu rozproszenia, związków aplikacyjnych i określenia przepływów informacji i rządzących nimi reguł podstawą jest wspomniany zestaw IEC61499 (zob. 2.5), czyli wysoko abstrakcyjne podejście systemowe. Natomiast kwestie programowania konkretnych akcji i przetwarzania w węzłach określa norma IEC61131 (zob. 2.6).

❖ **Miejsce i rola w ISP**

W celu ulokowania stacji procesowych DCS w omawianych zagadnieniach ISP należy je odnieść do dziedziny systemów rozproszonych oraz wspomnianych wyżej norm. Powstaje tym samym system rozproszony, gdzie działanie aplikacji wirtualnej opisane jest przez bloki

funkcyjne oraz przepływy danych i zdarzeń z IEC61499, a na poziomie węzłów działają programy utworzone z użyciem języków znanych z IEC61131-3 [M21]. Współpraca między kontrolerami DCS a innymi elementami systemu (np. PLC) jest możliwa na poziomie wymiany danych procesowych oraz nadzoru, czyli mogą zachodzić między nimi zarówno wymiany poziome, jak i pionowe. Technicznie nie ma przeciwskazań, aby budować ISP na bazie zarówno jednych, jak i drugich kontrolerów, tworząc tym samym układy mieszane.

Uważny czytelnik dostrzeże, że budowa ISP na bazie PLC niewiele się różni od budowy ISP na bazie stacji DCS. Różnice sprowadzają się do idei działania narzędzi i przepływu informacji (zob. 1.2), przy czym przepływ informacji jest realizowany przy użyciu sieci komputerowych tak samo jak w przypadku systemów opartych na PLC. Sposoby wykonywania zadań niewiele różnią się od tych stosowanych w PLC. Głównie bazuje się na zdarzeniowym i cyklicznym uruchamianiu programów i mechanizmie szeregującym. Ponadto, przepływ informacji często jest dokonywany przez wspólne repozytorium danych procesowych i systemowych, jakim najczęściej jest baza danych. Baza może być centralna lub rozproszona i obsługiwana przy użyciu serwerów dedykowanych dla konkretnego systemu.

3.1.5. Sterowniki na bazie PC

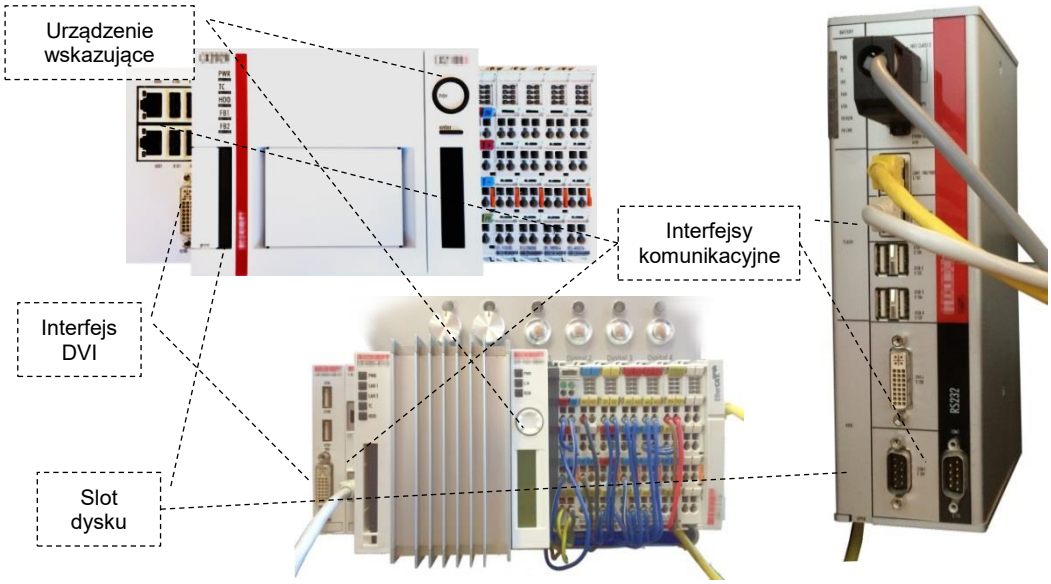
(PCC – ang. PC based Controller)

Wraz ze wzrostem złożoności aplikacji przemysłowych, pojawieniem się konieczności integracji systemów, włączaniem coraz to nowych i złożonych usług[51] w obszary działania ISP oraz potrzebą uzyskiwania ich wysokiej elastyczności (ang. flexible, resilience), pojawiła się potrzeba zwiększania wydajności i uniwersalności systemów cyfrowych zajmujących się obsługą informacji procesowej. Jednym ze sposobów na osiągnięcie tego jest poszerzanie dziedziny zastosowań komputerów sterujących. Niestety, stosowanie klasycznych komputerów PC lub nawet IPC w środowiskach systemów automatyki nie jest dobrym pomysłem, co opisano w 3.1.1 i 3.1.2. Cenne byłoby zatem umieszczenie urządzeń i funkcjonalności PC w architekturze PLC. Stało się to możliwe dzięki rozwojowi technologii mikroprocesorowej. Istnieje na rynku pewna grupa urządzeń sterownikowych utworzonych na bazie platformy PC. Nie jest tutaj mowa o IPC, lecz o komputerach przystosowanych do pełnienia roli PLC zarówno od strony konstrukcyjnej, jak i programowej, ale z zaletami rozwiązań klasy PC (por. 3.1.1). Odnosząc różnice do użytkownika, można stwierdzić, że IPC jest komputerem przemysłowym klasy PC dedykowanym dla człowieka, a PCC komputerem przemysłowym dla procesu. Urządzenia klasy PCC są często klasyfikowane jako tzw. elementy automatyza-

[51] Komunikacja zorientowana na usługi (np. SOAP, http), internet rzeczy (ang. Internet of Things), bazy danych przy procesie, HMI itp.

cji na bazie PC (ang. PC-based Automation). Na rysunku 95 przedstawiono przykładowe urządzenia PCC.

Rozpatrując tę klasę urządzeń, warto przyjrzeć się idei danego rozwiązania. Można uruchomić oprogramowanie klasy „sterownik programowy" (ang. Soft PLC, zob. 3.2.3) na komputerze klasy IPC lub stworzyć kontroler klasy PLC na bazie architektury PC i na nim uruchamiać procesy sterujące. Różnica wiąże się z zagadnieniami konstrukcji elektroniczno-mechanicznej, jak i zastosowanych systemów operacyjnych. W opisach dostępnych na rynku często nie jest to rozróżniane. Konsekwencje uderzają bezpośrednio w dwa aspekty pracy. Po pierwsze, nie każdy komputer IPC jest dostosowany do pracy w uciążliwych warunkach przemysłowych (por. 3.1.2), zwiększając tym samym awaryjność rozwiązania. Po drugie, charakterystyka czasowa pracy oprogramowania uruchomionego pod kontrolą systemu non-RT będzie inna niż procesów kontrolowanych przez OS klasy RT, szczególnie gdy system taki staje się zestresowany.

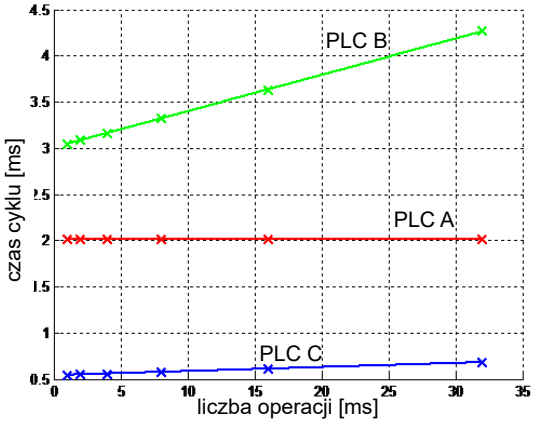


Rys. 95. Przykłady sterowników PC
Fig. 95. Examples of PC controllers

❖ Dziedzina zastosowań

Na wstępie dyskusji o dziedzinie zastosowań należy wyraźnie zaznaczyć, że dla wielu aplikacji ISP stosowanie sterowników PC nie ma sensu, wystarcza klasyczny PLC. PCC są to urządzenia, które sprawdzą się, gdy funkcjonalności węzła muszą być elastyczne oraz gdy istnieje potrzeba większej mocy przetwarzania względem sterowników klasycznych. Problem ograniczenia mocy obliczeniowej wspomniany przy omawianiu PLC (por. 3.1.3) jest tu eliminowany przez stosowanie chłodzenia aktywnego. Najczęściej są to wentylatory o konstrukcji umożliwiającej szybką wymianę bez wyłączania urządzenia i/lub układy pa-

sywne z ciepłowodami i odpowiednio dużymi radiatorami i obudową umożliwiającą wymuszony, grawitacyjny przepływ powietrza. Konsekwencją tego jest zmniejszona odporność urządzeń na zanieczyszczenia i zwiększona awaryjność w sytuacji aplikacji sterownika bez należytej ochrony zewnętrznej, np. stosownej szafy oraz braku konserwacji. Na rysunku 96 przedstawiono porównanie czasów realizacji rozkazów z użyciem klasycznych PLC i sterownika PC. Z jednej strony można zaobserwować, że czas cyklu sterownika A nie ulega widocznej zmianie wraz ze wzrostem liczby operacji. Wynika to z dużego zapasu mocy obliczeniowej urządzenia. Zmienność mogłaby być obserwowana poniżej przyjętej wartości czasu cyklu. Z drugiej natomiast, przy pomiarach czasu realizacji konkretnych zadań można zaobserwować większą niestałość niż w przypadku klasycznych PLC. Wynika to ze sposobu szeregowania zadań sterujących i systemowych. Oczywiście charakterystyki czasów zależą od konkretnych produktów oraz ich konfiguracji.



Rys. 96. Porównanie czasów cykli dla operacji mnożenia
Fig. 96. Comparison of cycles for multiplication command

Sterowniki PC mają z reguły konstrukcje modułowe i pracują pod kontrolą RTOS podobnie jak klasyczne PLC. To co odróżnia sterowniki PC od tych klasycznych to jednostka centralna, która traktuje funkcjonalność PLC jako jedną z potencjalnie wielu. Ponadto, ponieważ konstrukcja tych sterowników bazuje na idei PC, są one wyposażane w klasyczne interfejsy wejścia i wyjścia dla człowieka, czyli dla klawiatury, myszy i ekranu, a także w dyski SSD bądź nawet HDD. Aktywność PLC jest realizowana przez sterownik programowy (ang. Soft PLC) dostępny jako program uruchamiany i stanowiący jeden z podprocesów użytkownika w środowisku OS. Zwykle w tego typu sterownikach pracują RTOS podnoszące uniwersalność i prostotę zastosowań, czyli systemy klasy Windows[52] czy QNX. Pod kontrolą tych systemów uruchamiane mogą być różne, zgodne z danym OS programy, rozszerzające funkcjonalność węzła.

[52] W tym wypadku mowa jest o Windows w wersji CE lub embedded.

❖ **Zadania**

Sterownik PC realizuje takie same zdania jak klasyczny PLC. Zwykle jest programowalny, zgodnie z normą IEC61131-3 [M21]. Typowe zadania dodatkowe to:

- obsługa HMI (np. uniwersalne lub dedykowane programy HMI, lub aplikacje SCADA),
- obsługa baz danych (np. serwery SQL lub dedykowane bazy plikowe),
- obsługa klienta/serwera danych (np. OPC, http, SOAP itp.),
- zadania użytkownika (np. specyficzne programy dedykowane dla danej aplikacji, uruchamiane w węźle i niestanowiące programów sterujących),
- zadania integracji – dotyczy integracji wewnętrznej, lokalnej (zob. 3.5), jak i międzysystemowej (zob. oprogramowanie middleware 3.5.5).

❖ **Miejsce i rola w ISP**

Rola urządzenia w ISP jest taka sama jak innych programowalnych kontrolerów. Może ono stanowić główny komputer systemu scentralizowanego lub węzeł systemu rozproszonego. Z racji relatywnie dużej mocy obliczeniowej, zasobów pamięci oraz dostępności różnorodnych interfejsów doskonale nadaje się on jako komputer centralny systemu. Z powodu zastosowanej architektury należy jednak zawsze rozważać charakterystykę czasową realizacji zadań RT. Sterowniki PC mogą też służyć jako węzły interfejsowe zarówno dla człowieka, jak i dla innych systemów. Możliwe jest zatem uruchamianie funkcjonalności klasy HMI (zob. 3.2.3) oraz programów serwerowych (zob. 3.2.8).

3.1.6. Sterowniki wbudowane

(EPLC – ang. Embedded PLC)

Istnieje osobna klasa komputerowych urządzeń sterujących nazywana sterownikami wbudowanymi. Są to urządzenia o funkcjonalności PLC, tyle że nie stanowią odrębnych i niezależnych produktów, a są wbudowywane w inne urządzenia takie produkty stanowiące.

Rys. 97. Przykłady płyt sterowników dedykowanych
Fig. 97. Examples of dedicated controllers boards

❖ **Dziedzina zastosowań**

Sterowniki wbudowane stanowią gotowe do użycia urządzania w postaci układów elektronicznych lub jednoukładowych kontrolerów[53]. Znajdują zastosowanie jako urządzenia dołączane do konstrukcji elektroniki innych wyrobów, takich jak maszyny, narzędzia, złożone układy wykonawcze itp. Istnieje cały szereg takich produktów stosowanych w sterowaniu i automatyzacji procesów, z punktu widzenia rynkowego niszowych, dedykowanych dla systemów specyficznych, krótkoseryjnych czy też tworzonych na potrzeby konkretnej aplikacji. Wymagają one dysponowania funkcjonalnością PLC, ale ze względów handlowych lub marketingowych nie mogą być wyposażane w dodatkowe, odrębne urządzenia sterujące. Najczęściej rozszerzenie konstrukcji takiego urządzenia o funkcjonalności PLC nie jest opłacalne, gdyż wymaga dodatkowych nakładów zarówno czasu, jak i środków. Wbudowanie modułu EPLC w urządzenie jest relatywnie proste i tanie, dlatego daje szybki i prosty sposób zintegrowania gotowych konstrukcji. Często EPLC są produktami typu OEM.

❖ **Zadania**

EPLC może realizować takie same zadania jak zwykły PLC, z dokładnością do swoich możliwości wynikających z konstrukcji sprzętowej, czyli dostępności zasobów przetwarzających oraz interfejsów procesowych i komunikacyjnych. Osadzenie PLC w innym urządzeniu wynika zawsze z konkretnie określonych potrzeb tego urządzenia i zadań, jakie ma ono realizować. Stąd na etapie projektowania można dobrać taki EPLC, aby zapewnić wymagania funkcjonalne aplikacji urządzenia. Zwykle w takich przypadkach rezerwa zasobów sterownika nie jest potrzeba z racji ścisłego dopasowania konstrukcji. Zwykle EPLC są programowalne zgodnie z normą IEC61131-3 [M21].

❖ **Miejsce i rola w ISP**

EPLC jako element ISP niczym nie różni się od zwykłego PLC. Ewentualne ograniczenia w integracji z innymi elementami systemu wynikają z konkretnych konfiguracji, co ma również miejsce w przypadku klasycznych PLC bądź sterowników dedykowanych. Z jednej strony, EPLC może często stanowić komputer główny systemu scentralizowanego. Ma to miejsce, gdy system informatyczny jest trywialny, a urządzenie, w którym sterownik jest osadzony, tworzy jego główny element lub gdy stanowi ono niezależne urządzenie typu maszyna, linia produkcyjna, obrabiarka, robot itp. Z drugiej strony, nawet jeśli taki produkt stanowi rozwiązanie pozornie zamknięte, to dzięki EPLC produkty takie można włączyć do ISP lub

[53] Na przykład PLC on Chip firmy Divelbiss, konstrukcje na bazie Raspberry Pi lub innych układów, albo też rozwiązania na bazie FPGA firmy Altera czy Siemens [artykuły z art. Trybusa IEEE], lub polskie prace, np. [153] czy też zagraniczne, np. [134].

utworzyć ISP z takich produktów, zwiększając tym samym zakres kontroli nadrzędnej. Zatem zastosowanie EPLC sprzyja tworzeniu węzłów systemu rozproszonego i ich integracji.

3.1.7. Sterowniki i urządzenia dedykowane

(dedykowany układ sterowania, ang. dedicated controller)

W tej klasie produktów znajdują się wszelakie urządzenia komputerowe mające określoną, nieuniwersalną funkcjonalność, współpracujące z procesem z jednej strony, a z systemami klasy ISP z drugiej.

❖ **Dziedzina zastosowań**

Urządzenia te spełniają podobną funkcję jak PLC. Są to komputery przeznaczone do realizacji zadań sterowania dla konkretnych zastosowań. Różnica polega na uniwersalności lub dedykowalności urządzenia. Sterowniki dedykowane nie są reprogramowalne przez użytkownika, w przeciwieństwie do PLC. Zwykle są tworzone na potrzeby konkretnej aplikacji lub konkretnej klasy aplikacji, przez co sterowniki mają bardzo wąską dziedzinę zastosowań. Dla przykładu, mogą to być sterowniki kotłów, pomp, pras filtracyjnych, liczniki ciepła, moduły telemetryczne, pomiarowe czy też nawet sterowniki seryjnie produkowanych obrabiarek, maszyn i innych urządzeń sterowanych komputerowo.

Do zalet rozwiązań dedykowanych można zaliczyć bardzo dobre dopasowanie do konkretnego zastosowania, do wad małą uniwersalność i niską skalowalność. Koszty zależą od skali produkcji. Gdy urządzenie jest masowo produkowane, należy spodziewać się relatywnie niskich kosztów zakupu i wdrożenia, gdy jednak jest wysoce specjalizowane lub wykonywane na zamówienie, koszty mogą okazać się bardzo wysokie. Uwzględniając w projekcie ISP rozwiązania dedykowane, należy bardzo intensywnie przemyśleć użycie urządzeń niskoseryjnych lub wykonywanych na zamówienie.

Utrzymanie urządzeń dedykowanych w działaniu wymaga serwisu, ciągłości ich produkcji oraz dostaw części zamiennych. Pomijając koszty inwestycji, ich utrzymanie może okazać się kosztowne z racji specjalizowanego serwisu oraz zagrożeń braku długoterminowego wsparcia. Jeżeli ich utrzymanie wymaga rozwoju oprogramowania, to niezbędne jest, aby mieć dostęp do zagwarantowanych na określony czas aktualizacji lub mieć dostęp do stosownych narzędzi i projektów (np. kwestia własności źródeł programów i bibliotek, licencje open source itp.). Do stosowania urządzeń problematycznych należy przystąpić, tylko gdy są ku temu istotne przesłanki (np. brak alternatywy, perspektywy rozwoju itp.).

Systemy operacyjne wykorzystywane do pracy z urządzeniami dedykowanymi są to mikrosystemy RTOS (zob. 3.2.1) lub dedykowane oprogramowanie typu firmware.

❖ Zadania

Zakres zadań, jakie realizują urządzenia klasy sterowniki dedykowane, jest bardzo obszerny. Wchodzą w to wszystkie aktywności na obiekcie przemysłowym, które mogą być wspomagane przez urządzenia komputerowe. Mogą to być bardzo proste zadania, np. obsługa pomiarów, logowania danych, dedykowane sterowania i regulacje itp., a także zadania złożone, jak obsługa komunikacji, sterowanie maszynami, kompleksowe i wielokryterialne regulacje itp. Zadania te są jednak konkretnie zdefiniowane dla konkretnego typu urządzenia.

Rys. 98. Przykład zastosowania prostej synoptyki i sterowników dedykowanych[54]
Fig. 98. Examples of simple synoptic and dedicated controllers usage

❖ Miejsce i rola w ISP

Sterowniki dedykowane często mają interfejsy (sieciowe, IO, analogowe itp.) umożliwiające włączenie ich w system i oddziaływanie na ich pracę przez parametryzację

[54] Fotografie dzięki uprzejmości firm Proloc Sp. z o. o. i Coal Control GmbH.

i konfigurację. Z punktu widzenia ISP mogą być traktowane jako źródło informacji oraz jako urządzenia wykonawcze. Programy w nich pracujące uczestniczą w tym samym co inne węzły ISP przepływie informacji, ale stanowią odrębne i niezależne aplikacje. Przez to są one centralnym komputerem dla środowiska, dla którego są dedykowane. Stąd, z punktu widzenia ISP, nadają się one tylko do pełnienia roli zintegrowanego z systemem dostawcy i/lub odbiorcy informacji, na którego warstwę aplikacji projektant ISP nie ma wpływu.

3.1.8. Sterowniki bezpieczeństwa

(ISC – ang. Industrial Safety Controller)

Dla zachowania wysokiego poziomu bezpieczeństwa funkcjonalnego (por. 3.5.4) stosuje się niezależne od układów sterujących systemy bezpieczeństwa (ang. safety system). Uzyskuje się tym samym systemy zapewniające bezpieczeństwo funkcjonalne, nawet w sytuacji awaryjnej (ang. fail-safe control system). Zwykle takie systemy są tworzone, gdy błędne funkcjonowanie systemów sterowania, w tym ISP, zagraża życiu, mieniu lub środowisku. Jednymi z elementów takiego systemu są kontrolery bezpieczeństwa (ang. safety controller, fail-safe conroller).

❖ **Dziedzina zastosowań**

Urządzenia tego typu stosuje się jako układy zabezpieczające proces przemysłowy przed nieprawidłowym prowadzeniem zagrażającym zachowaniu jego bezpieczeństwa funkcjonalnego. Elementy systemu bezpieczeństwa stanowią niezależne produkty i działają niezależnie od ISP i jego elementów. Kontrolery bezpieczeństwa mogą stanowić urządzenia kompaktowe, jak i urządzenia o konstrukcji modułowej. Z wyglądu oraz względem konstrukcji fizycznej są to urządzenia bardzo podobne do normalnych kontrolerów. Posiadają układy IO i interfejsy sieciowe, obsługują elementy czujnikowe i wykonawcze oraz współpracują ze sobą tworząc system. Różnica polega na oferowanych funkcjonalnościach i zakresie działania. Sterowniki oraz inne elementy systemu muszą być przeznaczone tylko i wyłącznie do realizacji specyficznych funkcji bezpieczeństwa, związanych z detekcją i obsługą stanów zagrożeń w procesach przemysłowych. Inne funkcje związane z obsługą procesu są poza zakresem działania ISC. Funkcje bezpieczeństwa są natywnie wbudowane w urządzenia. Systemy operacyjne tych sterowników są takie jak w innych kontrolerach. Jednak zadania systemu i wsparcie dla programisty poza podstawowymi rozkazami obejmuje specjalne funkcje bezpieczeństwa.

Przykładowo mogą to być:

- E-Stop (ang. Emergency Stop) – funkcja obsługi wyłączenia awaryjnego z logiką potwierdzeń i opóźnień.

- S-Door (ang. Safety Door) – monitorowanie stanu otwarcia drzwi, osłon itp., z logiką potwierdzeń.
- Wyciszanie (ang. Muting) – obsługa czasowej dezaktywacji kurtyn chroniących obszary niebezpieczne, np. na potrzeby wprowadzenia surowców.
- Załączanie dwuręczne – funkcja obsługi załączania układu z użyciem dwóch przycisków wciskanych w określonym odstępie czasu. Wyjście jest aktywowane tylko, gdy rozbieżność czasu jest mniejsza od zadanej.
- Analiza rozbieżności zdarzeń – uogólniona funkcja załączania dwuręcznego. Funkcja aktywuje zdarzenie, gdy rozbieżność czasu nadejścia sygnałów jest akceptowalna.
- Sprzężenie zwrotne (ang. Feedback) – funkcja monitorująca stan obiektu. Zdarzenie jest aktywowane, gdy sygnał zwrotny z obiektu będzie zgodny z sygnałem sterującym.
- I inne, jak potwierdzanie rozpoczęcia reintegracji, różne odmiany powyższych itp.

Istnienie dostępu do urządzeń i specyficznych funkcji bezpieczeństwa, nie gwarantuje bezpieczeństwa. Sterowniki muszą być oprogramowane zgodnie z konkretnymi wymaganiami dotyczącymi obsługi sytuacji awaryjnych. Niepoprawne lub błędne programy obsługi spowodują nieprawidłową reakcję systemu bezpieczeństwa.

Działanie sterowników charakteryzuje się również zwiększonym poziomem zabezpieczeń dostępu. Z reguły programy bezpieczeństwa są zabezpieczone hasłem, niezależnym od blokad dostępu do programów sterujących.

Ponadto, w sterownikach bezpieczeństwa istnieją dodatkowe mechanizmy diagnostyczne. Ich dostępność zależy od konkretnego urządzenia, ale istnieje kilka typowych. Sygnały IO mogą być testowane na okoliczność awarii, np. zwarcia[55]. Kanały IO mogą być dublowane, czyli stosuje się dwukanałowe sygnały (dwa niezależne tory sygnałowe) w kontekście jednego sygnału logicznego[56]. Dla obrazów IO alokowane są dodatkowe struktury statusowe dostępne dla programisty. Program w pamięci jest dublowany i kontrolowany na bieżąco[57] przez OS sterownika.

Urządzenia oraz poszczególne moduły zawsze oznacza się wizualnie innym kolorem niż moduły systemów sterowania, najczęściej żółtym lub czerwonym. Wyróżniane są też programy, które są w nich uruchamiane. Jednostki programowe tworzy się z użyciem takich sa-

[55] Zależnie od klasy SIL oraz rodzaju układu przełączającego. Dokonuje się tego przez cykliczne i krótkotrwałe (np. 1 ms) zwieranie sygnału do niezależnego obwodu zasilania. OS jest na tej podstawie w stanie wykryć, czy obwód sygnału jest nieuszkodzony.

[56] Obsługa awarii jest realizowana przez OS sterownika na podstawie zdefiniowanych parametrów, np. czasu rozbieżności w kanale.

[57] Wykonywanie kodu programu jest niejako dublowane, czyli weryfikowane z kopią kodu celem wykrycia ewentualnego błędu w zapisie programu. Funkcje kontrolne są osadzane w kodzie automatycznie.

mych języków jak kod sterujący, z tym że w wersji dla sterowników bezpieczeństwa[58]. Różnica dotyczy rodzaju POU oraz dostępnych rozkazów i bibliotek. Niemożliwe jest tym samym uruchomienie kodu sterującego na jednostkach „safety" i odwrotnie[59].

Na rynku istnieje wiele rozwiązań[60], a ich dobór zależy od preferencji lokalnych, związanych zwykle z kompatybilnością.

❖ Zadania

Zadaniem takich urządzeń jest niezależna od zadań sterowania obserwacja tych atrybutów procesu i ich zależności, które mogą świadczyć o wystąpieniu zagrożenia i odpowiedniej reakcji na nie. W sytuacji wykrycia takiego zagrożenia, tzn. spełnienia/niespełnienia zaprogramowanych warunków, następuje automatyczne zatrzymanie zadań sterujących, przejęcie kontroli nad procesem przez system bezpieczeństwa i wykonanie procedur obsługi zagrożenia (ang. fault reactions), przewidzianych na daną okoliczność. Może to być wykonanie jakiejś akcji, zachowanie ostatnich poprawnych stanów lub wprowadzenie procesu w tzw. stan bezpieczny (tzw. pasywacja, ang. passivation, safe state). Po eliminacji przyczyny problemu sterownik bezpieczeństwa może wykonać tzw. reintegracje (ang. reintegration) celem przywrócenia procesu w normalny tryb sterowania. W uproszczeniu można stwierdzić, że kontrolery procesu realizują zadania sterowania i regulacji w sytuacji poprawnego przebiegu procesu oraz obsługują potencjalne błędy wynikające z takiego funkcjonowania. Natomiast kontrolery bezpieczeństwa obsługują stany, które wychodzą poza zakres przewidziany normalną pracą i nie mogą zostać obsłużone przez układy sterowania.

❖ Miejsce i rola w ISP

Sterownik bezpieczeństwa jest elementem systemu bezpieczeństwa, na który podobnie jak w systemie sterowania składają się inne węzły oraz komunikacja, w szczególności sieci komputerowe. Kontroler bezpieczeństwa nie jest węzłem ISP, o ile dany ISP nie jest systemem bezpieczeństwa. W nowoczesnych rozwiązaniach często systemy sterowania i systemy bezpieczeństwa pracują na tych samych sieciach i urządzeniach. Rozdzielone są tylko funkcjonalności i moduły. Mimo to nie należy postrzegać węzłów systemów bezpieczeństwa jako węzłów ISP, a jedynie jako węzły podsystemu niezależnie współpracującego przy obsłudze danego procesu.

---

[58] Dla przykładu zamiast języka FBD wykorzystuje się odpowiednik FBD w wersji „safety' (np. F-FBD).

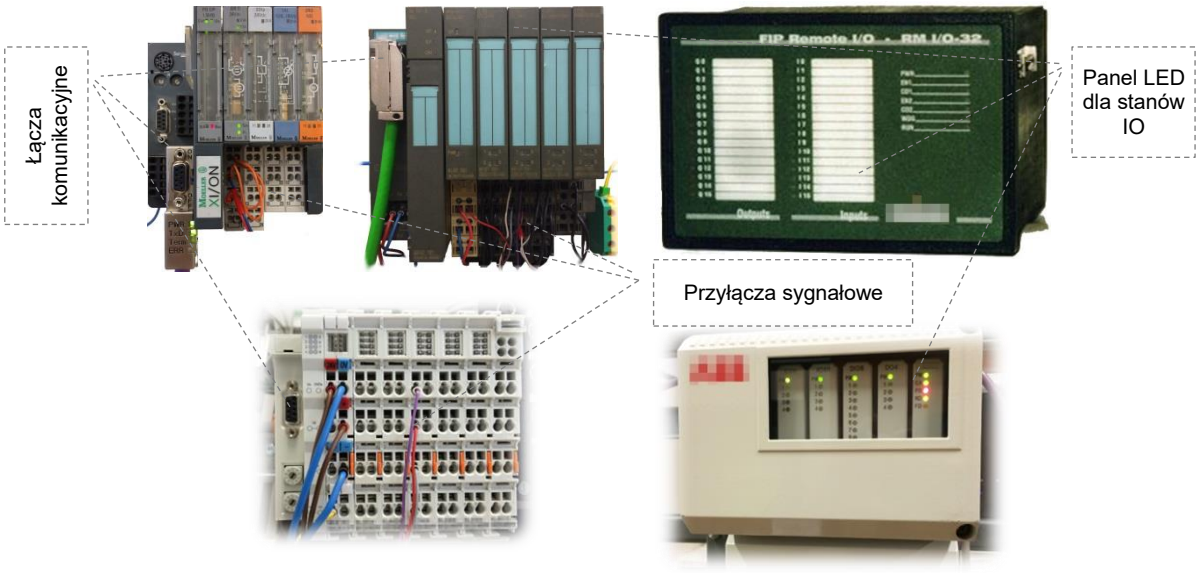
[59] Zależy to jednak od danego sterownika. Istnieją konstrukcje pracujące jako sterownik procesu oraz jako sterownik bezpieczeństwa z rozseparowanymi sygnałami, programami i komunikacją.

[60] kontrolerów (np. produkty Omron, Banner, Rockwell, ABB, Siemens, Pliz) oraz rozszerzeń profili dla sieci przemysłowych np. [5], [M6], [W17].

3.1.9. Zdalne wejścia/wyjścia

(wyspa IO, ang. Remote Input-Output, RIO, RMIO)

Są to urządzenia przeznaczone do zdalnej akwizycji informacji z obiektu i zdalnego oddziaływania na obiekt. Pojęcie działania zdalnego dotyczy oddalenia od współpracującego kontrolera poza obszar rozszerzeń jego magistrali lokalnej. Urządzenia połączone są z kontrolerem łączem cyfrowym, najczęściej przemysłową siecią komputerową, i stanowią zdalne moduły wejściowe i wyjściowe, rozszerzając tym samym jego dostępną lokalnie przestrzeń adresową IO. Aktualizacja danych związana z układami zdalnymi nie podlega mechanizmowi zachowywania stanu lokalnych IO (obsługi obrazów procesu). Podlega natomiast pracy sieci i aktualizacji obrazu w oknie czasowym współpracy przetwarzania lokalnego oraz działania cyklu sieci. Zatem charakterystyka czasowa związana z obsługą IO wynika z charakterystyki cyklu sieci, a nie z cyklu MPU. Przykłady kilku modułów zdalnych IO wraz z opisem ich podstawowych elementów przedstawiono na zdjęciu 99.



Rys. 99. Przykłady zdalnych modułów IO
Fig. 99. Examples of remote IO modules

❖ **Dziedzina zastosowań**

Stosowanie modułów zdalnych umożliwia eliminację konieczności prowadzenia długich tras kablowych dla przewodów sygnałowych pomiędzy obiektem a kontrolerem. Sprzyja również rozproszeniu terytorialnemu ISP. Sieć komputerowa może jednak okazać się wąskim gardłem współpracy. Dla sieci polowych oraz sieci RTE poniżej klasy 4 (zob. 3.3.1) czasy wymian są znacząco niższe od wymian na lokalnej magistrali systemowej, na której pracują lokalne IO. Istnieje jednak kilka rozwiązań (np. Profinet IO IRT v.2.3, EtherCAT), gdzie czas

aktualizacji danych IO może być porównywalny do czasów lokalnych (zob. 3.3). Problem wąskiego gardła dla wejść mogą też rozwiązywać moduły „inteligentne", które można swobodnie programować. Wówczas możliwe jest wykonanie lokalnego przetwarzania danych i zmniejszenie liczby niezbędnych danych aktualizowanych cyklicznie pomiędzy modułem a kontrolerem. Tak czy inaczej, przy projektowaniu rozproszenia układów IO należy brać pod uwagę opóźnienia związane z przetwarzaniem lokalnym i komunikacją.

Urządzenia klasy RMIO są na tyle proste, że przeważnie nie posiadają systemu operacyjnego, a jedynie dedykowany firmware pracujący w czasie rzeczywistym. Charakteryzują się dość dużą uniwersalnością zastosowań. Z racji niezależności od kontrolera nie muszą być dobierane pod względem konkretnego producenta, serii czy rodziny, a jedynie względem kompatybilności komunikacji i wymagań technicznych obsługi IO.

❖ **Zadania**

Głównym zadaniem zdalnych wejść-wyjść jest akwizycja informacji z obiektu i oddziaływanie na obiekt. Nie są to urządzenia programowalne, a jedynie konfigurowalne i parametryzowane. Moduły RMIO wykonują lokalnie elementarne przetwarzanie informacji związane z komunikacją i obsługą zakresów. Jest to dostosowywanie danych dotyczące zmiany reprezentacji, skalowania wartości, obcinania, filtrowania itp. Przetwarzanie na tym poziomie nie dotyczy wirtualnej aplikacji systemu, choć dzieje się na jej rzecz. Konfiguracja może dotyczyć adresacji przestrzeni IO oraz adresacji sieciowej, nazwy urządzenia, specyficznych ustawień sprzętowych danych układów IO. Parametryzacja dotyczy wspomnianego dostosowywania danych i/lub działania aplikacji lokalnej – w przypadku modułów „inteligentnych".

❖ **Miejsce i rola w ISP**

RMIO stanowią węzły ISP. Niezależnie od tego, czy moduły są podłączane do kontrolera siecią niezależną, czy też są ujęte w scenariuszu sieci systemowych, stanowią one abonentów uczestniczących w obiegu informacji systemowej. Ich rola sprowadza się jednak tylko do producenta i konsumenta informacji. Dlatego, podobnie jak w przypadku sterowników dedykowanych, projektant czy programista ma ograniczony wpływ na konstrukcję ich warstwy aplikacji. Niezależnie od sposobu usieciowienia ich oddziaływania w systemie wiążą się silnie z ruchem sieciowym. Oddziaływanie na przetwarzanie informacji w warstwie aplikacji systemu objawia się tylko w dziedzinie czasu. Moduły RMIO mają wpływ na opóźnienia i podlegają zarządzaniu jakością danych użytecznych.

3.1.10. Panele HMI

(ang. Human Machine Interface Panel, Panel HMI, Industrial Panel PC, Industrial Panel Computer itp.)

Najczęściej są to komputery panelowe konstruowane w sposób umożliwiający pracę w warunkach przemysłowych oraz montaż płaski na ścianach, szafach, obudowach itp. Komputer taki wyposażony jest w interfejsy dla człowieka. Najczęściej jest to ekran dotykowy, choć można spotkać rozwiązania z wbudowaną klawiaturą membranową, a nawet z manipulatorami typu mysz. Zależnie od modelu panele mogą dodatkowo posiadać interfejsy procesowe (IO) i systemowe (sieci), umożliwiając tym samym ich użycie blisko procesu.

❖ **Dziedzina zastosowań**

Panele HMI służą do zbudowania interfejsów lokalnych między maszyną a człowiekiem (operatorem). Stanowią bazę sprzętową dla uruchamiania oprogramowania HMI lub innego przeznaczonego do tworzenia interfejsów tego typu. Panele HMI nie powinny być stosowane do uruchamiania systemów SCADA. Panele ze względów konstrukcyjnych są dedykowane do montażu na szafach sterowniczych, pulpitach lub innych elementach znajdujących się blisko procesu. Z takich miejsc trudno jest prowadzić nadzór, który jest podstawą funkcjonalną systemów SCADA. Systemem operacyjnym paneli HMI jest często RTOS, choć nie jest to regułą, szczególnie jeśli panel nie spełnia roli sterownika. Jednak jeśli w działającym oprogramowaniu są realizowane zadania sterowania, to RTOS jest niezbędny, aby dochować wymagań czasowych ich realizacji i współpracy programów z innymi węzłami RT.

❖ **Zadania**

Urządzenia panelowe można podzielić względem oferowanych przez nie lokalnych funkcjonalności:

- Panele sterujące z funkcją wizualizacji. Pełnią funkcję kontrolerów (np. sterowników PLC, 3.1.3) z ekranem, klawiaturą, interfejsem dotykowym itp. Umożliwiają realizacje zadań sterowania i stworzenie interfejsu pomiędzy człowiekiem a urządzeniem, procesem, systemem lub jego częścią. Dzięki temu możliwe jest działanie panelu jako kontroler sprzężony z graficznym interfejsem użytkownika.

Rys. 100. Przykłady paneli sterujących
Fig. 100. Examples of control panels

- Panele wizualizacyjne. Urządzenia nie mają funkcjonalności umożliwiającej prowadzenie sterowania. Służą tylko do wizualizacji. Są to jednak komputery, które można zaklasyfikować jako IPC lub PCC zorientowane na lokalny interfejs użytkownika wraz z interfejsami komunikacyjnymi. Umożliwiają dzięki temu uruchomienie zarówno prostych programów HMI (3.2.8), jak i bardziej złożonych programów nadzorczych bezpośrednio przy procesie.

Rys. 101. Przykłady paneli wizualizacyjnych
Fig. 101. Examples of visualization panels

- Konsole. Z punktu widzenia przetwarzania informacji są to urządzenia pasywne, gdyż stanowią tylko interfejs użytkownika dla konkretnego węzła lub ich grupy. Są to proste komputery umożliwiające podłączenie lokalne i wyświetlanie tekstu lub obrazu. Dane do prezentacji są przesyłane przez te węzły, np. PCC, PLC. Połączenie najczęściej jest realizowane siecią komputerową (np. MPI, Profibus, CAN, Ethernet, i inne) lub bezpośrednio z węzłem łączem szeregowym (USB, RS232/485 itp.). Panel taki może mieć interaktywny interfejs użytkownika (np. klawiatura, dotykowy). Oddziaływanie interfejsu na funkcjonowanie ISP jest pośrednie i dokonuje się przez aplikację połączonego węzła.

Rys. 102. Przykłady paneli konsolowych
Fig. 102. Examples of consoles

- Panele ekranowe. Z punktu widzenia systemowego przepływu informacji są to urządzenia pasywne. Nie są to komputery, ale zdalne monitory dla wyświetlania tekstu lub obrazu generowanego na komputerach stanowiących węzły systemu, np. PC, IPC, PCC, PLC. Połączenie jest realizowane łączem wideo (np. VGA, DVI, HDMI itp.). Panel taki może mieć interfejs dotykowy. Obsługa interfejsu jest zdalna przez połączenie z komputerem (np. RS232, USB, sieć).

Obecnie rynek paneli operatorskich jest duży, a ich dobór do konkretnego zadania można prowadzić względem parametrów technicznych, popularności rynkowej lub też zakresu funkcjonalnego. Wskazówki można znaleźć w [M84].

Rys. 103. Przykłady paneli ekranowych
Fig. 103. Examples of screen panels

❖ Miejsce i rola w ISP

Panele są typowymi węzłami interfejsowymi użytkownika w systemach ISP. Od strony systemu zapewniają środki umożliwiające współpracę z obiegami informacji charakteryzującymi się ograniczeniami czasowymi. Natomiast od strony użytkownika umożliwiają uruchomienie niemal dowolnego oprogramowania. Poza panelami sterującymi nie są one samowystarczalne i wymagają innych urządzeń do pracy. Z reguły, niezależnie od rodzaju, bez współpracy z innymi węzłami ISP panel jest nieprzydatny.

3.1.11. Sterowniki napędów

(MC, ang. Motion Controller, Drive Controller)

Urządzenia napędowe są odpowiedzialne za kinetykę elementów procesu technologicznego, czyli innymi słowy za obsługę ruchu maszyn. Do kontroli napędów wykorzystuje się specjalne sterowniki. Są to urządzenia dedykowane do zastosowań w swojej klasie. Wykorzystanie sterowników napędowych dotyczy wszelkich kinetycznych układów wykonawczych, umożliwiających zamianę energii zasilania w energię kinetyczną.

❖ **Dziedzina zastosowań**

Współczesne obiekty przemysłowe wymagają określonej, a nierzadko precyzyjnej, kontroli działania napędów (np. położenia, obrotu, posuwu), a nie tylko ich sterowania dwustanowego typu załączenie i wyłączenie. Dlatego podstawowym zadaniem sterowania napędami jest zapewnienie regulacji kinetyki układu (np. przemieszczenia, wirowania) z dużą dokładnością i wysoką dynamiką. W systemach przemysłowych stosuje się różnego rodzaju napędy. Ze względu na sposób zasilania najczęściej wykorzystywane są napędy elektryczne i pneumatyczne. Natomiast co do realizacji ruchu do najczęściej stosowanych należą napędy obrotowe i liniowe. Najbardziej powszechnymi silnikami są silniki elektryczne, w tym głównie silniki obrotowe i krokowe. Sterowniki napędów wykorzystywane są we wszystkich gałęziach przemysłu, gdzie występuje potrzeba regulacji pracy napędów.

Sterowanie napędami zależy od rodzaju napędu oraz zastosowanego silnika. Wśród napędów elektrycznych do najczęściej spotykanych można zaliczyć sterowanie silnikami prądu zmiennego (ang. AC – Alternating Current), stałego (ang. DC – Direct Current), silnikami krokowymi oraz serwonapędami. Przy napędach pneumatycznych steruje się różnego rodzaju pneumatycznymi układami wykonawczymi, w tym silnikami, siłownikami i serwomechanizmami.

Wszystkie sterowniki napędów są urządzeniami pracującymi w trybie HRT (zob. 1.1.4), a czasy reakcji często muszą być bardzo krótkie (rzędu dziesiątek μs) z bardzo niewielką niestałością (rzędu 1μs). Zatem cykle pętli sterujących oraz cykle ewentualnej komunikacji muszą również być na tym poziomie. System operacyjny kontrolera musi być typu RTOS. W praktyce w sterownikach napędów nie wykorzystuje się uniwersalnych systemów operacyjnych, lecz specyficzny dla danego modelu firmware.

❖ **Zadania**

Sterowanie napędami liniowymi, zależnie od rodzaju, sprowadza się do obsługi silników krokowych, silników DC, enkoderów, wyłączników krańcowych oraz zaworów. Dla napędów liniowych istnieją przeważnie sterowniki dedykowane, choć czasem, przy prostych

układach, pojawia się konieczność ich obsługiwania przez sterowniki uniwersalne. Wybór dedykowanego sterownika lub dedykowanego modułu sterującego w większości przypadków będzie najlepszym rozwiązaniem. Sterownik taki wymaga konfiguracji i parametryzacji pracy, natomiast oprogramowanie zadań sterująco-regulacyjnych jest wbudowane i nie wymaga modyfikacji. W przypadku konieczności stworzenia oprogramowania dla sterowników uniwersalnych, to główne zdania sterowania sprowadzają się do obsługi regulacji typu PID (zob. 1.2.4). Regulator w tym wypadku bazuje na informacji zwrotnej z impulsatorów, enkoderów, krańcówek lub innych urządzeń dających odczyt informacji opisującej rzeczywisty ruch. Czasami podstawą regulacji może być czas, ale należy pamiętać, że np. czas posuwu w silniku liniowym odzwierciedla położenie tylko z dokładnością wynikającą z prędkości oraz przy założeniu, że napęd działa sprawnie. Uruchomienie silnika lub otwarcie zaworu i odczekanie określonego czasu nie gwarantuje, że położenie elementu ulegnie zmianie zgodnie z oczekiwaniami, gdyż w algorytmie nie uwzględnia się sprzężenia od rzeczywistego ruchu. W skrajnym przypadku sterowany napęd może być uszkodzony, a brak sprzężenia spowoduje, że obraz procesu w urządzeniu sterującym będzie skrajnie niepoprawny. Dlatego czas powinien być wykorzystywany do oprogramowania zabezpieczeń działających na zasadzie kontroli warunku osiągnięcia określonego celu w określonym czasie. Podobnie z wyłącznikami krańcowymi. W regulacji położenia można bazować na odczytach z przetworników (np. zliczanie impulsów), a krańcówki potraktować tylko jako sygnalizację położeń skrajnych. Przy wykorzystaniu przetworników dających informację o ruchu w postaci impulsów należy pamiętać, że sterowniki uniwersalne (PLC) mają z reguły dość ograniczone możliwości, jeśli chodzi o akwizycję sygnałów szybkozmiennych. Z reguły standardowe wejścia dyskretne działają do częstotliwości kilku kHz, ale bywa, że minimalne czasy przy przełączaniu stanu dyskretnego są wymagane na poziomie nawet kilkudziesięciu milisekund. Wobec tego, do akwizycji sygnałów dyskretnych szybkozmiennych należy używać szybkich wejść licznikowych (ang. HSC – High Speed Counter) (zwykle od ok. kilku kHz do kilku MHz) lub specjalnych, tzw. inteligentnych, programowalnych modułów szybkich wejść reagujących już na poziomie rzędu nanosekund [291], [328], [M63]. Wykorzystanie sieci komputerowych w dziedzinie sterowania napędami również wymaga użycia rozwiązań gwarantujących cykl wymiany informacji na poziomie przynajmniej setek mikrosekund. Z reguły stosuje się dedykowane rozwiązania sieci polowych (np. Sercos, SynqNet, WorldFIP) [245], [169] lub uniwersalne protokoły RTE (np. EtherCAT, Profinet IO IRT, Sercos III, EPL) [52] (zob. 3.3.2).

Ze względów konstrukcyjnych sterowanie elektrycznymi silnikami obrotowymi jest dość złożone i wymaga zaawansowanych i specjalistycznych urządzeń wykonawczych. Przy sterowaniu takimi silnikami, poza zapewnieniem kontroli prędkości obrotowej i momentu obrotowego istotne jest sterowanie rozruchem oraz hamowaniem. W silnikach AC, przy rozruchu

bezpośrednim, prąd podawany jest na elementy indukcyjne napędu (uzwojenia), powodując powstawanie udaru prądowego, a w konsekwencji udaru momentu obrotowego. Jest to wysoce niekorzystne z punktu widzenia pracy silnika i jego żywotności. Rozruch typu gwiazda-trójkąt również powoduje, choć mniejszy, skok prądu rozruchowego. Pojawiają się też problemy związane z długim czasem trwania rozruchu oraz ze spadkami napięcia w sieci zasilającej. Wspomniane specjalistyczne urządzenia sterujące eliminują te problemy [342].
Stosowanie układów sterujących silnikami ma również pozytywny wymiar ekonomiczny, gdyż umożliwia ograniczenie zużycia energii elektrycznej. Jednak, znaczące uzasadnienie oszczędnościowe względem ceny inwestycji pojawia się dopiero, gdy w układzie rzeczywiście zachodzi regulacja. Gdy napęd jest wysterowywany stale na 100%, to nie należy spodziewać się oszczędności. Natomiast oszczędności energii rosną, im dłużej napęd pracuje z regulacją poniżej obrotów znamionowych. Najważniejsze dla określenia wymiernych zysków jest określenie charakterystyki pracy napędu w badanym okresie czasu, a nie jego moc. Typowe zakresy regulacji w procesach wykorzystujących pompy, wentylatory czy sprężarki to 55% do 100%. Są też układy pracujące w całym zakresie regulacji (np. −100% do +100%[61]) lub takie, które są stosowane ze względu na uzyskiwanie małych prędkości (regulacja 5% – 40%).

W procesach przemysłowych elektryczne układy napędowe są najczęściej sterowane przez następujące trzy rodzaje urządzeń:

- Przekształtniki częstotliwości (przekształtnik, falownik, przetwornica, inwerter, ang. speed drive) – są to urządzenia, które umożliwiają sterowanie napędami elektrycznymi AC przez zmianę częstotliwości napięcia zasilania. Głównym elementem wykonawczym przekształtnika częstotliwości jest falownik. Dokonuje on zamiany prądu stałego na zmienny, zgodnie z przyjętym sterowaniem. W praktyce, często stosuje się nazwę „falownik" wymiennie z przekształtnikiem i przemiennikiem. Inne nazwy do użytku w kontekście sterowania napędami AC są dwuznaczne i niewskazane. Przemiennik częstotliwości może być postrzegany jako swoisty pośrednik pomiędzy zasilaniem sieciowym a zasilaniem napędu. Dokonuje on prostowania napięcia zasilającego AC, a następnie przez układ sterowanego falownika wytwarza w wyjściowym układzie mocy, pożądane w danej chwili, napięcie zmienne sterujące napędem [13], [37]. Urządzenia tego typu są niezależne i samowystarczalne, jeśli brać pod uwagę obsługę napędu. Zwykle jednak mają na tyle rozbudowane możliwości integracyjne, że mogą stanowić element ISP.

[61] Dla napędów dwukierunkowych, czyli takich gdzie podłączone urządzenie pozwala na pracę w obu kierunkach. Pełny zakres regulacji dla napędów jednokierunkowych to 0 – 100%.

- Przemienniki prądu stałego – są to urządzenia sterujące wartością napięcia zasilania silników DC. Jednak względem przemienników częstotliwości są one dużo prostsze. Nie wymagają stosowania falownika. Podobnie jak przemienniki mogą pracować samodzielnie lub stanowić elementy większych systemów. Podobnie jak powyżej, z reguły mogą one stanowić element ISP zintegrowany przez układy IO lub interfejsy sieciowe.
- Układy łagodnego rozruchu (miękkiego rozruchu, ang. softstart) – są to urządzenia eliminujące wspomniany wyżej gwałtowny rozruch napędu. Urządzenia typu softstart kontrolują prąd rozruchowy silnika, zgodnie z predefiniowaną charakterystyką. Mogą one stanowić element ISP, o ile posiadają interfejs komunikacyjny i możliwości współdziałania, np. zdalnej rekonfiguracji.

W praktyce najczęściej stosowane są silniki AC, a do ich sterowania przemienniki częstotliwości [144], [13]. Aby osiągnąć żądaną charakterystykę pracy napędu, układy przekształtnikowe wymagają parametryzacji. Ogólnie należy przyjąć, że działanie przekształtnika wymaga parametryzacji dokonywanej w kontekście konkretnego obiektu oraz konkretnej aplikacji. Ze względu na mnogość parametrów opisujących pracę napędu i atrybutów śledzących jego charakterystykę pracy, konfigurację lokalną czy też określenie zasad współpracy z węzłami ISP, parametryzacji powinien dokonać ekspert od systemów napędowych.

Typowy zakres konfiguracji w kontekście algorytmizacji lokalnej dotyczy określenia zasad normalnej pracy napędu, jego rozpędzania i hamowania oraz obsługi sytuacji awaryjnych. Algorytmy sterowania i regulacji napędu może realizować komputer przekształtnika przy użyciu regulatorów wbudowanych lub dedykowany regulator zewnętrzny, zwalniając tym samym programistę z konieczności ich oprogramowywania. Komputer sterujący (układ sterowania przekształtnika) posiada pamięć konfiguracji oraz układy IO. W pamięci umieszczone są rejestry odpowiadające za poszczególne parametry jego pracy. W rejestrach znajdują się zmienne[62] sterujące, jak i zmienne aktualizowane w czasie rzeczywistym przez OS komputera i umożliwiające monitorowanie pracy urządzenia oraz podłączonego napędu. Dla przykładu mogą to być zmienne odpowiedzialne za określenie:

- mocy znamionowej silnika sterowanego w [kW],
- prądu znamionowego silnika sterowanego w [A],
- prędkości obrotowej silnika sterowanego w [obr/min],
- czasu przyspieszania w określonych jednostkach czasu na zakres obrotów,
- czasu hamowania w jednostkach czasu na zakres obrotów,

[62] O reprezentacji bitowej lub słowowej.

- źródła regulacji silnikiem, czyli wskazanie interfejsu sterującego, np. IO analogowe/dyskretne, sieć i protokół komunikacyjny, lokalny panel sterowania itp.,
- źródła sterowania silnikiem dla funkcji startu i stopu.

Rys. 104. Przykłady przekształtników częstotliwości[63]
Fig. 104. Examples of speed drives

Każdy parametr charakteryzujący pracę napędu może być postrzegany w ISP jako zmienna typu słowowego, liczbowego lub boolowskiego. Dostęp do odczytu lub zapisu jest ściśle określany i oddaje charakter produkcji i konsumpcji informacji opisanej w 1.1.1 oraz 2.4.1. Zatem, węzły ISP mogą odczytywać i zapisywać wybrane rejestry, oddziałując tym samym na pracę przekształtnika. Niektóre przekształtniki umożliwiają uruchamianie prostych programów użytkownika, udostępniając tym samym przetwarzanie lokalne.

Na rysunku 104 przedstawiono przykładowe przekształtniki zamontowane w dedykowanych szafach. Oprogramowanie sterowania silnikami AC/DC bez użycia przekształtników nie daje dużego pola manewru. Sterowniki uniwersalne nie posiadają układów IO odpowiednich do bezpośredniego sterowania takimi urządzeniami. Możliwa jest jedynie obsługa załączenia (bezpośrednie, trójkąt-gwiazda) [36] przez układy przełączające oraz obsługa urządzeń towarzyszących typu sprzęgło, hamulec, sensory.

Więcej na temat napędów i ich sterowaniu można znaleźć w [173], [348], natomiast na temat układów przekształtnikowych, np. w [37], [342]. Istnieje też wiele źródeł online, np. [W18].

❖ Miejsce i rola w ISP

W kontekście systemowym kontrolery napędów umożliwiają współpracę z innymi elementami systemu. Integracja i wynikająca z niej interakcja innych węzłów

[63] Zdjęcia dzięki uprzejmości PPHW Proloc Sp. z o. o.

z przekształtnikiem wymaga określenia interfejsów komunikacyjnych i protokołów, zasad i zakresu oddziaływania oraz monitorowania i diagnostyki. Większość parametrów pracy sterowników napędowych może być modyfikowana z zewnętrznego źródła sterowania. Zatem na działanie przekształtnika można wpływać manualnie i/lub przez nadrzędne algorytmy pracujące w innych węzłach. Komunikacja odbywa się najczęściej przez sieci komputerowe lub standardowe układy IO. Dlatego przemienniki i inne sterowniki napędów mogą być traktowane w ISP jako węzły systemu lub jako elementy wykonawcze innych węzłów sterujących systemu (kontrolerów, sterowników).

Układy napędowe stwarzają potencjalnie duże zagrożenia życia i zdrowia, ze względu na wysokie wartości prądów, jakie występują w urządzeniu. Osoby bez specjalnego przeszkolenia nie powinny znajdować się w bliskości otwartych urządzeń ani nie powinny ich obsługiwać od strony elektrycznej.

Od strony informatycznej nie różnią się one od innych urządzeń i można na nie wpływać od strony węzłów systemu rozproszonego. Jednak w większości przypadków funkcjonowanie napędów ma duże znaczenie względem bezpieczeństwa funkcjonalnego procesu, dlatego takie oddziaływanie powinno być dobrze przemyślane i zabezpieczone względem bezpieczeństwa dostępowego (zob. 3.5.4).

3.1.12. Sterowniki robotów

(RC – ang. Robot Controller)

Roboty przemysłowe są uniwersalnymi urządzeniami mechatronicznymi wykonującymi określone powtarzane czynności. Umożliwiają wykonywanie prac mechanicznych z użyciem ramienia i różnego rodzaju narzędzi. Ich uniwersalność jest ograniczona daną klasą zadań, dla której są dedykowane, oraz dostępnymi narzędziami.

Współczesne roboty posiadają własne sterowanie komputerowe. Sterowanie elementami wykonawczymi robota jest dokonywane przez dedykowany wbudowany sterownik stanowiący tzw. kontroler robota oraz wymienny program pracy. Zdarza się, że tym sterownikiem jest klasyczny PLC.

❖ Dziedzina zastosowań

Istnieją różnego rodzaju roboty, a prężny rozwój robotyki tworzy ciągle nowe konstrukcje, znajdując dla nich nowatorskie zastosowania. Do najbardziej znanych należą zastosowania robotów w eksploracji środowisk, militariach i gospodarstwach domowych. Największe jednak zastosowanie znajdują w dziedzinie przemyśle. Ich możliwości funkcjonalne silnie zależą od dziedziny. Roboty dedykowane dla procesów przemysłowych są urządzeniami pra-

cującymi w trybie HRT (zob. 1.1.4) i wykonują z reguły powtarzalne i monotonne prace wymagające dużej precyzji, jak np. spawanie, przenoszenie elementów, zgrzewanie, klejenie itp. Często na liniach produkcyjnych współpracuje ze sobą wiele robotów wykonujących kolejne etapy produkcji. Najczęściej roboty wykorzystywane są w przemyśle motoryzacyjnym i spożywczym.

System operacyjny kontrolerów robotów musi być systemem typu RTOS. Regulacje, jakie zachodzą przy sterowaniu serwonapędami, są szybkozmienne, dlatego współdziałanie komponentów robota i wszelka komunikacja musi być izochroniczna. Stosowane systemy to uniwersalne systemy RTOS lub dedykowane oprogramowanie typu firmware.

❖ **Zadania**

Sterowanie robotem sprowadza się do sterowania ramieniem we wszystkich jego osiach swobody w czasie rzeczywistym i z wymaganą precyzją. Pojawiają się zatem zagadnienia sterowania napędami ze sprzężeniem zwrotnym, zgodnie z algorytmem opisującym ruchy i czynności podłączonego narzędzia. Jako urządzenia wykonawcze ruchu robota wykorzystuje się z reguły serwonapędy.

Programowanie robota nie jest związane z programowaniem kontrolera, lecz z oprogramowaniem czynności, jakie ma wykonać. Zatem w zakresie swoich możliwości ruchu i obsługi narzędzi robot jest swobodnie programowalny. Programowanie sprowadza się do określania sekwencji ruchów ramienia i akcji narzędzia w czasie. Dokonuje się tego na poziomie abstrakcji układu współrzędnych w przestrzeni, w której może operować robot. Obsługą serwonapędów zajmują się wbudowane funkcje kontrolera.

Języki programowania robotów różnią się od języków programowania PLC. Narzędzia deweloperskie dla robotów dokonują generacji kodu sterownika na podstawie trzech metod:

- pokazywania (uczenia) ruchów – fizyczne lub wirtualne przemieszczanie robota i zapamiętywanie położeń,
- programowania bezpośredniego – użycie języków niskopoziomowych programujących każdą akcję robota, czyli opisujących drogę i sposób poruszania,
- programowania zadaniowego – użycie języków wysokopoziomowych opisujących ruch na zasadzie wskazania zadania, jakie ma być wykonane, czyli opisu celu, jaki ramię ma osiągnąć, a nie drogi.

Notacje wywodzą się z języków uniwersalnych, lecz zawierają instrukcje dostosowane do wspomagania opisu jego ruchów[64].

[64] Przykładami mogą być języki AR-Basic, Robot-Basic, Jars, Karel bazujące na Pascalu, oraz TPE, AS, VAL, AML, V/V+ i inne ([298], [246], [M97], [W13]).

❖ **Miejsce i rola w ISP**

Roboty są urządzeniami samodzielnymi i nie wymagają do pracy innych urządzeń lub systemu. Jednak podobnie jak w przypadku robotów, dzięki interfejsom komunikacyjnym kontrolera roboty mogą stać się elementem ISP. Mogą one w dużo większym stopniu stanowić węzły ISP niż opisywane dalej maszyny produkcyjne, gdyż cechują się dużo większą uniwersalnością zastosowań i same stanowią element procesu. Zakres współpracy jest jednak nieduży. Ze względów konstrukcyjnych oraz bezpieczeństwa kontroler robota jest centralny i znajduje się przy urządzeniu. Wszelkie algorytmy sterujące ramieniem nie są rozpraszane. Istnieją prace związane ze zdalnym lub rozproszonym sterowaniem robotami [34], ale wyniki nie są zadowalające dla zastosowań HRT. Robot, jako węzeł ISP może głównie dostarczać informacji diagnostycznych oraz opisujących przebieg procesu. Oddziaływanie na taki węzeł sprowadza się do parametryzacji pracy, wybrania receptur produkcyjnych lub przeprogramowania serwisowego.

3.1.13. Skomputeryzowane maszyny

(CNC – ang. Computerized Numerical Control, Computerized Machine)

W procesach produkcyjnych stosuje się często maszyny skomputeryzowane, zwane też maszynami sterowanymi numerycznie. Stanowią one konstrukcyjną i funkcjonalną całość i mają na celu wytworzenie lub obróbkę konkretnego produktu lub półproduktu przy użyciu zaprogramowanej metody oddziaływania na materiał [167].

❖ **Dziedzina zastosowań**

Podobnie jak robotów, ich działanie jest uniwersalne w pewnej klasie zastosowań, a z racji sterowania numerycznego konkretne działanie w tej klasie zależy od wymiennego programu. Doskonałym przykładem mogą być samodzielne obrabiarki sterowane numerycznie, takie jak frezarki czy tokarki. Równie dobrym przykładem są maszyny typu wtryskarki, obcinarki, etykieciarki, i inne maszyny przeznaczone do realizacji podprocesu technologicznego określonej klasy, a sterowane z użyciem wbudowanego komputera lub własnego, wbudowanego ISP.

Maszyny posiadają często dwa komputery. Jeden jest kontrolerem sterującym pracą urządzeń wykonawczych i działa pod kontrolą RTOS w rybie HRTS. Natomiast drugi stanowi interfejs dla operatora i jest rodzajem panelu HMI. Ponieważ kontrolerami maszyny są często sterowniki dedykowane lub uniwersalne PLC, to stosowane OS wynikają z ich typów i konstrukcji. Systemem komputera interfejsu jest najczęściej jakiś system popularny typu Windows.

❖ **Zadania**

Maszyny skomputeryzowane stanowią rozwiązanie dedykowane dla realizacji konkretnych czynności. Ponieważ podobnie jak roboty są one programowalne i mogą pracować z jednym lub z wieloma narzędziami, więc są w zakresie swoich możliwości wszechstronne. Mogą przez to realizować wiele różnych zadań obrabiających na rzecz jednego obrabianego elementu. Ograniczenie zakresu realizowanych zadań wynika z zestawu obsługiwanych narzędzi oraz rodzajów i rozmiarów materiału, na którym maszyna może pracować. Maszyna dostarcza kompletu środków technicznych do realizacji zadań, w tym własny podsystem ISP odpowiedzialny za sterowanie, regulacje, diagnostykę, interfejs dla operatora i inne funkcjonalności (por. 1.1.5, 1.2.6) oraz podsystem zabezpieczeń technologicznych (ang. safety, por. rozdział 3.5.4). W OS panelu oraz uruchomionej aplikacji HMI powinny być skonfigurowane i używane mechanizmy bezpieczeństwa dostępowego (zob. 3.5.4), zgodne z polityką bezpieczeństwa w zakładzie oraz zgodne z ustalonymi zasadami dostępu pracowników do maszyn.

Najczęściej maszyny tego typu posiadają dedykowany komputer, oprogramowanie użytkowe i narzędzia. Języki programowania maszyn CNC[65] różnią się od języków uniwersalnych lub języków dla PLC. Dostarczają one głównie środków do opisu czynności, jakie maszyna ma w danym momencie wykonać. W nowoczesnych maszynach i narzędziach programujących, wiele z czynności programujących jest zautomatyzowana dzięki możliwości wykorzystywania plików CAD/CAM opisujących wyrób z fazy jego projektowania. Podobnie jak dla robotów programowanie nie dotyczy kontrolera maszyny, lecz programu, jaki ten kontroler ma wykonać na rzecz obrabianego materiału.

Programowanie maszyn niestanowiących obrabiarek CNC wynika z zastosowanych kontrolerów (np. PLC i ich języki zob. 3.1.3) lub z dedykowanych rozwiązań (zob. 3.1.6). Zadania programistyczne, z jakimi można się zetknąć przy tworzeniu kodu, to oprogramowanie maszyn stanów, czyli automatów sekwencyjnych oraz obsługa zaworów, krańcówek, różnych pomiarów, sygnałów sterujących, wartości nastaw, parametrów itp., oraz oprogramowania algorytmów bezpieczeństwa i trybów serwisowych. Doskonale nadają się do tego języki opisu sekwencji[66] wraz z uniwersalnymi językami logicznymi[67].

❖ **Miejsce i rola w ISP**

Skomputeryzowane maszyny, a w szczególności obrabiarki numeryczne klasy CNC, są z założenia układami zamkniętymi stanowiącymi sprzętowo i funkcjonalnie całość przeznaczoną do konkretnych zadań. Jednak dzięki interfejsom komunikacyjnym, podobnie jak ste-

[65] Przykładem języków dla CNC mogą być: G code (ustandaryzowany ISO6983, PN73M55256, PN92M55251), APT, EXAP, ADAPT.
[66] SFC, Grafcet, Graph itp.
[67] IL, LD, FBD, ST itp.

rowniki napędów, sterowniki maszyn CNC mogą stać się elementem ISP. Nie należy się jednak spodziewać, że zakres integracji będzie duży. Maszyna CNC, a zatem i jej sterownik nie są pomyślane jako węzły ISP. Możliwości integracyjne są zorientowane na uruchomienie kilku maszyn w procesie wsadowym lub składającym się z etapów obrabiania elementu na różnych maszynach. Można zatem zbudować linię produkcyjną z nadrzędnym ISP, gdzie, upraszczając, węzłami produkcyjnymi będą maszyny, do przekazywania elementów między maszynami wykorzystane zostaną roboty, taśmociągi, podajniki itp., a nad całością będzie czuwał sterownik.

Ponadto, integracja maszyn CNC w informatyczne struktury zakładowe dotyczy systemów pracujących powyżej ISP, czyli na systemy kontroli produkcji i wytwarzania (zob. 1.2.1). Podobnie jest z wszelkimi innymi maszynami stanowiącymi zamknięte rozwiązania dostarczające kompletny proces lub podproces produkcyjny. W takich przypadkach maszyna ma swój własny ISP, który tylko w pewnym zakresie wynikającym z filozofii działania maszyny może zostać zintegrowany z nadrzędnym ISP lub innym systemem IT.

3.1.14. Komputery serwerowe

Są to dedykowane komputery do pracy w roli serwerów, czyli jako baza sprzętowa dla uruchamiania oprogramowania serwerowego. Ich konstrukcja nie jest przeznaczona do pracy w trudnych warunkach przemysłowych. Powinny jednak odznaczać się podwyższoną trwałością ze względu na fakt, że z reguły pracują non stop, a ich awaria może powodować uniemożliwienie normalnego przebiegu produkcji. Komputery serwerowe powinny być umieszczane w specjalnych pomieszczeniach tzw. serwerowniach, które zapewniają kontrolowane warunki środowiskowe (np. klimatyzacja), zabezpieczenie zasilania (np. zasilacze awaryjne) oraz zabezpieczają dostęp fizyczny i mogą stanowić strefy bezpieczne EM.

Rys. 105. Przykłady komputerów serwerowych do montażu w szafach
Fig. 105. Examples of servers mounted in cabinets

❖ **Dziedzina zastosowań**

Komputery serwerowe służą do zapewnienia zasobów dla realizacji zadań programów serwerowych. Ich znaczenie dla ISP może być kluczowe. Od realizowanych przez nie usług często zależy poprawne funkcjonowanie obiegu informacji w systemie, jej dostępności i możliwości składowania. Jednak z racji, że komputery serwerowe nie stanowią bezpośredniego elementu ISP, pracują w miejscach oddalonych od procesu oraz często są poza administracją utrzymania ruchu, mogą być niedostępne. Dlatego, mimo ogólnych wymagań względem trwałości, przy projektowaniu rozwiązań z wykorzystaniem serwerów należy zawsze rozważyć działanie systemu przy chwilowej niedostępności takiego komputera.

❖ **Zadania**

Funkcjonalność komputera serwerowego może być wieloraka i zależy od oprogramowania na nim uruchomionego. Najczęściej platform serwerowych, jako hostów dla typowo wymaganych usług w ISP, używa się do obsługi: baz danych, serwisów web, jak również do realizacji usług serwerów zasobów (ang. NAS – network attached/access storage) czy też serwerów pośredniczących w wymianie danych i w zdalnym dostępie, uruchamianych w celach zwiększenia bezpieczeństwa (np. serwery DMZ, serwery dostępowe, ang. hopping/jump itp.) (zob. 4.2.1).

Urządzenie i jego system operacyjny powinien zapewniać standardowe i zaawansowane funkcjonalności układów serwerowych (np. backup, failover, storage, hot-swap, hot-plug itp.). Bardzo ważne jest, aby konstrukcja sprzętowa serwera umożliwiała ciągłą i bezawaryjną pracę (wysokie MTBF, zob. 4.2.5) oraz szybkie przywrócenie funkcjonalności w razie wystąpienie problemów (zob. 4.2.3). Im bardziej odpowiedzialną funkcję pełni w systemie, tym bardziej funkcjonalności serwerowe powinny być zapewnione i używane.

❖ **Miejsce i rola w ISP**

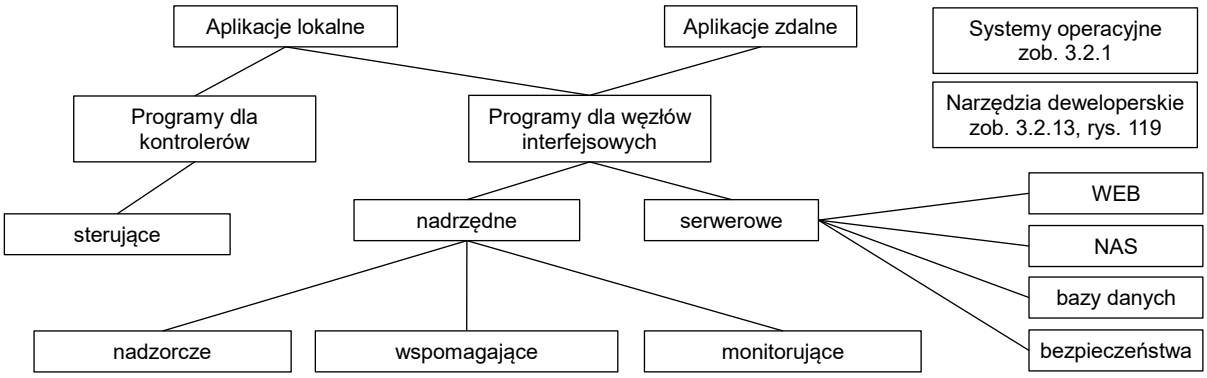
Serwery współpracują z węzłami ISP, które działają z ograniczeniami czasowymi, przy czym oprogramowanie serwerowe (por. 3.2.3) bardzo rzadko oferuje pracę z takimi ograniczeniami. Dlatego, aby zapewnić możliwie najlepszą responsywność serwera względem ISP, należy w miarę możliwości wydzielić sieć do komunikacji z ISP oraz zapewnić komponenty sprzętowe, umożliwiające przetwarzanie bez groźby wystąpienia braku zasobów. Upraszczając, serwery działające we współpracy z ISP powinny być wyposażone w niezależne karty sieciowe, redundantne dyski, wystarczającą przestrzeń pamięci operacyjnej, dostateczną moc procesora oraz dostęp do nośnika dla kopii zapasowych.

3.2. Oprogramowanie

Każde urządzenie komputerowe musi posiadać program, według wykonania którego działa. Dla prostych urządzeń dedykowanych wystarczy pojedynczy program z funkcjami obsługi przerwań. Natomiast dla bardziej złożonych konstrukcji potrzeba wielu programów odpowiedzialnych za realizację konkretnych zadań w urządzeniu. Podstawowym z nich jest wykonanie zadań funkcjonalnych związanych z aplikacją ISP, czyli wypełnienie roli, jaką węzeł pełni w systemie. Jednak, dla zapewnienia działania węzła zgodnego z wymaganiami ISP, szczególnie względem ograniczeń czasowych, muszą istnieć zadania:

- kontrolujące wykonanie programów użytkownika,
- komunikujące programy ze sobą i z zasobami sprzętowymi węzła (por. 2.4.2),
- zarządzające pracą węzła, interfejsowe i diagnostyczne.

Zadania takie są wykonywane w środowisku systemu operacyjnego osadzonego w urządzeniu lub pod kontrolą specjalnych programów wbudowanych w urządzenie tzw. programów układowych (ang. firmware). Rzadko w ISP spotyka się urządzenia programowane działające bez jakiegoś rodzaju oprogramowania „pokładowego”, a tym samym programowane bezpośrednio na poziomie warstwy sprzętu. Tworzenie takich rozwiązań dla urządzeń swobodnie programowalnych nie jest wskazane z powodu naturalnie pojawiającej się w takich przypadkach swoistości rozwiązań. Pociąga ona za sobą utrudnione tworzenie aplikacji ze względu na specjalistyczną obsługę, narzędzia i serwis oraz nietypowe lub niskopoziomowe (tzn. z reguły trudne i czasochłonne) sposoby programowania. Węzły oprogramowywane na poziomie sprzętu są jednak powszechnie spotykane jako urządzenia dedykowane, gdzie nie ma potrzeby dokładania warstwy aplikacyjnej użytkownika.



Rys. 106. Ogólny podział funkcjonalny dedykowanych programów aplikacyjnych
Fig. 106. General functional classification of dedicated application programs

Tak jak istnieje wiele rodzajów węzłów, tak i istnieje wiele funkcjonalności programów uruchamianych w takich węzłach. W ISP można wyróżnić kilka kategorii programów, któ-

rych dziedzina zastosowań jest wyraźnie oddzielona od innych. Na rysunku 106 zilustrowano ogólny podział funkcjonalny typów oprogramowania ISP i ich zależności. Opis wymienionych typów zawarto w dalszej części podrozdziału. Warto zauważyć, że programy sterujące nie stanowią elementów aplikacji zdalnych tak jak i elementów aplikacji urządzeń interfejsowych (por. 1.2.5 oraz 4.1.2).

Poniżej opisano najczęściej spotykane rodzaje oprogramowania używane w komponentach ISP. Uwzględniono systemy operacyjne oraz środowiska deweloperskie, które z punktu widzenia programisty stanowią raczej narzędzia, a poniższą listę spinają swoistą klamrą.

3.2.1. Systemy operacyjne

W ISP stosuje się różne systemy operacyjne i nie ma zasady, która faworyzowałaby użycie jakiegoś konkretnego typu. Istotne jest, aby dobrać system operacyjny do wymagań węzła, a dokładnie jego roli w ISP. Jeśli dany węzeł ISP działa z ograniczeniami czasowymi, to system powinien być systemem operacyjnym czasu rzeczywistego (ang. RTOS – Real Time Operating System). Da to możliwość przydziału gwarantowanych szczelin czasowych zasobów przetwarzających na wykonanie zadań krytycznych czasowo. W przeciwnym wypadku inne zadania systemowe lub użytkownika mogą przejąć zasoby przetwarzające na nieokreślony czas, uniemożliwiając wykonanie zadań sterowania w wymaganym limicie. W RTOS zadania są szeregowane w sposób umożliwiający utrzymanie ich aktywności, nawet gdy system jest maksymalnie obciążony. Dodatkowo mechanizm szeregujący (ang. scheduler) i wywłaszczający (ang. preemptive multitasking) musi gwarantować przewidywalny czas oczekiwania na taką obsługę. Jednak co do dotrzymywania ograniczeń czasowych na konkretnym poziomie, to należy być świadomym, że moc procesora „nie jest z gumy”. Jeżeli zadanie ma się wykonywać częściej, to w sytuacji, gdy nie ma już zapasu mocy, musi się to odbywać kosztem innych zadań. Dlatego RTOS nie zapewnią wykonania dowolnie dużej liczby zadań w dowolnie małym czasie. Jedynie mogą zagwarantować, że zadania będą sprawiedliwie szeregowane, zgodnie z przyjętą regułą. Typowym algorytmem szeregowania jest algorytm rotacyjny, tzw. round-robin i EDF (zob. 2.1.2). Do popularnych RTOS można zaliczyć:

- QNX – system z mikrojądrem (ang. microkernel), w którym nie rozdziela się operacji na warstwę jądra i użytkownika. W tym systemie każdy sterownik sprzętu, program czy nawet system plików działa w separowanej i zabezpieczanej przestrzeni użytkownika. Zadania są wywłaszczane na bazie priorytetów. Priorytety są dziedziczone.
- VxWorks – RTOS przeznaczony dla systemów wbudowanych. Jądro zarządza szeregowaniem i wywłaszczaniem zadań. Programy użytkownika w trybie RT są separowane

i chronione w pamięci od programów non-RT i zadań systemowych. Priorytety są dziedziczone. Bardzo często stosowany jest również w PLC.

- RT-Linux – bazuje na mikrojądrze, które szereguje zarówno zadania RT, jak i zadania standardowego systemu Linux jako wywłaszczalne procesy. Procesy użytkownika RT są traktowane jako moduły jądra systemu RT i uruchamiane z priorytetem jądra. Proces systemu Linux jest uruchamiany z niskim priorytetem. Dlatego procesy RT działają niezależnie od tego, jakie zadania systemu operacyjnego są wykonywane.
- Windows Embedded – wcześniej oznaczany jako Windows CE. RTOS przeznaczony jest dla systemów wbudowanych. Szeregowanie i wywłaszczanie zadań bazuje na dziedziczonych priorytetach. System zabezpiecza procesy użytkownika i jądro w wirtualnej przestrzeni pamięci. Wcześniejsze wersje systemu nie spełniały wymogów RTOS ze względu na niezdeterminowaną obsługę przerwań.

Nie jest to kompletna lista dostępnych rozwiązań. Istnieje wiele innych mniej lub bardziej popularnych systemów i mikrosystemów RTOS dedykowanych dla systemów wbudowanych, jak FreeRTOS, ThreadX, LynxOS, BeRTOS, Keil RTX, NuttX itp. [350], [280], [21], [W6], [W21]. Są to rozwiązania głównie otwarte (ang. open source) rozwijane przez publiczne grupy wsparcia.

W PLC najczęściej spotyka się systemy dedykowane, których konstrukcja wewnętrzna i charakterystyka pracy nie są publikowane. Systemy te szeregują zadania systemowe w trybie RT, natomiast programy użytkownika wraz z zsynchronizowanymi z nimi akcjami uruchamiają w cyklach o kontrolowanym czasie wykonania. Z systemów uniwersalnych najpopularniej stosowane w PLC to systemy VxWorks oraz Windows Embedded (CE, XP embedded).

Dla przykładu, uruchomienie programu sterującego obiektem w warstwie użytkownika popularnego systemu operacyjnego klasy Windows spowoduje, że, co prawda, przez większość czasu zadania sterowania będą prawdopodobnie realizowane poprawnie, ale w pewnych momentach pracy ich realizacja może zostać zaburzona. Wynika to z możliwości dowolnego wstrzymania działania wątków przez wątki o wyższym priorytecie lub wręcz przez procesy systemowe, nad którymi programista ma tylko ograniczoną kontrolę.

Należy programistom zwrócić uwagę, że popularne systemy operacyjne, nawet te klasyfikowane jako non-RTOS, mają w swych jądrach wbudowane mechanizmy wspomagające zarządzanie wykonywaniem procesów i ich wątków. Dla przykładu, API popularnego systemu Windows udostępnia możliwość uruchamiania procesów w kilku klasach priorytetów, a ich wątki z wieloma priorytetami w klasie. Ponadto, istnieją funkcje zarządzające wykonaniem, synchronizacją i komunikacją międzyprocesową i międzywątkową. Z reguły są też dostępne możliwości wbudowania programu w jądro systemu, co również może poprawić

jego charakterystykę działania w czasie. Warto te możliwości poznać i stosować, szczególnie gdy charakterystyka czasowa pracy węzła ma znaczenie. Twierdzenie, że program posiada charakterystykę niezgodną z oczekiwaniami z powodu systemu operacyjnego, jest często wykrętem leniwych lub niedouczonych programistów.

3.2.2. Programy dedykowane

Specyficzne środowisko przemysłowe (por. 1.1.2) determinuje tworzenie dość nietypowych, względem środowiska biurowego i domowego, programów i to nie tylko w kwestii ich funkcji, ale i możliwości ponownego zastosowania. Trudno jest znaleźć dwa identyczne obiekty przemysłowe, nawet jeśli funkcjonalnie są bardzo zbliżone. Wynika to z faktu, że maszyny i linie produkcyjne nie są kupowane „z półki", lecz produkowane na zamówienie pod konkretnego klienta. Ponadto, proces zakupowy z punktu widzenia danego zamawiającego realizowany jest sporadycznie. Obiekty przemysłowe są technicznie złożone i z biegiem czasu konstrukcyjnie ewoluują w ramach rozwoju i dostępności elementów składowych, a także samej technologii. Dlatego zestawy zakupione w jednym czasie będą identyczne lub bardzo podobne, a zakupione z odstępem np. roku mogą się już znacząco różnić, zarówno w kwestii konstrukcji mechaniczno-elektrycznej, wyposażenia AKP oraz komputerów i układów sterujących, w tym PLC i oprogramowania. W efekcie, nawet dwie linie produkcyjne, od tego samego producenta, produkujące ten sam produkt, na tej samej hali produkcyjnej wymagają często innego oprogramowania. Zależnie od zastosowanego sprzętu różnice dotyczą systemów operacyjnych, programów aplikacyjnych, lokalnych baz danych, wizualizacji i programów narzędziowych. W praktyce zdarza się, że integrator, ze względu na wprowadzone modyfikacje, staje przed problemem gruntownej adaptacji programu dla sterownika, rozbudowy wizualizacji, dostosowania komunikacji itp. Bywa, że stopień trudności jest porównywalny ze stworzeniem nowej aplikacji, a bywa również, że w ogóle nie może tego zrobić, gdyż zmiany wymagają użycia niedostępnych narzędzi lub są w sprzeczności z przyjętymi w istniejącym ISP założeniami. Dlatego tworząc oprogramowanie dla ISP, dość rzadko spotyka się możliwość powielenia aplikacji i większość kodu ma charakter dedykowany dla konkretnego przypadku.

Programy dedykowane tworzone na potrzeby ISP i jego elementów można podzielić ze względu na funkcje, jakie pełnią w systemie oraz na zakres terytorialny oddziaływania. Poniżej przedstawiono zgrubny, choć oddający istotę zastosowań, podział.

- programy sterujące

  Programy sterujące są tworzone dla komputerowych urządzeń sterujących i są odpowiedzialne za realizację zadań obsługi procesu (akwizycji, sterowania, regulacji).

Kod jest tworzony w językach uniwersalnych[68] i/lub w specjalnych językach programowania[69], wspomagających programowanie algorytmów i struktur danych typowych dla obiektów przemysłowych. Forma wynikowa kodu jest tworzona w postaci wykonywalnej, binarnej lub interpretowanego pseudokodu. Czasami, dla niektórych urządzeń i narzędzi, możliwy jest zapis pseudokodu wraz z symbolami i innymi pomocniczymi informacjami opisującymi dany program. Umożliwia to późniejszy odczyt informacji źródłowych programu, w tym kodu, symboli i komentarzy, z urządzenia.

Programy mogą być zabezpieczane przed odczytem i/lub modyfikacją zarówno na poziomie narzędzia deweloperskiego (por. 3.2.13), jak i urządzenia. Blokowanie dostępu do kodu nie jest jednak dobrą praktyką inżynierską i działa na niekorzyść klienta (użytkownika programu) w dłuższej perspektywie użytkowania. Nikt nie zagwarantuje, że autor programu (firma, programista) będzie dostępny po danym okresie czasu, i tym samym podejmie się akcji serwisowej lub modyfikacji kodu. Zatem, jeśli deweloper nie przekaże dostępu do źródeł programów sterujących, to tym samym skazuje użytkownika na realne ryzyko unieruchomienia procesu. Poza tym, wymuszanie powiązania serwisowego przez blokady dostępu lub zaciemnianie kodu świadczy raczej o nieporadności konkurowania dewelopera na gruncie jakości produktu i świadczonych usług. Dobrym rozwiązaniem kompromisowym jest blokowanie dostępu do kodu tylko w okresie gwarancyjnym.

W praktyce, do najczęściej spotykanych zastosowań programów sterujących należą programy tworzone dla urządzeń klasy PLC. Jako przykład można przytoczyć program sterownika sterującego pracą konkretnej maszyny, np. wtryskarki, maszyny tnącej, etykietującej, pieca, filtra itp., sterującego linią produkcyjną, np. produkcji kabla, papieru, karbidu itp., lub sterującego procesem, np. ciepłowniczym, demineralizacji wody, pasteryzacji, czy choćby przepompowywania ścieków. Programy sterujące pracują w większości systemów przedstawionych w rozdziale 7.1.

- programy nadrzędne

  Są to programy, które służą do celów realizacji pozostałych zadań ISP (zob. 1.1.6), czyli diagnostyki, zarządzania, monitorowania, raportowania oraz innej obróbki i prezentacji danych na poziomie wyższym niż procesowy. Przeważnie są mocno zintegrowane ze strukturą i ideą działania danego ISP. Nie istnieją typowe programy aplikacyjne. Ich zakres działania wynika z danego ISP a motywacja istnienia pochodzi przeważnie od wymaganych zadań funkcjonalnych, których nie można zrealizować lub nie opłaca się

[68] Zwykle C lub Basic.
[69] Zgodnych z normą implementacji LD, FBD, ST, IL, SFC, ich implementacji specyficznych oraz innych specyficznych języków, jak STL, FUPLA, S-Edit, CFC, SCL, Graph, Grafcet, Hi-Graph i inne.

realizować z użyciem dostępnych programów i narzędzi uniwersalnych klasy HMI, SCADA, MES, ERP (zob. 3.2.7, 3.2.8, 3.2.9, 3.2.10). W praktyce programy te tworzone są na podstawie specyfikacji wymagań i z użyciem typowych narzędzi i środowisk deweloperskich (zob. 3.2.13) lub uniwersalnych środowisk narzędziowych (np. MS Office, Open Office, Matlab, LabView itp.) spersonifikowanych dla danego klienta/aplikacji.

Przykładami mogą być dedykowane programy do specjalizowanych wydruków, generacji raportów, zestawień i dokumentów, dedykowanych wizualizacji i interfejsów użytkownika, specjalizowanego przetwarzania danych, czy też innych programów służących do interakcji człowieka z urządzeniami i maszynami.

- aplikacje lokalne

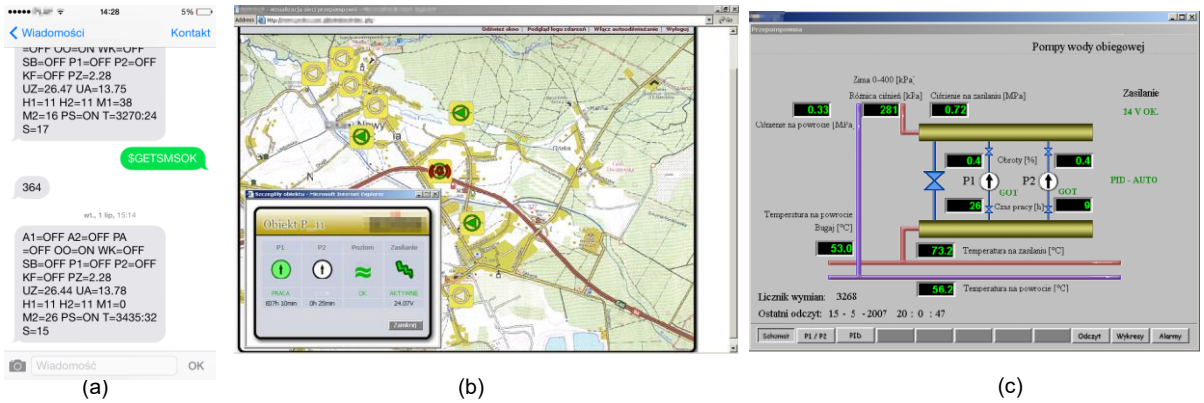
  Pojęcie „aplikacji” dotyczy w tym przypadku tworzenia konfiguracji, zależności i powiązań oraz skryptów na potrzeby działania uniwersalnych programów funkcyjnych różnego rodzaju, np. narzędziowych, wizualizacyjnych, bazodanowych itp. środowisk uruchomieniowych (ang. developement and run-time tool, environment) dla aplikacji. Termin aplikacji w tym kontekście nie jest szczególnie szczęśliwym określeniem, gdyż może mylić z pojęciem aplikacji użytkownika (por. 2.4.5). Aplikacje, o których mowa, nie są programami w rozumieniu informatycznej definicji programu, a jedynie zestawem danych parametryzujących program do właściwego działania. Programy, na rzecz których powstają aplikacje, mają zbiór uniwersalnych funkcjonalności w ramach swego przeznaczenia, i są do niczego nieprzydatne w swej podstawowej formie. Dopiero aplikacja powoduje, że program zaczyna działać zgodnie z oczekiwaniami względem wymagań danego ISP. Mimo to nazwa „aplikacja” jest powszechnie stosowana i nie jest błędna, gdyż to właśnie ona urzeczywistnia warstwę aplikacji użytkownika w programie, dla którego jest przeznaczona.

  W praktyce, do najczęściej spotykanych programów funkcyjnych tego typu należą programy typu SCADA, HMI, MES, ERP (punkty poniżej). Aplikacjami dla nich są dostosowane do konkretnych potrzeb zakładu wizualizacje, interfejsy użytkownika, zgrupowane funkcjonalności zarządzania, obsługi działów, integracje itp.

- aplikacje zdalne

  Do programów dedykowanych należy zaliczyć również programy umożliwiające uruchomienie aplikacji webowych i mobilnych. W tej dziedzinie w ISP najczęściej wykorzystuje się standardową, internetową usługę www. Nie wymaga ona instalacji specjalistycznego oprogramowania po stronie klienta. Większość współczesnych pakietów oprogramowania do wizualizacji procesów przemysłowych udostępnia

funkcjonalność publikowania lokalnych ekranów wizualizacyjnych w przestrzeni publicznego Internetu z użyciem języka HTML i JavaScript. Są to wygodne i proste w użyciu funkcje, niewymagające dużej wiedzy z zakresu technologii internetowych. Czasami spotyka się również wykorzystanie innych usług, jak FTP (generacja raportów, zestawień itp.), poczty elektronicznej (sygnalizacja zdarzeń, raporty, dokumenty itp.), telnet (zdalna administracja, diagnostyka, monitoring itp.) [15]. Szeroko natomiast stosowane są usługi sieci komórkowych. Najprostsze i bardzo popularne jest wykorzystanie usługi krótkich wiadomości tekstowych (SMS) do przesyłania informacji o zdarzeniach, stanie pomiarów lub stanie całych obiektów.



(a) (b) (c)

Rys. 107. Przykłady interfejsów dostępu do danych
Fig. 107. Examples of interfaces for data access

Ponadto, do ciągłego monitorowania obiektów często wykorzystuje się transmisje pakietowe na bazie protokołu GPRS [260]. Wówczas jednak na urządzeniu mobilnym musi znajdować się aplikacja umożliwiająca obsługę takiej transmisji. Podobnie z innymi urządzeniami mobilnymi, niekoniecznie GSM. Tablety, smartfony bądź inne urządzenia pracujące w zakładowej sieci WiFi stanowią ciekawą i wygodną alternatywę dla stacjonarnych paneli HMI. Widoki interfejsów użytkownika dla przykładowych aplikacji zdalnych przedstawiono na rysunku 107: (a) na bazie sygnalizacji SMS, (b) wizualizacji web i (c) lokalnego ekranu wizualizacji.

Niezależnie od powyższych kategorii można rozróżnić programy dedykowane ISP, dzieląc je według obszarów działania technologii informatycznych. Są to programy:

- dla pracy obiektowej (sterujące)
  Zgodnie z opisem powyżej. Technologie tutaj stosowane to głównie specjalizowane technologie informatyki przemysłowej związane z pracą w HRTS i FRTS.

  Bardzo ważne są tutaj zagadnienia bezpieczeństwa w rozumieniu funkcjonalnym (ang. safety, zob. 3.5.4). Ponadto, ze względu na istnienie nowych zagrożeń należy rozważać również zagrożenia bezpieczeństwa dostępu do zasobów sprzętowo-programowych.

- dla pracy biurowej (desktopowe)
  W rozwiązaniach desktopowych oraz we wszystkich nieobiektowych, których praca z założenia nie jest ograniczana czasowo, stosuje się powszechne technologie IT. Istnieją podsystemy programowe dedykowane dla współpracy z ISP, jak systemy SCADA, MES czy ERP. Są one optymalizowane pod kątem wymagań ISP, ale bazują na ogólnych technologiach IT. Programy desktopowe nie powinny realizować funkcji sterowania. Urządzenia, na których takie programy są uruchamiane, czyli z reguły komputery osobiste (PC, IPC), nie gwarantują obsługi w trybach HRT czy FRT. Uzależnianie działania procesu od informacji wypracowywanej w takich trybach musi wiązać się ze źródłem informacji gwarantującym przetwarzanie w takim reżimie.

  Z poziomu pracy biurowej zagrożenia dla bezpieczeństwa funkcjonalnego procesu są dużo niższe niż na poziomie programów sterujących. Jednak program desktopowy współpracujący z programami sterującymi, i posiadający błędy, może doprowadzić do zagrożeń funkcjonalnych. Podobnie może skutkować brak walidacji działań użytkownika, w tym kontroli bezpieczeństwa od strony zakresu dostępu i jego uprawnień.

- dla urządzeń mobilnych (mobilne)
  Dobre programy działające zdalnie z poziomu urządzeń mobilnych nie powinny zawierać funkcji programów sterujących. Wynika to z bezpieczeństwa prowadzenia procesu. Urządzenia mobilne są z natury tymczasowo obecne w systemie, a ich dostęp do infrastruktury procesowej nie jest kontrolowany względem ograniczeń czasowych. Ponadto, urządzenia mobilne bardzo rzadko dysponują systemami operacyjnymi klasy RTOS. Zastosowaniem tego typu programów jest dziedzina nadzorcza z dostępem do ISP o charakterze tymczasowym i incydentalnym (na żądanie).

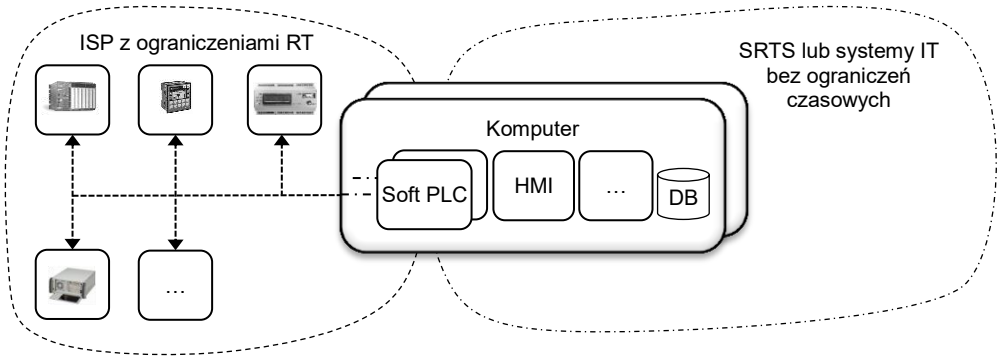
  Zagrożenia bezpieczeństwa od strony funkcjonalnej, przy dobrze napisanej aplikacji zdalnej, praktycznie nie istnieją. Istnieją natomiast zagrożenia bezpieczeństwa względem nieupoważnionego dostępu do zasobów danych procesowych. Zagrożenia pochodzą od niewłaściwego użycia programu (lub jego błędów) oraz od dostępu z sieci bezprzewodowej. Nie należy zapominać, że siecią urządzeń mobilnych może być nie tylko bezprzewodowa sieć zakładowa, ale i sieć publiczna (np. GSM).

- dla środowiska webowego (webowe)
  Podobnie jak dla urządzeń mobilnych, dostęp do ISP z poziomu sieci publicznej nie może się wiązać z zadaniami sterowania. Technologie stosowane dla tego typu aplikacji są takie same jak dla innych aplikacji webowych. ISP nie odznacza się żadnymi rozwiązaniami dedykowanymi dla ISP. Nawet jeśli ktoś udostępnia gotowe produkty dla aplikacji ISP, to bazują one na typowych technologiach web.

Zagrożenia bezpieczeństwa od strony zdalnego dostępu są znacząco większe niż przy programach mobilnych. Urządzenia mobilne są z reguły przypisane do użytkownika i przez to chroniony jest do nich bezpośredni dostęp. Aplikacje zdalne działające w przestrzeni sieci publicznych, jakim jest sieć Internet, umożliwiają potencjalny atak każdemu, kto ma ochotę spróbować. Dlatego zagadnienia bezpieczeństwa dostępowego są kluczowe przy aplikacjach webowych.

3.2.3. Sterowniki programowe

Funkcjonalności oferowane przez PLC można oderwać od fizycznego urządzenia i potraktować jako zadania uruchamiane w środowisku innego komputera, tzn. nie PLC. Jeśli taki komputer wykorzystuje RTOS umożliwiający zarządzanie czasem wykonywania takich zadań, to uzyskuje się sterownik programowy działający w czasie rzeczywistym z ograniczeniami wynikającymi z własnej konstrukcji oraz z możliwości środowiska, w którym został uruchomiony. Gdy zadanie jest uruchomione w środowisku OS bez mechanizmów RT, wówczas uzyskuje się funkcjonalność sterownika z ograniczeniami miękkimi. Niezależnie od uzyskanych właściwości względem RT popularna nazwa takich rozwiązań to sterownik programowy (ang. Soft PLC), gdzie słowo „soft" pochodzi od angielskiego „software", czyli programowy a nie od Soft-RT (por. 1.1.4). Na rysunku 108 przedstawiono schemat wykorzystania Soft PLC jako jednego z wielu funkcjonalności uruchamianych na komputerze pracującym w różnych podsystemach RT. Warto zauważyć, że na komputerze może zostać uruchomionych więcej niż jeden sterownik oraz że może być więcej komputerów niż jeden oferujących „hostowanie" dla sterowników.

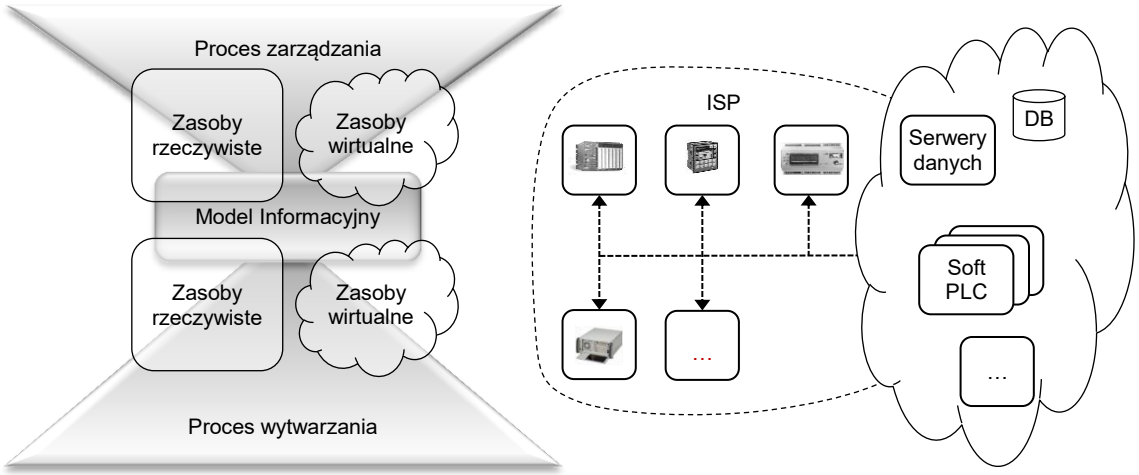


Rys. 108. Lokalizacja sterownika programowego w ISP
Fig. 108. Location of soft PLC within ICS

Od strony funkcjonalnej, a także od strony oprogramowania aplikacyjnego sterownik programowy nie różni się od klasycznego PLC. Wykorzystywane są takie same strefy pamięci, języki programowania, wejścia i wyjścia, cykl i możliwość definiowania zdarzeń acyklicznych. Różnica sprowadza się do fizyczności sprzętu. Soft PLC może być uruchamiany

w węźle ISP typu sterownik PC, IPC, a nawet komputer klasy PC, o ile zachowane zostaną względy bezpieczeństwa.

Ponadto, jeśli węzeł na to pozwala, to można uruchomić wiele sterowników wirtualnych w jednym węźle fizycznym. W skrajnym przypadku, funkcjonalność PLC może nawet zostać przeniesiona do zakładowej chmury obliczeniowej (ang. cloud networking, cloud computing), o ile charakterystyka komunikacji i bezpieczeństwo funkcjonalne na to pozwalają. Dzięki temu przy odpowiednio wydajnej infrastrukturze informatycznej PLC może podlegać wirtualizacji. Jest to jednak technologia znajdująca się obecnie w sferze prac badawczo-rozwojowych, a w rzeczywistych aplikacjach spotykana śladowo. Częściej pojęcie sterownika wirtualnego pojawia się w kontekście symulatora PLC, zdalnego dostępu edukacyjnego [7] lub symulacji środowiska w celach wykrywania zagrożeń dostępu [335]. Do ciekawych prac z tej dziedziny można zaliczyć ideę control-as-a-service [141] lub [142] (zob. 3.2.12). Na rysunku 109 przedstawiono ideę sterowników wirtualnych.



Rys. 109. Idea wirtualnego sterownika
Fig. 109. Idea of virtual PLC

Biorąc pod uwagę model systemu „diabolo” (por. 1.2.1), zakładano, że działanie procesów odbywa się w środowisku fizycznych urządzeń zlokalizowanych w konkretnych miejscach infrastruktury zakładowej. Biorąc pod uwagę współczesny stan technologii wirtualizacji zasobów i przetwarzania w chmurach oraz ich dynamiczny rozwój, można pokusić się o stwierdzenie, że zasoby, w kontekście których wykonywane są zadania procesów, są wirtualne, a rozproszenie przetwarzania dotyczy środowiska chmurowego. Nie budzi to emocji, jeśli mowa jest o zakładowych systemach IT. Jednak to samo można uczynić dla dolnej części modelu, czyli procesów wytwarzania.

Rozbudowa zasobów klasycznych sterowników PLC nie zawsze jest możliwa lub generuje znaczące koszty. Istnieje również wiele zadań w ISP, które nie wymagają obsługi w reżi-

mie Hard-RT. Utrata ważności danych po określonym czasie lub nieokreślona niestałość czasu realizacji zadań są akceptowane z racji tolerancji opóźnień po stronie procesu oraz istnienia ich obsługi po stronie PLC. Dlatego koncepcja Soft PLC zyskuje na popularności. Jej stosowanie powoduje uzyskanie redukcji kosztów przy zadowalających rezultatach względem charakterystyki czasowej przetwarzania w czasie rzeczywistym. Ponadto, wiele z nowych konstrukcji komputerów sterujących bazuje na idei sterownika PC (por. 3.1.5), co pociąga za sobą stosowanie oprogramowania Soft PLC.

3.2.4. Programy serwerowe

Istnieje grupa programów, które mają na celu serwowanie usług na rzecz programów klienckich. Między programem serwera a jego klientami istnieją połączenia logiczne, czyli zestawione są związki aplikacyjne. Do obsługi takich związków wykorzystywane są różnego rodzaju mechanizmy komunikacji międzyprocesowej (ang. IPC – interproces communications), bazujące na lokalnych mechanizmach systemów operacyjnych czy też na dedykowanych, lub ustandaryzowanych mechanizmach dokładanych do węzła. W przypadku fizycznego rozproszenia procesów nieodzownym składnikiem obsługi wymian Klient-Serwer są również sieci komputerowe i dostarczane przez nie usługi komunikacyjne.

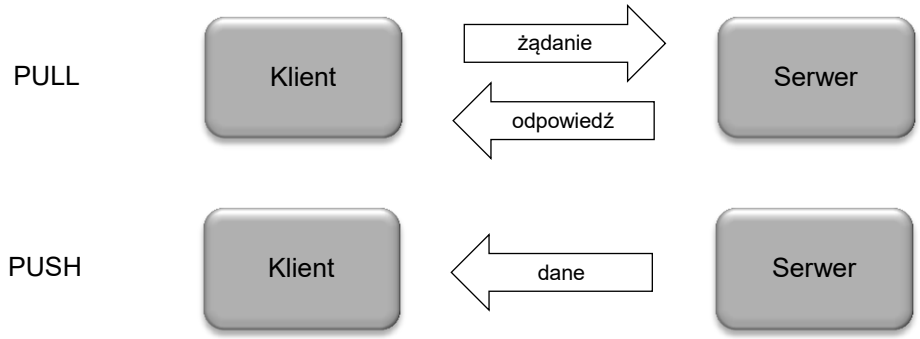
Klientami są aplikacje ISP pracujące zarówno z ograniczeniami czasowymi, jak i bez, a także aplikacje zewnętrzne korzystające z danych obsługiwanych przez serwer. Wszystkie korzystają z usług serwera w celu realizacji własnych funkcjonalności, w tym również zadań integracji, o ile takie występują. W ISP najczęściej wykorzystywane są usługi:

- wymiany danych (dostarczanie danych procesowych, konfiguracji i parametryzacji oraz obsługi danych potrzebnych do integracji),
- składowania danych (obsługa baz danych i logów),
- plikowe (obsługa współdzielonych plików),
- aplikacyjne (dostarczanie specyficznych usług i aplikacji).

Najczęściej wykorzystuje się programy serwerów baz danych, serwerów związanych z usługami typowo sieciowymi (np. DHCP, FTP i pochodne, www, poczta), serwerów informacyjnych (np. raportowe), serwerów terminalowych (np. zdalny dostęp) oraz serwerów danych procesowych. Wykorzystanie baz danych (zob. 3.2.5) i serwerów usług sieciowych nie odbiega od typowych rozwiązań znanych z innych dziedzin. Jedyna zauważalna specyficzność dotyczy wbudowanych mechanizmów integracji tych serwerów z podsystemami programowymi ISP – najczęściej oprogramowania nadzorczego (np. SCADA, MES itp.). Integracja dokonywana jest na poziomie współdziałania usług podsystemów, np. automatycznej generacji stron www na podstawie ekranów wizualizacji, plików raportów na podstawie stanu monitoringu, wspomagania tworzenia struktury bazy

oraz zapytań na podstawie struktury i typów danych aplikacji lokalnych, scenariuszy wymian na bazie dostępnych zmiennych itp.

Najistotniejszym zadaniem serwera jest realizacja dostarczania „czegoś" w odpowiedzi na żądanie. Żądania są generowane przez programy klienckie (model pull) cyklicznie lub zdarzeniowo. Gdy dostarczanie jest realizowane przez serwer automatycznie (model push), wówczas serwer nie czeka na żądanie od klienta, a wysyła dane zgodnie ze zdefiniowanym wyzwalaczem, np. cyklicznie lub zdarzeniowo. Jest to zilustrowane na rysunku 110.



Rys. 110. Modele wymian pull i push przy komunikacji klient-serwer
Fig. 110. Pull and Push models of exchanges in client-server communication

Niezależnie od modelu realizacji dostarczania programy klienckie traktują serwery jako pewne źródło informacji odznaczające się takimi cechami (por. 2.3.2), jak:

- niezawodność,
- trwałość,
- spójność informacyjna,
- przewidywalność.

Dlatego bardzo istotne jest, aby oprogramowanie serwera było przystosowane do realizacji niezaburzonego procesu dostarczania oraz aby dane niezbędne dla tego procesu zarówno bieżące, jak i historyczne były zabezpieczane przed uszkodzeniem i utratą. Większość programów serwerowych zarówno po stronie aplikacyjnej, jak i systemów operacyjnych daje wsparcie dla wykonywania i przywracania kopii zapasowych danych, ich struktur oraz ustawień. Mechanizmy te są dostępne, o ile platforma sprzętowa pozwala na ich wykorzystanie (por. 3.1.14).

Interfejsy dostępu do usług serwerów są z reguły standardowe i zachowują cechę przezroczystości (por. 2.3.2). Umożliwiają współpracę komponentów programowych od różnych producentów. Komercyjne tworzenie serwerów z zamkniętymi interfejsami jest bardzo dobrym przepisem na klęskę produktu.

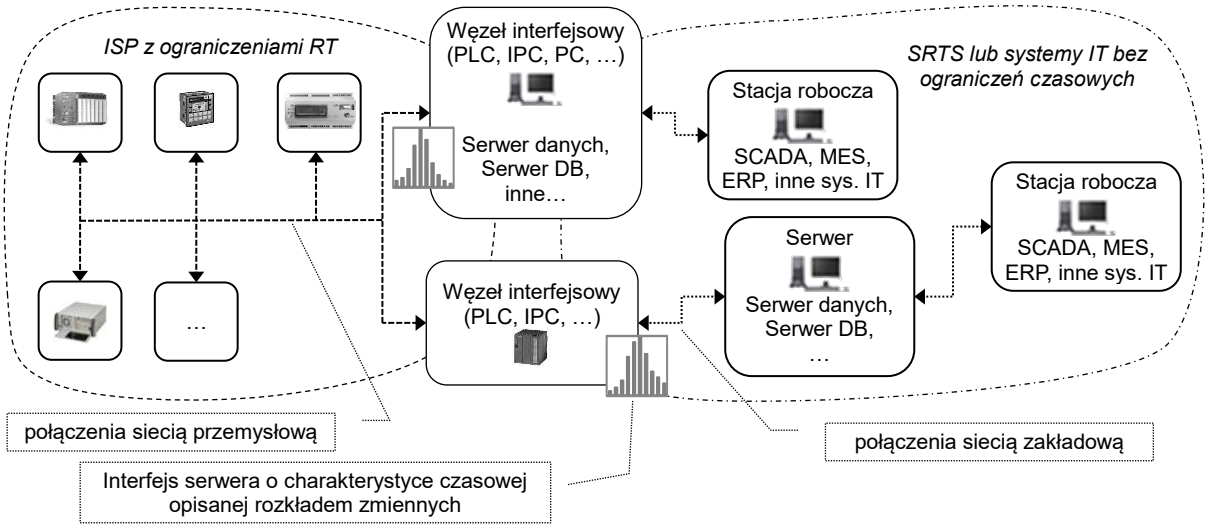
Programy serwerowe uruchamia się na platformach serwerowych (por. 3.1.14) znajdujących się zwykle poza obszarem działania z ograniczeniami czasowymi, a dostęp realizuje

przez węzły interfejsowe. Oprogramowanie serwerowe pracuje zatem przeważnie bez mechanizmów zapewniających realizacje zadań w określonym czasie, czyli cecha punktualności nie jest zachowywana (por. 2.3.2). Są to rozwiązania klasy SRT. Nie oznacza to jednak, że czas nie jest tu ważny. Czasami programy serwerowe pozostają w bezpośredniej interakcji z węzłami ISP pracującymi w trybie RT. Wówczas dla uproszczenia struktury systemu i zmniejszenia liczby elementów pośredniczących w przekazywaniu danych serwery uruchamiane są bezpośrednio na węzłach integracyjnych lub nawet na węzłach procesowych. Ma to miejsce np. przy wykorzystaniu serwerów danych. W ISP separacja funkcjonalności względem czasu jest jednak dość wyrazista, a szczególnie podział na obszary RT i pozostałe. Zatem ze względów na ten podział i wynikających z niego kwestii bezpieczeństwa funkcjonalnego zaleca się uruchamianie programów serwerowych na węzłach lokalnych ISP tylko wówczas, gdy istnieją bezpośrednie interakcje serwera z innymi węzłami lokalnymi ISP. Jeśli aktywności związane z komunikacją typu Klient-Serwer stanowią podsystem non-RT lub SRT, to musi być wówczas dokonana analiza współzależności czasowych oparta na analizie procesów stochastycznych (por. 1.1.4).

Responsywność serwera względem aplikacji ISP ma znaczenie o tyle, o ile aplikacja może czekać na uzyskanie odpowiedzi. Dla przykładu, podczas produkcji dane o parametrach produktów mogą być dostarczane przed rozpoczęciem wytwarzania partii produkcyjnej. Nie jest istotne, czy oczekiwanie na dane będzie trwało 500 ms czy 5 sekund, ale dłuższe czasy oczekiwania bądź brak komunikacji może doprowadzić do nieplanowanych przestojów produkcyjnych. Dlatego podczas projektowania interakcji z serwerami należy rozważać opóźnienia wynikające z komunikacji oraz z przetwarzania. Z racji braku działań zdeterminowanych w czasie, rozważań takich nie można oprzeć na analizie najgorszego przypadku. Można jednak wykonać analizę bazującą na badaniach czasu odpowiedzi i klasyfikującą dane rozwiązanie względem dopasowania zmierzonych bądź wyliczonych rozkładów czasów odpowiedzi układu (por. 1.1.4, rys. 4) do wymagań aplikacji. Przykład metody można znaleźć w [381]. Na rysunku 111 zilustrowanio przykładowe połączenia oraz wyróżniono interfejsy serwerów odznaczające się brakiem cechy punktualności, a mogących być opisanych rozkładem prawdopodobieństwa wybranych zmiennych istotnych dla współdziałania serwera z ISP.

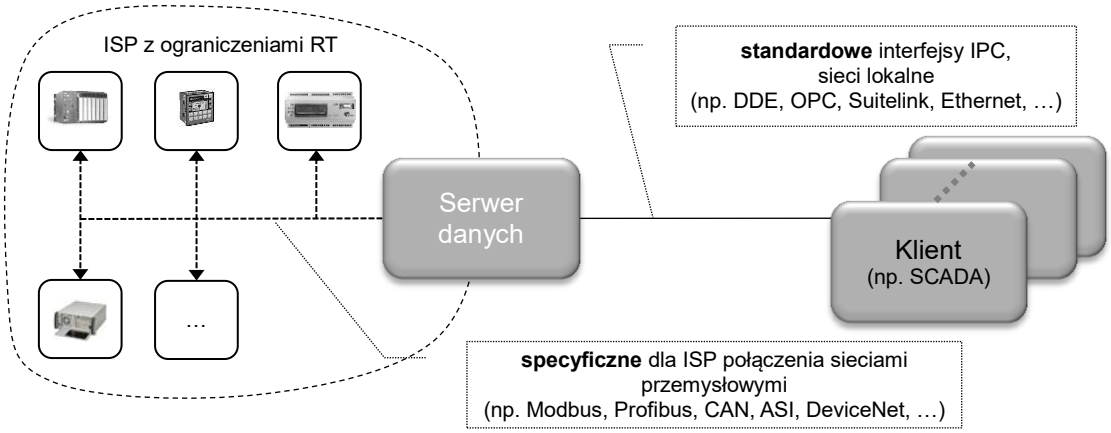
W przeciwieństwie do szerokiej gamy różnorakich serwerów ogólnego przeznaczenia dość specyficzną grupę stanowią serwery danych procesowych (DS, PDS – ang. Process Data Server). Są to rozwiązania dedykowane dla ISP i praktycznie tylko w tego typu systemach są stosowane. Zadaniem serwera danych jest dystrybucja danych pomiędzy węzłami ISP a ustandaryzowanym oprogramowaniem klienckim przeznaczonym do obróbki tych danych. Dobrą analogią mogą być systemy DMP (ang. Data Management Platform) dedykowane głównie do zarzadzania przepływem danych z wielu źródeł na potrzeby działania systemów

handlowych, przy czym DMP dla ISP zarządza wymianą danych pomiędzy elementami tego systemu, jego podsystemami lub systemami zewnętrznymi.



Rys. 111. Ilustracja współdziałania serwerów z aplikacją ISP
Fig. 111. Illustration of servers and ICS application cooperation

Dane są opisane strukturą i typem zmiennych systemowych, a ich wartości stanowią głównie dane procesowe. Oprogramowanie serwera służy jako pośrednik w wymianie i musi dbać o pozyskiwanie i dostarczanie bieżących wartości oraz prowadzić diagnostykę i statystykę połączeń. Serwer danych nie musi pracować w czasie rzeczywistym klasy HRT, gdyż przeważnie służy do skomunikowania z podsystemami pracującymi w trybie SRT lub FRT. Powinien jednak spełniać wymogi podsystemów FRT. Nadmierne opóźnienia przy przekazywaniu danych mogą skutkować utratą przydatności, a utrata danych lub utrata informacji o kolejności zdarzeń jest niedopuszczalna. Serwery danych często oferują ustandaryzowane interfejsy komunikacji międzyprocesowej, zapewniając tym samym możliwość dołączania szerokiej gamy klientów. Natomiast od strony procesu oferują moduły komunikacji specyficzne dla środków komunikacyjnych dedykowanych dla ISP. Często interfejsy w tym zakresie są uniwersalne, umożliwiając dołączanie odpowiednich modułów obsługujących rozwiązania standardowe lub wręcz dołączania modułów specyficznych tworzonych na potrzeby konkretnej aplikacji, bądź podsystemu. Najczęściej serwery danych wykorzystuje się do skomunikowania aplikacji kontrolerów z oprogramowaniem nadzorczym na stacjach PC (rys. 112).



Rys. 112. Ogólna idea działania serwera danych
Fig. 112. General idea of data server operation

Typowym przykładem serwera danych, z racji dużej popularności, są serwery OPC [176]. Geneza skrótu pochodzi z początku wieku, gdzie technika OPC bazowała tylko na technikach wymiany danych OLE, COM i DCOM firmy Microsoft, stąd OPC – OLE for Process Control. Obecnie rozwiązania te są znane jako OPC Classic. Kontynuacją rozwoju OPC na platformie technologii firmy Microsoft jest OPC .NET znana też jako OPC EI lub OPC XI (OPC Express Interface). Bazuje ona na technologiach .NET Framework oraz WCF (Windows Communication Foundation). Z biegiem czasu technika OPC ewoluowała również w rozwiązania niezależne od technologii firmy Microsoft i aktualnie rozwijana jest jako OPC Unified Architecture [254]. OPC UA bazuje na otwartym, zorientowanym obiektowo i niezależnych od producenta modelach usług, danych i ich wymiany [161]. Przykładami niezależnych serwerów OPC mogą być TOPServer lub MatrikonOPC, przy czym rynek tych produktów jest bardzo duży, a większość podsystemów nadzorczych oferuje swoje natywne rozwiązania. Istnieje sporo źródeł traktujących o użyciu technik OPC oraz ich rozwoju [60], [255], [238], [339], [300]. Technika OPC UA sprzyja tworzeniu otwartych i elastycznych rozwiązań systemowych [140].

3.2.5. Bazy danych

(DB – ang. Database)

Działanie ISP często wiąże się z koniecznością składowania bieżących i historycznych danych opisujących przebieg procesu w czasie, danych parametryzujących przebieg procesu lub innych danych, które są współdzielone pomiędzy rozproszonymi aplikacjami systemu lub między różnymi systemami i podsystemami.

Współczesne serwery baz danych to programy do bezpiecznego gromadzenia i przetwarzania danych wraz z interfejsem umożliwiającym uzyskanie dostępu do tych danych wraz z funkcjonalnościami ich przetwarzania oraz narzędziami do lokalnego i zdalnego

zarządzania (ang. DBMS – database management system). Obecne serwery baz danych oferują cały szereg funkcji związanych z zapisem, przechowaniem, odczytem i przetwarzaniem danych [M76]. Funkcjonalności ISP korzystają tylko z ich części, takich jak:

- zapis danych do bazy (bezpieczne zachowanie danych pochodzących od klienta w odpowiednim repozytorium),
- odszukanie danych w odpowiedzi na zapytania (kwerendy, ang. data query),
- wykonanie przetwarzania na serwerze (np. przez procedury składowane, funkcje, ang. stored procedure, function),
- odczyt danych (dostarczenie do klienta wybranych rekordów stanowiących wynik odszukania),
- diagnostyka połączeń i operacji.

Aby użycie powyższych funkcji serwera było możliwe, konieczne jest również niejawne skorzystanie z mechanizmów bezpieczeństwa, transakcyjności, indeksacji, optymalizacji czy też z innych specyficznych operacji kontrolujących dostęp i strukturę danych. Dlatego z punktu widzenia systemu operacyjnego działanie procesów obsługujących bazy danych jest dość złożone. Wchodzi w to przetwarzanie danych zarówno w warstwie OS, jak i użytkownika, w tym operacje dyskowe, pamięciowe, współpraca międzyprocesowa i międzywątkowa oraz komunikacja sieciowa.

Współczesne serwery baz danych nie działają z ostrymi lub mocnymi (HRTS, FRTS, por. 1.1.4) ograniczeniami czasu rzeczywistego. Platforma systemu operacyjnego, na którym jest uruchomiony serwer, również nie jest z reguły klasy RTOS. Ponadto podłączenia serwerów DB do ISP wykonywane są najczęściej na bazie zakładowych sieci lokalnych, które oparte na klasycznym Ethernecie, nie gwarantują zdeterminowanej w czasie komunikacji. Dlatego nie należy traktować bazy danych jako źródła danych, które może dostarczyć żądane informacje w określonym czasie. Podobnie z zapisem do bazy. Czas pomiędzy zleceniem zapisu a zakończeniem transakcji, mimo że z reguły krótszy niż wykonanie zapytania i odczytu danych, jest niedeterministyczny. Deterministyczna jest natomiast kolejność wykonywanych operacji i ich spójność, co wraz z mechanizmami buforującymi w większości przypadków wystarcza na potrzeby użycia DB w ISP.

Serwer może pracować lokalnie przy źródle danych (np. przy serwerze OPC) lub zdalnie na innej maszynie i w innej sieci. Praca lokalna, przy procesie, nie jest dobrym pomysłem. Ze wspomnianych względów ograniczeń czasowych warstwa bazy danych nie jest elementem procesu sterowania, zatem powinna być od niego oddzielona. Ponadto, serwery DB zajmują dość dużo zasobów komputera, na którym pracują, co przy węzłach ISP może nie być wskazane. Osadzenie serwera DB poza węzłami ISP upraszcza również jego administrację, zwięk-

sza bezpieczeństwo i umożliwia łatwiejsze współdzielenie dostępu i integracje danych. Dobrym rozwiązaniem może być umieszczenie serwera DB na maszynie wirtualnej [130].

Mimo braku wsparcia dla oczekiwanych w ISP ograniczeń czasowych istnieją proste jak i mniej proste mechanizmy, umożliwiające skonfigurowanie serwera w sposób zadowalający względem uzyskiwanych opóźnień. Mowa tutaj o stosowaniu dobrych zasad projektowania baz danych oraz odpowiednich dla danej bazy środków w celu uzyskania najlepszej możliwej charakterystyki czasowej współpracy serwera z ISP. Jest zatem wysoce wskazane, aby inżynier zajmujący się ISP posiadał wiedzę z zakresu baz danych. Przydatne zagadnienia dotyczą doboru odpowiednich rodzajów organizacji baz do typów zadań (bazy relacyjne, nierelacyjne, obiektowe, temporalne, strumieniowe, dyskowe, pamięciowe itp.), doboru baz do rodzaju obsługiwanych danych (typy danych, model danych, struktura powiązań, charakterystyka wyszukiwań itp.), konfiguracji serwera (struktura danych/obiektów, kopie bezpieczeństwa (ang. backup), wdrożenie polityki bezpieczeństwa, określenie zadań lokalnych, optymalizacje itp.) oraz umiejętności tworzenia zapytań (wybór i użycie konstrukcji językowych dających najlepsze rezultaty w danym przypadku).

Warto zwrócić uwagę, że mimo wielu informacji marketingowych traktujących o przemysłowych bazach danych, tak naprawdę ich „przemysłowość" sprowadza się do zastosowań, a nie do konstrukcji. Wsparcie, jakie się pojawia, dotyczy ułatwień integracji z innymi podsystemami ISP, obsługi stempli czasowych, gromadzenia i buforowania (w pamięciach podręcznych, kolektory, ang. cache, collectors) i optymalizacji wydajnościowej silników, czyli względem odczytu i zapisu z minimalnym możliwym opóźnieniem[70]. Generalnie, do współpracy z ISP można wykorzystać dowolną bazę, choć gdy istotne jest gromadzenie dużej liczby szybko zmieniających się danych bądź zapytania kierowane do bazy bazują na ciągłym a nie dyskretnym charakterze danych, to bazy orientowane na systemy przemysłowe są wysoce wskazane. Wraz z rozwiązaniami serwerowymi baz danych często spotyka się wbudowane platformy towarzyszące, wspierające usługi dodatkowe umożliwiające łatwe integracje podsystemów oraz generacje raportów produkcyjnych. Jako przykłady popularnych baz „przemysłowych" mogą służyć IndustrialSQL Server skonstruowany na bazie Microsoft SQL Server oraz Proficy Historian firmy Wonderware [391], [M96].

Najczęściej używanymi bazami danych w kontekście ISP są bazy o architekturze relacyjnej i działające na bazie języka SQL [326], [305]. Bazy temporalne czy przestrzenno-temporalne, które z założenia obsługują stemple czasowe zapisu i czas ważności zapisanych danych, są specyficzną odmianą baz relacyjnych. Nie są jednak bazami danych czasu rzeczywistego w rozumieniu zdeterminowania czasowego obsługi danych. Bazy pamięciowe

[70] Na przykład baza Historian w pakiecie Wonderware 2014 R2 może logować do 1000 parametrów na sekundę [M85].

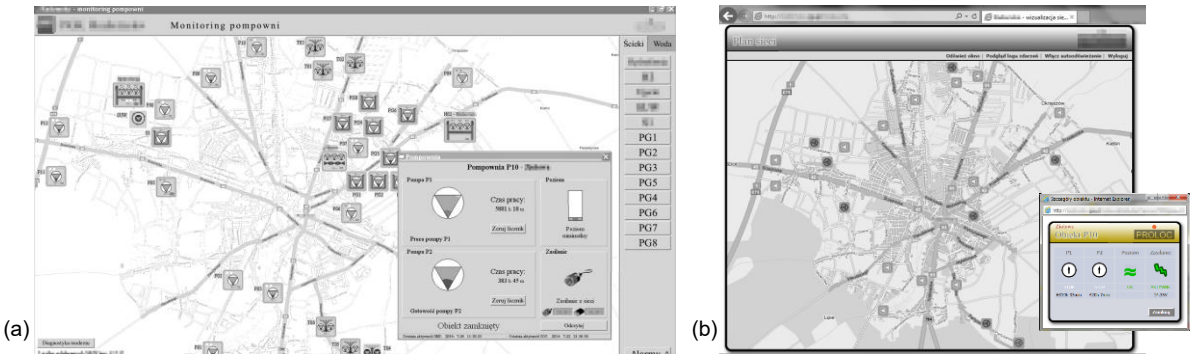
(ang. in-memory) oferują dużo lepszą wydajność niż dyskowe, ale są droższe we wdrożeniu i utrzymaniu. Dlatego nie znalazły zbyt dużego uznania w ISP. Niestety, zakres niniejszej książki nie obejmuje rozwinięcia teorii i praktyki baz danych. Więcej można znaleźć np. w [251].

Przykładem zastosowania bazy danych w ISP może być podsystem zbierający w czasie rzeczywistym dane produkcyjne z przebiegu procesu produkcyjnego prowadzonego na wielu liniach i zapisujący je w lokalnych i centralnej bazie danych, przedstawiony w dodatku 7.1.3.

3.2.6. Programy monitorujące

Przeważnie w ISP wykorzystywany jest przynajmniej jeden program przeznaczony do interakcji z człowiekiem. Mogą to być programy stanowiące interfejs interaktywny jak programy SCADA lub HMI, omówione w następnych rozdziałach, albo programy tylko obserwujące proces. Do tych drugich należą programy monitorujące. Służą one do śledzenia wybranych parametrów pracy obiektu lub systemu. Dodatkowo dokonują ich logowania i/lub informowania użytkownika o stanach bieżących, niepoprawnych czy też awaryjnych. Najczęściej spotykany sposób ich działania to cykliczny odczyt parametru, analiza wartości, zapis do bazy i wyświetlenie jego wartości na ekranie komputera. W razie detekcji nieprawidłowości następuje sygnalizacja z użyciem atrybutów grafiki (np. kolor, miganie itp.) i/lub dźwięku. Czasem do sygnalizacji stosuje się pocztę elektroniczną lub usługi sieci telekomunikacyjnych (np. SMS, komunikaty głosowe).

Programy monitorujące są najczęściej dedykowane dla danego procesu. Przykładem może być program zbierający dane z rozproszonych przepompowni i informujący o ich stanie użytkowników zarówno w dyspozytorni, przez www, jak i na urządzeniach mobilnych (przykład na rysunku 113 odpowiednio (a) i (b)).



Rys. 113. Przykład monitoringu w formie programu dedykowanego (a) i serwisu www (b)
Fig. 113. Example of monitoring in the form of a dedicated program (a) and webpage (b)

3.2.7. Programy SCADA

(SCADA – ang. Supervisory Control and Data Acquisition)

Programy klasy SCADA są przeznaczone do zbierania danych z przebiegu działania procesu i sprawowania nad nim nadrzędnej kontroli przez człowieka. Stanowią najczęściej aplikowane oprogramowanie nadzorcze. Dostarczają zadań i funkcji umożliwiających użytkownikowi sprawowanie nadzoru nad systemem przez:

- wizualizację przebiegu procesu,
- manualne oddziaływanie na elementy systemu,
- parametryzację działania węzłów,
- parametryzację przebiegu procesu,
- monitorowanie parametrów procesu,
- alarmowanie o stanach nieprawidłowych,
- przetwarzanie lokalne danych,
- zbieranie, zapis i analizę danych procesowych,
- integrację z innym oprogramowaniem na poziomie nadzorczym.

SCADA integruje zatem funkcje programów monitorujących z innymi, związanymi z zapewnieniem interakcji z użytkownikiem. Popularnie, programy SCADA nazywa się często systemami lub programami do wizualizacji lub wizualizacyjnymi. Jednak wymienione powyżej zadania wskazują, że zakres funkcjonalny działania tego typu oprogramowania jest dużo szerszy niż tylko wizualizacja przemysłowa. Ma to dobre strony, dając inżynierowi jeden produkt wyczerpujący potrzeby użytkownika, ale ma również i złe strony, gdy integrator zbyt entuzjastycznie skorzysta z dostępnych możliwości. Często nadużywaną funkcjonalnością jest przetwarzanie lokalne. W systemach SCADA realizowane jest ono z użyciem skryptów (programy użytkownika ang. user program) uruchamianych cyklicznie lub zdarzeniowo. Jeśli charakterystyka czasowa przetwarzania zawartego w skryptach ma wykazywać cechę zdeterminowania czasowego (np. programy sterujące), to ich użycie jest dobrą drogą do problemów. Podsystemy tego typu nie powinny realizować zadań sterowania ze względu na ograniczenia konstrukcyjne komputerów i systemów operacyjnych, na których są uruchomione. OS typu non-RT nie jest w stanie zarządzać procesami oprogramowania SCADA w reżimie mocnych lub twardych ograniczeń czasowych. Ponadto, konstrukcja wewnętrzna takich systemów jest z reguły niedostosowana do szeregowania lokalnych zadań przetwarzania z ograniczeniami czasowymi. Systemy SCADA należy traktować jako element ISP, ale stanowiący interfejs dla użytkownika, czyli działający jako podsystem SRTS. Wbudowanie w niego funkcji wymagających reżimu HRTS może skutkować

nieprzewidywalnymi opóźnieniami, szczególnie w sytuacji przeciążenia systemu hosta (brak zasobów, lawina zdarzeń, ang. overloading).

Istnieje bardzo szeroki rynek systemów SCADA, zarówno znanych produktów pochodzących od producentów ogólnoświatowych, popularnych produktów lokalnych, jak i produktów niszowych aplikowanych z sukcesem, ale bez zaplecza marketingowego. Dla przykładu, popularne systemy na polskim rynku to np.: InTouch, Proficy PE, Indusoft, WinCC, ControlMaestro (WizCon), Intellution iFIX, Asix, Kronos, Zenon i wiele innych.

Konstrukcja współczesnych systemów SCADA jest zróżnicowana. Dominują rozwiązania uniwersalne, znajdujące zastosowanie praktycznie w każdej branży przemysłu. Najbardziej zaawansowane z nich stanowią elementy większych, kompleksowych pakietów, na które składają się programy serwerowe, bazodanowe i integracyjne. Zwykle działają na bazie modelu centralnego zarządzania danymi (por. 1.2.2, rys. 17). Parametry jakościowe takich systemów mogą być rozpatrywane w kontekście charakterystyk czasowych lub wygody użycia (ang. usability). Dostosowanie względem czasu sprowadza się, podobnie jak w bazach danych, do zapewnienia minimalnych opóźnień w wykonywaniu zadań. Natomiast wygoda użycia sprowadza się do tworzenia mechanizmów dostosowujących funkcjonalności interfejsu do potrzeb prezentacji przemysłowych systemów automatyki i klasycznych interakcji. Są to różnego rodzaju biblioteki obiektów graficznych i symboli wraz z mechanizmami ich tworzenia, współdzielenia, kontroli i nawigacji. Ponadto, są to narzędzia zapewniające tworzenie zmiennych i adresów, generacje odpowiednich widoków interfejsów, ich konwertery, a także narzędzia zarządzające aplikacjami w trybie off-line jak i on-line.

Ze względu na dużą uniwersalność dostępnych produktów i mnogość oferowanych licencji systemy SCADA wykorzystywane są zarówno do obsługi dużych i złożonych systemów, jak również do tworzenia lokalnych interfejsów HMI (por. 3.2.8). Z reguły o możliwościach aplikacyjnych danego produktu decyduje licencja określająca dostępną przestrzeń zmiennych IO, czyli takich, które służą do skomunikowania ze źródłami danych. Czasami, gdy przy rozbudowie systemu istniejące możliwości komunikacyjne okazują się niewystarczające, ograniczenia wynikające z limitu zmiennych IO można ominąć z użyciem dynamicznej zmiany ich powiązań, podobnie jak dla problemu braku wystarczającej pojemności komunikacyjnej sieci polowych wspomnianych w [23].

Oprogramowanie SCADA wspomaga pracę utrzymania ruchu. W praktyce, czasami kompetencje nadzorcze są również przekazywane dla obsługi procesu. Przykładem zastosowania oprogramowania SCADA jako HMI może być aplikacja w systemie przedstawionym w dodatku 7.1.2 lub 7.1.3.

3.2.8. Programy MMI/HMI

(MMI – ang. Man-Machine Interface, HMI – ang. Human Machine Interface)

Koncepcja interfejsu maszyny z człowiekiem została opatentowana w 1986 roku jako MMI. Ze względu na sukcesywny rozwój i obniżanie kosztów technologii interfejsów GUI (ang. Graphic User Interface) pomysł zyskiwał na popularności i z biegiem czasu przerodził się w koncepcję HMI.

HMI jest to ogólna kategoria programów umożliwiających tworzenie i wykorzystywanie lokalnych interfejsów pomiędzy człowiekiem a maszyną (procesem, podprocesem). W praktyce często oprogramowanie HMI stanowi programy dla paneli HMI (por. 3.1.10). Zadania funkcjonalne w zakresie budowy i działania interfejsu są tu dowolne. Jednak w zakresie uruchamiania zadań sterowania, wymagających z reguły co najmniej mocnych ograniczeń czasowych, należy brać pod uwagę, tak jak w rozdziale powyżej, cechy urządzenia, jego systemu operacyjnego i oprogramowania HMI. Jak wspomniano w 3.1.3, istnieją sterowniki zintegrowane z panelami graficznymi, tym samym umożliwiając uruchomienie oprogramowania HMI wraz z programami sterującymi. Integracja interfejsu z pamięcią sterownika i jego funkcjami dokonuje się wówczas na poziomie systemu operacyjnego urządzenia, umożliwiając realizację zarówno zadań z twardymi ograniczeniami czasowymi, jak i z miękkimi.

Z założenia oprogramowanie HMI stanowi interfejs lokalny przy maszynie. Dlatego najczęściej programy HMI są kojarzone jako relatywnie proste interfejsy prezentujące lokalne parametry pracy maszyny, procesu lub jego fragmentu. Jednak czasami spotyka się uogólnienia i pod nazwą programu klasy HMI klasyfikuje się dowolny program, który zapewnia interfejs z człowiekiem, niezależnie czy pracuje lokalnie, blisko urządzeń jako panel, czy też zdalnie jako inny komputer.

Oprogramowanie HMI wspomaga pracę operatorów i utrzymania ruchu. Podstawowym zadaniem takiego oprogramowania jest udostepnienie użytkownikowi zrozumiałego interfejsu do parametryzacji i obserwacji pracy. Warto jednak wspomnieć, że współczesne możliwości integracyjne dotyczące systemów automatyki, ISP oraz systemów IT zakładu, i co za tym idzie ich wzajemna współpraca, umożliwiają utworzenie narzędzi aktywnie wspomagających użytkownika w obsłudze i diagnostyce procesu. Ma to szczególne znaczenie przy stosowaniu produkcji oszczędnej (szczupłej, ang. Lean Manufacturing), gdzie część kompetencji zarządzania produkcją przekazywana jest właśnie do utrzymania ruchu, liderów procesu czy nawet operatorów. Dzięki temu systemy HMI z prostych interfejsów stają się systemami ekspertowymi, mogącymi aktywnie wspomagać prowadzenie procesu. Czasem można spotkać określenie M4H – Machine for Human. Przykładowe zastosowanie prostego HMI pokazano w dodatku 7.1.4, natomiast rozbudowanego w 7.1.3.

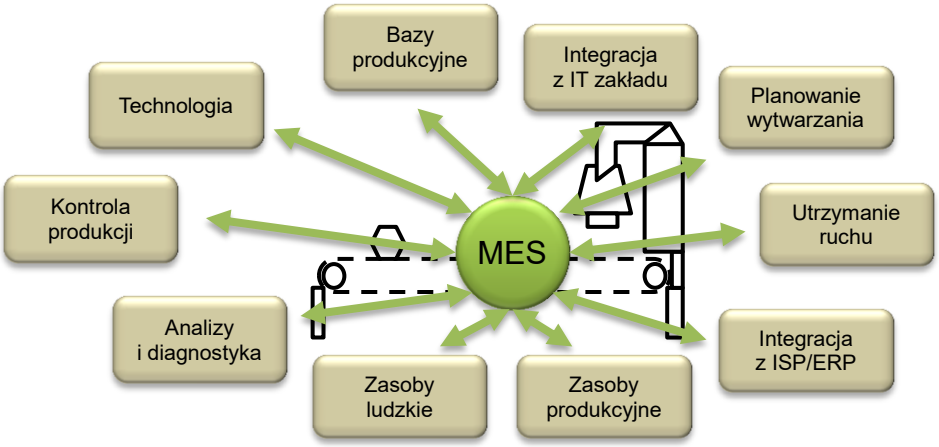
3.2.9. Programy MES

(MES – ang. Manufacturing Execution Systems)

Rozwiązania klasy MES stanowią program lub zestaw programów o charakterze systemowym służących do wspomagania zarządzania konkretną produkcją. Ich funkcjonalności nie są skupione na funkcjach sterowania czy regulacji procesem, lecz na zarządzaniu i utrzymaniu produkcji oraz jej analizach jakościowo-wydajnościowych. Zatem, działanie MES dotyczy całych jednostek produkcyjnych, a nie tylko danego procesu. Zapewniana jest obsługa zarówno od strony niezbędnych i bezpośrednio powiązanych procesów biznesowych, jak i od strony parametryzacji technologicznej produkcji i jej kontroli. Systemy MES umożliwiają m.in.:

- Śledzenie i rejestrację danych na temat bieżącej produkcji (tzw. produkcji w toku). Nie chodzi tu o dane procesowe tylko produkcyjne, odnoszące się do jednostek i procesów produkcyjnych.
- Analizę danych charakteryzujących produkcję w określonym czasie. Dotyczy danych statystycznych produkcji i jej jakości.
- Zarządzanie zasobami ludzkimi i produkcyjnymi oraz ich alokacją. Określanie co, kiedy i przez kogo ma być wytwarzane.
- Integracja produkcji z zarządzaniem. Połączenie poziomu sterowania i poziomu nadzorczego z systemami wyższego poziomu, np. z systemami wspomagania zarzadzania klasy ERP.

Na rysunku 114 przedstawiono schematycznie podstawowe funkcjonalności MES i symboliczną linię technologiczną jako dziedzinę ich oddziaływania.



Rys. 114. Ilustracja funkcjonalności MES
Fig. 114. Illustration of MES functionality

Programy MES są oprogramowaniem wspomagającym pracę głównie kierowników działów produkcyjnych (ang. production manager, production supervisor). Pozwalają na parametryzacje jednostek produkcyjnych (zob. 1.1.1) oraz śledzenie ich działań o jeden poziom abstrakcji wyżej niż systemy HMI czy SCADA, dając wgląd nie tylko w realizację zadań na poziomie procesu, ale przede wszystkim w parametry jakościowe i wydajnościowe ich przebiegu. Dzięki temu kierownicy mogą śledzić raporty dotyczące bieżącej charakterystyki produkcji, jak i je analizować za wybrany okres czasu.

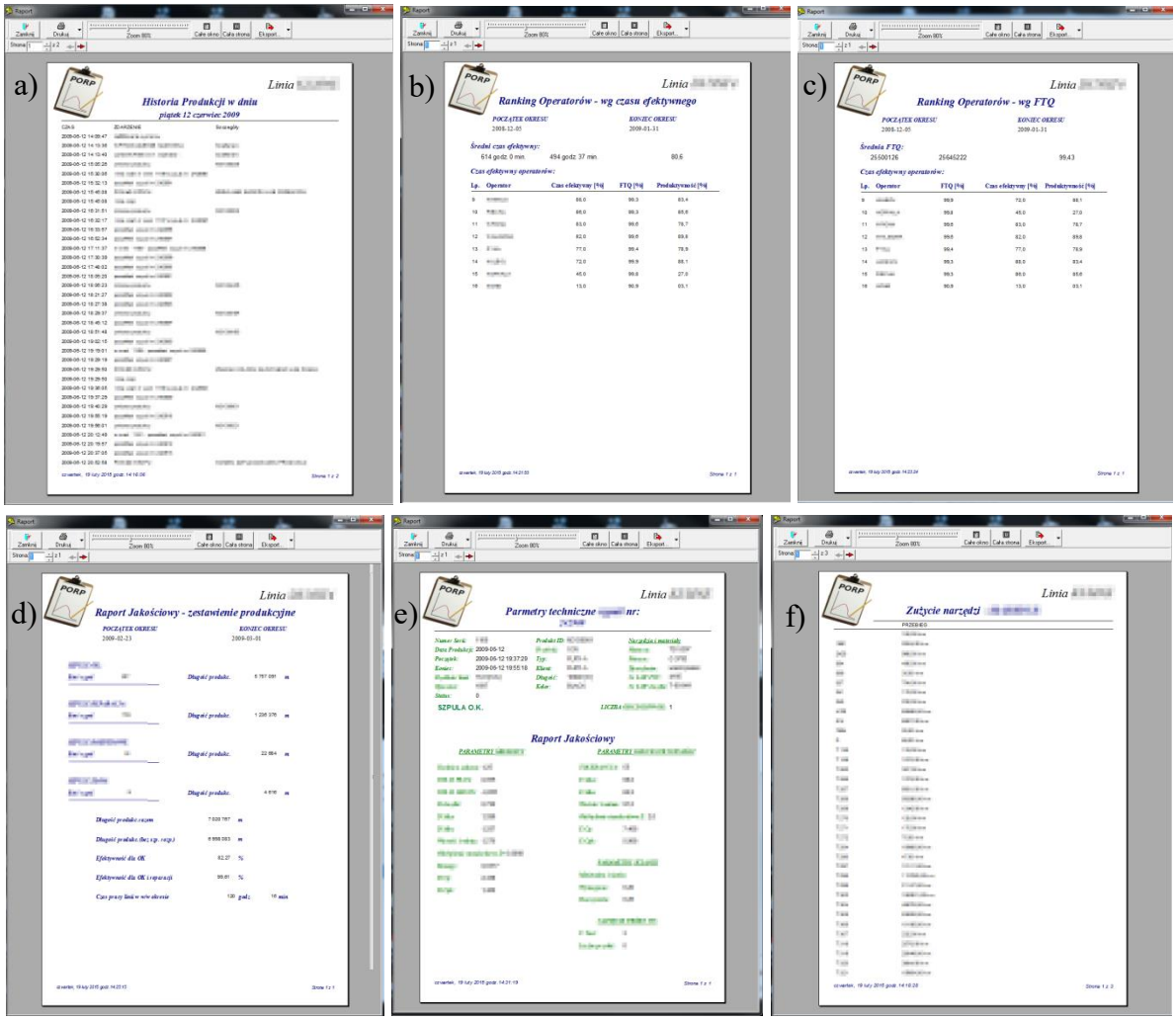
Zatem, MES dają możliwości śledzenia historii produkcji (tzw. genealogii lub śladu produkcji, ang. tracebility). Dystrybuując zlecenia produkcyjne i receptury, rejestrując przepływ materiałów (surowców, półproduktów i produktów), wartości charakterystyczne z przebiegu produkcji, zdarzenia planowane i nieplanowane, wszelakie aktywności i błędy, w tym działania i błędy operatorów, użytkownik może prześledzić dowolną produkcję z dokładnością do rozdzielczości czasowej i znaczeniowej zebranych danych. Do wystawienia atestu produkcyjnego danego produktu wystarczy prześledzić poprawność ciągu rejestrowanych danych względem wymagań ze zlecenia. Natomiast, w przypadku wykrycia problemu po fakcie, na przykład w wyniku reklamacji, użytkownik jest w stanie przeanalizować wstecznie ścieżkę produkcyjną danego wyrobu i odszukać przyczynę problemu oraz określić niejawne implikacje wystąpienia tego problemu.

Systemy MES sprawują również kontrolę nad przepływem dokumentów związanych ze zleceniami produkcji, parametrami produkcyjnymi (nastawami, recepturami, programami działań, korektami itp.), przekazywaniem jednostek produkcyjnych między okresami organizacji pracy (dni, zmiany, serie itp.) oraz z harmonogramowaniem ich pracy. Raporty produkcyjne generowane na podstawie zarejestrowanych w bazie danych mogą być udostępniane operatorom w postaci klasycznego wydruku, etykiety lub innej formy jako dokument atestacji zakończonego procesu. Raporty mogą dotyczyć przeróżnych aspektów kontroli produkcji od zdarzeń (np. detekcji błędów, postojów, niezgodności itp.), przez raporty jakościowo-wydajnościowe (np. FTQ, PPM, DPU itp. zob. 1.1.2), po analizy statystyczne (np. pareto itp.).

Na rysunku 115 przedstawiono przykładowe raporty: produkcyjnego (a), parametrów jakościowych (b, c, d), technicznych (e, f). Przykładowy diagram „pareto” przedstawiono na rysunku 116.

Ponadto MES dają możliwości analizy globalnych parametrów jakościowych jak różnego rodzaju wskaźniki związane z oszczędnym gospodarowaniem (tzw. szczupła produkcja, z ang. Lean Manufacturing) (zob. 1.1.2) czy też z innymi metodami zarządzania jakością, jak metoda „sześć sigma” (ang. Six Sigma) [W37], metoda „5 Why” itp. Stanowią one narzędzie do analizy limitów JIT – just in time, danych dla procesów Kaizen i Kanban [91] i innych

wskaźników określających produkcję w wymiarze ekonomicznym. Niektóre systemy dają również możliwości analizy dynamicznej przez definiowanie i śledzenie, np. wąskich gardeł procesu lub generacji niestandardowych, wynikających z potrzeby chwili, raportów.

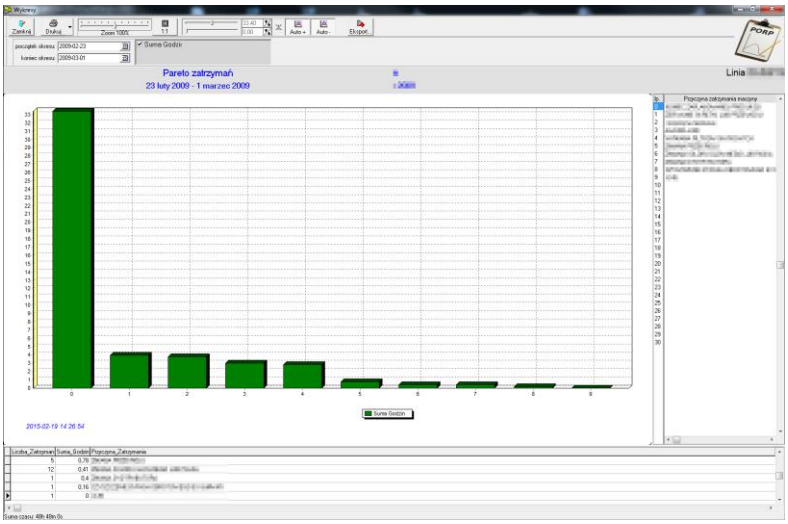


Rys. 115. Ilustracja przykładowych raportów
Fig. 115. Illustration of sample reports

Systemy MES mogą dostarczać wielu innych funkcji, które w teorii stanowią często funkcje innych podsystemów IT wymienionych w rozdziale 1.2.1 i opisanych w 3.2.11. Wynika to z faktu, że sztywne przypisania i podziały nie sprawdzają się w środowisku przemysłowym. Jak zostało wspomniane w rozdziale 1.2.1, przy projektowaniu systemów zarządzania sensowniej jest podejść do integracji wymaganych funkcjonalności w ramach tworzenia danego systemu, niż tworzyć i integrować systemy o określonych funkcjonalnościach.

Podobnie podział funkcji między SCADA a MES nie jest w praktyce bardzo ostry. Programy MES mogą wspomagać również pracę operatorów procesu. Dotyczy to zwykle parametryzacji linii w zakresie niezwiązanym z ingerencją w parametry technologiczne, bizneso-

we lub z odczytem informacji niejawnych. Jednak podobnie jak w przypadku systemów SCADA, należy zawsze pamiętać o charakterystyce czasowej styku obszarów RT. Interakcje programów sterujących z programami MES nie będą się odznaczały zdeterminowaniem czasowym. Zatem uzależnianie działań sterujących w funkcji odpowiedzi systemu MES, podobnie jak przy systemach nadzorczych, jest błędem konceptualnym tworzonego rozwiązania.



Rys. 116. Ilustracja przykładowego raportu pareto
Fig. 116. Illustration of a sample pareto report

Na rynku stosuje się sporą gamę produktów od dedykowanych, przystosowanych na miarę do danego procesu, po uniwersalne jak np. Wonderware MES 2014. Przykład systemu MES może stanowić [306] lub podsystem MES z dodatku 7.1.2 lub 7.1.3.

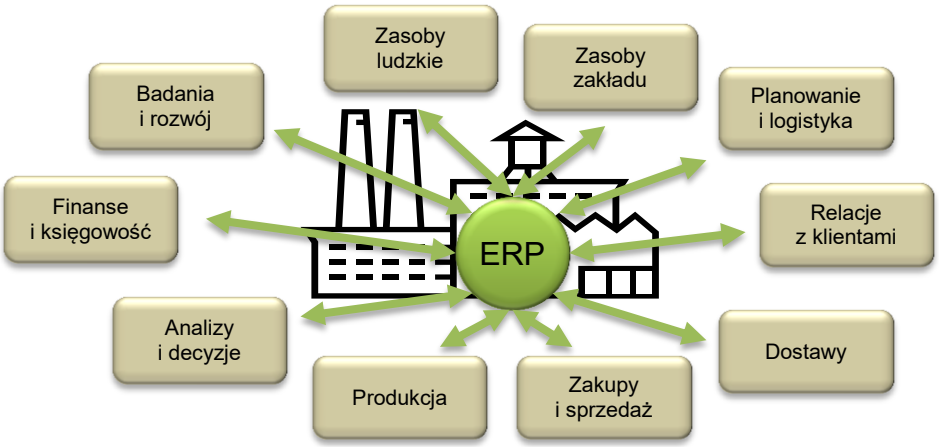
3.2.10. Programy MRP/ERP/APS

(MRP – ang. Manufacturing Resource Planning, ERP – ang. Enterprise Resource Planning, APS – ang. Advanced Planning and Scheduling)

Zgodnie z nazwą, systemy informatyczne tej klasy służą zarządzaniu zasobami przedsiębiorstwa. Nie stanowią zatem bezpośrednich składowych omawianych systemów ISP. Są to systemy pracujące na wyższym poziomie abstrakcji względem przetwarzania danych na produkcji i nie mają bezpośredniego kontaktu z danymi procesowymi. Widać to dobrze na rysunku 14. Stanowią one jednak źródło i miejsce docelowe dla informacji wykorzystywanych w ISP, i tym samym wchodzą z nim w interakcje.

Podobnie jak MES, podsystemy ERP stanowią oprogramowanie wspomagające zarządzanie, z tym że wspomaganiu podlegają procesy biznesowe w kontekście całego przedsiębiorstwa lub oddziału, a nie tylko konkretnej produkcji. W zakres działania zwykle wchodzi wsparcie procesów:

- planowania materiałowego (np. zaopatrzenie, magazynowanie, zapasy, łańcuchy i śledzenie dostaw, transport itp.),
- zarządzania produkcją (np. zdolności, zlecenia, planowanie, monitorowanie jakości itp.),
- obsługi sprzedaży (np. analizy, bilanse, dystrybucja, handel, reklamacje itp.),
- podejmowania decyzji strategicznych (np. księgowych, finansowych, relacji biznesowych, analizy efektywności, inwestycje, projekty itp.),
- zarządzania zasobami ludzkimi (np. płace, kadry, oceny i pomiary itp.),
- zarządzania zasobami zakładu (np. narzędzia i środki, jednostki produkcyjne, serwis, remonty itp.).



Rys. 117. Ilustracja funkcjonalności ERP
Fig. 117. Illustration of ERP functionality

Systemy ERP stanowią najbardziej rozwiniętą i kompleksową klasę systemów informatycznych wspomagających zarządzanie przedsiębiorstwem. Stanowią rozwinięcie standardu MRP pierwszej i drugiej generacji o funkcje ekonomiczno-planistyczne. Systemy dostarczają narzędzi pozwalających na automatyzowanie wymiany danych między komórkami zakładu, przez co upraszcza procesy przepływu informacji biznesowych zarówno wewnątrz, jak i na zewnątrz zakładu. Zobrazowano to na rysunku 117, przy czym podane funkcje są przykładowe i nie wyczerpują zbioru funkcji dostępnych w produktach komercyjnych.

Rozwinięciem systemów klasy ERP są systemy planowania klasy APS (ang. Advanced Planning and Scheduling), dostarczające mechanizmów do symulacji i optymalizacji procesów biznesowych. Są to systemy wspomagające zarządzanie dynamiczne (ang. Resilience Management), dostosowując działanie przedsiębiorstwa do potrzeb chwili.

Centralnym elementem systemów ERP jest baza danych oraz serwer. Funkcjonalności są realizowane przez usługi serwera oraz rozproszone programy klienckie. Komunikacja odbywa się lokalnymi sieciami zakładowymi, zwykle z użyciem protokołów aplikacyjnych i usług

opartych na stosie TCP/IP. Dostęp do modułów jest zabezpieczany przez wbudowane mechanizmy uwierzytelniania użytkownika i autoryzacji jego działań, i/lub przez mechanizmy i polityki domenowe zakładu. Podobnie komunikacja między modułami z reguły jest zabezpieczana z użyciem szyfrowania, sieci VPN (ang. Virtual Private Network) lub innych powszechnie stosowanych mechanizmów. Dane przekazywane między modułami systemu są niejawne i często wrażliwe, dlatego kwestie bezpieczeństwa dostępu są tu bardzo istotne. Wpływ na bezpieczeństwo funkcjonalne jest niewielki lub żaden.

Funkcjonalności systemów ERP są opisane w standardach i katalogach pojęć organizacji APICS [W12], jak również są one definiowane przez producentów (np. SAP, Comarch, Sage, IFS, Microsoft, Oracle i wiele innych). Każdy producent stara się dopasować swój produkt do zapewnienia funkcjonalności podstawowych, ale też wprowadza swoje funkcje i terminy, które mają go wyróżniać na tle konkurencji. Dlatego systematyzacja szczegółowa nazewnictwa i zakresu działań nie ma zbyt wielkiego sensu. Dobrym zobrazowaniem tego jest wiele systemów wymienionych w rozdziale 1.2.1 i opisanych w 3.2.11, które w praktyce stanowią często podsystemy ERP, a ich funkcjonalności wzajemnie się przenikają.

3.2.11. Inne oprogramowanie systemów IT

Systemy, a nawet ich rodzaje czy klasy można bezkarnie mnożyć, szczególnie w zakresie wsparcia procesów biznesowych. Na rynku funkcjonuje wiele rodzajów systemów wspomagania, obsługi, zarządzania itp. Ze względów marketingowych często wokół nowych produktów pojawiających się na rynku tworzy się otoczkę przełomowości i nowej jakości proponowanego rozwiązania. W większości przypadków są to działania obliczane na efekt sprzedażowy. Dla twórcy bądź integratora systemów ISP nie ma wielkiego sensu omawiać chwilowego stanu produktów na rynku. Stosowane w nich technologie informatyczne zwykle bazują na aktualnie popularnych technikach, technologiach, standardach i usługach (np. SOA, XML, WPP, WWW, WSN, CPS, IoT itp. zależnie od potrzeb i rozważanego obszaru) oraz na klasycznych środkach, jak urządzenia, systemy operacyjne, bazy danych, języki programowania, sieci i różnego rodzaju aplikacje omówione powyżej. Kwestia klasyfikacji jest wtórna i nie dotyczy sedna implementacji.

Dla ilustracji różnorodności istniejących klasyfikacji można wymienić przykładowe klasy systemów IT, pracujących w ramach obsługi informatycznej przedsiębiorstwa. Są to:

- CMMS (ang. Computerised Maintenance Management Systems) – Ogólna klasa systemów przeznaczona do wspomagania utrzymania ruchu. Stanowią często podsystemy MES/MRP lub są silnie z nimi zintegrowane.
- BI (ang. Business Intelligence) – Systemy analityki biznesowej przeznaczone do pozyskiwania wiedzy z wytworzonej informacji otrzymanej z zebranych danych. Wspomagają

kierowanie firmą względem zdefiniowanych celów. Do działania wykorzystują przeważnie bazy i hurtownie danych wraz z mechanizmami eksploracji danych (ang. data mining) czy narzędziami analitycznymi opartymi, np. na technologii OLAP (ang. Online Analitycal Processing).

- CRM (ang. Custromer Relationship Managenet) – klasa systemów pozwalająca na tworzenie baz klientów oraz przeróżnych zależności biznesowych wiążących aktywności przedsiębiorstwa z tymi klientami. Umożliwia planowanie działań na podstawie wnioskowania ze zgromadzonych danych.
- SCM (ang. Supply Chain Management) – systemy umożliwiające zamodelowanie i kontrolę tzw. łańcucha dostaw, czyli procesów związanych z przepływem towarów. System wspomaga logistykę działania przedsiębiorstwa i firm współpracujących.
- DSS (ang. Decision Support System) – klasa systemów przeznaczonych do wspomagania podejmowania złożonych decyzji, stosowanych gdy istnieją znane zasady wnioskowania. Zwykle systemy te działają jak systemy ekspertowe, wykorzystując metody sztucznej inteligencji do zautomatyzowania procesów decyzyjnych.
- EIS (ang. Executive Information System) – system informowania kierownictwa. Służy wspomaganiu podejmowania decyzji jak DSS, ale jest funkcjonalnie dostosowany dla tzw. kierownictwa szczebla wyższego, czyli uwzględnia cele strategiczne firmy.
- Systemy PLM (ang. Product Lifecycle Management) – systemy tej klasy wspomagają zarządzanie cyklem życia produktów. Umożliwiają kontrolę nad danym produktem od etapów jego tworzenia, przez produkcję, po wycofanie z rynku.
- PI(M)S (ang. Plant Information (Management) System) – klasa systemów globalnego gromadzenia, zarządzania, analiz i prezentacji danych. Klasa dość ogólna, do której można zaliczyć wszelkie systemy informacyjne dedykowane dla przedsiębiorstw.

Można doszukiwać się wielu innych podsystemów czy też rodzajów takich systemów. Trywializując, można stworzyć system utrzymania automatów z napojami działający w funkcji pogody, o ile klient go potrzebuje. Można utworzyć klasę takich systemów (np. Food Supplying Management Systems) i przekonywać, że taki czy inny produkt należy lub nie należy do tej klasy. Jednak poza łatwiejszym ogarnięciem pojęciowym danego produktu nie wpływa to na zagadnienia deweloperskie. Bardziej istotna jest analiza produktu w kontekście wymagań funkcjonalnych i czasowych oraz możliwości integracji.

3.2.12. Wirtualizacja węzłów

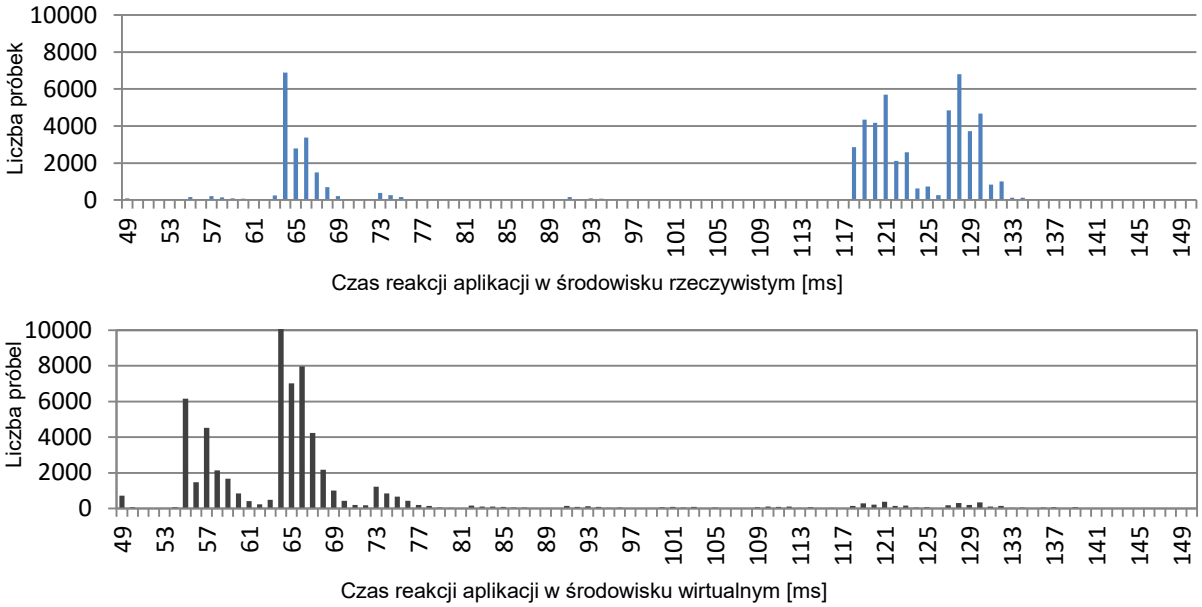
Rozwój technik wirtualizacji spowodował, że dzisiaj prawie każdy element systemów IT może być wirtualny. W kontekście ISP wirtualizacja wykorzystywana jest do zapewnienia wirtualnych węzłów systemu oraz wirtualnej komunikacji sieciowej.

Oprogramowanie dedykowane dla ISP i wspierające techniki wirtualizacji jest jak na razie w fazie badań i rozwoju i nie ma większego znaczenia aplikacyjnego[71]. Nie oznacza to, że technik wirtualizacji nie można wykorzystać w ISP, a wręcz przeciwnie można doszukać się sensownego ich zastosowania z bardzo interesującymi rezultatami. Główne zastosowanie dotyczy wirtualizacji węzłów i uruchamiania na nich programów obsługi klasycznych zadań ISP. Wirtualizacja kontrolerów (np. PLC) ma wielu zwolenników oraz istnieją badania z tym związane (zob. 3.2.3), jednak ze względów ograniczeń praktycznych i braku motywacji aplikacje zwykle nie wychodzą poza laboratoria. Główny problem przy wirtualizacji kontrolerów to zapewnienie ograniczeń czasowych. Co innego z wirtualizacją węzłów interfejsowych. Obecnie nie ma ograniczeń technicznych dla wirtualizacji węzłów z aplikacjami zorientowanymi na ISP, np. typu SCADA, programy wspomagające zarządzanie, serwery baz danych czy nawet serwery danych, np. OPC. Celem wirtualizacji zasobów węzłów interfejsowych jest rozproszenie ich funkcjonalności. Rozdzielenie zadań względem kompetencji użytkowników skutkuje zwiększeniem bezpieczeństwa dostępu. Natomiast rozdzielenie aplikacji wśród węzłów wirtualnych powoduje łatwiejsze ich utrzymanie i ewentualną naprawę. Dodatkowo, wirtualizacja stacji roboczych PC może wprowadzać lepszą gospodarkę zasobami fizycznymi niż uzyskiwaną na maszynach fizycznych. W efekcie możliwe jest uzyskanie lepszych parametrów czasowych pracy aplikacji w środowisku wirtualnym niż w środowisku rzeczywistym. Dla przykładowej aplikacji SCADA, pracującej w transakcji z PLC w środowisku rzeczywistym i wirtualnym, czas reakcji zilustrowano na rysunku 118. Przedstawione wyniki testów były prowadzone dla hosta opartego na Xen oraz dla dwóch maszyn wirtualnych z Windows 7.

W środowisku wirtualnym czas reakcji dla zdecydowanie większej liczby próbek był krótszy niż dla identycznego układu w środowisku rzeczywistym. To pozornie niemożliwe do uzyskania przyśpieszenie nastąpi tylko wówczas, gdy rozpatrywane są aktywności ISP odwołujące się do wybranych, podatnych na wirtualizacje usług systemu operacyjnego. W przypadku odwołań np. do współdzielonych z innymi procesami zasobów sprzętowych, efekt nie będzie tak spektakularny lub wręcz nastąpi pogorszenie charakterystyk czasowych pracy układu. Więcej wyników badań zagadnień rozpraszania funkcjonalności stacji PC

[71] Dotyczy roku wydania publikacji.

z użyciem wirtualizacji zamieszczony jest w [130]. Zagadnienia wirtualizacji komunikacji sieciowej przedstawiono w 3.3.5.

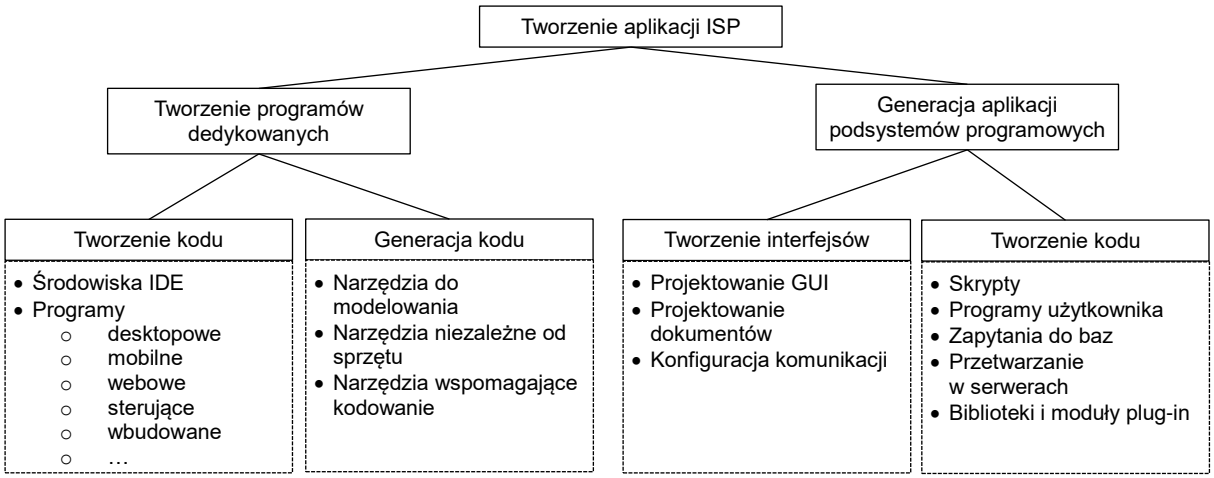


Rys. 118. Przykład charakterystyki czasowej responsywności warstwy aplikacji
Fig. 118. Example of temporal characteristics of application layer responsiveness

3.2.13. Narzędzia deweloperskie

Do tworzenia oprogramowania ISP niezbędne są narzędzia, efektem działania których jest utworzenie aplikacji dla węzła, węzłów lub całego ISP. Tworzenie aplikacji może odbywać się drogą kreacji lub generacji produktu (rozwiązania programowego). Otrzymywanym produktem nie musi być program. Może to być również zestaw plików (tzw. aplikacja, projekt, konfiguracja, ang. application, configuration itp.) konfigurujących i określających przepływ, kodowanie, składowanie oraz przetwarzanie danych w istniejących podsystemach programowych ISP (np. bazy danych, SCADA, MES, ERP, HMI itp.). Wszystkie narzędzia umożliwiają tworzenie tzw. projektu (ang. solution, project itp.) w rozumieniu zbioru danych niezbędnych do utworzenia, parametryzacji i rozwoju produktu.

Pomijając aspekty uruchomieniowe i wdrożeniowe, praca nad utworzeniem aplikacji sprowadza się do konfiguracji narzędzia, konfiguracji produktu (projektu dla platformy docelowej, ang. target) oraz utworzenia kodu. Konfiguracja określa cały szereg parametrów umożliwiających określone działanie narzędzia i uzyskanie produktu o żądanych właściwościach. Natomiast kod służy do zapewnienia specyficznego przetwarzania. W zależności od platformy docelowej kod może być kompilowany do postaci wykonywalnej przez MPU węzła i uruchamiany jako program lub prekompilowany do pseudokodu i uruchamiany w formie interpretowanych skryptów.



Rys. 119. Ogólny podział funkcjonalny narzędzi deweloperskich ISP
Fig. 119. Common functional classification of development tools of ICS

W zależności od danego narzędzia, docelowej platformy OS oraz przeznaczenia tworzonego kodu dostępne są różne języki programowania. Współcześnie, do tworzenia oprogramowania ISP dostępny jest duży wybór języków wspierających różne paradygmaty programowania, od deklaratywnych, takich jak programowanie logiczne czy funkcyjne, przez programowanie strukturalne, proceduralne i modularne do paradygmatów imperatywno-obiektowych (zob. 2.1). Na rysunku 119 przedstawiono ogólny podział narzędzi deweloperskich ISP względem zadań, jakie one realizują. Rysunek nie oddaje sekwencji działań, a jedynie podział. Podczas tworzenia aplikacji ISP może być użyte tak jedno, jak i wiele różnych narzędzi.

Ze względów funkcjonalnych można wyróżnić kilka typów narzędzi deweloperskich. Głównie są to narzędzia do tworzenia programów oraz do adaptacji gotowych podsystemów.

- tworzenie programów dedykowanych
  Do tworzenia programów dedykowanych (zob. 3.2.2) przeznaczone są narzędzia deweloperskie, umożliwiające translacje kodu/opisu/modelu formalnego zrozumiałego dla człowieka na produkt programowy zapisany w zrozumiały dla elementów ISP sposób. Dostępne są narzędzia zarówno bardziej, jak i mniej popularne, ale w większości dobrze udokumentowane i powszechnie dostępne, w wersjach darmowych, jak i płatnych. Stanowią one najczęściej formę zintegrowanych środowisk deweloperskich (tzw. IDE, ang. Integrated Developement Environment) wraz z odpowiednim zestawem bibliotek. Są to narzędzia uniwersalne i umożliwiają tworzenie dowolnych elementów programowych w oferowanym zakresie. Zakres ich działania jest określony natywnie lub zależy od tzw. wtyczek (ang. plug-in) zainstalowanych w danym środowisku. Program tworzony jest dla konkretnej platformy docelowej, na określoną platformę OS i o funkcjonalnościach osiągalnych w ramach możliwości i dostępności określonych zasobów i środków dla

danego narzędzia. Rozwiązania multiplatformowe lub niezależne od platformy systemowo-sprzętowej w lokalnych programach ISP spotykane są rzadko. Głównie ze względu na specyfikę obsługi interfejsów oraz statyczne przypisanie konkretnych urządzeń komputerowych do konkretnych funkcji.

W dziedzinie ISP nie ma specjalnych wymogów względem wyboru narzędzi. Istotniejszy jest wybór odpowiedniego języka i wsparcia. Każdy język ma związaną dziedzinę zastosowań i dlatego warto zastanowić się, przed podjęciem programowania, czy stosowanie danego rozwiązania programistycznego dla danego problemu będzie optymalne względem czasu, kosztów i problemów, które potencjalnie może przysporzyć oraz czy nie ograniczy tworzonego rozwiązania w perspektywie czasu i rozwoju. Interesujący przegląd języków znajduje się w [W27]. Wśród programistów często istnieje tendencja do trwałego wiązania się z rozwiązaniami, które uznawane są za najlepsze, z jednoczesną negacją innych. Nie jest to zdrowe podejście, a już na pewno nie profesjonalne. W zdecydowanej większości przypadków jest to bardziej kwestia mody, uprzedzeń, irracjonalnych opinii i obserwacji itp. niż merytorycznej analizy przydatności w kontekście danego problemu.

W podstawowy zakres funkcjonalny narzędzi wchodzi:

- projektowanie interfejsów użytkownika,
- programowanie,
- symulacja/debugowanie,
- konfiguracje.

W szczególności, narzędzia umożliwiają tworzenie i uruchamianie wszelkich rodzajów programów dedykowanych dla ISP (zob. 3.2.2) w postaci:

- programów desktopowych

  Narzędzia dają wsparcie dla programowania w językach uniwersalnych wysokiego poziomu[72] i wszelkiego rodzaju VPL (ang. Visual Programming), z użyciem różnych bibliotek i frameworków[73]. Przykłady narzędzi mogą stanowić dowolne środowiska IDE[74] umożliwiające kompilacje kodu źródłowego i linkowanie z innymi elementami celem utworzenia programu uruchamianego w danym lokalnym środowisku systemu operacyjnego stacjonarnego węzła klasy PC. Wybór narzędzia oraz wspierających technik i technologii zależy przede wszystkim od docelowej platformy systemowej.

[72] Na przykład: C++, C#, Objective C, Java, Visual Basic, Pascal, Julia itp.

[73] Liczba dostępnych na rynku bibliotek, frameworków, modeli programowania, standardów kodu i opisu danych jest bardzo dużo. Ich opis i systematyzacja leżą poza zakresem niniejszej książki a sam w sobie mógłby posłużyć do stworzenia osobnej. Przykłady to .NET, WPF, WCF, Silverlight, Cocoa, Swing, JavaFX, QT i wiele innych.

[74] Na przykład: MS Visual Studio, Borland Developer Studio, SharpDevelop, Dev-C++, Xcode, Momentics Tool Suite, Eclipse, Code::Blocks Studio itp.

Często podyktowany jest jednak subiektywnym wyborem programisty wynikającym z przyzwyczajeń i doświadczeń. W przypadku programów dedykowanych należy zawsze brać pod uwagę możliwości rozwoju produktu na przestrzeni kilkunastu lat.

- programów dla platform mobilnych

  Wsparcie oferowane dla urządzeń mobilnych dotyczy również języków uniwersalnych wysokiego poziomu[75] wraz z bibliotekami i frameworkami wspierającymi programowanie[76]. Przykłady narzędzi mogą stanowić, tak jak w punkcie powyżej, uniwersalne i zaawansowane narzędzia deweloperskie lub popularne darmowe środowiska programistyczne[77]. Wybór narzędzia, języka i wsparcia jest z reguły podyktowany platformą sprzętowo-systemową urządzenia mobilnego. Aktualnie nie ma tu problemów długowieczności programów z powodu szybkich zmian na rynku urządzeń mobilnych. Warto jednak rozważać takie tworzenie kodu, aby był on łatwo utrzymywalny, a nawet migrowalny na inne platformy. Prawdopodobnie dane urządzenie mobilne i dana wersja jego oprogramowania nie będzie żyła zbyt długo, ale długi czas eksploatacji ISP wymusza długowieczność aplikacji, i tym samym konieczność zapewnienia jego konserwacji i ewolucji.

- programów webowych

  Narzędzia podobne jak wyżej i wykorzystanie przeróżnych technologii webowych i języków zależnie od rodzaju aplikacji webowej, jaki jest wymagany. W ISP najczęściej wykorzystuje się środki i metody umożliwiające realizacje względnie dobrej interakcji[78] wraz z opisami dokumentów webowych w powszechnie znanych i standardowych językach[79] i przetwarzaniem opisanym w różnych językach uniwersalnych lub przeznaczonych dla środowiska web[80]. Nie istnieją konkretne techniki, technologie i języki dedykowane dla aplikacji webowych w ISP, a istniejące nie ograniczają wymaganego zakresu funkcjonalnego. Dlatego wskazanie najlepszego podejścia względem narzędzi nie jest tu możliwe. Stosowalność danej techniki, języka, metody, modelu czy narzędzia wynika najczęściej z ich popularności oraz z przyzwyczajeń i ograniczeń programistów tworzących dla tej dziedziny zastosowań. Warto rozważyć techniki umożliwiające prostą rozbudowę stron webowych o nowe, często powtarzalne, elementy dynamiczne oraz możliwości rozszerzenia dostępu do nowych źródeł danych.

[75] Głównie C, Objective C, Java ME, Python, AS3.
[76] Na przykład: .NET, Cocoa, Swing, Starling, Feathers, QT, Node.JS itp.
[77] Na przykład: MS Visual Studio, Borland Developer Studio, Xcode, NetBeans, Eclipse itp.
[78] Na przykład: ASP, JSP, AJAX, CGI itp.
[79] HTML, XHTML, XML itp.
[80] Na przykład: PHP, C#, Java lub JavaScript, rzadziej w Ruby, Pearl, Python, VisualBasicScript itp.

- programów dla swobodnie programowalnych urządzeń sterujących

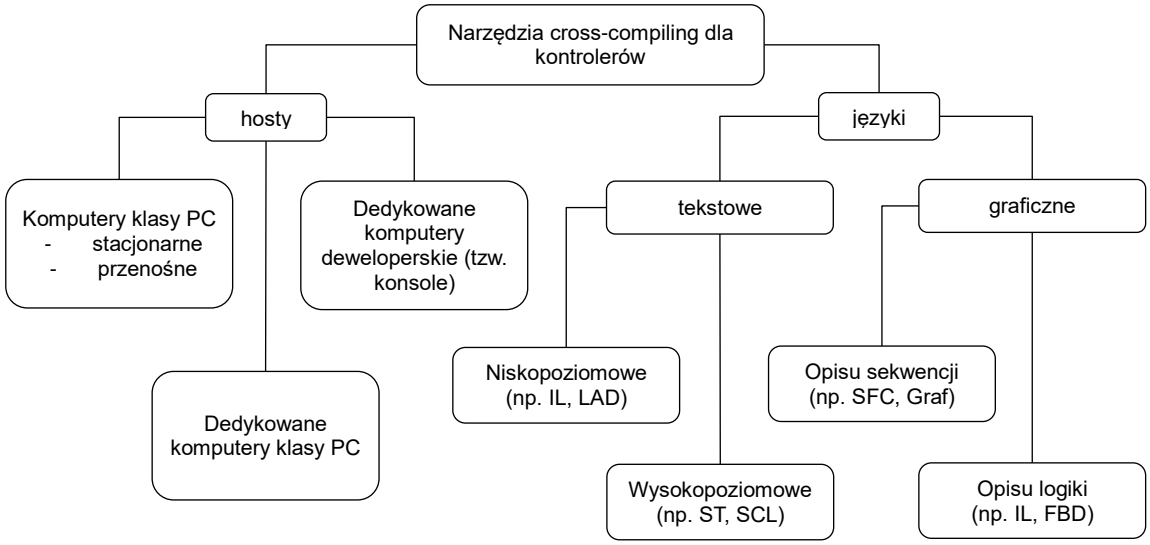
  Mowa tu głównie o narzędziach do tworzenia programów dla swobodnie programowalnych kontrolerów (komputerów sterujących zob. 3.1.3, 3.1.4). Są to środowiska specjalizowane dla konkretnych urządzeń lub ich rodzin uruchamiane na tzw. hostach, czyli na uniwersalnych lub dedykowanych komputerach klasy PC, a także na specjalizowanych konsolach. W ostatnich latach konsole straciły na znaczeniu z racji popularyzacji komputerów przenośnych. Działanie narzędzi opiera się na tzw. cross-compiling, czyli przygotowaniu kodu i kompilacji na sprzętowo programowej platformie deweloperskiej innej niż platforma docelowa urządzenia. W narzędziu definiuje się tzw. target, czyli określa platformę docelową i przeprowadza wszelkie prace programistyczne w kontekście tej definicji. Działanie mechanizmów translacyjnych, linkujących i komunikujących również zależy od tej definicji. Finalnie, produkt programistyczny jest przesyłany z platformy deweloperskiej (ang. download) do urządzenia i na nim uruchamiany.

  Z reguły, narzędzia tego typu umożliwiają zdefiniowanie w projekcie wielu platform docelowych oraz ich zmianę. Jednak kod źródłowy i konfiguracja poprawne dla jednej platformy mogą być błędne dla innej, co wynika bezpośrednio z różnic konstrukcyjnych urządzeń.

  Typowym przykładem mogą być narzędzia do programowania PLC, paneli, falowników, stacji procesowych itp. Oferują one wsparcie dla programowania z użyciem języków danej platformy docelowej, edytory interfejsów graficznych, tabele parametryzacyjne i diagnostyczne, symulatory itp. Zwykle obsługują kilka języków programowania specyficznych dla urządzeń sterujących i często zgodnych z normą IEC61131-3 [M21]. Są to języki zarówno tekstowe[81], jak i graficzne[82] (rys. 120).

  Przeważnie narzędzia tego typu są produktem specjalizowanym dla danych urządzeń sterujących, niemniej jednak, często bazują również na wspólnej platformie programowej, adaptując wejście, wyjście i specyfikę konfiguracji do konkretnej, docelowej platformy sprzętowej. Adaptacja może być wbudowana lub dołączana przez moduły typu plug-in.

[81] Na przykład: IL, ST, STL, SCL itp.
[82] Na przykład: LD, FBD, SFC, LAD, AS, CFC, Graph i inne.



Rys. 120. Klasyfikacje urządzeń deweloperskich i języków programowania dla kontrolerów
Fig. 120. Classification of development devices and programming languages for controllers

Przykładem popularnych narzędzi mogą być narzędzia bazujące na niezależnej sprzętowo platformie CoDeSys Automation Alliance[83], czy też pakiety dedykowane dla konkretnych rodzin kontrolerów[84]. Na rynku znajdują się również narzędzia uniwersalne, gdzie programowanie oddzielone jest od platformy sprzętowej i odbywa się na abstrakcji urządzeń wirtualnych. Narzędzia takie umożliwiają tworzenie kodu zgodnego z IEC61131-3 jak i IEC61499 (por. 1.2.6). Przykładem może być ISaGraf Workbench bazujący na technologii Automation Cooperative Platform [W28], Fbanch bazujący na OOONEIDA Workbench [W3], FBDK [W22] czy produkt CORFU Uniwersytetu w Patras [W9]. Są to nowoczesne narzędzia z dużym potencjałem rozwoju, ale obecnie o niewielkiej popularności (zob. 3.1.4).

W podstawowy zakres funkcjonalny narzędzi dla programowania kontrolerów wchodzi:

- tworzenie i modyfikacja konfiguracji sprzętowej danego węzła (ang. HWC – hardware configuration; określanie, czym fizycznie jest węzeł),
- konfiguracja elementów sprzętowo-programowych (np. adresacja, przerwania, powiązania itp.),
- programowanie (tworzenie kodu i danych[85] oraz ich struktury),

[83] Na przykład: easy Soft, TwinCat, SX-Programmer, MXpro i inne.
[84] Na przykład: Proficy ME dla GE IP, Step7 i TIA Portal dla Simatic, Unilogic i Visilogic dla Unitronics, Automation Studio dla B&R czy PG5 Controls Suite dla Saia Burgess.
[85] Dotyczy alokacji zmiennych, określania stałych, rezerwacji miejsca na składowanie danych itp.

- funkcje uruchomieniowe (walidacja kodu, kompilacja, komunikacja z węzłem, debugowanie, diagnostyka, wizualizacja i formalizacja kodu [400], [401], [402] itp.),
- funkcje migracyjne (przenaszalność kodu pomiędzy językami: translacja i rzutowanie kodu; przenaszalność programów pomiędzy platformami: eksport i import projektów, formalizacja kodu [401], [403]).

Do szczególnych cech narzędzi dla kontrolerów należy możliwość definiowania jednostek organizacyjnych kodu (tzw. modułów, sekwencji, bloków, ang. POU – Program Organization Unit, OB – Organization Block) i tworzenia tego kodu w różnych językach w ramach jednego projektu. Ponadto, często spotyka się możliwość przechodzenia między językami, zarówno przez automatyczną konwersję, jak i przez nakładkowanie edytora na kod zapisany w postaci jednego języka podstawowego. Należy się jednak spodziewać, że mechanizmy konwersji wprowadzają nadmiarowości, a czasami nawet i błędy, a mechanizmy nakładkowe nie działają dla każdej konstrukcji programowej.

Inna cechą narzędzi jest możliwość zapisu projektu nie tylko w zasobach stacji deweloperskiej, ale i w samym kontrolerze. Jest tu mowa nie tylko o zapisie programów wykonywalnych, ale i innych danych projektu, takich jak źródeł programów wraz z tabelami symboli, komentarzami, a nawet dokumentacją. Odczyt takiego projektu jest możliwy do postaci źródłowej. Jest to opcja, która jest możliwa dzięki temu, że program w kontrolerze nie jest kompilowany do postaci binarnej tylko do interpretowalnego pseudokodu lub dzięki temu, że w kontrolerze istnieje specjalna pamięć, najczęściej typu flash, na którą można zapisać źródłowe i dodatkowe pliki projektu. Nie wszystkie narzędzia i nie wszystkie urządzenia wspierają takie funkcje.

Ponadto programy narzędziowe dla kontrolerów umożliwiają ustanowienie i zarządzanie kontrolą dostępu do projektu, jak i jego elementów. Są to wygodne mechanizmy, które dają możliwość zabezpieczenia projektu, kodu lub jego fragmentu przed dostępem osób i firm trzecich. Jednak jeśli nie istnieją specjalne przesłanki, aby z nich korzystać, to warto rozważyć rezygnację z zabezpieczania dostępu (por. 3.2.2). Z punktu widzenia klienta na pewno będzie to pozytywnie odebrane, choćby ze względu na cechę trwałości rozwiązania (por. 2.3.2).

- programów dla urządzeń dedykowanych

Urządzenia dedykowane są tworzone dla konkretnej aplikacji lub dla konkretnego zastosowania. Dlatego programowanie urządzeń dedykowanych można porównać do programowania systemów wbudowanych (ang. embedded systems). Narzędzia są pochodną tej dziedziny. Zatem wybór narzędzia zależy od platformy sprzętowej oraz

dostępności systemu operacyjnego na urządzeniu. Brak systemu zmusza programistę do użycia narzędzi umożliwiających generację kodu dla danej architektury procesora. Jeśli natomiast urządzenie dedykowane jest wyposażone w system operacyjny, to z reguły narzędzia będą zależały od tego systemu. Idea użycia narzędzi deweloperskich opiera się w tej kategorii, podobnie jak w przypadku urządzeń sterujących, na metodzie cross-compilling. (np. Atmel Studio, CodeVisionAVR, Arduino IDE, Momentics IDE, Eclipse itp.).

- innych elementów programowych
  Tworzenie wszelkich elementów programowych typu biblioteki, komponenty, aplety, skrypty, moduły, sekwencje itp. ułatwia i poszerza możliwości twórcze oraz skraca czas tworzenia rozwiązań.

- narzędzia do generacji aplikacji
  Zagadnienie automatycznego generowania aplikacji dotyczy generacji zestawu plików konfiguracyjnych dla innych narzędzi (podsystemów programowych ISP), jak np. systemy SCADA, HMI czy klienci serwerów baz danych. Tworzenie aplikacji dokonywane jest przy użyciu specjalnych narzędzi wspomagających twórcę w projektowaniu interfejsu graficznego, składowaniu i przetwarzaniu danych, tworzenia związków między elementami rozproszonymi itp. Najprostsze z nich dają możliwość formalnego opisania działania danej aplikacji (np. przez pliki tekstowe, binarne, XML, AutomationML, CAEX itp.) [401], a najbardziej złożone dają duże wsparcie zarówno od strony kreatorów, edytorów, jak i mechanizmów walidacji, symulacji i testowania. Proste narzędzia są często integrowane z narzędziami deweloperskimi dla programów sterowania (np. easy Soft CoDeSys). Architektura tych złożonych bywa jednak rozbudowana. Najbardziej zaawansowane platformy dostosowują standardowe funkcjonalności dostępnych technologii IT dla potrzeb ISP, oferując tworzenie i zarządzanie obiektami aplikacji, obsługę wspólnej przestrzeni nazw dla różnych funkcjonalności, zintegrowane panowanie nad zagadnieniami bezpieczeństwa dostępu i bezpieczeństwa funkcjonowania, obsługę komunikacji i interfejsów integracyjnych. Często narzędzia te mogą stanowić elementy ISP, dając możliwość modyfikacji aplikacji w trakcie pracy systemu. Przykładem narzędzi mogą być systemy SCADA wymienione w 3.2.7, narzędzia baz danych (3.2.5) lub szerzej, całe koncepcje architektur oprogramowania wspomagającego tworzenie aplikacji, jak Archestra firmy Wonderware lub Totally Integrated Automation firmy Siemens.

- narzędzia do generacji kodu

  Automatyczna generacja kodu wiąże się z narzędziami do modelowania, symulacji i generacji kodu źródłowego dla wybranych węzłów (np. PLC) lub całego systemu docelowego. Stanowią one wyrafinowane narzędzia umożliwiające opis rozwiązań i ich testowanie na wysokim poziomie abstrakcji. Można w nich znaleźć funkcje wspomagające zadania:

  - projektowania i modelowania aplikacji,
  - symulacji elementów systemu, jak i stworzonych modeli działania,
  - testowania całych aplikacji w środowisku symulowanym na bazie modeli,
  - generacji kodu i uruchamiania na urządzeniach.

  Przykład mogą stanowić dobrze znane z innych zastosowań narzędzia, jak Matlab i Simulink [47]. Matlab jest językiem wysokiego poziomu służącym do realizacji złożonych zadań obliczeniowych. Narzędzie dostarczane jest wraz z zaawansowanym i wygodnym w użyciu środowiskiem interaktywnym. Simulink natomiast jest uniwersalnym środowiskiem dla symulacji i projektowania opartego na modelach, dla zastosowań wielodziedzinowych. Przeznaczony jest dla symulacji systemów dynamicznych i wbudowanych, w tym różnorodnych systemów zależnych od czasu. Dostarcza interaktywnego środowiska graficznego oraz bibliotek różnego rodzaju, tzw. bloków, umożliwiających realizacje wymienionych wyżej zadań, a w szczególności tworzenia i weryfikacji modeli funkcjonowania elementów systemu oraz samego procesu.

  Do działania ze środowiskiem ISP wykorzystuje się specjalne moduły dodatkowe, tzw. toolbox'y dostarczające narzędzi do modelowania oraz tzw. kodery, czyli moduły do generacji kodu źródłowego. Przykładem może być Control System Toolbox oraz SimulinkPLC Coder działający z wieloma narzędziami deweloperskimi dla PLC, jak: Automation Studio, TwinCAT, Step 7, PC Worx i wiele innych.

  Generatory kodu oparte na modelowaniu systemów są doskonałymi narzędziami dla automatyków. Informatycy, z racji niedostatku doświadczenia z modelowaniem systemów, mogą być zniechęceni do używania tego typu narzędzi. Wydaje się jednak, że przyszłością tworzenia rozwiązań programistycznych dla dziedziny ISP będą narzędzia generujące rozwiązania na podstawie opisów formalnych lub nawet pseudoformalnych. Obecnie narzędzia do generacji kodu warto stosować, gdy problem jest złożony i dotyczy regulacji ciągłych. Użycie tego rodzaju narzędzi jak Matlab i Simulink do tworzenia względnie prostych aplikacji nie znajduje specjalnego uzasadnienia.

ISP nie wprowadzają ani ograniczeń ani specjalnych wymagań dotyczących narzędzi deweloperskich. Każde narzędzie umożliwiające uzyskanie żądanego produktu jest dobre. W praktyce nie zaleca się stosowania narzędzi starych. Produkty związane z ISP są aktualnie

silnie rozwijane, a starsze wersje mogą mieć irytujące ograniczenia i błędy. Jednak przejście z narzędzia starszego na nowsze jest nieodwracalne. Projekty zapisane w nowszym narzędziu przeważnie nie są obsługiwane w narzędziach starszych. W przypadku narzędzi dla kontrolerów dotyczy to zarówno projektów źródłowych przechowywanych w zasobach stacji deweloperskiej, jak i projektów produkcyjnych przechowywanych w węzłach ISP (np. PLC, DCS), czyli dla przykładu projekt zapisany do sterownika może nie być odczytywalny z użyciem wcześniejszych generacji narzędzi.

3.3. Infrastruktura komunikacyjna

Przesyłanie informacji jest kluczowe w systemach rozproszonych. ISP są systemami rozproszonymi terytorialnie. Ich węzły znajdują się przy procesie, który z reguły prowadzony jest na stosunkowo dużym obszarze. Informację z procesu można przekazywać do systemu analogowymi kanałami przemysłowymi typu pętla prądowa itp. oraz przy użyciu sieci komputerowych. W praktyce unika się kanałów analogowych, eliminując je przez zastąpienie łączami cyfrowymi. Jeśli nie ma możliwości eliminacji, to kanały analogowe prowadzi się jak najkrótszymi przewodami sygnałowymi, odpowiednio ekranowanymi (por. 3.4.2). Wynika to z racji dużych kosztów ich budowy oraz możliwości zaburzeń elektromagnetycznych (EMI). Do realizacji kanałów cyfrowych używa się najczęściej elementów opisanych w kolejnych podrozdziałach. W kwestii doboru rozwiązań więcej można znaleźć w [118].

3.3.1. Sieci komputerowe

Jak wspomniano przy opisie abstrakcji sieci (por. 2.4.4), sieci komputerowe dedykowane do wykorzystania w ISP to sieci polowe, sieci RTE oraz mniej lub bardziej niestandardowe rozwiązania dedykowane. Głównie stosuje się lokalne sieci kablowe, choć ostatnimi laty wzrasta wykorzystanie sieci rozległych oraz bezprzewodowych, w tym sieci lokalnych (najczęściej tzw. WiFi: IEEE 802.11), publicznych (np. Internet, GSM), personalnych (ang. PAN – personal area network: Bluetooth, ZigBee itp.) oraz technik transmisji bliskiego zasięgu (NFC, RFID itp.).

Każde z wymienionych rozwiązań znajduje zastosowanie w ISP. Różni się jednak zdecydowanie dziedziną zastosowań. Zastosowanie sieci personalnych to głównie chwilowe połączenia diagnostyczne, programowania oraz debugingu, czy też ewentualnie lokalne usługi ISP dołączane dynamicznie oraz zastosowania WSN (ang. Wireless Sensor Networks) [151]. Zakres zastosowania sieci bliskiego zasięgu to głównie detekcja obecności oraz identyfikacja produktów i elementów systemu.

Sieci polowe (ang. fieldbus) rozwijane są od lat 80. ubiegłego wieku. Pojęcie dotyczy wszelkich sieci komputerowych przeznaczonych do obsługi urządzeń na poziomie procesu technologicznego w czasie rzeczywistym. Są to sieci lokalne pracujące na medium przewodowym typu kable miedziane lub światłowody. Nazwa pochodzi od działania sieci w pewnym polu (obszarze, terenie, miejscu ang. field) w kontekście przemysłowym kojarzonym jako teren, na którym prowadzony jest proces i działa sieć, stąd czasami sieci te nazywane są sieciami lub magistralami terenowymi, lub miejscowymi. Zadania sieci polowych sprowadzają się do:

- łączenia urządzeń polowych (węzłów ISP) w celu zapewnienia wymiany informacji między nimi,
- rozwiązywania i upraszczania zagadnień systemowych (okablowanie, standaryzacja, wymiana informacji, rozwój urządzeń inteligentnych),
- realizacji komunikacji zdeterminowanej czasowo dla szerokiego spektrum aplikacji na poziomie procesu.

Ze względu na swą popularność, technologia sieci polowych jest bardzo dobrze udokumentowana i sprawdzona w praktyce [253], [385], [387]. W ostatnich latach rozwój sieci polowych został zatrzymany (por. 2.4.4 p. Generacje). Powodem stał się rozwój techniki Ethernet [160] i jej adaptacja do potrzeb ISP [102], [414], [119], [122], [129]. Korzyści, jakie płyną z wykorzystania sieci Ethernet, wynikają z aspektów ekonomicznych produkcji komponentów i aplikacji, zachowania spójności infrastruktury sieci zakładowych, łatwiejszego utrzymania i serwisowania, znaczącej popularności, standaryzacji, ale przede wszystkim z jakości standardu Ethernet.

Dla uproszczenia klasyfikacji RTE względem dziedziny zastosowań wprowadzono klasy działania sieci Ethernet[86] (ang. RT class). Klasy definiuje się względem możliwości zapewniania zdeterminowanej w czasie obsługi pakietów na rzecz aplikacji czasu rzeczywistego. Rozróżnia się następujące klasy sieci:

- Class 0 – działanie sieci bez ograniczeń czasu rzeczywistego (ang. NRT – Non Real Time, non-RT);
- Class 1 – działanie w trybie czasu rzeczywistego (ang. RT – Real Time) ze skalowalnym cyklem powyżej 10 ms (ang. SRT, Low End) ze scenariuszem wymian działającym na warstwie transportowej TCP/IP i bez wymagań dotyczących innych komponentów sieci Ethernet;

[86] Określane przez międzynarodowe stowarzyszenie producentów i użytkowników systemów automatyki IAONA.

- Class 2 – działanie sieci w trybie RT z cyklem od 1 ms do 10 ms (ang. HRT, High End) ze scenariuszem wymian działającym na standardowej warstwie MAC i z użyciem innych komponentów sieci obsługujących RTE;
- Class 3 – działanie sieci w trybie izochronicznym (ang. IRT – isochronosus RT) z cyklem poniżej 1 ms, z synchronizacją wymian i zegarów abonentów, małą niestałością cyklu rzędu 1 μs oraz z użyciem innych komponentów sieci obsługujących RTE. Rozwiązania takie często działają na własnych, modyfikowanych programowo warstwach, wprowadzających nowe funkcjonalności do standardowego łącza Ethernet.
- Class 4 – klasa działania taka jak dla Class 3 z możliwością implementacji nowych funkcjonalności w sprzęcie. W tym wypadku dopuszcza się niestandardowe rozwiązania sieci Ethernet. Rozwiązania takie nie będą współpracować z węzłami, które nie są wyposażone w interfejsy danego typu.

Wśród kilkudziesięciu rozwiązań klasy RTE bazujących na technice kablowej 100Base-TX znajduje się kilka, które są objęte standaryzacją wg norm PN-EN61158 i PN-EN 61784 (2.2.6). Wspomniane normy dotyczą sieci polowych, jak również sieci Ethernet wykorzystywanych w kontekście sieci polowych. W szczególności norma IEC61784-2 [M40] dotycząca profili bazujących na IEC8802-3 [M75], odnosi się do rozwiązań RTE [103]. Sieci RTE nie rozwiązują problemów związanych z tzw. „wąskim gardłem" ISP, czyli przepustowością sieci, choć ze względu na relatywnie dużą jej wartość na warstwach łącza i fizycznej znacząco go niwelują. Z poziomu warstwy aplikacji sprawność zależy od wielu czynników, a Ethernet, nawet gigabitowy, nie musi rozwiązać problemu [180]. Niestety, ani pojawienie się technologii RTE, ani wprowadzenie jej protokołów do standardów międzynarodowych nie rozwiązało problemu mnogości niekompatybilnych rozwiązań [107]. Szczegóły dotyczące technologii RTE można znaleźć w dokumentacjach konkretnych rozwiązań oraz w [385], [414], a jej analizy np. w [121], [126], [90], [75], [109], [155].

Sieci bliskiego zasięgu (personalne, pikosieci) wykorzystywane są głównie do działań administracyjnych, zapewniania alternatywnych kanałów transmisji oraz identyfikacji użytkowników i obiektów. W środowisku gęstego rozmieszczenia elementów i na zamkniętym obszarze sieci PAN wydają się ciekawą alternatywą dla sieci kablowych. Mogą one stanowić element interfejsów multisieciowych i multiusługowych, tworząc konwergentne rozwiązania komunikacyjne trzeciej generacji (3G, por. 2.4.4), czasem też określane jako otaczające (ang. ambient). Umożliwiają one dostosowywanie się do bieżących potrzeb komunikacyjnych systemu oraz jego dynamicznej struktury (np. mobilnej). Dla przykładu, dostępność określonego zakresu usług może zależeć od lokalizacji danego abonenta w przestrzeni, w której pracuje ISP. Zatem, dziedziną zastosowań mogą tu być podsystemy bezpieczeństwa, kontekstowe usługi lokalne oraz zadania diagnostyczno-serwisowe. Rozwiązania bliskiego zasięgu wyko-

rzystuje się również w podsystemach lokalizacji obiektów i ostrzegania przed kolizją elementów ruchomych (zob. np. [147]).

3.3.2. Protokoły

(protokoły sieciowe, ang. Network Protocol)

Protokół jest to zbiór reguł i formatów, mających zastosowanie w komunikacji między procesami przy wykonaniu danego działania (por. [33]). Definicja określa ciąg komunikatów, które należy wymienić, oraz format danych w komunikatach, czyli składnie (zasady) dialogu i semantykę (znaczenie) jego treści. Pojęcie protokołu zawsze odwołuje się do modelu (węzeł, stos, por. 2.2, 2.4.2) i jego elementów. Dlatego błędem jest twierdzenie, że protokół umożliwia dialog urządzeń. Dany protokół umożliwia skomunikowanie programów pracujących w węzłach na danych warstwach zastosowanego modelu. Inaczej mówiąc, proces obsługujący dany protokół na danej warstwie stosu protokołów węzła umożliwia skomunikowanie się z jego odpowiednikiem w innym węźle. Z punktu widzenia zadań funkcjonalnych systemu działanie protokołu w ISP dotyczy skomunikowania aplikacji pracujących w węzłach. Wchodzi w to ustanowienie zależności aplikacyjnych, transferu danych użytecznych przez wykorzystanie odpowiednich usług (lub transakcji), i obsługi sytuacji awaryjnych (zapewnienie trwałości i niezawodności) (por. 2.4.2, 2.4.4, 2.4.5). W celu zapewnienia poprawnego działania ISP kluczową sprawą jest nie tylko skomunikowanie aplikacji węzłów w celu realizowania zadań, ale i właściwy dobór protokołów pasujących do tych konkretnych realizowanych zadań [118].

W sieciowych środowiskach heterogenicznych może wystąpić problem kompatybilności urządzeń ze względu na niezgodność obsługiwanych protokołów, a czasem nawet ich wersji, lub niezgodności mediów transmisyjnych. Prostym rozwiązaniem dla medium są różnego rodzaju konwertery. Przy stosowaniu takich urządzeń należy jednak brać pod uwagę wnoszone opóźnienia. Dla konwerterów protokołów, w przypadku sieci zdeterminowanych czasowo, integracja często nie jest w pełni możliwa. Wynika to z problemów zsynchronizowania cyklicznych obiegów informacji w integrowanych, różnych sieciach (por. 2.4.2). Może się okazać, że ze względu na różne cykle i zjawiska opisane w 2.4.2 informacja będzie musiała być nadpisywana lub gubiona. Innym problemem jest fakt, że niektóre protokoły mogą nie dostarczać informacji wymaganej przez inny protokół, np. wymaganej adresacji. Dlatego stosując konwertery protokołów, należy pamiętać, że konwerter nie jest złotym środkiem na łączenie wszystkiego ze wszystkim. Do działania konwerter wymaga przemyślanej konfiguracji, a czasami konwersja nie jest możliwa. Dla przykładu, konwersja Modbus na Profinet wymaga mapowania adresów i danych. Ze względu na różnice w adresacji niemożliwe jest wykonanie połączenia na poziomie którejkolwiek z warstw stosów protokołów. Możliwe jest to

jedynie w warstwie aplikacji użytkownika przy przyjętych określonych założeniach, które stanowią swoistą konfigurację bramy. Ponadto, przy łączeniu ISP poprzez sieci klasyczne następuje utrata własności czasowych związanych z przekazywanymi danymi.

Poniżej wymieniono wybrane rozwiązania z ustandaryzowanych rodzin protokołów przemysłowych wg wspomnianych wcześniej norm. Rodziny te obejmują protokoły polowe oraz protokoły dla sieci Ethernet [125]. Podano odniesienia do norm i profili normy IEC61784 [M39], [M40] (ang. CFP – Communication Profile Family). Są to przykłady subiektywnie wybrane względem popularności. Pełna rodzina protokołów sieci przemysłowych jest zdecydowanie większa. Zakres opisu jest podstawowy, aby dać wgląd w ogólne przeznaczenie i charakterystykę. Szczegóły można znaleźć w specyfikacjach i opracowaniach, np. w [387], [68], [101], [225], [253], [385], [281].

- Foundation Fieldbus

  Grupa protokołów wspieranych przez międzynarodową organizację Foundation Fieldbus (FF)[87] powstałą z połączenia InterOperable System Project oraz WorldFip North America.

  Dla sieci polowych zdefiniowany jest protokół H1 (CPF1/1) oraz H2 (CPF1/3). H1 oferuje transmisje z prędkością 31,25 kb/s na potrzeby automatyki (zastąpienie pętli prądowych), a H2 oferuje 1 lub 2,5 Mb/s i przeznaczona jest do systemów automatyzacji. Wykorzystywane modele architektury oprogramowania dla obsługi wymian to Client-Server, Publisher-Subscriber i tzw. Report Distribution [410], [316], [168], [237], [M93]. Przykład aplikacji można znaleźć np. w [61].

  Dla sieci Ethernet jest przeznaczony protokół HSE (CPF1/2, IEC61158 Type 5). Protokół wykorzystuje standardowy stos Ethernet oraz daje wsparcie dla TCP/IP. Dedykowany jest do integracji i tworzenia spójnej platformy łączącej różne komponenty systemu, jak również może działać z funkcjonalnością sieci polowej. Protokół może pracować w trybie czasu rzeczywistego na zamkniętym segmencie, może również integrować różne sieci polowe, wykorzystując specjalne urządzenia łączące (tzw. ang *link devices*). Obsługa wymian bazuje na modelach z H1 i H2 [170].

- CIP

  Grupa protokołów (ang. *Common Industrial Protocol*) wspieranych przez międzynarodową organizację Open DeviceNet Vendors Association (ODVA)[88] założoną w 1995 roku.

  Protokoły CIP w warstwie aplikacji są jednolite, a w warstwach niższych wykorzystywane są różne technologie. Dla sieci polowych zdefiniowane są protokoły

[87] http://www.fieldbus.org
[88] http://www.odva.org

ControlNet (CPF2/1) oraz DeviceNet (CPF2/3). DeviceNet jest uniwersalną siecią do łączenia urządzeń automatyki i wymiany danych procesowych. Bazuje na modelach Master-Slave i Producent-Konsument. W warstwie dostępu do łącza wykorzystuje magistralę CAN. Oferuje prędkości 125 kb/s, 250 kb/s i 500 kb/s. ControlNet jest typową siecią polową z dość dużą, jak na taki rodzaj sieci, prędkością transmisji wynoszącą 5 Mb/s. Kontrola wymian bazuje na tych samych modelach co DeviceNet. Zastosowanie jest uniwersalne, od utrzymywania stanu w czasie rzeczywistym po usługi konfiguracyjne i diagnostyczne urządzeń. Ponadto, w ramach CIP zdefiniowana jest jeszcze sieć CompoNet dedykowana dla obsługi prostych urządzeń automatyki, gdzie rozmiar danych użytecznych jest niewielki (pojedyncze bity), cykle relatywnie krótkie (do 1ms) i liczba urządzeń dość duża (maksymalnie dla pojedynczej stacji typu master to 385 węzłów). Prędkości transmisji wynoszą 93,75 kb/s, 1,5 MB/s, 3 Mb/s i 4 Mb/s. Dla zapewnienia determinizmu czasowego wykorzystywany jest w niej model z podziałem czasu (TDMA) [M87], [243]. Przykład ciekawej aplikacji można znaleźć w [200] lub [392].

Dla sieci Ethernet jest przeznaczony protokół EtherNet/IP (CPF2/2) i można go zaklasyfikować jako Class3. Protokół wykorzystuje standardowy Ethernet oraz standardowe protokoły TCP, UDP i IP. Daje również wsparcie dla protokołów aplikacyjnych stosu TCP/IP, jak np. DHCP, FTP czy HTTP. Protokół przeznaczony jest podobnie ja HSE do pracy z funkcjonalnością pojedynczej sieci polowej, jak również jako sieć szkieletowa dla innych rozproszonych sieci systemowych. Model komunikacji przyjęty w EtherNet/IP to Publisher-Subscriber z synchronizacją zegarów na bazie protokołu PTP według IEC61588 [284], [283], [M12].

- Profibus/Profinet

  Grupa protokołów wspieranych przez międzynarodową organizację Profibus & Profinet International (ang. PI, niem. PNO)[89] założoną w 2003 roku.

  Dla sieci polowych zdefiniowano protokoły Profibus-DP (CIP3/1) oraz Profibus-PA (CIP3/2). Protokół DP (ang. Decentralized Peripherals) jest typowym rozwiązaniem sieci polowej, natomiast PA (ang. Process Automation) dla obsługi urządzeń automatyki. Oba rozwiązania bazują na tym samym protokole, ale mają różny standard medium kablowego. DP pracuje na RS485, natomiast PA na specjalnym dwuparowym ekranowanym kablu techniki MBP (ang. Mechester Bus Powered), który w wersji IS może pracować w środowisku zagrożonym wybuchem. Technologia ta jest również wykorzystywana w innych sieciach polowych, jak np. FF. DP może pracować z wieloma prędkościami od 9,6 kb/s do 12 Mb/s, natomiast dla PA jest dedykowana prędkość

[89] http://www.profibus.com

31,25 kb/s. Kontrola ruchu w sieci odbywa się zgodnie z modelem Multimaster-Slave, z aktywacją masterów przez przekazywanie żetonu (ang. token), czyli wirtualnego znacznika uprawnień [104], [M2], [W8]. Aplikacji sieci Profibus jest bardzo wiele. Przykładem może być [374] lub [57].

Dla sieci Ethernet jest przeznaczony protokół Profinet (CPF3/3), przy czym dla funkcjonalności związanych z integracją systemów i komponentów dedykowany jest Profinet CBA, natomiast dla sieci polowych Profinet IO (CPF3/2) działający na warstwie MAC jako Class3 lub 4. Aktualnie CBA nie jest rozwijany. Można spodziewać się, że ze względu na brak rynku dla tego protokołu, przy kolejnej aktualizacji będzie on usunięty ze standardu. Protokół IO wypełnia funkcjonalność sieci polowych, znajduje szerokie zastosowanie i ciągle się rozwija. Wykorzystywany w nim jest standardowy stos Ethernet i TCP/IP oraz standardowy sprzęt z wyjątkiem trybu izochronicznego, który wymaga przełączników specjalizowanych (ang. *ASIC hardware*). Wykorzystywany model dostępu do medium to rodzaj TDMA (ang. *time division multiple access*) z aplikacyjnym modelem wymian Publisher-Subscriber wraz z możliwością przesyłu komunikatów z priorytetami (ang. *prioritised messaging*) [282], [288], [290], [126], [M83].

- P-NET

  Protokół polowy wspierany przez międzynarodową organizację The Interntional P-NET User Organization (IPUO)[90] utworzoną w 1990 roku w Danii.

  P-NET jest to protokół sieci polowych bazujący na magistrali RS485 (CPF4/1) i RS232 (CPF4/2) i modelu wymian typu Multimaster-Slave z przekazywaniem żetonu. Stosowana prędkość transmisji to 76,8 kb/s. Sieci P-NET pracują w topologii pierścienia i znajdują uniwersalne zastosowanie w dziedzinie sieci polowych [184], [356], [357], [261], [270], [M1], [W35].

  Dla sieci Ethernet powstał protokół P-NET on IP (IEC61784 CPF4, IEC61158 Type 4). Można go zaklasyfikować jako Class1. Protokół ten działa dokładnie tak samo jak P-NET, używając datagramów UDP i adresacji IP. Aplikacyjny model wymian to Client – Server [281].

- Interbus

  Protokół dostępny od 1987 roku i wspierany przez międzynarodową organizację Interbus Club[91] założoną w 1992 roku.

  Interbus (CPF6, IEC61158 Type8) jest siecią polową o topologii pierścienia opartą na łączu RS485 pracującym z użyciem modelu wymian typu Master-Slave z rozgłoszeniem

[90] http://www.p-net.org
[91] http://www.interbusclub.com

pojedynczej ramki i powtarzaniem sygnału w każdym węźle sieci. Podobna idea komunikacji stosowana jest dla Ethernetu w sieci EtherCAT. Prędkość transmisji to 500 kb/s lub 2 Mb/s. Interbus może być stosowany w systemach bezpieczeństwa [263], [83], [M64].

Dzięki specjalnym modułom interfejsowym możliwa jest integracja sieci Ethernet z segmentami sieci polowej Interbus, celem poszerzenia możliwości funkcjonalnych komunikacji, jednak brak jest dedykowanego protokołu do wykorzystania w sieci Ethernet. Sieć Ethernet jest traktowana jako narzędzie do tworzenia warstwy integrującej środowiska sieciowe Interbus pracujące z ograniczeniami czasowymi. Idea wykorzystania Interbus na sieci Ethernet wspierana jest przez współpracę Interbus Club z organizacją Profibus & Profinet International [303].

- CC-Link

  Grupa protokołów wspieranych przez międzynarodową organizację CC-Link Partner Association[92], utworzoną w 2000 roku. Protokoły te są głównie popularne w Azji.

  CC-Link (CPF8) jest otwartym protokołem polowym na bazie modelu Master-Slave z rozgłoszeniem. Wykorzystywana prędkość to 156 kb/s, 625 kb/s, 2,5 Mb/s, 5 Mb/s i 10 Mb/s. Typowy cykl sieci to 4-16 ms dla maksymalnej liczby węzłów (64) przy 10 Mb/s. [162]. Przykład aplikacji można znaleźć w [341].

  Do pracy z siecią Ethernet powstało rozwiązanie CC-Link IE. Wykorzystywany jest standardowy 1 Gbps Ethernet (802.3z) w połączeniu dupleksowym w pętli, gdzie abonenci są połączeni w fizycznej topologii łańcucha. Sieć może, ale nie musi pracować w trybie czasu rzeczywistego. Protokół bazuje na cyklicznej wymianie obszaru pamięci pomiędzy abonentami oraz na transakcjach acyklicznych do innych zadań. Wymiany danych są realizowane deterministycznie dzięki przekazywaniu żetonu (ang. token-passing) [W14].

- EtherCAT

  Otwarty protokół wspierany przez międzynarodową organizację EtherCAT Technology Group (ECTG)[93], założoną w 2003 roku.

  Protokół dedykowany jest tylko dla sieci Ethernet (CPF12) i można go zaklasyfikować jako Class3 lub 4. Protokół działa na standardowym stosie Ethernet i standardowym sprzęcie warstwy MAC, ale tylko jeśli chodzi o stację Master i warstwę fizyczną interfejsów stacji Slave. Ze względu na specyficzną obsługę pakietów „w locie” i połączenia łańcuchowe point-to-point interfejsy stacji Slave muszą mieć specjalną

[92] http://www.cc-link.org
[93] http://www.ethercat.org

dwuportową konstrukcję i funkcjonalność nadążnego odczytu, kopiowania i zapisywania danych podczas obsługi ramki w interfejsie. Obsługa i integracja TCP/IP jest możliwa przez specjalne bramy (ang. gateway). Wykorzystywany model dostępu bazuje na podziale czasu z pojedynczą ramką rozgłoszeniową. W warstwie aplikacji pracuje Client-Server ze wspólnym obszarem pamięci. Protokół dostarcza bardzo krótkich cykli sieci na poziomie dziesiątek mikrosekund z niewielką wartością niestałości (ang jitter, 1.1.4) [275], [309], [293], [W16].

- Ethernet Powerlink
  Otwarty protokół wspierany przez międzynarodową organizację Ethernet Powerlink Standardization Group (EPSG)[94], założoną w 2003 roku.

  Dedykowany jest tylko dla sieci Ethernet (CPF13). Protokół jest całkowicie oparty na standardowym stosie Ethernet i TCP/IP oraz wykorzystuje standardowe rozwiązania sprzętowe sieci Ethernet. Można go zaklasyfikować jako protokół Class3. Komunikacja odbywa się w trybie RT ze scenariuszem wymian działającym na warstwie MAC. Dane są transmitowane cyklicznie i acyklicznie wg scenariusza przechowywanego przez stację zarządzającą w synchronizowanych cyklach. Dane NRT są obsługiwane z użyciem stosu TCP/IP. Protokół pracuje poprawnie tylko na zamkniętych segmentach wg modelu dostępu do medium z podziałem czasu SCNM (ang. Slot Communication Network Managemant) i modelu aplikacyjnego Publisher-Subscriber dla danych RT i Client-Server dla NRT. Uzyskuje się relatywnie krótkie cykle na poziomie setek mikrosekund i niewielki jitter. Ponadto, EPL stanowi warstwę aplikacji przy wykorzystywaniu protokołu CANOpen w środowisku sieci Ethernet [394], [205], [330], [121].

- Modbus
  Rozwiązania wspierane przez organizację Modbus Organization[95], założoną w 2004 roku.

  Modbus jest najstarszym z przytaczanych protokołów. Pochodzi z 1979 roku i jest bardzo popularnym, choć nieustandaryzowanym, protokołem dedykowanym głównie dla łączy szeregowych (RS422, RS485). Występuje w dwóch odmianach: RTU (transmisja binarna) i ASCII (transmisja znakowa), a wymiana danych bazuje na modelu Master-Slave.

  Dla sieci Ethernet dedykowane są dwa standardowe rozwiązania: Modbus-TCP oraz Modbus-RTPS (CPF15). Można je zaklasyfikować jako Class1. Protokół dla sieci Ethernet działa na standardowym stosie Ethernet, TCP/IP i standardowym sprzęcie. W przypadku Modbus-TCP ramki Modbus są kapsułkowane w ramce Ethernet. Kontrolę

[94] http://www.ethernet-powerlink.org
[95] http://www.modbus.org

wymian prowadzi mechanizm powyżej warstwy transportowej TCP wg modelu Client-Server. Kontrolę dostępu do medium w przypadku Modbus-RTPS sprawuje dodatek RTPS działający za pomocą mechanizmów Publisher-Subscriber oraz Composite State Transfer na bazie protokołu UDP/IP [38], [415], [385].

- Sercos

  Rodzina protokołów Sercos jest wspierana przez międzynarodową organizację Sercos International[96] od 1990 roku. Pierwszy protokół tej rodziny powstał w 1987 roku.

  Protokoły polowe tej rodziny (Sercos I i II) są dedykowane głównie dla kontroli układów napędowych i bazują na światłowodach w topologii pierścieniowej. Wykorzystywany jest model Master-Slave. Uzyskiwane cykle są znacząco krótkie jak na protokoły polowe (od 62 μs do 65 ms). Oferowane prędkości to 2 Mb/s i 4 Mb/s dla Sercos I oraz 2 Mb/s, 4 Mb/s, 8 Mb/s i 16 Mb/s dla Sercos II [287], [M92].

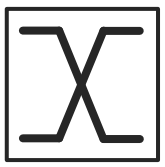
  Sercos III (CPF16) jest protokołem otwartym zaprojektowanym dla sieci Ethernet i dla dziedziny, jaką jest sterowanie napędami. Można go zaklasyfikować jako Class3. Sercos III wymaga wbudowanych przełączników bezpośrednio w każdego abonenta sieci (ang. *embedded switches*), przez co nie da się go użyć na standardowym sprzęcie, ale dzięki temu nie są wymagane inne urządzenia infrastruktury sieciowej (switch, hub). Używana topologia to łańcuch lub pierścień z połączeniami point-to-point, podobnie jak w EtherCAT. Protokół dopuszcza użycie standardowych protokołów działających na Ethernet, np. TCP/IP w kanale NRT. Używany mechanizm dystrybucji danych bazuje na podziale czasu przy transmisji ramek z tzw. telegramami i obsłudze ich „w locie". Protokół wykorzystuje kilka kanałów aplikacyjnych, m.in. Master-Slave dla danych użytecznych. Uzyskuje się krótkie cykle i bardzo niewielki jitter. [323], [157].

3.3.3. Urządzenia infrastruktury sieciowej

Niektóre sieci wymagają dodatkowej infrastruktury [215], [295]. Stosuje się różne urządzenia, które stanowią elementy niezbędne do działania sieci, ale nie stanowią elementów warstwy aplikacji ISP. W ISP można wydzielić kilka typów urządzeń pracujących jako elementy infrastruktury sieci. Poniżej przedstawiono opis podstawowych typów. Pokazane symbole są przykładowe, używane powszechnie, ale jednoznacznych norm na oznaczenie takich elementów jest brak.

[96] http://www.sercos.org

- Przełączniki (ang. switch)

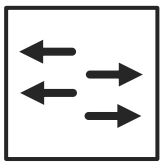


Urządzenia wieloportowe służące do przekierowywania pakietów z jednego portu (wejściowego) na jeden (wyjściowy) na podstawie adresów w warstwie łącza. Przekierowanie może się odbywać na zasadzie:
  - retransmisji buforowanej (ang. store-and-forward), gdzie ramki są kompletowane i składowane w pamięci urządzenia, a następnie przekierowywane do właściwych portów; w tej metodzie ramki są kontrolowane pod względem poprawności składniowej oraz poprawności przenoszonych danych (suma kontrolna),
  - retransmisji bezpośredniej (ang. cut-through), gdzie dane nie są kompletowane i składowane lokalnie, lecz są bezzwłocznie przekierowywane „w locie"; kompletowane są jedynie adresy, aby przekierowanie było możliwe,
  - retransmisji izochronicznej (ang. time-triggered), gdzie wszelkie akcje przekazywania między portami są precyzyjnie synchronizowane sygnałem synchronizacyjnym sieci.

  Najmniejsze opóźnienia uzyskuje się na switchach typu cut-through. Natomiast switche izochroniczne są niezbędne dla pracy z sieciami izochronicznymi, jak np. Profinet czy Sercos.

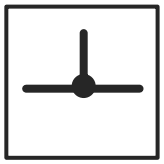
  Przełączniki mogą też wspomagać diagnostykę ruchu przez umożliwienie zwielokrotniania portów (ang. mirroring). Cały ruch na określonym porcie może być wówczas kierowany na inny wybrany, co umożliwia podłączenie węzła diagnostycznego.

- Rozdzielacze (koncentrator, ang. hub)



Rzadko obecnie stosowane urządzenia powielające pakiety z portu wejściowego na wszystkie pozostałe. Tak jak w przypadku przełączników transmisja może być z użyciem bufora lub z natychmiastową retransmisją znaków. Rozdzielacze wzmacniają sygnał, o ile są zasilane. Współcześnie mogą być przydatne do diagnostycznego podglądania ruchu. Połączenia point-to-point utrudniają diagnostykę ruchu, gdyż nie przewidują, z punktu widzenia topologii, dodatkowych przyłączy dla węzłów diagnostycznych. Zastosowanie huba może rozwiązać ten problem, przy ograniczeniach wynikających z opóźnień i ewentualnego lokalnego przetwarzania (np. ignorowania błędnych pakietów).

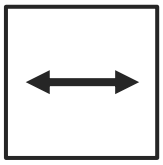
- Przyłącza (ang. tap)



Układy bierne służące do podłączenia urządzenia do konkretnego miejsca w kablu sieciowym. Stosuje się głównie dla magistral. Urządzenie nie wzmacnia sygnału. W układach magistralowych występują ograniczenia

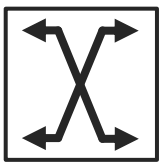
na minimalne i maksymalne odległości między punktami przyłączeniowymi, jak i ograniczenia długości odgałęzienia.

- Repetery (ang. repeter)



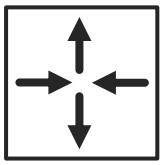
Urządzenie służące do wzmacniania sygnału na medium. Urządzenie przechwytuje sygnały na medium i retransmituje je w powiązanym segmencie. W efekcie odtwarza sygnał i tym samym umożliwia przedłużenie dopuszczalnej, ze względów ograniczeń fizycznych propagacji, długości segmentu. Liczba repeaterów nie może być dowolna ze względu na sumowanie opóźnień wnoszonych przez te urządzenia. Przeważnie w ISP zakłada się istnienie maksymalnie trzech przełączników w segmencie, ale zależy to od danej techniki sieciowej.

- Bramy (ang. gateway)
(łączniki, ang. proxy, link-device)



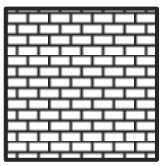
Urządzenia służące do łączenia różnych sieci, czyli do zmiany protokołu i ewentualnie medium. Są to dedykowane komputery dokonujące przetwarzania pakietów i wymagające konfiguracji. W ISP stosuje się je tylko w przypadku konieczności integracji na poziomie sieci (zob. 3.5.3). Wprowadzają one jednak wiele ograniczeń i w systemach z silnymi ograniczeniami czasowymi nie powinny być brane pod uwagę, o ile nie jest to absolutnie konieczne.

- Rutery (trasery, ang. router)



Służą do przekierowywania określonych pakietów do konkretnych sieci. W sieciach przemysłowych nie wykorzystuje się routerów poza przypadkami integracji z sieciami rozległymi oraz wówczas, gdy w ISP wykorzystywane są heterogeniczne środowiska sieciowe. W przypadku routerów dość rzadko rozważa się ograniczenia czasowe. Należy przyjąć, że charakterystyka czasowa ruchu pakietów obsługiwanych przez te urządzenia podlega raczej polityce QoS niż rygorom determinizmu czasowego sieci przemysłowych. Obecnie, z ruterem często integrowane są punkty dostępowe sieci bezprzewodowych (WiFi) oraz funkcjonalności mostów (ang. bridge) dla ruchu między segmentem kablowej sieci lokalnej i sieci bezprzewodowej.

- Ściana ogniowa (ang. firewall)



Ściany ogniowe służą do odfiltrowywania pakietów nieprzeznaczonych dla danego węzła lub dla usług zdefiniowanych jako dozwolone do komunikacji w danym węźle. Urządzenia tego typu są niezwykle ważne

przy integracji systemów ze względu na szeroko rozumiane bezpieczeństwo zarówno dostępu, jak i bezpieczeństwo funkcjonalne.

Przeważnie nie dostrzega się złożoności i ważności problemu kontroli ruchu sieciowego przy przekazywaniu danych z jednego ISP do drugiego, szczególnie gdy medium i urządzenia infrastruktury bazują na tej samej technice sieciowej. W ISP kontrola przepływu danych przez readresacje i eliminację obcych pakietów nie jest wystarczająca. Aby nie zakłócać cyklu pracy sieci deterministycznej, należy zapewnić uwzględnianie transmisji danych spoza danego ISP, zgodnie z cyklem pracy jego sieci. Taką funkcjonalność zapewniają czasem routery i gatewaye [321]. Można też traktować ją jako specyficzną funkcjonalność firewalla [128] lub obowiązkową węzła integracyjnego.

Przy projektowaniu systemu uwzględniającego urządzenia infrastruktury komunikacyjnej należy mieć na uwadze, że stosowanie powyższych elementów, poza pasywnym przyłączem, wprowadza znaczące opóźnienia, dlatego ich liczba w systemie nie może być dowolna.

3.3.4. Koprocesory sieciowe

Koprocesory sieciowe są niezależnymi urządzeniami (modułami sterownika) lub układami wbudowanymi w jednostkę centralną PLC. Działają w celu odciążenia głównego procesora od zadań obsługi sieci, a tym samym uniezależnienia przetwarzania zadań funkcjonalnych węzła od zadań sieciowych. Można spotkać, choć coraz rzadziej, koprocesory swobodnie programowalne. Programista może dowolnie oprogramować taki moduł w oferowanym przez niego zakresie. Dzięki temu można stworzyć koprocesor sieci ze specyficznymi dla danej aplikacji funkcjami i protokołem. Osiąga się to przez oprogramowanie koprocesora obsługą żądanego stosu i przetwarzania danych z użyciem tzw. programów użytkownika (ang. user program) tworzonych w uniwersalnych językach typu Basic lub C.

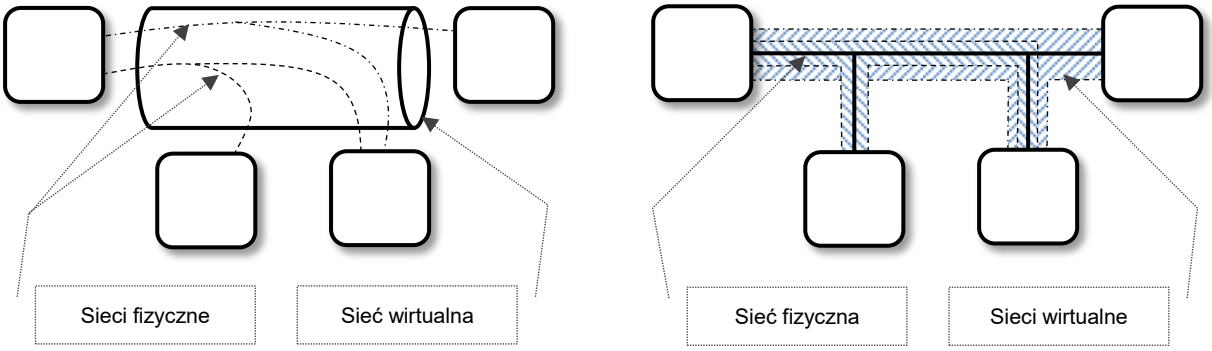
Obecnie, zamiast fizycznych koprocesorów, częściej spotyka się koprocesory wirtualne w postaci specjalnych zadań obsługi sieci uruchamianych w jednostce centralnej. Jest to możliwe, gdy system operacyjny jest na tyle zaawansowany, aby deterministycznie szeregować procesy w czasie, a procesor jest w stanie je przetworzyć z wymaganymi ograniczeniami czasowymi.

Gdy istnieje potrzeba bezpośredniego dostępu do nieaplikacyjnych usług sieci i protokołów, należy zwracać uwagę na możliwości współpracy danego koprocesora z kodem użytkownika. Na rynku istnieje wiele urządzeń, które nie umożliwiają wykorzystania dowolnego protokołu (czy też usługi) z zaimplementowanego stosu z poziomu programów aplikacyjnych. Dla przykładu, w sterownik może być wbudowana obsługa TCP/IP wraz z kilkoma

protokołami aplikacyjnymi RTE. Programista ma dostęp do usług protokołów RTE, ale nie może skorzystać bezpośrednio np. z transportu TCP.

3.3.5. Wirtualizacja środowiska sieciowego

Jak wspomniano w 3.2.11, technologie wirtualizacji można, warto i czasami należy stosować w ISP. Wirtualizacji mogą zostać poddane nie tylko zasoby sprzętowe węzłów, ale i sieci, a dokładnie protokoły sieciowe. Aktualnie podejścia są dwa (rys. 121).



Rys. 121. Sposoby wirtualizacji sieci przemysłowych
Fig. 121. Ways of virtualization of industrial networks

Jedno polega na wykorzystaniu wielu rzeczywistych sieci do zbudowania dynamicznego wirtualnego kanału komunikacyjnego. Drugie polega na wykorzystaniu jednej sieci rzeczywistej do przenoszenia pakietów pochodzących od wielu wirtualnych sieci. Praktyczne zastosowanie obu nie jest zbyt duże. Przemysłowe sieci wirtualne są klasyfikowane jako sieci przemysłowe 3G (por. 2.4.4) i w powszechnym użyciu jeszcze niepopularne. Natomiast ich potencjał jest wart uwagi, szczególnie biorąc pod uwagę cechę dostępności i trwałości transmisji, a także możliwości pracy w środowiskach heterogenicznych.

Przypadek z wirtualizacją kanału dla dwóch sieci jest pokazany np. w [123]. Rozwiązanie takie jest dobre, gdy istnieje warstwa protokołu dynamicznie zarządzająca alokacją zasobów fizycznych (ruchem, obciążeniem, usługami itp.). W przeciwnym razie jest to tylko specyficzny rodzaj redundancji. Natomiast zalety, jakie przynosi, wynikają z zalet redundantnych łączy oraz z możliwości dopasowywania funkcjonowania kanału do wymagań aplikacji, przez co daje możliwość budowania systemów ze zmienną charakterystyką ruchu sieciowego, dostosowującą się do bieżących wymagań aplikacji. Dynamiczne zarządzanie ruchem może również umożliwić tworzenie systemów bardzo dynamicznych w swej strukturze logicznej.

Przypadek drugi nie ma nic wspólnego ze zwielokrotnianiem sieci fizycznych. Wręcz przeciwnie. Wykorzystuje się spójną infrastrukturę jednej sieci, najlepiej o wysokiej niezawodności i przepustowości. Ogólna idea działania polega na kapsułkowaniu ramek różnych

protokołów w pakietach sieci bazowej i zgodnie z regułami transmisji tej sieci. Uzyskuje się tym samym możliwość przekazywania ramek dowolnych protokołów z użyciem jednej sieci nadrzędnej. Warunkiem powodzenia jest, aby sieć fizyczna oferowała wystarczająco dużą przepustowość. Jest to niezbędne, aby udało się wykonać transakcje sieci wirtualnych bez zachwiania ich ograniczeń czasowych. Typowym rozwiązaniem jest wykorzystanie sieci RTE do wirtualizacji sieci polowych. Dość znanym rozwiązaniem jest technika VAN (ang. virtual automation networks) powstała w ramach projektu badawczego UE [406]. Umożliwia ona pracę wielu urządzeń z różnymi interfejsami sieci polowych przez scalenie transmisji w ramach sieci fizycznej opartej na Ethernecie.

3.4. Elementy bierne

Dobierając technologie i elementy systemu, należy mieć na uwadze cechy wynikające ze specyfikacji wymagań danego obiektu i ogólnej wiedzy inżynierskiej. Jednak przy wyborze elementów systemów informatycznych łatwo zapomnieć o ich „przyziemnych" cechach nie-informatycznych. Dlatego należy zawsze rozpatrzeć nie tylko parametry informatyczno-elektryczne, ale i mechaniczno-środowiskowe. W praktyce można doświadczyć bardzo boleśnie sytuacji, gdy „superzestaw", zaprojektowany, dobrany, oprogramowany, i doskonale działający w laboratorium, np. nie mieści się w szafie lub ma obudowę niespełniającą wymagań środowiska. Dlatego na etapie projektowania, przy doborze elementów, należy brać pod uwagę również i takie cechy, które nie są bezpośrednio związane z wymaganiami funkcjonalnymi ISP. Są to:

- wymiary – elementy muszą zostać zamontowane w stosownych miejscach względem wymagań użytkownika, technologii i BHP,
- warunki pracy – środowisko pracy wymaga określonych odporności (por. 1.1.2),
- zasilanie – analiza dostępności określonej sieci zasilającej oraz jej parametrów i stabilności,
- temperatura – zapewnienie możliwości odprowadzenia ciepła i dogrzania obiektu w różnych porach roku (wymiany z otoczeniem lub urządzenia chłodzące i grzewcze),
- dostępność – zapewnienie dostępu serwisowego i utrzymania ruchu, również w sytuacjach awaryjnych związanych zarówno z systemem, jak i procesem,
- oznakowanie – zapewnienie możliwości oznaczenia elementów oraz swobodnego odczytu oznaczeń (szybka, jednoznaczna identyfikacja),
- kolory – dobór kolorów zgodnych z ustandaryzowaną lub powszechnie stosowaną dla danych rozwiązań kodyfikacją.

Ponadto, istotny jest dobór elementów wchodzących w bezpośrednią interakcję z działaniem urządzeń aktywnych i mających wpływ na to funkcjonowanie, takich jak media sieciowe i okablowanie.

3.4.1. Media sieciowe

(medium, nośnik fizyczny, ang. network media)

W praktyce, dla sieci przemysłowych stosuje się trzy rodzaje mediów transmisyjnych. Są to przewody dla sieci przewodowych, światłowody dla sieci optycznych oraz fale elektromagnetyczne dla sieci bezprzewodowych [63], [64]. O medium bezprzewodowym mówi się czasem jako o tzw. „eterze", co jest historyczną zaszłością (XIX w.), i jest równie niepoprawne jak twierdzenie, że medium bezprzewodowym jest powietrze. Fale elektromagnetyczne rozchodzą się w dowolnych ośrodkach umożliwiających ich propagacje. Po szczegóły w tej kwestii warto zajrzeć do literatury, dotyczącej opisu natury zjawisk fizycznych, w tym teorii względności i transformacji Lorentza [84], [308].

W aplikacjach przemysłowych najczęściej stosuje się przewody w formie specjalnych kabli na bazie wspomnianej wyżej skrętki miedzianej, światłowodów jedno- lub wielomodowych oraz fal radiowych [295]. Trasy kablowe, ich struktura, umiejscowienie urządzeń infrastruktury sieciowej, przyłącza, anteny i wszelkie cechy elektryczno-mechaniczne zawsze powinny być przedmiotem projektu i wynikać z dogłębnych przemyśleń, pomiarów, a nawet badań.

- radio

  Fale radiowe nie są jedynymi falami, jakie można wykorzystać do transmisji bezprzewodowych. Istnieją rozwiązania korzystające z podczerwieni, światła widzialnego i dźwięku, niemniej jednak współcześnie dla ISP praktyczne znaczenie mają tylko fale radiowe. Najczęściej używa się częstotliwości ISM (ang. Industrial, Scientific, and Medical), częstotliwości sieci komórkowych (GSM[97]) oraz częstotliwości dedykowanych (pozwolenia UKE[98]). W zakresach ISM pracują sieci w popularnych technologiach IEEE802.11 (WiFi, WLAN) oraz IEEE802.15 (WPAN). Pasmo ISM może być wykorzystywane bez licencji i pozwoleń, jednak dopuszczalne zakresy jego częstotliwości oraz emitowanej mocy różnią się w zależności od regionu świata. Podobnie użycie pasm licencjonowanych, np. na potrzeby budowy radiolinii modemowych dużej mocy, jest regulowane lokalnie (UKE). Sieci GSM zyskują na popularności w ISP, szczególnie w dziedzinie zastosowań, jaką jest zdalne

[97] Global System for Mobile Communications.
[98] Urząd Komunikacji Elektronicznej.

monitorowanie obiektów [3]. Sieci komórkowe stanowią rozwiązanie wygodne z racji niemal globalnego pokrycia, a ich użycie jest relatywnie tanie.

Stosowanie połączeń bezprzewodowych jest wygodne, ale obarczone wieloma ograniczeniami. Z racji charakteru nośnika nie sposób zapewnić wyłączności dostępu i detekcji kolizji [242]. Występuje zatem problem zarządzania i kontrolowania dostępu do medium. Ponadto, dostęp do częstotliwości pasma jest limitowany, urządzenia osób trzecich będące w zasięgu mają swobodny dostęp do transmisji oraz fale radiowe są podatne na zakłócenia. Istnieją rozwiązania, w których starano się wyeliminować kolizyjność transmisji przez ich separacje na różnych kanałach lub przez rozpraszanie widma, jednak nie rozwiązują one wszystkich problemów z kolizyjnością i zaburzeniami [272], [20], [408]. Mimo to, większość znaczących rozwiązań sieci przemysłowych daje wsparcie dla połączeń bezprzewodowych [203]. Niestety, realizacja komunikacji z twardymi ograniczeniami czasowymi jest praktycznie niemożliwa.

Medium bezprzewodowe może być również stosowane jako element środowiska wielosieciowego [123]. Dzięki wykorzystaniu wielu różnych mediów można zwiększyć niezawodność względem odporności na zaburzenia EMC. Podatność na zaburzenia wynika ze stosowanego rodzaju nośnika oraz z parametrów stosowanego sygnału (np. częstotliwość, kodowanie itp.). Dlatego, jeśli np. sieć oparta na Ethernecie zostanie zaburzona przez odziaływania przewodzone indukowane w przewodzie, to te same zaburzenia mogą nie wpłynąć na sieć bezprzewodową. Tego rodzaju zabezpieczenie może być stosowane nie tylko dla medium radiowego, ale i dla kanałów multisieciowych opartych na różnych sieciach kablowych.

Więcej na te tematy można znaleźć w [272], [56], [340], [312], [388], [345], [11].

- kable

  W ISP kable miedziane są najpopularniejszym rozwiązaniem. Znajdują one zastosowanie w większości aplikacji, są proste w montażu i tanie. Ograniczenia ich stosowania wynikają z parametrów elektrycznych linii transmisyjnej [27], możliwości oddziaływania zaburzeń i podatności na wystąpienia zakłóceń transmisji, znaczącej tłumienności sygnału oraz innych parametrów powiązanych z powyższymi [146].

  Zależnie od wymagań warstwy fizycznej danego protokołu kabel musi spełniać określone wymagania. Do najważniejszych parametrów elektrycznych należą impedancja i czas propagacji, do konstrukcyjnych ekranowanie, temperatura pracy, średnica, rodzaj przewodu, rodzaj izolacji itp. Sieci polowe z racji niskich prędkości transmisji (od kilku kb do kilku Mb) mogą przeważnie pracować na zwykłych przewodach typu skręcona para przeznaczonych do transmisji cyfrowej. Rozwiązania klasy RTE, wymagają kabli Ethernetowych o oznaczeniu TX. Dla uproszczenia doboru stosuje się ich kategoryzacje.

Dla standardu 100Base-TX, która jest najczęściej stosowana w rozwiązaniach RTE wystarczającą kategorią kabla jest 5e w wersji ekranowanej (STP). W środowisku przemysłowym występują jednak zagrożenia dla mediów transmisyjnych w postaci wpływu środowiska, uszkodzeń mechanicznych oraz zaburzeń elektromagnetycznych (EMI). Dlatego dla wielu zastosowań wskazane jest używanie kabli 5e o specjalnej konstrukcji przeznaczonej dla danej technologii. Dotyczy to:

- budowy ekranu,
- średnicy kabla,
- rodzaju rdzenia,
- średnicy rdzenia,
- koloru zewnętrznej izolacji.

Istnieją klasyfikatory kabli Ethernetowych. Wygodna, bo stosowana międzynarodowo, jest klasyfikacja AWG (ang. American Wire Gauge). Kod AWG określa parametry kabla i pośrednio jego przeznaczenie. Przykłady przedstawiono w tabeli 5.

Tabela 5

Przykładowe parametry kabli Ethernetu przemysłowego

Kod AWG	*Przekrój rdzenia*	*Średnica rdzenia*	*Rezystancja statyczna*	*Przeznaczenie*
25	0,162 mm^2	0,455 mm	106 Ω/km	…
24	0,205 mm^2	0,511 mm	84,2 Ω/km	Patch kable kat. 5
23	0,259 mm^2	0,574 mm	66,6 Ω/km	Kable instalacyjne kat. 5
22	0,324 mm^2	0,643 mm	53,2 Ω/km	Ethernet przemysłowy
21	0,411 mm^2	0,724 mm	41,9 Ω/km	…

Kabel AWG22 dla Ethernetu przemysłowego może mieć kilka wersji, np. względem konstrukcji żyły, a konkretnie liczby nici w żyle (i ich przekroju):

AWG22/1 – kabel montażowy stacjonarny – powinien pozostać nieruchomy po instalacji (tzw. drut),

AWG22/7 – kabel przyłączeniowy giętki – względna odporność na okazjonalny ruch i na wibracje (tzw. linka),

AWG22/19 – kabel mocno giętki odporny na permanentny ruch i skręcanie, nadający się do podłączanie urządzeń ruchomych (tzw. giętka linka).

Dobrym zabezpieczeniem przed uszkodzeniami mechanicznymi jest zastosowanie redundantnych tras kablowych, w tym topologii pierścieniowych [399]. Zwielokrotnianie tras w sieciach lokalnych może być realizowane przez natywne mechanizmy urządzeń

i protokołów przemysłowych (np. HSB [M91], RNA [M3], PRP [M86], HSR [M50], [202] itp.) lub przez standardy ogólne (np. IEEE 802.1D). W celu eliminacji zakłóceń związanych z EMC należy stosować środki ograniczające zaburzenia. Mogą to być np. ekranowane przewody i metalowe koryta. Wskazane jest unikanie zbędnie długich połączeń, czyli tworzenie możliwie krótkich tras kablowych, ich ułożenie z dala od źródeł zaburzeń itp. W skrajnych przypadkach należy zamienić medium miedziane na światłowody.

Przy doborze sieci i medium istotne jest również rozważanie dotyczące minimalnych odległości między węzłami, maksymalnej liczby węzłów w segmencie, zasad dotyczących terminacji linii (terminatory), zasad rozgałęzień (ang. T-branch) oraz liczby dopuszczalnych zwielokrotnień segmentów (powielonych segmentów przez repetery itp.). Wszystkie te cechy są zawsze zdefiniowane dla danej sieci i danego kabla, a wynikają z ograniczeń elektrycznych tego kabla względem zastosowanej częstotliwości i innych parametrów sygnału elektrycznego.

- kolory
  Często organizacje wspomagające dany standard wspierają również stosowanie konkretnej kolorystyki kabli i przewodów oraz znaczenia (sposobów łączenia) przewodów o określonym kolorze. Dlatego np. dla sieci Profinet czy EtherCAT spotyka się kable zielone lub żółte, a dla Profibus fioletowe. Istnieją normy opisujące zagadnienia doboru i montażu kabli, np. IEC61918 [M45], IEC61935 [M46], IEC11801 [M68].

- światłowody
  Światłowody (ang. fiber optic) są kablami umożliwiającymi przesyłanie wiązki światła. W oznaczeniach kabli Ethernetu występuje jako FX. Nośnikiem sygnału jest włókno wykonane ze szklanego lub plastikowego rdzenia i polimerowego pokrycia. Plastikowe konstrukcje są tańsze od szklanych, ale oferują mniejszy zasięg transmisji. Dla ISP jest on jednak przeważnie wystarczający. Wynosi on 50 m dla plastikowego rdzenia (ang. POF, Plastic Optical Fiber) i 100 m dla szklanego rdzenia i plastikowego płaszcza (ang. HCF, Hard-Cladded silica Fiber, H-PCF, Hard-Plastic-Cladded Fiber).

  Kable światłowodowe stosuje się tam, gdzie występują istotne zaburzenia przewodzone lub indukowane, lub tam gdzie odległość jest zbyt duża, aby stosować kable miedziane. Ich ograniczenia to głównie wrażliwość mechaniczna oraz wyższy koszt montażu i ograniczona dostępność urządzeń względem rozwiązań dla kabli miedzianych.

W sieciach przemysłowych ISP stosuje się przewody światłowodowe typu:

- jednomodowe – włókno przenosi jedną wiązkę światła (promień, mod). Zaletą światłowodów jednomodowych jest duża odległość transmisji wynoszącą powyżej 10 km,
- wielomodowe – do włókna wpuszczanych jest wiele wiązek światła (promieni, modów) pod różnym kątem. Oferuje szersze pasmo, ale mniejszą odległość transmisji (do 2 km).

W ISP stosuje się kable z wieloma rdzeniami, z których z założenia wykorzystuje się tylko część. Wybór takich kabli daje większe bezpieczeństwo instalacji, gdyż w razie uszkodzenia włókna można przejść na inne, zapasowe. Ponadto, daje to szanse na łatwą rozbudowę systemu.

- konwersja medium

  W praktyce często stosuje się konwertery mediów. Sprawa jest prosta, gdy z racji konwersji nie następują istotne opóźniania zarówno w konwerterach, jak i z powodu użycia innych mediów niż oczekiwane przez urządzenia. Jednak przy drastycznych przejściach, np. z Ethernetu na radio, należy rozważać istotne opóźnienia. Może okazać się, że protokół Modbus doskonale działający na kablu przy użyciu w kanale radiowym nie będzie dobrze pracował na tej samej konfiguracji węzłów z powodu timeoutów.

  Konwertery medium nie konwertują protokołów, zatem sam konwerter działa jak repeter i nie wystarczy do połączenia urządzeń wykorzystujących inny protokół na innym medium. Dla przykładu, przejście z RS485 na skrętce na Ethernet wymaga konwersji przynajmniej adresów w warstwie łącza. Dlatego konwerter medium jest tylko jednym z elementów niezbędnych dla zapewnienia takiego przejścia. Na rynku spotyka się zintegrowane urządzenia typu gateway czy proxy konwertujące zarówno protokoły, jak i media. Jednak przy projektowaniu systemu z wykorzystaniem takiego urządzenia należy zawsze dokonać analizy czasowej przejścia danych użytecznych.

3.4.2. Przewody i kable sygnałowe

(kabel sygnałowy/sterowniczy, ang. signal/analog wire/cable)

Do transmisji sygnałów analogowych (zob. 1.2.4) z obiektu do układów przetworników IO stosuje się przewody w formie różnego rodzaju kabli sygnałowych. Kable sygnałowe nie są elementami sieci informatycznych. Jednak to z nich buduje się kanały analogowe, którymi informacja jest dostarczana do ISP. Jeśli zastosowany nie spełni wymagań elektrycznych i konstrukcyjnych, to dostarczony sygnał będzie zakłócony lub wcale nie dotrze, a co za tym idzie informacja będzie przekłamana. Każdy informatyk wie, co oznacza przetwarzanie błędnej informacji. Powinno się zatem stosować przewody odpowiednie do transmitowanych

sygnałów pod względem elektrycznym i konstrukcyjnym zarówno mechanicznym, jak i odporności na wpływ środowiska, w którym pracuje.

Charakterystyka elektryczna kabla jest dość złożona i obejmuje cały szereg różnych wartości fizycznych. Do najważniejszych należą charakterystyka EMC, odporność napięciowa, pojemność właściwa, rezystancja, impedancja, tłumienie, i wiele innych. Dobór powinien być dokonany do rodzaju sygnału.

Parametry mechaniczne dotyczą wymiarów, konstrukcji wewnętrznej, kolorów itp. Istotna jest konstrukcja rdzenia. Przewody jednodrutowe (jedna nić, ang. solid core) są sztywne i nadają się do montażu permanentnego i ewentualnie kilkakrotnych korekt położenia. Zbyt częste wyginanie kabla może spowodować jego uszkodzenie. Ponadto sztywny kabel jest niewygodny do manipulowania. Kable wielodrutowe (wiele nici w rdzeniu, ang. stranded core) zwane linkami są giętkie i można nimi swobodnie manipulować. Odznaczają się lepszą przewodnością i większą odpornością mechaniczną. Są jednak droższe od kabli sztywnych. Kolejną ważną cechą konstrukcyjną kabla sygnałowego jest splecenie żył i ich odseparowanie. Przepływ prądu w przewodzie generuje pole elektromagnetyczne, które rozchodzi się w postaci fal elektromagnetycznych. Splecione pary przewodów (tzw. skrętka) zapewnia redukcję EMI i przesłuchów między parami. Dodatkowo, można wykorzystać kable z ekranowaniem, w tym z osobnym ekranowaniem każdej z par. Dobór parametrów mechanicznych powinien być dokonany względem konkretnego obiektu.

Użycie kolorów jest objęte pewną dowolnością, jednak warto stosować się do norm i lokalnych regulacji. Przykładem może być DIN47100 (dot. kolorów kabli, wycofana choć ciągle stosowana), EN60445 (dot. oznaczania i identyfikacji przy współdziałaniu człowieka z maszyną), IEC11801 (dot. okablowania w budynkach), EN50173 (dot. okablowania strukturalnego). Kabel żółto-zielony jest zawsze stosowany do uziemiania.

Odporności dotyczą możliwości pracy kabla w wodzie, wilgoci, wysokiej lub niskiej temperaturze, środowisku agresywnym chemicznie, narażeniu na promieniowanie UV, w strefach zagrożonych wybuchem itp. Dotyczy to zwykle konstrukcji płaszcza i wszelkich „zbrojeń”. Dobór odporności powinien być dokonany względem wymagań środowiska.

3.4.3. Zagadnienia ekranowania

Tak jak wspomniano w 1.1.2, działanie elementów ISP może się sprzęgać z działaniem elementów obcych przez oddziaływania indukcyjne (pola magnetyczne) i pojemnościowe (pola elektryczne, elektrostatyczne). Podstawową zasadą dobrego okablowania jest dobór właściwych tras kablowych, minimalizujących możliwość oddziaływań pól, i tym samym wystąpienia zaburzeń transmitowanych sygnałów. Ponadto, w celu eliminacji obcego wpły-

wu na przesyłanie sygnału elektrycznego bardzo ważne jest stosowanie ekranowania przewodów.

Przy korzystaniu z sygnałów dyskretnych nie ma potrzeby przejmowania się ekranowaniem przewodów. Oddziaływanie sygnału dyskretnego na układ wejścia ma charakter ilościowy, a nie jakościowy, zatem zakłócenie sygnału ma niewielki wpływ na niesioną informację i stąd jej zakłócenie jest mało prawdopodobne. Warto w tym przypadku rozważać problem indukowanych przepięć, które mogą prowadzić do uszkodzenia układu wejścia. Jednak w zdecydowanej większości konstrukcji układy wejścia są izolowane galwanicznie, co w znacznym zakresie eliminuje problem uszkodzeń z powodu przepięć. Ekranowanie ma natomiast duże znaczenie przy elektrycznych sygnałach cyfrowych i analogowych. Nieekranowane przewody elektryczne w środowisku przemysłowym są w tym przypadku źródłem problemów. Również użycie kabla ekranowanego bez właściwego podłączenia ekranu oraz użycie niewłaściwych wtyczek i gniazd skutkuje podobnymi problemami jak użycie kabla bez ekranu.

Problem ekranowania dotyczy zatem tylko mediów przewodowych dla sieci komputerowych oraz przewodów sygnałowych transmitujących analogowo wartości zmiennych (por. 1.2.4). Ekrany przewodów sieciowych są konstruowane jako płaszcz na przewodzie (lub przewodach), przeważnie w formie plecionki miedzianej i/lub folii aluminiowej. Ekran jest w stanie wytłumić zarówno pole elektryczne, jak i elektromagnetyczne wysokiej częstotliwości pochodzące ze środowiska przemysłowego. Ekran utrzymywany na stałym potencjale, najlepiej ziemi lub wspólnej masy, chroni przed możliwością wyindukowania niechcianego przepływu prądu w przewodzie znajdującym się w oddziaływaniu pola elektrycznego. Natomiast tłumienie pola elektromagnetycznego jest możliwe dzięki efektowi przewodzenia naskórkowego (ang. skin effect). Powoduje on, że gęstość wyindukowanego zmiennego prądu elektrycznego będzie większa przy powierzchni kabla wystawionego na takie oddziaływanie niż w jego wnętrzu. Powstające na powierzchni prądy wirowe generują pole magnetyczne chroniące wnętrze przewodu przed polem zewnętrznym. Szczeliny w ekranie, o ile występują, muszą być jednak mniejsze od długości fali zaburzającej. Głębokość wnikania pola w przewodnik maleje wraz ze wzrostem częstotliwości i dla kabla miedzianego i częstotliwości 1 MHz wynosi poniżej 0,07 mm, a dla 1 GHz poniżej 2,2 μm. Wynika stąd, że dla tłumienia zaburzeń elektromagnetycznych ekrany lepiej się sprawdzają przy wysokiej częstotliwości fal zaburzających. Aby eliminować oddziaływania elektrostatyczne oraz elektromagnetyczne w kablach Ethernetu przemysłowego, stosuje się ekran w postaci oplotu i folii.

Dla sygnałów cyfrowych relatywnie niskiej częstotliwości wskazane są kable ekranowane z ekranem podłączonym do masy w jednym punkcie segmentu. Zatem ekran w sieciach po-

lowych należy podłączać w jednym punkcie, najlepiej do uziemienia szafy sterowniczej lub do masy sterownika (zob. rys. 122 (a)). Umożliwia to spływanie wyindukowanych prądów do jednego punktu bez wzajemnego ich oddziaływania. Łączenie ekranu w wielu punktach może powodować przepływ prądu w ekranie (tzw. pętle masy) z racji zjawiska wyrównywania potencjałów zasilania. W warunkach przemysłowych zdarza się, że urządzenia są zasilane z różnych sieci, brak jest masy systemowej lub pewnego dostępu do odniesienia stałopotencjałowego (np. uziemienia). W efekcie brak jest punktu wspólnego w obwodach zasilających i potencjały mas w węzłach ISP mogą się różnić. Zjawisko to jest bardzo niekorzystne dla poprawnego przepływu sygnału w przewodzie [264], [53].

Podobnie dla sygnałów pomiarowych wysoce wskazane są przewody ekranowane z ekranem podłączonym do masy w jednym punkcie. Sygnały analogowe należy chronić przed wyindukowaniem zaburzeń. Ekran podłączony w jednym punkcie umożliwia odprowadzanie zaburzeń indukowanych do masy, zanim spowodują powstanie zakłóceń w transmitowanym sygnale. Należy również zwrócić uwagę na ekranowanie wtyczek. Wszelkie złącza są elementem kanału transmisyjnego i w ich wnętrzu fizyka nie przestaje działać.

Dlatego dla przypadku wysokich częstotliwości transmitowanego sygnału ekran należy podłączać do masy z obu końców każdego kabla i w każdym punkcie przyłączeniowym w każdym segmencie.

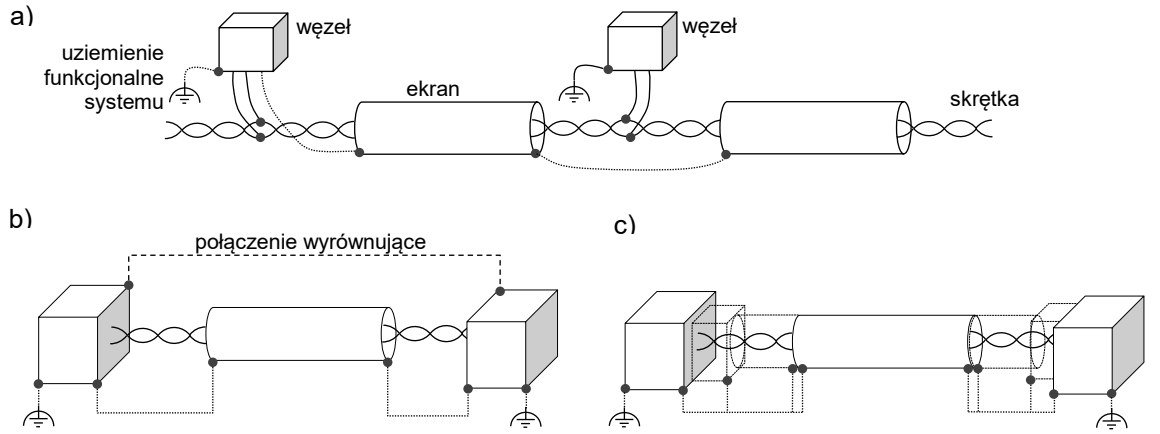
Gdy odpowiednio długi przewodnik zostaje podłączony do masy na swoim końcu, otrzymujemy niezamierzony efekt anteny, która zarówno zbiera sygnały z otoczenia, jak i może takie sygnały emitować. Szczególnie problem jest istotny przy wysokich częstotliwościach sygnałów, emitując niepożądane zaburzenia w postaci fal elektromagnetycznych i zbierając zakłócenia z fal otaczających. W efekcie ekran stanowi barierę z zerową tłumiennością (ang. attenuation) dla zaburzeń. Można nawet stwierdzić, że dla Ethernetu przemysłowego lepiej jakby ekranu nie było wcale, niż miałby być uziemiony tylko na jednym końcu.

Podobny problem objawi się przy przejściu kabla pomiędzy strefami (ang. zones), np. z tzw. pola (ang. field) do wnętrza szafy sterowniczej. Wówczas efekt anteny zniszczy strefę chronioną szafy. Aby temu zapobiec, należy drugi punkt uziemienia ekranu zapewnić przynajmniej przy wejściu do szafy. Bardzo użyteczny rysunek ilustrujący to zjawisko można znaleźć na rysunku 11 w [M20].

Kolejny powiązany problem to potencjalnie nieekranowane fragmenty kabla przy jego końcach. Taki, nawet bardzo krótki fragment staje się źródłem potencjalnych zaburzeń, w tym również nieekranowane gniazda i wtyki.

Zatem w przypadku sieci RTE, aby ograniczyć wpływ pola, stosuje się pary splecione i ekran łączony do masy węzłów na obu końcach przewodu, czyli przy każdym podłączonym

urządzeniu aż do samego wejścia sygnału w obwody elektroniczne urządzenia. Gniazda i wtyki przyłączeniowe powinny być metalizowane, i tym samym stanowić ekran dla końcówek przewodów. Zilustrowano to na rysunku 122 (c), przy czym w praktyce ekran i elementy wtyku, gniazda i węzła nie są łączone przewodami, a ciągłość obwodu ekranu jest zapewniana przez odpowiednie konstrukcje stykowe tych elementów.



Rys. 122. Ilustracja zagadnienia ekranowania
Fig. 122. Illustration of shielding issue

Niestety, problem wyrównywania potencjałów nie znika. Aby zabezpieczyć ekran kabla Ethernetu przemysłowego przed przepływem prądów wyrównawczych, należy zapewnić stały potencjał masy wszystkich węzłów. Można tego dokonać tworząc niezależne, względnie wysoko obciążalne, połączenia wyrównawcze, wyrównujące różnice potencjałów (ang. equipotential bonding) między węzłami sieci (zob. rysunek 122 (b)). Połączenia wyrównawcze powinny być podłączone do uziemienia funkcjonalnego systemu i układane razem z kablem sieciowym. Ponadto, elementem projektu ISP powinien być podsystem zasilania węzłów ze spójnym podsystemem uziemienia funkcjonalnego dla całego ISP. Dzięki niemu przez zapewnienie podłączenia masy każdego urządzenia z masą funkcjonalną (lokalną) systemu (szafy, obudowy, instalacji linii produkcyjnej itp.) można wyeliminować różnice potencjałów pomiędzy masami poszczególnych węzłów. Masa taka musi być uziemiona połączeniem niskorezystancyjnym (<1Ω) w jednym punkcie do masy zakładowej (uziemienia budynku, hali produkcyjnej, ziemi itp.). Tworzy się tym samym połączenia wyrównawcze zapewniające taki sam potencjał w całym systemie (wszystkich jego węzłach). Uziemianie masy systemu do masy zakładowej w wielu punktach jest błędem i nie powinno mieć miejsca. Ewentualne koryta metalowe powinny być zbudowane z tego samego materiału oraz podłączane do uziemienia funkcjonalnego, gdy tylko jest ku temu możliwość. Spójne umaszenie zasilania węzłów oraz połączenia equipotencjałowe zabezpieczają w wystarczającym stopniu przed występowaniem prądów wyrównawczych w ekranach. Każda technologia sieci przemysłowych dostarcza szczegółowych wytycznych dla okablowania, np. [M81], [M82], [M83].

3.5. Integracja

Integracja elementów w ISP może być prowadzona na płaszczyznach abstrakcji wymienionych w 2.4. Systematyzując integrację, można wydzielić dwa podejścia:

- integracja logiczna
  Dotyczy koncepcji integracji systemów rozpatrywanej na poziomie przyjętych modeli przepływu informacji i bez wnikania w strukturę wewnętrzną integrowanych systemów. Wykorzystywane są wbudowane mechanizmy interfejsowe ISP (por. 1.1.5, 1.1.7, 2.4.2) i jego wektor zewnętrzny (por. 2.4.6). Do realizacji takiej integracji często niezbędne jest przeprowadzenie również integracji fizycznej, najczęściej na poziomie sieci.
- integracja fizyczna
  Dotyczy aspektów technicznych połączenia elementów systemu. Wchodzą w to zagadnienia przetwarzania, reprezentacji i kodowania, składowania oraz przesyłania informacji. Najczęściej wektory informacyjne węzłów nie ulegają zmianie w sensie struktury i typów zmiennych. Następuje jedynie ich modyfikacja względem liczby zmiennych i powiązań. Dlatego najistotniejszym zagadnieniem przy integracji fizycznej jest integracja mechanizmów przekazywania informacji między węzłami, czyli sieci.

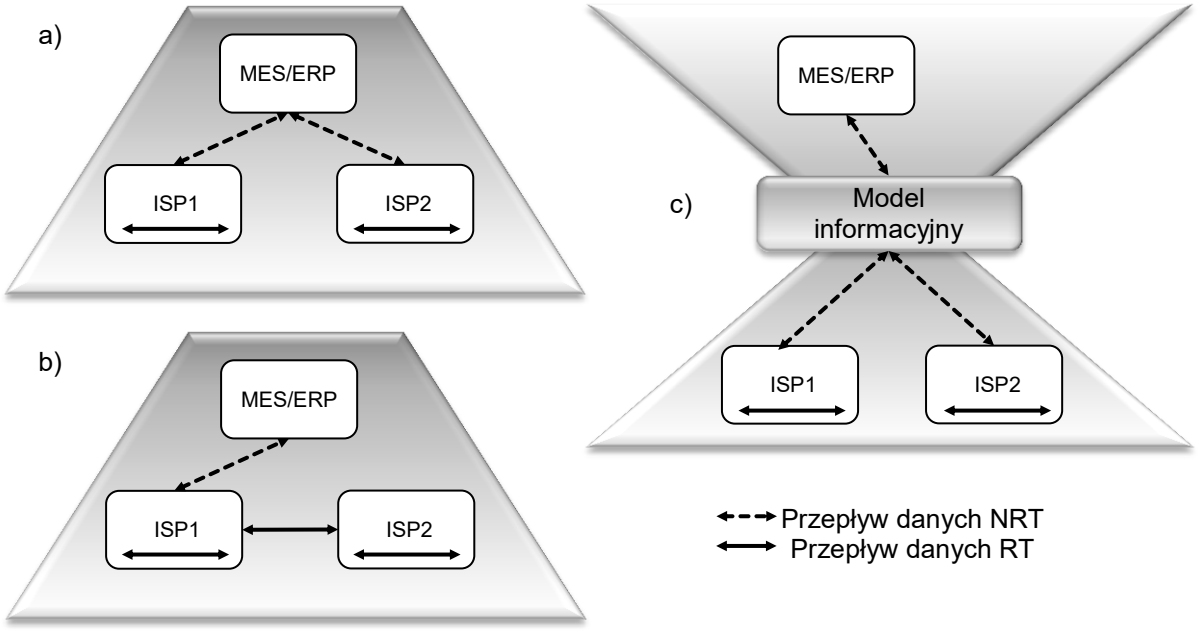
Biorąc po uwagę powyższy podział i względy praktyczne (techniczne), proces integracji opisuje się najczęściej względem systemów, aplikacji i sieci. Integracja na poziomie systemów dotyczy łączenia przepływu informacji i ewentualnego przetwarzania na rzecz drugiego systemu. Systemy mają oddzielny zakres funkcjonalny, ale istnieje podzbiór informacji wprowadzanej i wyprowadzanej, która jest zależna. W przypadku integracji na poziomie aplikacji integracja zachodzi między systemami, które po zintegrowaniu uzyskują nowy, poszerzony zakres funkcjonalny. Natomiast integracja na poziomie sieci dotyczy rozbudowy czy też modyfikacji systemu o węzły (wszelkie elementy z interfejsami sieciowymi), które muszą zostać włączone do istniejącego obiegu informacji.

Należy zwrócić uwagę, że nie istnieją uniwersalne przepisy na integracje. Sposób integracji wynika z lokalnych wymagań i dostępności sprzętu oraz oprogramowania, a także z istnienia (lub nie) możliwości modyfikacji istniejących rozwiązań. Dla poprawnego przeprowadzenia integracji niezbędna jest kompleksowa wiedza umożliwiająca skorzystanie z dostępnych środków i technologii informatycznych, z zachowaniem cech i wymagań ISP. Integracja jest jednym z kluczowych obszarów badawczych i aplikacyjnych [158].

3.5.1. Integracja systemów

W zakładach przemysłowych na poziomie ISP (zob. rysunek 14) może pracować wiele systemów. Często takie ISP są niezależne funkcjonalnie. Każdy jest dedykowany do realiza-

cji pewnych zadań i na poziomie wymiany informacji służącej do obsługi danego procesu mogą działać samodzielnie. Jednak z punktu widzenia informatycznego istnieją parametry (zmienne), które pochodzą spoza takiego systemu. Jest to informacja niezwiązana bezpośrednio z procesem produkcji i stanowi przeważnie informację pochodzącą z procesu zarządzania. Zwykle jest ona wprowadzana/wyprowadzana ręcznie przez operatorów, nadzór i służby utrzymania ruchu. Informacje te nie pochodzą jednak z znikąd, lecz są efektem procesów biznesowych lub działania innych ISP.



Rys. 123. Ilustracja zagadnienia integracji systemów
Fig. 123. Illustration of system integration issue

W pierwszym przypadku integracja sprowadza się do współpracy z systemami klasy MES/ERP. Zaletą integracji ISP z systemami zarządzania jest eliminacja potencjalnych błędów w przepływie informacji i oszczędność czasu pracy pracowników. Problemy pojawiające się przy integracji tego typu są w zdecydowanej większości przypadków łatwo rozwiązywalne, gdyż nie dotyczą użycia środków informatycznych, a ich doboru. Systemy MES/ERP pracują w przestrzeni biurowo-serwerowej, gdzie nie występują poważne ograniczenia w dostępie do zasobów, środków i technologii informatycznych. Integracja polega na takim doborze środków sprzętowo-programowych, aby zapewnić niezawodny przepływ danych, ale już niekoniecznie z ograniczeniami czasowymi wymaganymi na poziomie danego ISP. Najczęściej dokonuje się tego przez integracje aplikacji węzłów z użyciem zakładowej sieci lokalnej, bazy danych i jej standardowych interfejsów (np. ODBC, ADO, DDE, COM, OPC itp.) lub interfejsów specjalizowanych, stworzonych na potrzeby danej integracji (np. serwer danych ATD). Przykładem może być np. współpraca systemu MES, systemu wsparcia na-

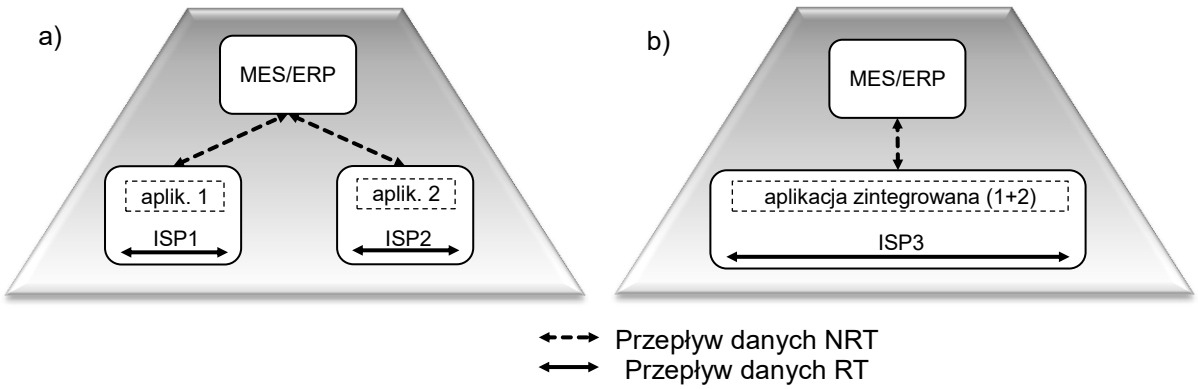
praw i systemu ERP przez interfejs bazy danych (zob. 7.1.2) lub integracje zastosowane w systemach opisanych w załączniku 7.1.

Z punktu widzenia procesu technologicznego, proces może być obsługiwany przez kilka niepowiązanych bezpośrednio ISP. Przepływ informacji między nimi odbywa się przez operatora i zgodnie z lokalnymi procedurami działań dla danego procesu. Jednak, w zależności od technologii procesy produkcyjne mogą osiągnąć większą wydajność lub jakość, gdy informacja będzie krążyć między systemami automatycznie. Jest to drugi przypadek integracji systemów, którego celem jest przekazywanie informacji stanowiącej efekt pracy danego systemu do innego systemu. Przykładem mogą być systemy produkcji wsadowej, gdy efekt pracy jednego jest przekazywany do drugiego wraz z kompletem danych o półprodukcie.

Ilustracja zagadnienia integracji systemowej została przedstawiona na rysunku 123, który jest modyfikacją rysunku 17 i rysunku 14. Dla modelu z centralnym repozytorium (rys. 123 (c)) integracje z założenia odbywają się przez model informacyjny. Bez naruszenia takiej idei integracje poziome są możliwe tylko na poziomie aplikacyjnym.

3.5.2. Integracja aplikacji

W efekcie integracji aplikacyjnej osobne systemy zostają połączone i przekształcone w podsystemy ISP obejmujące działaniem większy obszar terytorialny i szerszy zakres funkcjonalny. Integracja na tym poziomie jest wskazana o ile podsystemy mają współzależne zadania wykonywane w czasie rzeczywistym. Różnica względem integracji typu (b) z rysunku 123 polega na rodzaju przekazywanej informacji. Przy integracji systemów przekazywany jest wektor zewnętrzny a nie wewnętrzne wektory stanu (por. 2.4.6). Dlatego z reguły nie istnieje potrzeba przekazywania takich informacji w trybie HRT.



Rys. 124. Ilustracja zagadnienia integracji aplikacji
Fig. 124. Illustration of application integration issue

Gdy istnieje potrzeba wymiany zmiennych stanowiących elementy przetwarzania na rzecz realizacji zadań funkcjonalnych systemu, wówczas niezbędne jest połączenie systemów na poziomie funkcjonowania ich aplikacji. Polega to na scaleniu warstw aplikacyjnych integrowanych systemów. W proces ten wchodzi głównie ustanowienie nowych powiązań aplikacyjnych i komunikacyjnych, a także modyfikacja istniejących i stworzenie nowych zadań. Zostało to zilustrowane na rysunku 124 (a), na którym pokazano dwa systemy przed integracją i (b) system po integracji.

3.5.3. Integracja sieci

Integracja na poziomie sieci sprowadza się do zapewnienia komunikacji między węzłami z zachowaniem cech systemu rozproszonego, jakie są wymagane dla modyfikowanych lub nowych połączeń aplikacyjnych. Konieczność integracji na tym poziomie występuje przeważnie z powodu rozbudowy systemu o nowe węzły. Stanowi element integracji fizycznej i często występuje podczas integracji logicznej systemów i aplikacji. Może być dokonywana w środowisku sieciowym kompatybilnym lub w środowisku heterogenicznym.

❖ **Środowisko kompatybilne**

Integracja systemów w środowisku kompatybilnym jest przeważanie możliwa i nie przysparza znaczących problemów integratorom. Odbywa się ona na poziomie dowolnej warstwy modelu sieci przemysłowej. W praktyce najczęściej wymagane są zmiany w warstwie aplikacji (powiązania, zob. 2.4.7) i w warstwie fizycznej (media, zob. 3.4.1).

Dołączenie nowych węzłów lub modyfikacja wymian w węzłach istniejących pociąga za sobą konieczność modyfikacji scenariusza wymian. Nowe węzły wymuszają również modyfikację okablowania – najczęściej tworzenia punktów przyłączeniowych, rozbudowy i zmian w istniejącej infrastrukturze, a czasem zmiany struktury okablowania. Dlatego potencjalne problemy są dwa:

- brak wolnego pasma na przesłanie nowych informacji,
- ograniczenia rozbudowy warstwy fizycznej.

W przypadku przewodowych sieci przemysłowych stosuje się techniki wykorzystujące stałą szerokość pasma. Zajętość pasma kontrolowana jest przez scenariusz wymian według predefiniowanego cyklu sieci (zob. 1.1.7). Wyczerpanie pojemności informacyjnej pasma sprowadza się do niemożności rozbudowy cyklu w sposób niezmieniający ograniczeń czasowych transmitowanych danych. Rozwiązaniem jest optymalizacja istniejącego scenariusza pod kątem zmniejszenia rozmiaru przesyłanej informacji. Jeśli nie jest to możliwe, to należy rozważyć modyfikację dynamiczną cyklu, czyli taką, aby zachować

charakterystykę czasową obsługi tych zmiennych, które nie mogą ulec zmianie, i zmodyfikować charakterystykę tych, które na to pozwalają. Zostało to opisane w [23].

Ograniczenia zmian w okablowaniu wynikają z parametrów fizycznych danego medium i użytej techniki sieciowej. Przeważnie sprowadzają się do ograniczeń długości segmentów, liczby punktów przyłączeniowych i zagadnień adresacji oraz do problemów wytyczenia nowych tras kablowych i zmian w strukturze okablowania.

❖ **Środowisko niekompatybilne**

Integracja dwóch lub więcej różnych sieci przemysłowych jest dość złożonym zagadnieniem. Dotyczy zarówno zagadnień wynikających ze składni, semantyki samego protokołu, jak i działania aplikacji, utworzenia powiązań oraz dotrzymania ograniczeń czasowych. Dlatego integracje sieci należy rozpatrywać od strony aplikacji oraz od strony protokołu.

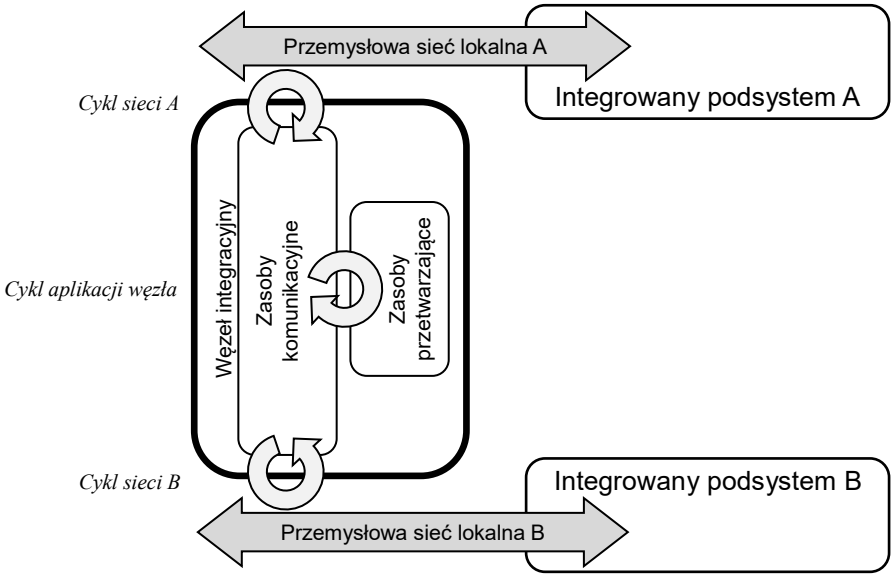
Zakładając, że rozpatrywane dane są reprezentowalne w warstwie aplikacji integrowanych elementów i że nie istnieją problemy z przetwarzaniem danych w tej warstwie, to podczas integracji ISP w niekompatybilnym środowisku sieciowym istnieją trzy zagadnienia, mogące przysporzyć problemów:

- brak wolnego pasma (w co najmniej jednej integrowanej sieci),
- konieczność konwersji protokołów,
- konieczność konwersji sygnałów i medium.

Brak „miejsca” na przesłanie dodatkowych lub zmodyfikowanych informacji stanowi taki sam problem jak opisany powyżej dla przypadku środowiska kompatybilnego, z tą różnicą, że w przypadku integracji różnych sieci zmiany w scenariuszu wymuszane są nie tylko dołączaniem nowych węzłów i związanych z tym związków, ale również dokładaniem zmiennych pochodzących z innego podsystemu komunikacyjnego. Z punktu widzenia opisu informacyjnego sprowadza się to do tego samego, gdyż w obiegu pojawią się nowe zmienne, ale od strony aplikacyjnej zmienne te niekoniecznie stanowią bezpośrednią reprezentację danych dla powiązań aplikacyjnych ustanawianych z nowymi węzłami.

Powiązania aplikacyjne często muszą być tworzone z urządzeniami pośredniczącymi w integracji, np. bramami (ang. gateway), proxy lub routerami. Urządzenia te umożliwiają integracje cykli, co zostało zilustrowane na rysunku 125. Warto tu zauważyć analogie do rysunku 57. W przypadku pośrednictwa cykl sieci i związki aplikacyjne będą wyglądały zupełnie inaczej niż w przypadku pojawienia się takich samych węzłów w środowisku kompatybilnym. W danej sieci związki są ustanawiane z aplikacją i protokołem węzła pośredniczącego. Dla aplikacji w węźle systemu aplikacja pośrednicząca powinna być przeźroczysta, o ile integracja ma być niezauważalna (ang. seamless) dla aplikacji węzłów ISP. Dla danej aplikacji ISP cały zewnętrzny system jest sprowadzany do wektora wejścia i wyjścia abonen-

ta pośredniczącego. Tym samym, między węzłami fizycznymi integrowanych systemów tworzone są wirtualne związki aplikacyjne, a węzły pośredniczące zapewniają obraz wektorów zewnętrznych i dostarczają integracji protokołów.



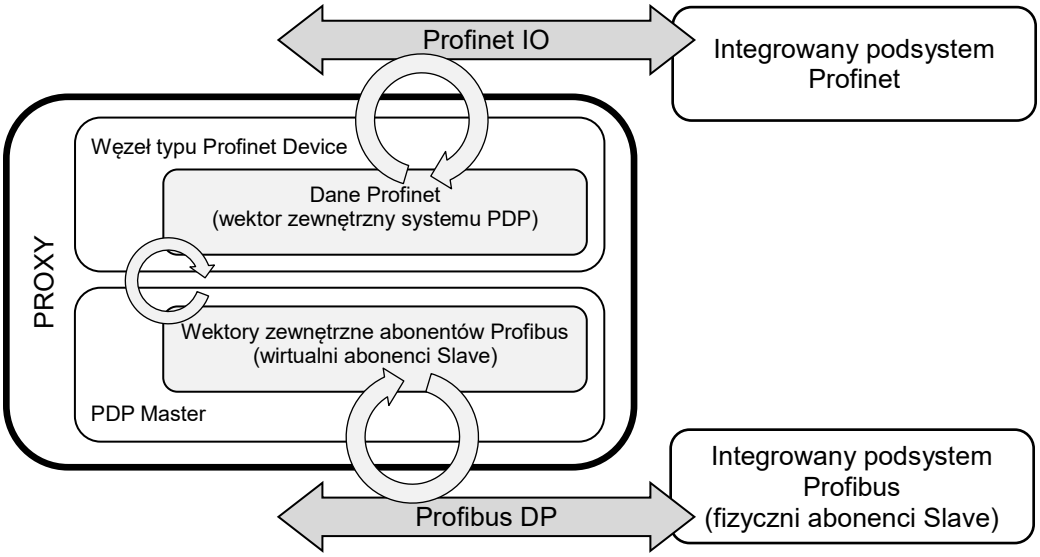
Rys. 125. Obiegi informacyjne węzła integracyjnego
Fig. 125. Informational circulation of an integration node

Przekazywanie zmiennych sieciowych pomiędzy węzłami ISP wykorzystującymi różne protokoły sprowadza się do niezależnego działania różnych scenariuszy wymian i lokalnego cyklu przetwarzania, co zostało zilustrowane w postaci pętli na rysunku 125. Zarządzają one transakcjami umożliwiającymi przekazanie danych pomiędzy instancjami zmiennych istniejącymi w przestrzeni alokacji lokalnych buforów komunikacyjnych każdego z tych protokołów. Integracja sieci polega zatem na pozyskaniu wymaganych wartości zmiennych z użyciem jednej sieci i aktualizacji tych wartości w ich odpowiednikach z użyciem drugiej. Od strony aplikacji sprowadza się to do identyfikacji zmiennych, pozyskania ich zrozumiałej reprezentacji, przedstawienia w reprezentacji docelowej i przekazania do obsługi w danej sieci. Standardowe zadania sieci sprowadzają te operacje do mapowania zmiennych sieciowych alokowanych przez lokalne stosy protokołów i synchronizowania ich wartości, a następnie na obsłudze wynikającej ze składni i semantyki docelowego protokołu. Z procesem mapowania nie wiąże się nic bardziej wyszukanego niż tabele powiązań. Proces przypisywania wartości to z reguły kopiowanie pamięci z ewentualnym prostym przetwarzaniem wynikającym z konieczności zmiany reprezentacji. Jednak do obsługi zmiennych protokół wymaga adresacji oraz zarządzania czasem. Adresacja może stanowić pewien problem, o ile adresy sieci źródłowej i docelowej nie są konwertowalne.

Przykładem może być wspomniana w 3.3.2 integracja Profinet i Modbus, gdzie niemożliwa jest bezpośrednia konwersja adresu IEEE 802.3 i powiązań IO do prostej adresacji urzą-

dzeń i specyficznej adresacji IO stosowanej w Modbus. Innym przykładem może być konwersja adresów sieci WorldFip i Profibus. W tym wypadku różnica jest w idei, gdyż w WorldFip adresuje się informację, a w Profibus węzły. Problemy te można jednak rozwiązać przez wprowadzenie dodatkowej informacji konfiguracyjnej do aplikacji węzła pośredniczącego, umożliwiającej jednoznaczne przemapowanie adresów.

Przykładem poprawnej integracji na poziomie aplikacji może być użycie tzw. proxy do integracji sieci Profibus z siecią Profinet. Zostało to pokazane na rysunku 126. Na zastosowanej tu zasadzie tworzenia wirtualnych abonentów w pamięci urządzenia integrującego można dokonać integracji dowolnych sieci przemysłowych. Wymaga to jednak użycia dedykowanych urządzeń integrujących lub stworzenia specjalnego węzła na bazie uniwersalnego (np. PLC), co nie zawsze jest trywialne.



Rys. 126. Integracja sieci Profibus z siecią Profinet
Fig. 126. Profibus and Profinet integration

Zdecydowanie gorzej jest z zachowaniem ograniczeń czasowych. Z racji specyficznego i ograniczonego zbioru usług sieci przemysłowych bazującego na cyklu sieci i scenariuszach wymian przekazywanie danych w danej sieci jest możliwe tylko w określonych momentach czasu. Cykle z natury nie są i nie mogą być zsynchronizowane. Pozostaje jedynie możliwość synchronizacji wartości danych przy użyciu wewnętrznego algorytmu aplikacji węzła integracyjnego. Gdy okresy aktualizacji zmiennych w obu sieciach są takie same, to działanie mechanizmu polega tylko na odpowiednim przepisywaniu wartości. Gdy okresy są różne, mogą wystąpić zjawiska nadpisywania wartości, czyli utraty danych. Nie ma dobrego rozwiązania tego problemu. W przypadku obsługi aperiodycznej można wprowadzić buforowanie danych. Jednak w przypadku aktualizacji cyklicznych bufory nie rozwiążą problemu, gdyż ich zajętość będzie rosła aż do nieuchronnego przepełnienia. Jedynym rozwiązaniem

w takim przypadku jest ustalenie reguł integracji niezależnie dla każdej zmiennej i uruchomienie algorytmu obsługującego te reguły.

3.5.4. Integracja węzłów

Węzły dostarczają informacji, która z racji rozproszenia dostarczanych funkcjonalności musi być dostępna w innych węzłach, w innych systemach, w szczególności w systemach nadrzędnych lub ogólnie na poziomie zarówno ISP, jak i na poziomach wyższych. Zakładając heterogeniczność środowiska urządzeń oraz wymóg elastyczności rozwiązania, ich integracja staje się koniecznością niezależnie od pełnionych funkcji i złożoności takich węzłów.

Integracja węzłów u swych podstaw sprowadza się do zagadnień opisanych wcześniej. Jednak trudno spodziewać się projektowania od podstaw i praktycznej implementacji mechanizmów integracji przy tworzeniu systemu na potrzeby jednostkowej aplikacji. Budowanie takiej warstwy sprzętowo-programowej jest ekonomicznie nieuzasadnione i nierealizowalne w typowych okresach czasu przeznaczonych na wdrożenie systemu. Dlatego w niniejszym podrozdziale zwrócono uwagę na istniejące techniki, które wspomagają integracje węzłów. Istnieją projekty, w ramach których grupy ośrodków komercyjnych i naukowych tworzą rozwiązania umożliwiające realizacje i zarządzanie przepływem informacji wśród węzłów ISP. Są to techniki dobrze znane, jak np. OPC, SOA, EPS, CBA, czy dopiero rozwijane, jak dla przykładu FDI [W19], [M8], [M7], EDDL [W15], FDT/FDT2 [M9], VAN [406] i inne. Technologie te dotyczą różnych aspektów funkcjonowania systemu, od usług komunikacji po ogólną architekturę wymiany danych i mogą wspólnie posłużyć do zbudowania ustandaryzowanego mechanizmu integracji.

W praktyce, użytkownika interesuje spójna koncepcja, gdzie każdy element systemu byłby postrzegany jednakowo, niezależnie od producenta i użytego narzędzia. Ogólna koncepcja takich technik bazuje na podejściu komponentowym, w którym informacje specyficzne urządzenia są grupowane w zbiór standardowy widziany jako wspólny i zrozumiały dla wszystkich interfejs. Jedną z nowszych propozycji w tym zakresie jest FDI (ang. Field Device Integration). Jest to technika, która umożliwia zintegrowanie grupy węzłów przez stworzenie systemu hostującego informacje i metody ich obsługi. U podstaw FDI stoją technologie proponowane przez grupy FDT[99], OPC Foundation[100] oraz EDDL Cooperation Team[101],

[99] Organizacja non profit zrzeszająca producentów sprzętu i oprogramowania (np. ABB, Endress+Hauser, Invensys, Metso, Siemens, Emerson, Honeywell i inne), której celem jest standaryzacja interfejsu komunikacyjnego i konfiguracyjnego dla urządzeń polowych (ang. field devices).

[100] Organizacja wspierająca utrzymanie, rozwój i standaryzacje technik OPC. W FDI wykorzystuje się specyfikację OPC UA.

[101] Grupa tworząca i rozwijająca specyfikację technik do tworzenia standardowych interfejsów umożliwiających konfigurację i diagnostykę urządzeń „inteligentnych" na bazie tzw. języka opisu urządzeń EDDL (ang. Electronic Device Description Language) [W15], [M44].

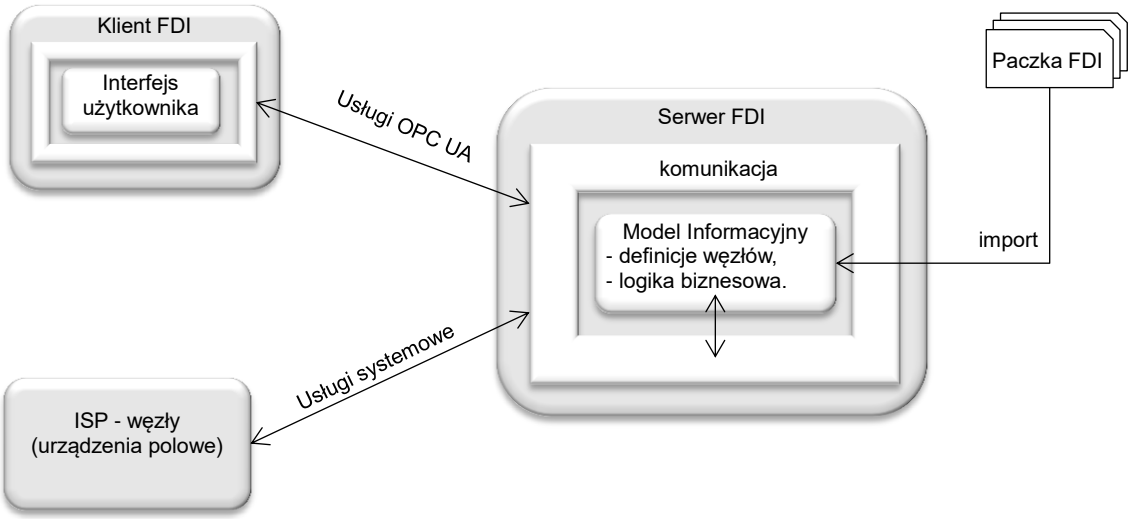
a koordynacją technologii zajmuje się organizacja FDI Cooperation[102]. Dodatkową organizacją zaangażowaną we współpracę użytkowników systemów klasy ISP oraz w rozwój technologii i ich standaryzacji jest Namur[103]. W 2007 roku organizacja określiła wymagania dla technologii integracji węzłów, które muszą być spełnione, aby dana technika była wspierana jako rozwiązanie standardowe[104]. FDI jest przykładem rozwiązania spełniającego takie wytyczne.

FDI zakłada, że z każdym węzłem systemu (urządzeniem polowym) związana jest jedna i tylko jedna definicja urządzenia, tzw. paczka FDI (ang. FDI Package) definiująca jego interfejs danych oraz funkcjonalność. Ponadto zakłada, że procesy serwerów i klientów danych bazują na standardowej, certyfikowanej i niezależnej od producenta technice komunikacyjnej OPC UA. Paczka FDI jest zatem rodzajem uniwersalnego sterownika (ang. driver) jednoznacznie identyfikującego zakres i możliwości funkcjonowania danego urządzenia. Paczka zawiera między innymi niezależne od konstrukcji urządzenia definicje określające działanie urządzenia, jego logikę biznesową, interfejs użytkownika oraz rozszerzenia. Paczki mogą być importowane przez specjalne programy, jakimi są „FDI Host" pracujące jako narzędzia integracyjne i spełniające rolę uniwersalnych serwerów lub klientów dla danych.

Dzięki temu uzyskuje się architekturę zgodną z modelem diabolo przedstawionym w 1.2.1, gdzie serwer FDI zawiera model informacyjny, definicje urządzeń, z którymi jest w stanie się komunikować oraz logikę działania tej komunikacji. Wymiana danych odbywa się przez standardowe interfejsy z użyciem standardowych mechanizmów komunikacji. W uproszczeniu zostało to przedstawione na rysunku 127. Elementy składowe wykorzystywane w FDI są częściowo ustandaryzowane przez IEC62769 (model, definicje, interfejs) [M14].

Komunikacja serwera FDI może być rozszerzana o dialog z klasycznymi klientami OPC UA czy też o komunikację przez specjalne serwery komunikacyjne obsługujące specyficzne interfejsy komunikacyjne urządzeń. Z założenia klientami FDI są procesy pracujące powyżej ISP, jednak jeśli w ISP istnieją węzły wspierające funkcje klienta FDI, to i one mogą stać się takimi klientami. Komunikacja z urządzeniami odbywa się z użyciem usług systemowych, czyli standardowej komunikacji OPC UA i/lub protokołów własnych danych urządzeń, np. sieci polowych.

[102] Od 2007 roku wspiera rozwój FDI zrzeszając również inne grupy wsparcia, jak np. Fieldbus Foundation, HART Communications Foundation, PROFIBUS & PROFINET International oraz wspomniane powyżej.
[103] Międzynarodowe stowarzyszenie użytkowników systemów automatyki [W31].
[104] FDI-User-Requirements 2011 [W31].



Rys. 127. Idea przepływu informacji w FDI
Fig. 127. Idea of information flow in FDI

3.5.5. Internet i praca w intersieciach

Systemy lokalne są podstawą w konstrukcjach ISP. Jednak w ostatnich latach liczba i zakres integracji systemów lokalnych z wykorzystaniem przestrzeni publicznej wzrasta. Mowa tu zarówno o węzłach pracujących w sieciach obcych, jak i współpracujących systemach integrowanych przez infrastrukturę takich sieci. Przy pracy z wykorzystaniem środków niestanowiących elementów systemu lokalnego pojawiają się dwa kluczowe problemy. Jeden to bezpieczeństwo, a drugi to charakterystyka czasowa wymiany danych. Zagadnienia bezpieczeństwa ISP zostały przedstawione w rozdziale 4. Rozważanie charakterystyk czasowych działania systemów rozproszonych w przestrzeni obcych infrastruktur, w tym publicznych, jest trudne i technicznie ograniczone. Analiza najgorszego przypadku jest praktycznie niewykonalna, gdyż intersieci nie dostarczają zdeterminowanych w czasie usług przekazywania danych. Nawet jeśli teoretycznie wykorzystywana sieć dysponuje mechanizmami kontroli ruchu determinującymi w czasie dostęp do przekazywanych danych, to w praktyce sieć taka leży poza kompetencjami administracyjnymi ISP i nie powinna być traktowana jako bezpieczna. Zatem wyniki ewentualnej analizy będą miały znaczenie poglądowe. Można natomiast wszelkie połączenia intersieciowe i wykorzystywane w nich urządzenia infrastruktury potraktować analizą statystyczną. Wyniki wówczas tylko szacują zachowanie danego rozwiązania, ale stanowi to założenie do analiz i nie sugeruje, że wymiana danych ma charakter zdeterminowany. Jako aparaty do analizy można wykorzystać opisy elementów dokonane z użyciem rozkładów wybranych zmiennych losowych i teorię kolejek [73], [1]. Możliwa jest analiza teoretyczna na etapie projektu, o ile charakterystyka pracy elementów jest znana. Można też dokonać wstępnych szacunków na podstawie przyjętych założeń, a następnie do-

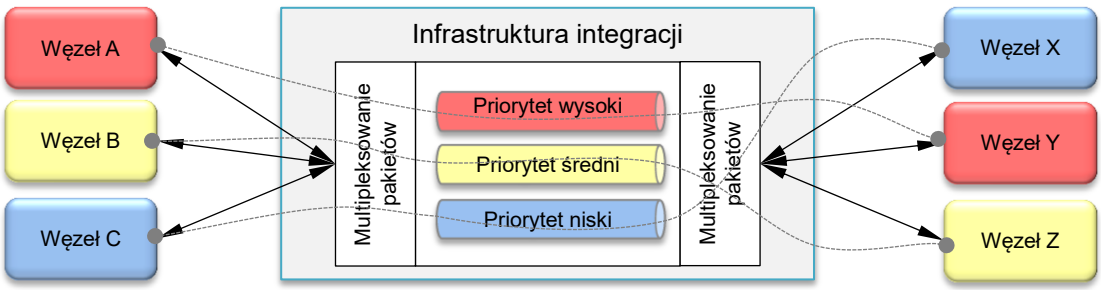
konać pomiarów charakterystyki pracy realnego układu i dostrojenia parametrów, o ile charakterystyka pracy elementów jest nieznana.

Dynamiczne pomiary charakterystyki czasowej układu, w tym wybranych wskaźników komunikacji w intersieciach, mogą posłużyć również do detekcji nieprawidłowości jego działania lub przynajmniej do stwierdzenia odstępstw od sytuacji przyjętej za normalną. Jest to możliwe w ISP, gdyż charakterystyki takie wykazują z reguły dużą samopowtarzalność w czasie. Wynika to z cykliczności zachowań układów w ISP, omówionej w 1.1.4.

❖ **Transmisja w czasie rzeczywistym**

Mechanizmem wspomagającym uzyskanie pożądanej charakterystyki czasowej transmisji w intersieciach jest mechanizm QoS (ang. Quality of Service). QoS jest pojęciem dość ogólnym i zawsze należy mieć na uwadze kontekst użycia. W omawianym przypadku, kontekstem jest przekazywanie danych między aplikacjami węzłów ISP w środowisku intersieciowym. Nie jest to jednak mechanizm gwarantujący zdeterminowany w czasie dostęp do medium, sieci, aplikacji czy jakiejkolwiek warstwy stosu protokołu komunikacyjnego. Zapewnia on zachowanie pewnej charakterystyki w ramach dostępnych zasobów. Dlatego już ogólnie można stwierdzić, iż jego stosowalność w ISP jest ograniczona do integracji wymagających podejścia soft-RT (por. 1.1.4).

Podejście QoS w ISP sprowadza się do dwóch mechanizmów. Pierwszy polega na prioretyzacji pakietów, a drugi na wyborze tras alternatywnych. Prioretyzacja sprawdzi się zarówno w środowisku sieci lokalnych, gdzie możliwa staje się separacja ruchu RT od innego, jak i w przypadku intersieci, gdzie teoretycznie wszystkie podsieci biorące udział w przekazywaniu danych mogą faworyzować pakiety konkretnego połączenia, np. wybranej sieci wirtualnej (ang. VPN). Wybór tras dla pakietów ma sens tylko dla połączeń, gdzie taki wybór istnieje, czyli dla odpowiednio złożonych intersieci.



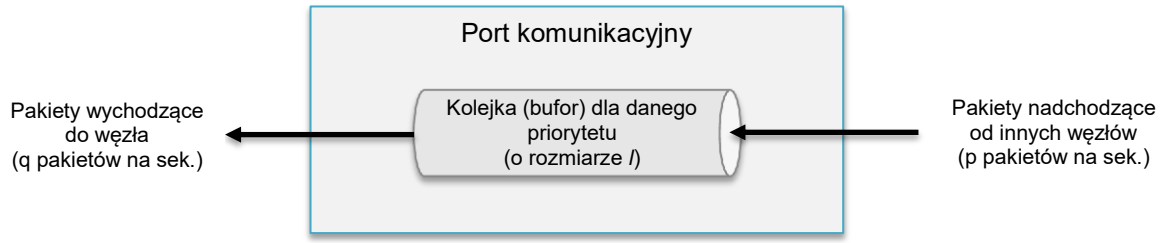
Rys. 128. Ilustracja QoS na bazie obsługi priorytetów
Fig. 128. Illustration QoS based on priorities service

Działanie QoS w integracjach wykorzystujących sieci lokalne wymaga budowania mechanizmu kolejkowania pakietów i mechanizmu określania jakości informacji użytecznej na

poziomie aplikacji. Na rysunku 128 przedstawiono ogólny schemat działania takiego mechanizmu.

Rozwiązanie takie redukuje opóźnienia i gwarantuje bezstratną, deterministyczną obsługę w zakresie czasu, w którym możliwe jest buforowanie danych. Przedstawione na rysunku bufory muszą być alokowane dla wszystkich fizycznych portów komunikacyjnych, a ich liczba dla danego portu zależy od liczby obsługiwanych priorytetów.

Pakiety z wyższym priorytetem są obsługiwane w pierwszej kolejności, pozostałe są buforowane do czasu, gdy zakończy się obsługa tych pilniejszych. Dla przykładu z rysunku, jeśli zachodzi potrzeba komunikacji w związku komunikacyjnym aplikacji węzła A oraz Y, to wymiany te będą zachodziły przed wymianami BZ, a te przed CX. Do każdego pakietu można dołożyć informację określającą czas życia przenoszonych jednostek danych, co da wgląd w jakość dostarczanej informacji względem czasu.



Rys. 129. Buforowanie pakietów w porcie
Fig. 129. Packets buffering in a port

Mechanizm buforowania działający w każdym porcie został zilustrowany na rysunku 129. Zakładając stały rozmiar pakietu, mechanizm taki jest w stanie zapewnić komunikację bez utraty danych dla *m* portów i *n* priorytetów, gdy kolejka nie rośnie

$$\left(\forall i = 1..m\right)\left(\forall j = 1..n\right)\left(p_{ij} \le q_{ij}\right)$$

lub dla danego portu i priorytetu *ij*, gdy rośnie $(p_{ij} > q_{ij})$, przez czas

$$t_{ij} = \frac{l}{p_{ij} - q_{ij}}[s]$$

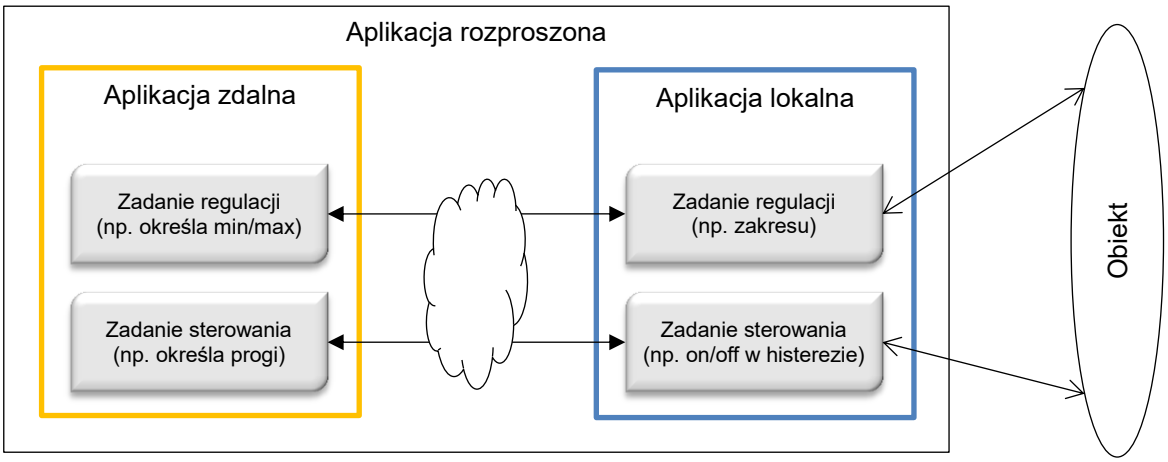
Oznacza to, że oprogramowanie przełączające umożliwi przekazywanie danych bez ich utraty przez t_{ij} sekund dla portu *i* oraz priorytetu *j* przy strumieniu wejściowym *p* pakietów na sekundę i strumieniu wyjściowym *q* na sekundę. Dla całego procesu przełączania czas ten stanowi minimum ze wszystkich portów i buforów. Określenie powyższych wartości dla protokołów deterministycznych jest możliwe ze względu na znaną pesymistyczną liczbę przesyłanych informacji. Dla sieci niedeterministycznych można szacować na bazie wartości średnich lub rozkładów zmiennych losowych.

W praktyce mechanizm priorytetów wykorzystują protokoły przemysłowych sieci Ethernet. Przykładowo w protokole Profinet IO [288] wykorzystuje się protokół IEEE 802.1P. Zatem realizacja odbywa się przez przełączniki (por. 3.3.3) obsługujące te protokoły.

Sprawa się komplikuje, gdy do integracji wykorzystuje się intersieci. Bezpieczeństwo w takim przypadku można w relatywnie prosty sposób zapewnić przez wykorzystanie prywatnych sieci wirtualnych (np. wspomniany Profinet IO wykorzystuje 802.1Q). Trudniej zapewnić wymaganą charakterystykę czasową przekazywania danych. Przeważnie, rozwiązania wspierające transmisje w czasie rzeczywistym w sieciach opartych na protokole TCP/IP określa się jako RTIP (ang. RT services over IP). Nie ma idealnych rozwiązań w tej dziedzinie. Większość sieci przemysłowych opartych na Ethernecie (por. 3.3.2) wspierają transmisję przez intersieci, oczywiście z wykorzystaniem QoS, ale z utratą determinizmu. W sieciach optycznych można wykorzystać standard GMPLS zarządzający ruchem w połączeniach VPN [209]. Wykorzystanie to jest jednak uzależnione od dostawców łączy. Ogólna idea tworzenia połączeń czasu rzeczywistego w intersieciach sprowadza się do wykorzystywania dostępnych usług QoS na bazie priorytetów oraz określenia tras dla danego połączenia, pomiaru opóźnień w takiej strukturze i zarządzaniu ruchem w tych trasach. Do tego wskazane jest posiłkować się wspomnianym określaniem jakości informacji względem czasu. Używanie takich mechanizmów daje możliwość kontrolowania połączenia względem wymaganej charakterystyki.

Istnieją próby tworzenia aplikacji rozproszonych, gdzie zadania sterowania i regulacji są wykonywane zdalnie względem układów wykonawczych przy użyciu integracji przez Internet [34], [71] lub wirtualizacji procesów sterujących w środowisku chmurowym [141]. Efekty są mocno dyskusyjne. Jeśli ktoś rozważa stworzenie aplikacji oddziałującej przez Internet na obiekt, to wskazane jest, aby to oddziaływanie było na zasadzie sterowania aplikacji lokalnej (parametryzacja, konfiguracja, rozkazy) ze sprzężeniem zwrotnym lub aby pozostać tylko przy zadaniach monitorowania. Regulacja przy niedeterministycznych połączeniach może skutkować opóźnieniami wychodzącymi poza parametry regulatora i wymagania obiektu, a w skrajnym przypadku ją całkowicie uniemożliwić lub zaburzyć. Sterowanie jest równie ryzykowne. Nawet najprostsze zadanie typu włącz/wyłącz może zadziałać załączając obiekt i nie mieć możliwości jego wyłączenia w wymaganym czasie lub wcale. Dlatego lokalna aplikacja pośrednicząca, realizująca regulacje i sterowanie lokalne lub przynajmniej funkcje bezpieczeństwa jest w takich przypadkach dobrym rozwiązaniem. Idea takiego rozproszenia zadań aplikacji została przedstawiona na rysunku 130. Stopień (głębokość) dekompozycji zadań i rozproszenia może być różny, ale nie pozostaje on bez wpływu na charakterystykę czasową działania całości. Zakładając niezależność tras, niestałość charakterystyk będzie rosła wprost proporcjonalnie do głębokości dekompozycji (liczby powiązanych zadań

rozproszonych). Jest to spowodowane faktem, że każde niezależne połączenie aplikacyjne przez intersieć wnosi niedeterministyczne opóźnienia, które w aplikacji rozproszonej zaczynają interferować, a ich rozkłady się splatać. Przy trasach zależnych, głębokość dekompozycji może nie wpływać znacząco na charakterystyki pracy, o ile przepustowość łącza nie ogranicza zwiększenia ruchu. Jednak nawet prze takim podejściu należy przeanalizować zależności czasowe pomiędzy zadaniami, sprawdzając, czy nie mają one charakteru HRT lub FRT. Sugeruje się, aby w środowiskach intersieciowych, szczególnie w Internecie, rozpraszać zadania powiązane aplikacyjnie tylko zależnościami SRT.



Rys. 130. Rozproszenie zadań aplikacji
Fig. 130. Distribution of application tasks

W systemach wykorzystujących połączenia komórkowe można używać różnych usług transmisji danych, od prostych komunikatów SMS, przez pakiety GPRS po zaawansowane usługi LTE. Względem czasu rzeczywistego niewykorzystanie takich sieci musi budzić wątpliwość i wymuszać odpowiedni dobór. W zakresie SMS jedyne co można rozważyć to tryb transferu: tekstowy albo binarny. Nie ma to jednak istotnego wpływu na czas dostarczenia komunikatu. Przy transmisji pakietowej istotne jest to, jaki protokół zostanie wybrany w warstwie transportowej oraz czy w ogóle wykorzystuje się warstwy wyższe ponad np. protokół PPP. Im bardziej złożona nadbudowa, tym większe wystąpią opóźnienia. Należy zawsze mieć na uwadze, że przy tego typu połączeniu nie występuje konieczność transmisji dużych pakietów danych użytecznych. Zatem potencjalne zalety złożonych protokołów, jak np. TCP będą niewykorzystane. Realny wpływ na zarządzanie czasem jest możliwy tylko przy usługach, które wspierają QoS jak LTE. Niestety, operatorzy sieci często nie udostępniają usług tego typu, mimo że standard je określa. W temacie adaptacji takich rozwiązań do dziedziny ISP prowadzone są prace, np. [93].

❖ **Rozwój**

W ostatnich latach zostało uruchomionych kilka projektów badawczych dotyczących innowacyjnego wykorzystania Internetu w kontekście systemów przemysłowych. Są to projekty, które głównie służą zintegrowaniu systemów i poszerzeniu ich zakresu funkcjonalnego. Ponadto nowe techniki wprowadzają nowe możliwości przepływu i analizy danych. Praktyczny wymiar zastosowań nie jest jednak wielki. Technologie te są na etapie badań i rozwoju, a na wdrożenia mogą sobie pozwolić tylko duże i bogate firmy. Do kilku przykładowych rozwijanych technologii można zliczyć:

- Industrial Internet

  Pojęcie Internetu przemysłowego jest co najmniej dwuznaczne. Podejście intuicyjne dotyczy wykorzystania sieci Internet do integracji systemów klasy ISP. Zostało to omówione powyżej. Drugie podejście jest bardziej abstrakcyjne i traktuje temat szerzej. Internet przemysłowy jest zestawem wszelakich technologii umożliwiających interakcję z elementami fizycznymi procesów produkcyjnych, celem pozyskania, składowania i przetwarzania danych na potrzeby szeroko pojętej analizy przebiegu procesów, ich optymalizacji wielokryterialnej oraz dostrojenia do potrzeb związanych z bieżącymi celami biznesowymi i wymaganiami produkcji. Oczywiście jest to również rodzaj integracji systemów, z tym że zakres działania wychodzi poza funkcjonalności ISP (zob. 1.2) i współpracujących systemów nadrzędnych (zob. 1.2.2, 3.2).

  W zakresie używanych technologii są standardowe protokoły, usługi i technologie internetowe, jak i całkiem nowe podejścia aktualnie będące przedmiotem badań i rozwoju, szukające dopiero swojego miejsca w rzeczywistych aplikacjach przemysłowych. Można do nich zaliczyć technologie chmurowe, eksploracje danych (ang. data mining), „big data" [411], IoT [22], M2M [26], [30] (por. 2.4.2, 2.4.4), CPS (por. 2.7) i inne.

- IoT – Internet of Things

  IoT jest pojęciem odnoszącym się do różnych dziedzin aplikacji, od sprzętu gospodarstwa domowego, przez infrastrukturę miejską, branżę samochodową, przemysł do osobistych gadżetów (ang. wearable). Dotyczy usieciowienia różnego rodzaju urządzeń z wykorzystaniem sieci publicznych na bazie Internetu. W ISP zastosowanie jest z założenia ograniczone do tych elementów, których uwidocznienie w przestrzeni publicznej ma jakiś cel [409]. Obecnie nie ma zbyt wiele takich potrzeb, choć istnieją dziedziny zastosowań ISP, gdzie filozofia IoT jest trafna i pożądana. Są to dla przykładu rozproszone systemy monitorowania obiektów infrastruktur przemysłowych, przemysłowe sieci sensorowe oraz systemy zarządzania sieciami przesyłowymi. Do

mniej przemysłowych można zaliczyć monitoring środowiskowy [99] czy automatykę budynkową.

Dopóki jednak nie nastąpi zmiana w charakterystyce komunikacji w sieciach publicznych, elementy IoT mogą pełnić rolę tylko systemów SRT lub wspomagającą te funkcjonalności ISP, które nie wymagają zestawiania zdeterminowanych w czasie połączeń aplikacyjnych.

Informacje dodatkowe o idei IoT można znaleźć np. w [395], [365], [271], o zagadnieniach integracji w [82], o bezpieczeństwie w [311], [366], a o korzyściach dla WSN np. w [333], [59].

- Industry 4.0

  Industry 4.0 jest pojęciem ogólnym dotyczącym i grupującym technologie umożliwiające stworzenie nowych jakości w komputeryzacji systemów produkcyjnych, zarówno klasy ISP jak i nadrzędnych. Zakres dotyczy zarówno funkcjonalności na poszczególnych warstwach modelu (por. 1.2.1, 2.2), jak i zagadnień integracji. Jak nazwa sugeruje, chodzi o stworzenie nowego podejścia i nowej generacji systemów opartych na najnowszych technologiach typu IoT (ang. Internet of Things), CPS (ang. Cyber-Physical Systems) (zob. 2.7), chmurowego, komunikacji 3G (por. 2.4.4) [371]. Zatem nie należy rozpatrywać tego podejścia jako technik integracyjnych dla istniejących rozwiązań, lecz jako środki dla tworzenia nowej klasy systemów, nazywanych też często Smart Factories lub Factory of the Future [317].

  Podstawą są systemy CPS (zob. 2.7). Dostarczają one danych, na podstawie których budowany jest opis informacyjny obiektu, stanowiący swoisty obraz procesu, reprezentujący rzeczywistość z określonym czasem ważności elementów tej reprezentacji. Można tu doszukać się analogii do obrazów procesu stosowanych w klasycznych rozwiązaniach PLC. Następnie na obrazie dokonywane są przetwarzania i na podstawie wyników podejmowane są decyzje. Idea dotyczy systemów informatycznych w skali przedsiębiorstwa. Podstawowe cechy to możliwość współdziałania elementów bez technicznych ograniczeń, rozproszenie i decentralizacja funkcjonalności w środowisku fizycznym, jak i wirtualnym, orientacja na usługi, modułowość rozwiązań oraz działanie w czasie rzeczywistym. Co do tego ostatniego należy pamiętać o rodzajach SCR (por. 1.1.4) i mieć świadomość, że większość usług w skali całego systemu dotyczy podejścia SRT. Zdeterminowany w czasie obieg informacji istnieje tylko w wybranych funkcjonalnościach systemu.

  Industry 4.0 jest rozwijane w Europie. Istnieje również podobny projekt amerykański pod nazwą SMLC (ang. Smart Manufacturing Leadership Coalition) [W10]. Industry4.0, IoT oraz idea SmartFactory są często rozwijane wspólnie [334]. Aktualnie, tworzenie

koncepcji zintegrowanych systemów umożliwiających całkowitą komputeryzację procesów wytwórczych jest głównym trendem rozwojowym w dziedzinie. Należy podkreślić: systemów zintegrowanych, a nie integracji systemów.

- IoT@work
  Jest to zakończony projekt UE łączący w pracach badawczych ośrodki naukowe[105] oraz centra rozwojowe (ang. R&D – Research and Development) producentów[106] komponentów ISP. Główny cel projektu był bardzo ambitny i godny poparcia w kontekście rozwoju dziedziny. Dotyczył on stworzenia mechanizmów umożliwiających automatyczne współdziałanie węzłów celem konfiguracji i uruchomienia osadzonej w nich aplikacji (ang. embedded intelligence). Była to zatem realna próba zaprzęgnięcia idei IoT do zadań ISP.

  Jako efekt prac powstały technologie integracyjne (ang. IoT@work Integration technology) związane z opisem informacyjnym obiektów, konfigurowaniem sieci, rozprowadzaniem i przetwarzaniem zdarzeń, bezpieczeństwem i inne.

  Więcej informacji na temat projektu i osiągnięć można znaleźć na [W26] oraz w [M66].

- ICT
  ICT – Information and Communications Technology nie jest technologią wiązaną bezpośrednio z systemami przemysłowymi, a nawet nie jest technologią w ogóle. Jest to ogólne pojęcie określające zestaw środków technicznych umożliwiających integrację współczesnych technologii informatycznych (IT) z wykorzystaniem cyfrowych technik komunikacyjnych, głównie nowoczesnych sieci komputerowych zarówno kablowych, jak i bezprzewodowych. Zatem, mogą one również posłużyć do integracji systemów w przedsiębiorstwach, w tym ISP w kontekście zarówno lokalnym, jak i zdalnym.

  Ponieważ zarówno technologii informatycznych, jak i sieciowych jest bardzo wiele, więc zastosowanie praktyczne ICT w ISP wiąże się z ustaleniem pewnych założeń, a zatem niezbędne jest udziedzinowienie ICT dla systemów przemysłowych. W efekcie otrzymuje się potrzebę tworzenia dziedzinowych technologii integracyjnych lub podejść zintegrowanych zasygnalizowanych powyżej.

- Podejście middleware
  Na koniec warto wspomnieć, że wszelkie integracje międzysystemowe, a szczególnie intersieciowe, wymagają stosowania oprogramowania pośredniczącego (ang.

[105] Akademickie ośrodki Lemgo‘s Institut Industrial IT z Niemiec i City University London z Wielkiej Brytanii, oraz ośrodki komercyjne: TXT-e Solutions z Włoch i European Microsoft Innovation Centre z Niemiec.
[106] Firmy Siemens AG oraz Fiat RC.

middleware). Klasyczne podejście dotyczy pośrednictwa pomiędzy OS i aplikacją użytkownika. W integracji ISP pośrednictwo występuje pomiędzy aplikacjami i służy ich skomunikowaniu [372]. Z punktu widzenia aplikacji nie ma znaczenia, gdzie znajduje się inna aplikacja. Dlatego oprogramowanie pośredniczące może działać zarówno w przestrzeni lokalnej systemu, jak i w przestrzeni zdalnej. Problemem jest oczywiście zapewnienie odpowiednich charakterystyk czasowych przekazywania zmiennych. Jest to jednak problem warstwy połączeń, a nie samego pośrednika, choć często w ISP oprogramowanie pośredniczące jest postrzegane jako oprogramowanie komunikacyjne. W pewnym stopniu tak jest, gdyż głównym celem jest komunikowanie aplikacji, ale zakres jego działania dotyczy głównie mapowania informacji pomiędzy aplikacjami oraz analiza i diagnostyka procesu przekazywania danych. Łatwo wyobrazić sobie, że i inne zadania można osadzić w warstwie pośredników.

Prace w dziedzinie integracji z użyciem middleware są prowadzone od dawna, więc nie jest to całkiem nowe podejście [267], [393], ale rozwój tego typu oprogramowania ciągle jest duży [98], [133], [247], [44]. Można zatem korzystać z gotowych pomysłów, jak również programiści nie powinni mieć problemów z tworzeniem własnych programów pośredniczących, dopasowanych do konkretnych aplikacji.

4. ASPEKTY BEZPIECZEŃSTWA W ISP

Wykorzystanie współczesnych systemów informatycznych sprawia, że praca w przemyśle staje się przyjemniejsza, bardziej wydajna i bezpieczniejsza dla człowieka. W efekcie człowiek uzależnił się od ich działania. Wpływają one zarówno bezpośrednio, jak i pośrednio na edukację, zadania i plany zawodowe jednostek, ale także na funkcjonowanie i planowanie rozwoju gospodarki zarówno w wymiarze pojedynczych zakładów, jak i w wymiarze ogólnym. Ten związek objawia się spektakularnie, gdy coś w nim idzie źle. Zdarza się słyszeć o awariach, szkodach, stratach i innych negatywnych skutkach materialnych, ludzkich i środowiskowych wynikających z błędów obsługi, zdarzeń losowych lub nieprawidłowości działania różnych systemów technicznych. W komputerowo wspomaganych systemach przemysłowych, z racji ich oddziaływania z fizycznymi procesami, należy wykazać szczególną troskę względom założeń działania takich systemów i ich bezpieczeństwa funkcjonowania, tym bardziej że technologie informatyczne używane do ich konstrukcji dają ku temu duże możliwości.

Każdego inżyniera produkcji interesuje poprawne prowadzenie procesu przemysłowego. Aby było to możliwe, należy brać pod uwagę nie tylko technologię, automatyzację i informatyzację całości, ale i zagadnienia bezpieczeństwa eksploatacji wszystkich elementów układu. Zapewnianie bezpieczeństwa dla procesu przemysłowego wiąże się z zapewnieniem prawidłowego działania samego procesu jak i zapewnieniem prawidłowego oddziaływania z nim systemu kontroli. Przekłada się to bezpośrednio na konstrukcje takich systemów oraz ich odporność na zagrożenia eksploatacyjne. W tym kontekście mówi się o właściwych do realizacji takich systemów technologiach informatycznych (ang. proper design) oraz o niezawodności i rzetelności działania (ang. reliability, dependability). Rozważenie tych zagadnień na etapie projektowania instalacji lub jej modernizacji może zapewnić spokojną pracę i niezawodne działanie procesu. Lekceważenie wcześniej czy później doprowadzi do problemów.

W kwestii wykorzystywanych technologii należy brać pod uwagę wsparcie, jakie dają one dla pracy układu w środowisku przemysłowym. Nie każda technologia informatyczna, szczególnie komunikacyjna, daje odpowiednie środki, aby maksymalizować bezpieczeństwo

działania systemu. Technologie powinny być dobrze dobrane na etapie projektowania. Błędy na tym etapie są z reguły nienaprawialne. Dlatego, przy decyzjach o wyborze, osoby decydujące oraz działy IT dostarczające opinii, powinny brać pod uwagę konkretną specyfikę procesu i wymogi z nią związane, szczególnie kwestie dotyczące charakterystyki czasowej przetwarzania i wymiany danych. W przypadku braku odpowiedniej kadry wskazany jest konsulting techniczny.

W kwestii zapewnienia odporności systemu na zagrożenia prowadzące do destabilizacji ruchowej procesu należy, jak zasygnalizowano w 1.1.1, rozpatrywać zagrożenia zasobów (ang. security – dotyczącego zagrożeń zewnętrznych), czyli niepożądanego dostępu do urządzeń i danych oraz od zagrożeń funkcjonalności (ang. safety – dotyczącego zagrożeń wewnętrznych), czyli niebezpieczeństw zaistnienia sytuacji niewłaściwych z punktu widzenia działania danej technologii. Celem działania mechanizmów bezpieczeństwa jest zawsze zapewnienie działania systemu w sposób niezagrażający procesowi i środowisku, w którym pracuje.

Branie pod uwagę aspektów bezpieczeństwa przy tworzeniu, modyfikacji czy utrzymaniu ISP jest niezwykle istotne. Ignorowanie zagrożenia w sposób świadomy lub często nieświadomy, a co za tym idzie pomijanie aplikacji systemów bezpieczeństwa, w efekcie prowadzi do przykrych konsekwencji.

Należy jednak pamiętać, że bezpieczeństwo lub jego brak jest swoistym stanem, a system informatyczny powinien taki stan *zapewniać*. Mówiąc o systemach bezpieczeństwa, zapewnianiu, zarządzaniu czy kontroli bezpieczeństwa, należy zawsze rozważać działanie pewnego **procesu** gwarantującego bezpieczeństwo, a nie statyczną ochronę. Ochrona taka jest skuteczna tylko przez pewien okres czasu, po czym jej skuteczność maleje [367]. Prostą ilustracją problemu z życia codziennego może być program antywirusowy, który nieaktualizowany powoli staje się bezużyteczny [186].

W niniejszym rozdziale dokonano przeglądu kluczowych zagadnień związanych z bezpieczeństwem systemów przemysłowych. Przedstawione aspekty bezpieczeństwa, w kontekście elementów składowych systemów oraz teorii ogólnej, powinny nakierować czytelnika na zagadnienia warte dalszych rozważań, a także przybliżyć wiedzę, gdzie i czego należy szukać w temacie współczesnych sposobów zapewniania bezpieczeństwa w ISP. Wskazane jest, aby inżynierowie odpowiedzialni za tworzenie systemów klasy ISP stosowali dobre praktyki zawarte w rozdziale lub dostępne w innych źródłach, np. rekomendacje US ICS-CERT (United States Industrial Control Systems – Cyber Emergency Response Team) [M94]. Ogólną analizę problematyki można znaleźć w [204].

4.1. Konstrukcja bezpiecznego systemu

Aby zwiększyć bezpieczeństwo projektowanego rozwiązania już na etapie jego konstrukcji, należy wprowadzić odpowiednie mechanizmy, wbudowując je w oprogramowanie, sprzęt i komunikację. Z wymienionych w 2.3 cech ISP, kilka wiąże się bezpośrednio z bezpieczeństwem funkcjonowania systemu. Są to: niezawodność, spójność informacyjna, punktualność, tolerancja awarii i trwałość. Istnienie tych cech jest w prostej linii związane z użytymi technologiami informatycznymi, dotyczącymi działania systemów operacyjnych, programów i sieci komputerowych. Istotny jest również dobór urządzeń i ich zadań w systemie.

4.1.1. Oprogramowanie

W ISP funkcjonuje oprogramowanie, będące w bezpośrednim oddziaływaniu z procesem i urządzeniami AKP, oprogramowanie oddziałujące z elementami systemu lub innymi systemami oraz oprogramowanie oddziałujące z człowiekiem. Człowiek, jako istota dość nieprzewidywalna nie wymaga obsługi zdeterminowanej czasowo. Ważne jest, aby systemy interfejsowe typu HMI czy SCADA nie gubiły danych i nie zmieniały kolejności zdarzeń. Jest to potrzebne do poprawnej diagnostyki i wglądu w stan procesu. Jeśli jednak chodzi o pozostałe grupy programów, to powinny wykazywać się zdolnością do działania w czasie rzeczywistym. Sposób działania wynika z ogólnej teorii systemów czasu rzeczywistego oraz wymogów procesu (por. 1.1.4, 1.2.3). Kluczowa jest tutaj zdolność do zarządzania czasem wykonywania zadań i jego kontroli oraz sposób reakcji na błędy przekroczenia dopuszczalnego czasu obsługi.

Administracja oprogramowania i jego utrzymanie w ISP charakteryzują się pewnymi problemami, które nie występują w systemach biznesowo-biurowych. Dotyczą one:

- Zachowania stanu (ang. retentiveness). Jest to problem programistyczny związany z zapamiętywaniem stanu aplikacji na okoliczność restartu, awarii lub wyłączenia zasilania. Zakładając, że program jest poprawny, a sprzęt umożliwia zachowywanie stanu, należy dbać, aby mechanizmy zachowywania działały. Dotyczy to cyklicznych wymian baterii, kontroli dysków, kart i pamięci, a także rozsądnej gospodarki zasobami o ograniczonej żywotności (np. EEPROM, flash).
- Aktualizacji (ang. upgrade, patching). W ISP często nie jest możliwa aktualizacja oprogramowania, ze względu na fakt, że pakietów aktualizacji nikt nie przygotowuje, a nawet jeśli, to nie można jej przeprowadzić w dowolnym momencie. ISP charakteryzuje się pracą ciągłą, a aktualizacje wymagają zatrzymania działania oprogramowania na bliżej nieokreślony czas. Dlatego wszelkie aktualizacje muszą być planowane na odpowiedni czas

przed jej wykonaniem. Ponadto, ze względu na możliwość wprowadzenia nieznanych wcześniej zaburzeń wynikających z nowego, zaktualizowanego kodu, należy aktualizacje wykonywać tylko w sytuacji, gdy aktualna wersja nie spełnia wymagań funkcjonalnych lub stwarza zagrożenia.

- Stosowania metod szyfrowania i protokołów bezpiecznego przekazywania danych. W ISP stosuje się urządzenia, które w porównaniu z systemami serwerowymi czy nawet desktopowymi nie mają dużych mocy przetwarzania oraz wielkich zasobów pamięci. Nie wszystkie algorytmy bezpieczeństwa da się zaimplementować. Dlatego wdrożenie mechanizmów tego rodzaju na poziomie węzłów ISP jest w praktyce ograniczone lub w ogóle nie występuje. Natomiast niezbędne jest ich użycie przy integracji ISP z podsystemami pracującymi w przestrzeni publicznej lub ją wykorzystującymi. Najpopularniejszą techniką w tej dziedzinie jest VPN [M71], [297], [172].

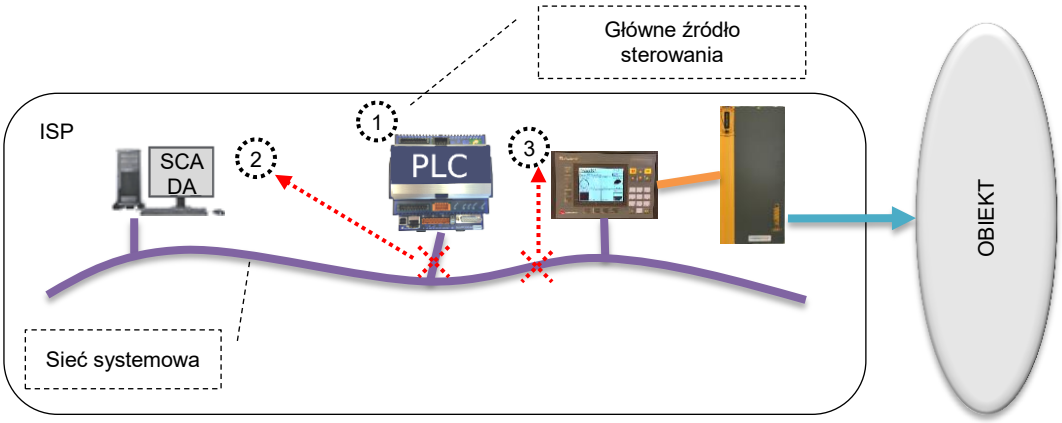
4.1.2. Urządzenia

Do obsługi procesów przemysłowych stosowane są specjalizowane komputery klasy PLC, DCS, IPC i inne. Mają one dostosowane interfejsy do procesu i urządzeń AKPiA oraz są w stanie zapewnić terminowe (punktualne) wykonanie zadań przetwarzania danych. Wykorzystanie takich urządzeń nie wprowadza konstrukcyjnych zagrożeń bezpieczeństwa. Problem pojawia się, gdy w ISP umieszczone zostają komputery przeznaczone dla człowieka. Zagrożenia i ograniczenia z tym związane zamieszczono w 3.1.

Dobór urządzeń powinien uwzględniać sytuacje awaryjne i ich obsługę. Najczęściej do zabezpieczania przed upadkiem (awarią) systemu z powodu awarii jego komponentów używa się układów redundantnych, opisanych szerzej w 4.2.3 oraz [228], lub wbudowanych mechanizmów zwiększających odporność na upadki [M77]. Urządzenia powinny również spełniać wymagania środowiskowe (1.1.2) danej aplikacji. Nawet najlepiej zaprojektowany ISP przestanie funkcjonować, gdy zastosowane urządzenia nie sprostają warunkom eksploatacji.

Ponadto, istotny jest dobór zadań i zakresu uprawnień w oddziaływaniu urządzeń na obiekt. Oddziaływania dotyczą wszelkich żądań i rozkazów związanych z sterowaniem, przekazaniem danych, parametryzacją, konfiguracją itp. Problem liczby źródeł sterowania i ich współpracy z danym elementem sterowanym dotyczy zagadnień automatyki z punktu widzenia prowadzenia procesu, ale i informatyki z punktu widzenia dynamicznego określania i walidacji działań takich źródeł. Przez źródło sterowania rozumiany jest element systemu, który ma uprawnienia do wydawania rozkazów dla urządzeń wykonawczych (por. 1.1.5). Ze względów bezpieczeństwa związanych z zapewnieniem istnienia takiego źródła może być ich wiele, co prowadzi do rozwiązań z funkcjonalnościami redundantnymi. Istnieje wówczas

problem współpracy takich źródeł, szczególnie gdy wydawane polecenia są wzajemnie sprzeczne.

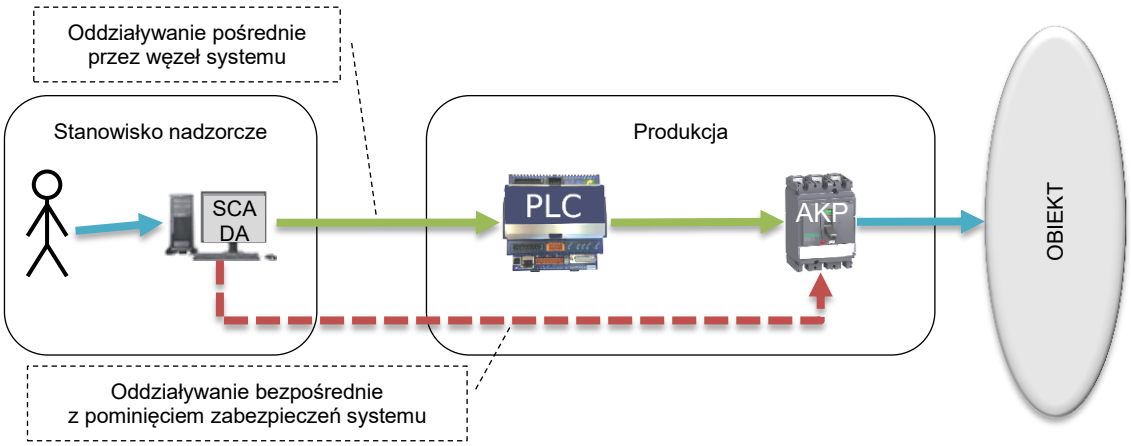


Rys. 131. Awaryjne źródła sterowania
Fig. 131. Backed up control sources

W dobrze skonstruowanym systemie nie może dochodzić do sytuacji, gdy różne węzły systemu wydają sprzeczne rozkazy. Dlatego z reguły przyjmuje się priorytety dla źródeł sterowania. Określają one nadrzędność funkcji sterujących danych węzłów (zadań aplikacji) względem innych węzłów. Najprostszą acz skuteczną metodą jest określenie kolejnych priorytetów dla źródeł stanowiących kolejne zapasowe węzły dedykowane do obsługi danego działania. Przy takim rozwiązaniu w danym momencie czasu istnieje tylko jedno aktywne źródło sterowania. Dla przykładu, najwyższy priorytet może mieć sterownik, w przypadku jego utraty (awaria, brak komunikacji, zasilania itp.) sterowanie przechodzi w tryb manualny i źródłem staje się program nadzorczy, a w przypadku jego awarii konsola lokalna. Zostało to zilustrowane na rysunku 131.

W przypadku wielu aplikacji węzłów interfejsowych, których zakres oddziaływania na inne elementy systemu się pokrywa, istotne jest zapewnienie synchronizacji stanu tych elementów oraz stanu oddziaływania, we wszystkich aplikacjach węzłów interfejsowych. W zależności od potrzeb i możliwości technicznych można tego dokonać przez rozgłoszenie wektora stanu elementów pomiędzy aplikacjami lub odczytu stanu tych elementów. Dodatkowo należy zadbać o przekazanie wektora żądań, aby inne aplikacje były w posiadaniu informacji opisującej nie tylko bieżący stan elementu, ale i bieżący stan oddziaływań względem tych elementów. Niebezpieczne jest natomiast dopuszczenie, aby stan elementów, na które oddziałuje dana aplikacja, był różny od stanu, na podstawie którego inna aplikacja może dokonać jakiegoś oddziaływania. Problem ten dotyczy wszystkich węzłów systemu, jednak ze względu na zasadę jednego źródła sterowania nie powinien wystąpić w węzłach innych niż interfejsowe. Węzły interfejsowe, jak nadzorcze czy integracyjne, z natury mogą być zwielokrotniane zarówno w swojej liczbie, jak i zakresie oddziaływań na elementy sys-

temu. Dla przykładu, komputery z wizualizacją procesu i możliwością ręcznego oddziaływania na jego parametry i działanie mogą być zlokalizowane w kilku miejscach nastawni lub różnych miejscach budynku. Mogą być też mobilne lub zdalne. Wykonanie oddziaływania na jednym musi być widoczne na każdym innym, czyli musi być uwidaczniany zamiar oraz efekt oddziaływania. Rozstrzyganie aktywnego źródła sterowania wiąże się z algorytmami stosowanymi w redundancji zasobów (zob. 4.2.3 oraz [228]).



Rys. 132. Oddziaływanie z człowiekiem
Fig. 132. Interactions with a human

Istnieje jeszcze problem ogólnej koncepcji sterowania związany ze zwielokrotnionymi źródłami sterowania. W przypadku pracy systemu w trybie automatycznym, czyli bez bezpośredniego udziału człowieka, sprawa jest prosta i sprowadza się do ustalenia zasad i wynikających z nich algorytmów. Jednak gdy jednym ze źródeł sterowania staje się człowiek, wówczas pojawia się czynnik niedeterministyczny. Jest to zatem problem związany z określeniem sposobu oddziaływania żądań pochodzących z węzłów interfejsowych. Operator czy też inny pracownik związany z prowadzeniem procesu nie jest zwykle w stanie postrzegać swoimi zmysłami istotnych danych i ich współzależności, nawet jeśli zjawiska są wolnozmienne. Dlatego jeśli udostępnia się funkcje sterujące człowiekowi, to algorytmy sterujące powinny być przełączone w tryb ręczny (manualny), a rozkazy wydawane przez człowieka powinny podlegać walidacji względem stanu wektora informacyjnego opisującego obiekt. Innymi słowy, rozkaz wydany przez człowieka z poziomu np. systemu SCADA, powinien zostać przekazany do odpowiedniego węzła (np. sterownika), który będzie w stanie przed realizacją tego rozkazu sprawdzić wszelkie niezbędne warunki bezpieczeństwa i warunki ograniczeń technologicznych. Zatem, procesem steruje sterownik, ale rozkazy dla sterownika może wydawać człowiek lub inny węzeł.

Istnieją zatem przynajmniej dwa przypadki obsługi oddziaływania z węzła interfejsowego. Zostało to zilustrowane na rysunku 132. Pierwszy – to sytuacja, gdzie dane z węzła inter-

fejsowego oddziałują bezpośrednio na obiekt, tym samym „mostkując" działanie ISP. Dokonuje się tego przez przekazanie danych z węzła interfejsowego do „inteligentnych" elementów wykonawczych (AKP) na obiekcie. Drugi – to oddziaływanie na „inteligentne" węzły ISP, czyli na aplikacje pracujące w węzłach, w celu aktywacji odpowiednich funkcji związanych z oddziaływaniem na obiekt. Efektem pierwszego jest możliwość bezpośredniego działania na urządzeniach automatyki. Pomijana jest logika i zabezpieczenia zawarte w urządzeniach sterujących systemu. Efektem drugiego jest oddziaływanie przez pośrednika w postaci programów ISP. W takim przypadku wszelkie działania z poziomu interfejsu są tylko zleceniem działania do systemu. Oddziaływanie rzeczywiste na obiekt odbywa się przez aplikacje i pod kontrolą algorytmów działających w węzłach ISP.

Problem ten jest szerszy i sprowadza się do pytania, czy jeżeli istnieje ISP obsługujący proces przemysłowy, to czy należy dać użytkownikowi możliwość wyłączenia funkcji tego systemu i sterowania ręcznego. Odpowiedź nie jest oczywista, gdyż zależy od wymagań technologicznych, funkcjonalnych i bezpieczeństwa, ale zalecane jest, aby tego nie robić. Powód jest prosty. Wektory informacyjne węzła interfejsowego opisujące obiekt są najczęściej tylko podzbiorem stanu obiektu. Ponadto, są one na tyle złożone, że ich poprawna interpretacja może być trudna, szczególnie dla człowieka. Tym samym decyzje podjęte na ich podstawie mogą być błędne i niebezpieczne.

Podejście drugie jest bardziej bezpieczne. Oddziaływania z poziomu węzła interfejsowego są wypracowywane na podstawie informacji dostępnych w tym węźle i wysyłane do odpowiednich węzłów systemu związanych z urzeczywistnieniem takich oddziaływań. Węzły te, z racji tego że odpowiadają za urzeczywistnienie operacji, muszą posiadać komplet informacji niezbędnej do wykonania takiego urzeczywistnienia. Zatem, wykonają daną akcję tylko wówczas, gdy spełnione są warunki na jej wykonanie. Dla przykładu, ręczne sterowanie w stacji SCADA zostanie zrealizowane, o ile warunki technologiczne i bezpieczeństwa zdefiniowane w ISP na to pozwolą. Innymi słowy, załączenie np. wentylatora z poziomu kliknięcia myszą nie spowoduje zadziałania przekaźnika załączającego obwody, a jedynie spowoduje wysłanie rozkazu do odpowiedniego węzła (np. PLC), który takie załączenie wykona lub nie, w zależności od analizy stanu obiektu. Realizacja tego w praktyce sprowadza się do stworzenia zadań walidujących dane oddziaływanie. Wprowadzane wartości, w tym wszelkie rozkazy, muszą być sprawdzane względem statycznych wartości progowych i/lub dynamicznych wartości wyliczanych na podstawie określonych algorytmów i stanu obiektu, uwzględniając nie tylko technologię, ale i bezpieczeństwo funkcjonalne.

Algorytmy walidacji wprowadzanych danych i rozkazów mogą być rozproszone pomiędzy węzeł interfejsowy a węzeł procesowy (lub węzły). Całkowita walidacja po stronie węzła interfejsowego jest bezpieczna tylko wówczas, gdy w tym węźle znany jest pełny obraz in-

formacyjny sterowanego obiektu zarówno od strony semantycznej tego opisu, jak i od strony regułowej powiązań. Przy analizach warto pamiętać, że węzły interfejsowe niekoniecznie pracują z takimi samymi wymaganiami czasowymi jak węzły procesowe oraz że zależności synchronizacyjne zdarzeń obiektowych i ich obsługi mogą w nich nie występować.

4.1.3. Sieci komputerowe

ISP są systemami przeważnie rozproszonymi terytorialnie. Wymaga to przekazywania danych pomiędzy rozproszonymi aplikacjami systemu z uwzględnieniem ograniczeń czasowych. Współcześnie, przekazywanie odbywa się najczęściej z użyciem komputerowych sieci przemysłowych. Istnieją trzy generacje takich sieci [319], [125]. Klasyczne rozwiązania stanowią sieci polowe (miejscowe, systemowe, ang. fieldbus), bardziej nowoczesne i uniwersalne są sieci klasy RTE (Ethernet przemysłowy, ang. Real Time Ethernet), natomiast do najbardziej wyrafinowanych aplikacji stosuje się sieci trzeciej generacji oparte na rozwiązaniach heterogenicznych [125]. W każdym przypadku istnieją specjalne protokoły umożliwiające zdeterminowaną w czasie obsługę danych na medium. Sama warstwa fizyczna z dostępem swobodnym (np. RS485 czy Ethernet) nie wystarczy do zapewnienia transmisji z ograniczeniami czasowymi [122]. Stosowanie „zwykłych" sieci prowadzi do niewydolności komunikacyjnej w sytuacjach stresowych (odbiegających od typowych), do występowania nieprzewidywalnych opóźnień w obsłudze danych, oraz w skrajnym przypadku do utraty danych. Zatem, niezbędne jest, aby w ISP do komunikacji między węzłami stosować komputerowe sieci przemysłowe. Ich wykorzystanie umożliwia uzyskanie zdeterminowanego w czasie dostępu do medium, a co za tym idzie zdeterminowaną w czasie wymianę danych między węzłami systemu. Więcej na ten temat zamieszczono w 2.4.4.

4.2. Niezawodność działania

Według współczesnych wymogów utrzymania ruchu kwestie związane z działaniem systemu powinny być traktowane szerzej niż tylko jako zadania regulacji i przekazywania danych. Szczególnie dotyczy to zadań komunikacyjnych w systemach rozproszonych, gdyż to ze strony sieci pojawiają się obecnie największe niebezpieczeństwa.

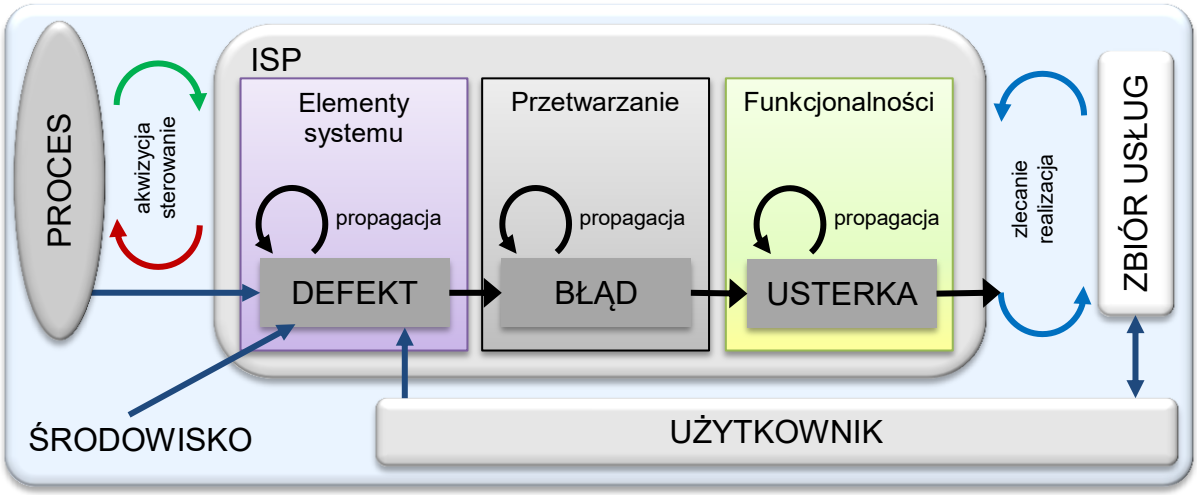
Zagrożenia poprawnego funkcjonowania systemu i kontrola bezpieczeństwa z tym związana wiążą się z teorią niezawodności systemów (ang. reliability, dependability). Występują w niej ogólne zagrożenia klasyfikowane jako:

- defekt (ang. fault) – występuje jako efekt zaburzeń lub upadków działania komponentów systemu i wynika z wad sprzętu lub oprogramowania bądź też z błędów, lub negatywnych działań użytkownika i/lub środowiska, w którym system pracuje,

- błąd (ang. error) – generowany jest jako efekt wystąpienia defektu i dotyczy procesu przetwarzania w węzłach systemu,
- usterka (awaria, ang. failure) – powodowane jest przez błędy i uniemożliwia poprawne wykonanie usług.

Defekt jest skutkiem działania człowieka, oddziaływania środowiska lub funkcjonowania procesu. Może to być pomyłka lub celowe działanie (błąd, pluskwa) w kodzie, oprogramowaniu, systemie czy dokumencie, nieprzewidziane zachowanie procesu lub nieprzewidziany wpływ środowiska. Jeśli element systemu z oddziałującym defektem wykona swoje zadanie (np. wykona się program), może wystąpić błąd. W efekcie system nie zrobi tego, co powinien lub zrobi coś, czego nie powinien robić, wywołując usterkę (awarię). Defekty w oprogramowaniu, systemach lub dokumentach mogą, lecz nie muszą skutkować awariami.

Wszystkie zagrożenia mogą ulegać propagacji w zakresie swojego oddziaływania, czyli defekty mogą powodować kolejne defekty bądź błędy, błędy mogą powodować kolejne błędy oraz awarie, a awarie mogą powodować kolejne awarie i wpływać na zbiór dostępnych usług systemu. Zostało to zobrazowane na rysunku 133.



Rys. 133. Zagrożenia i ich oddziaływania
Fig. 133. Threats and their influences

Zagrożenia te mogą wpływać na bezpieczeństwo integralności zasobów (ang. security), bezpieczeństwo funkcjonalne (ang. functional safety) oraz dostępność systemu (ang. availability) [55]. Zostały one opisane w kolejnych podrozdziałach.

4.2.1. Zagrożenia dostępu

Bezpieczeństwo w rozumieniu klasycznym (ang. security) jest cechą zabezpieczania informacji przed nieautoryzowanym ujawnieniem, przekazaniem, modyfikacją lub zniszcze-

niem, niezależnie czy przyczyna zagrożenia wynika z przypadku czy celowego działania [353], [346]. Problem kontroli integralności nie dotyczy tylko sieci komputerowej i jej izolowania od środowisk pozasystemowych. Dotyczy wszelkich środków zapobiegających nieupoważnionej interakcji z systemem.

Bezpieczeństwo dostępu do zasobów systemu było przez wiele lat ignorowane jako niedotyczące ISP. Wynikało to głównie ze szczególnych cech tych systemów i priorytetów bezpieczeństwa, różniących się od tradycyjnych, biurowych systemów komputerowych. ISP, z racji swojego fizycznego umiejscowienia i odizolowania, a zatem istnienia fizycznych utrudnień dostępu, były postrzegane jako systemy, do zasobów których nie ma realnej możliwości włamania. Ponadto nie rozpatrywano tego typu systemów jako atrakcyjnych dla hackerów lub innych osób znajdujących zysk lub radość z włamania do systemów komputerowych. Przykładem ignorancji może być stanowisko producentów dla przypadku ujawnionych sposobów włamań do systemów kontroli świateł ulicznych, gdzie stwierdzono, że systemy spełniają przyjęte standardy przemysłowe, które nie zawierają mechanizmów bezpieczeństwa [139].

W praktyce założenie o braku zagrożeń dla systemów zamkniętych było cały czas w znacznym stopniu błędne, gdyż nawet w odizolowaniu systemy te były narażone na bezpośrednie ataki wirusów i robaków pochodzących z umyślnych lub nieumyślnych infekcji lokalnych, z nośników przenośnych. Tego rodzaju infekcje nie wpływały z reguły na działanie systemów opartych na sieciach przemysłowych i PLC, gdyż nie istniało złośliwe oprogramowanie infekujące takie zasoby. W praktyce były one jednak w stanie utrudniać pracę współpracujących z nimi systemów nadrzędnych działających z użyciem komputerów klasy PC i sieci Ethernet. Rozwój dostępu zdalnego, w tym sieci Internet, sieci komórkowych, sieci bezprzewodowych itp., poszerzył ten problem, [120]. Integracja z sieciami publicznymi, prowadzona lepiej lub gorzej, spowodowała otwarcie ISP na ataki, które do tej pory były domeną sieci ogólnodostępnych (np. DoS, DDoS, File Inclusion, SQL Injection, MIM, XSS itp.). Ponadto poza typowymi i wyrafinowanymi atakami sfery biurowej, systemy ISP zostały wystawione na ataki, których motywacją są działania związane ze szpiegostwem gospodarczym, nieuczciwą konkurencją czy wręcz sabotażem i terroryzmem. W efekcie, ostatnimi laty można zaobserwować, że różne formy szkodliwego oprogramowania coraz bardziej dotykają zakresu działania systemów automatyki [232], [58], [45], [M90], [M79], [M95], [M98], [M11], [M10], [217], [139]. I chociaż zaawansowane techniki ochrony zasobów dla systemów domowych, biurowych i biznesowych rozwijały się i stale się rozwijają, to dla ISP rozwoju takiego nie ma, a wręcz istnieje brak dedykowanych technik ochrony dla tego typu systemów. Przykładami ostatnich ataków na systemy przemysłowe mogą być:

- Ataki na PLC. Dobrze znany medialnie atak dokonany został przez program robaka o nazwie Stuxnet [232]. Jest to rodzaj rootkita atakującego wybrane typy PLC i narzędzi deweloperskich, rozpowszechnianego właśnie przez przenośne nośniki pamięci i sieć. Złośliwe oddziaływanie polega na zmianie sterowania przekształtnikami częstotliwości sterującymi napędami np. pomp. Dokładna skala ataków nie jest znana, ale mówi się o tysiącach zakładów na całym świecie o różnym znaczeniu, od prostej produkcji po instalacje krytyczne (np. wirówki frakcjonujące stosowane przy wzbogacaniu uranu w elektrowniach) [217].
- Uzyskanie nieuprawnionego dostępu do systemu sterowania ruchem kolejowym. Zaburzony został system sterowania sygnalizacja świetlną, co skutkowało opóźnieniami pociągów [M90].
- Uzyskanie nieuprawnionego dostępu do obiektów stacji pomp wody. Akcje wielokrotnego włączania i wyłączania pompy spowodowały jej awarie [M79], [M95].
- Włamanie do systemu elektrociepłowni oraz routerów brzegowych [M98].
- Wielokrotne włamania do systemów metropolitalnych odpowiedzialnych za kontrolę urządzeń infrastruktur miejskich [M11].
- Atak na sygnalizację świetlną ulic, które podobno spełniały „standardy przemysłowe” [139].

Również planowo prowadzone audyty bezpieczeństwa prowadzą do wykrycia poważnych luk. Przykładem może być przepompownia wody [M10]. Dostęp uzyskano przez łącze bezprzewodowe bluetooth i hasło „0000”. Do poszukiwania zagrożeń można użyć internetowego serwisu Shodan (www.shodanhq.com). Wielu ekspertów bezpieczeństwa IT uważa go za wyszukiwarkę informacji dla hakerów. Serwis pozwala odnaleźć w sieci konkretne elementy systemów podatnych na ataki, w tym systemy ISP, np. SCADA.

Ostatnio też, szczególnie dla ISP, nasila się zagrożenie dość nietypowe. Dotyczy ono zdalnego oddziaływania na system przy użyciu ataku polem elektromagnetycznym (ang. EMW – electromagnetic weapon, EMA – EM attack). Urządzenia będące w stanie wygenerować wysokonapięciowe impulsy (np. ok. 1 ns ze zboczem narastającym 100 ps i polem EM o wartości tysięcy woltów na metr) mogą być wielkości przenośnej skrzynki [296]. Wystawienie elektroniki na impuls rzędu 5 kV na metr bez problemu i nieodwracalnie zniszczy typowy chip, a 2 kV/m może spowodować reset układu. Niezwykle istotne jest zatem, aby projektując ISP, mieć na uwadze tzw. higienę EM:

- używanie kabli światłowodowych zamiast miedzianych,
- ekranowanie kabli i koryt,

- zabezpieczanie kabli filtrami EM,
- budowanie bezpiecznych pomieszczeń dla krytycznych infrastruktur,
- dozbrojenie ścian uziemionymi okładzinami metalowymi (pręty, siatka itp.),
- dozbrojenie okien w uziemioną siatkę metalową,
- zapewnienie dystansu pomiędzy obszarem swobodnego dostępu a chronioną infrastrukturą,
- instalacja detektorów pola mogących ostrzegać przed próbami ataku.

Więcej szczegółów z tego zakresu można znaleźć w [296].

Dlatego zagadnienia ochrony bezpieczeństwa zasobów i ich integralności w rozumieniu ochrony informacji przed nieuprawnionym ujawnieniem, transferem, modyfikacją lub zniszczeniem, niezależnie przypadkowym lub celowym należy obecnie brać pod uwagę przy projektowaniu ISP. Istnieją trzy podstawowe wymagania bezpieczeństwa zasobów [92], [278]:

- dostępność (ang. availability) – zdolność do poprawnej realizacji zadań na rzecz innych zasobów,
- integralność (ang. integrity) – zdolność do zachowania poprawności i kompletności danych,
- poufność (ang. confidentiality) – gwarancja, że informacje nie będą udostępniane lub ujawniane nieuprawnionym osobom, podmiotom lub procesom.

Priorytety tych wymagań są różne od wymagań w systemach biurowych. Dla ISP najważniejszym wymaganiem jest dostępność [55]. Wynika to z ochrony procesu, gdyż bez dostępności zasobów prowadzenie procesu jest niemożliwe. Na drugim miejscu jest integralność danych. Poprawna realizacja zadań regulacji jest możliwa tylko z użyciem poprawnych i kompletnych danych zarówno w wymiarze ich wartości, jak i orientacji w czasie. Brak ochrony poufności nie uniemożliwia działania ISP, choć w wyniku takiego braku może dojść do zachwiania lub zaniku integralności, a nawet dostępności zasobów.

Zagrożenia tego typu nie są tylko problemem sieci, a proste rozdzielanie sieci zaporami ogniowymi (ang. firewall) nie jest wystarczające i nie umożliwia zachowania specyficznej obsługi informacji w ISP. Należy brać pod uwagę wszelkie środki, zapobiegające dostępowi nieuprawnionych ludzi lub oprogramowania do systemu krytycznego. Istnieją mechanizmy obronne niezwiązane z funkcjonalnością samego systemu i pracujące niejako obok. Mogą one zmniejszyć zagrożenia od nieuprawnionego dostępu do zasobów. Dzielą się one na trzy kategorie [55]:

- mechanizmy prewencji – są to podstawowe mechanizmy obrony, których celem jest uniemożliwienie skutecznego oddziaływania zagrożenia,

- mechanizmy detekcji – są to programy śledzące zachowanie ruchu sieciowego oraz aktywności węzłów i reagujące w sytuacjach, gdy odbiegają od przyjętego wzorca. Celem takich systemów jest wykrycie dokonanego już ataku,
- mechanizmy reakcji i przywracania – są to zadania w systemie, które mają za cel minimalizowanie szkód w przypadku wykrytego ataku oraz przywrócenie systemu do poprawnego działania, gdy w wyniku ataku pojawiły się jakieś defekty.

W przypadku komputerowych sieci przemysłowych stosuje się mechanizmy prewencji i detekcji. Najczęściej bezpieczeństwo zapewniają mechanizmy opisane w 2.4.4 w części dotyczącej wymagań.

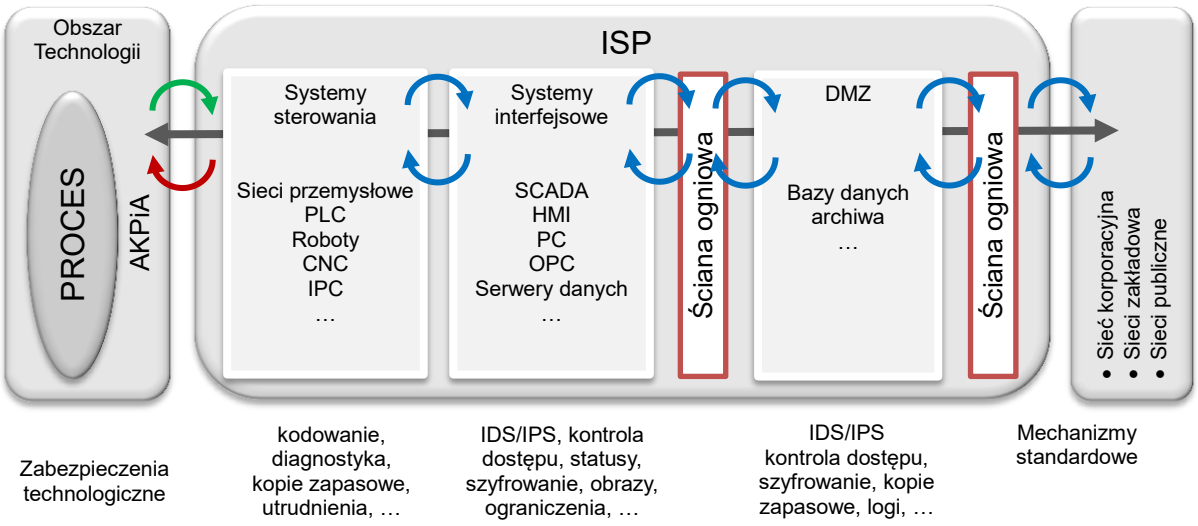
W przypadku węzłów systemu, czyli różnego rodzaju komputerów (PLC, ICS, DCS itp.) stosuje się wszystkie rodzaje mechanizmów dodatkowych, w tym wypunktowane poniżej i zilustrowane na rysunku 134.

- Utrudnienia lub ograniczenia w dostępie fizycznym do szafy, urządzeń i ich interfejsów (pamięć, urządzenia HID, sieci PAN itp.) oraz w uruchamianiu dodatkowych funkcji i kodu.
- Sprzęt i oprogramowanie kodujące, konwertujące, szyfrujące i dokonujące identyfikacji, autoryzacji i uwierzytelniania. Ponadto, wszelkie mechanizmy diagnostyczne sprzętu, jak wykrywanie awarii, zmian konfiguracji, sum kontrolnych, przeciwdziałania wykonywaniu kodu itp.
- Systemy klasy IDS (ang. Intruder Detection System). Są to systemy analizujące transakcje sieciowe węzła i sieci w czasie rzeczywistym pod kątem adresów i zawartości użytecznej. W ISP aktywności sieciowe są mało zmienne, dlatego działanie takich systemów może być bardzo przydatne i skuteczne. W praktyce nie ma specjalizowanych systemów tej klasy dedykowanych dla ISP, a te istniejące są uniwersalne i nie są dostosowane do specyfiki węzłów ISP.
- Systemy IPS (ang. Intrusion Prevention System). Działają tak jak IDS, z tym że dodatkowo mają funkcjonalności blokowania ataków w czasie rzeczywistym.
- Serwery dostępowe (pośredniczące, ang. hopping/jump server). Są to komputery serwerowe z odpowiednim oprogramowaniem[107] uruchomione w celu przekazywania usług/danych dostępnych w jednym segmencie sieci, np. sieci ISP, do systemów znajdujących się poza tym segmentem, np. do sieci zakładowej. Tworzy się tym samym obszar chroniony (bezpieczny). Serwery tego typu mogą separować ISP również od dostępu zdalnego uzyskiwanego z przestrzeni publicznej, np. przez tworzenie sieci wirtualnych

[107] Oprogramowanie specjalistyczne lub OS skonfigurowany na obsługę konkretnych usług i aplikacji.

(VLAN). Użytkownik może logować się do konkretnej usługi lub uzyskiwać dostęp do konkretnej aplikacji. DMZ są przykładem strefy tworzonej z użyciem takich serwerów.

- Strefy zdemilitaryzowane (DMZ – ang. Demilitary Zone). Jest to wydzielony obszar sieci znajdujący się pomiędzy siecią zewnętrzną a wewnętrzną strefą chronioną. W przypadku ataku, infekcji ulegają zasoby w strefie DMZ, a atak jest powstrzymany do obszaru tej strefy. Stosuje się głównie do zabezpieczania krytycznych obszarów wymiany danych. Dla ochrony pojedynczych węzłów ISP jest to mechanizm mało praktyczny. W DMZ nie powinny znajdować się systemy oddziałujące bezpośrednio z procesem, a jedynie te ich zasoby, które są wykorzystywane przez systemy współpracujące.
- Przywracanie obrazów węzłów. Proste i skuteczne działania reakcji przeważnie dokonywane ręcznie przez administratorów lub utrzymanie ruchu. Istotne jest aby nie tylko móc przywrócić komputer do działania, ale i mieć odpowiedni obraz do przywrócenia. Należy zatem dbać o cykliczne tworzenie takich obrazów. Dotyczy to zarówno kopii programów, konfiguracji, całej pamięci (w tym dysków), jak i backupów baz danych.



Rys. 134. Umiejscowienie środków ochrony dostępu do zasobów
Fig. 134. Location of security means for accessing resources

Ściany ogniowe, poza standardową funkcjonalnością filtracji pakietów, powinny mieć funkcjonalności, umożliwiające wprowadzanie i wyprowadzanie informacji bez zakłócania obiegu danych w sieciach przemysłowych [128].

Przy działaniach prewencyjnych ważne jest, aby ustalić kwestie kluczowe funkcjonowania zabezpieczeń [55] dotyczące:

- polityki bezpieczeństwa firmy,
- listy zagrożeń realnych i mało prawdopodobnych,

- ustaleń procedur kontrolnych weryfikujących działanie polityki bezpieczeństwa,
- ustaleń procedur testowych systemu weryfikujących, że wprowadzone mechanizmy nie wpływają negatywnie na działania ISP.

Czasami bezpieczeństwo uzyskuje się przez ukrycie detali architektury, struktury i implementacji systemu (ang. security through obscurity). Nie jest to jednak technika zalecana, gdyż bazuje tylko na potencjalnej niewiedzy atakujących i jest w sprzeczności z zasadą Kerckhoffsa [10]. Posadowienie bezpieczeństwa na założeniu, że ktoś czegoś nie wie, jak ktoś coś zrobił, jest błędne. W sytuacji jak ten ktoś się dowie, jak to coś jest zrealizowane, wówczas zabezpieczenia znikają. Współczesne metody inżynierii wstecznej (ang. reverse engineering) są na tyle rozwinięte, że zdecydowaną większość ukrywanych aspektów konstrukcyjnych można wykryć i zrozumieć.

4.2.2. Zagrożenia funkcjonalne

Bezpieczeństwo funkcjonalne (BF) jest to bezpieczeństwo w rozumieniu pewności działania (ang. safety), czyli że działanie ISP jest prawidłowe. Jest to cecha określająca konieczność eliminacji niedopuszczalnego ryzyka urazu fizycznego lub uszkodzenia zdrowia ludzi, bezpośrednio lub pośrednio w wyniku uszkodzenia mienia lub środowiska [M37]. Jest to jedna z najważniejszych cech działania przemysłowego systemu rozproszonego. Prawidłowe działanie spoczywa na projektantach i programistach. Błędy w tej dziedzinie mogą się pojawić tylko z powodu błędnych założeń, błędnych algorytmów lub błędów w kodzie. Systemy BF wykorzystujące środki sprzętowo-programowe do zabezpieczania procesu w nomenklaturze angielskojęzycznej nazywane są często SIS z ang. Safety Instrumented Systems [149].

Różnice w podejściach do tematu bezpieczeństwa funkcjonalnego wynikają z przyjętego punktu widzenia na umiejscowienie mechanizmów zapewniania poprawności działania. Teoria niezawodności, w zakresie zagrożeń funkcjonalnych, charakteryzuje usługi zapewniania bezpieczeństwa dostarczane przez system. Teoria bezpieczeństwa funkcjonalnego dotyczy natomiast wszelkich środków dodanych do systemu celem zapewnienia bezpieczeństwa jego działania. Istnieją zatem rozwiązania zwiększające bezpieczeństwo użytkowania układu, które są wbudowane w system i jego elementy oraz niezależne systemy bezpieczeństwa działające niejako obok ISP [125]. Cel działania w obu przypadkach jest taki sam, a funkcjonalności związane z bezpieczeństwem nie są związane z funkcjonalnościami ISP. Układy bezpieczeństwa wbudowane w system sterownia stanowią logicznie odrębna całość. Fizycznie pracują jako elementy urządzeń sterujących (np. moduł), ale logicznie nie są z nimi związane. Programowanie i konfiguracja oraz procedury związane z dokonywaniem zmian są odrębne.

Systemy bezpieczeństwa są systemami, których działanie nie objawia się przy poprawnej pracy systemu sterowania. Są one w pewnym sensie ukryte lub uśpione. Działanie jest obserwowalne dopiero po stwierdzeniu awarii lub podczas diagnostyki. W zależności od sposobu i zakresu działania systemy bezpieczeństwa można podzielić na:

- systemy zapewniające operacyjność urządzeń obiektowych (ang. fail-operational),
- systemy zapewniające bezpieczeństwo prowadzenia procesu (ang. fail-safe),
- systemy zapewniające bezpieczeństwo zasobów (ang. fail-secure).

ISP można określić jako bezpieczny, gdy jego praca nie powoduje **nadmiernego narażenia** na:

- negatywne oddziaływania na zdrowie i życie ludzi,
- negatywne oddziaływania na środowisko,
- straty materialne.

W praktyce nie ma systemów całkowicie bezpiecznych i wolnych od zagrożeń. Dlatego używa się pojęcia **ryzyka** wystąpienia zagrożenia, wartościując tym samym „nadmierne narażenie". Poziom ryzyka może być akceptowalny lub nieakceptowalny w danym przypadku.

Standardy BF (IEC61508) obejmują analizę zagrożeń (ang. hazard) i ryzyka (ang. risk) [315] oraz wymagania względem zabezpieczeń. Analiza zagrożeń dotyczy określania potencjalnych źródeł szkodliwego oddziaływania na człowieka, mienie lub środowisko, a także określania awarii (wypadków, ang. accident), czyli niepożądanych zdarzeń zachodzących w konsekwencji wystąpienia nieprzewidzianych zagrożeń. Analiza ryzyka dotyczy natomiast określania miary niebezpieczeństw płynących z wystąpienia awarii [39]. Formułuje się ją na podstawie częstości (lub prawdopodobieństwa) wystąpień oraz ich konsekwencji (dotkliwości, ang. severity) oddziaływania na otoczenie. Typowymi klasami częstości wystąpienia są:

- częste,
- prawdopodobne,
- okazjonalne,
- mało prawdopodobne,
- nieprawdopodobne,
- jednostkowe.

Typowymi klasami konsekwencji są:

- katastrofalne,
- krytyczne,

- marginalne,
- nieznaczne.

Działanie systemów BF (ang. safety-related systems) ma na celu zapewnienie określonego poziomu bezpieczeństwa układu przez zmniejszanie ryzyka. Wymagania względem bezpieczeństwa układu wynikają z analizy ryzyka i są określane przez prawdopodobieństwo wystąpienia awarii (PFD ang. Probability of Failure on Demand) lub akceptowalny poziom zagrożeń (THR – ang. Tolerable Hazard Rate). Najczęściej jednak wyznacza się je czterostopniową miarą jakościową SIL (ang. Safety-Integity Level) określającą w czterech przedziałach prawdopodobieństwo wystąpienia usterki lub prawdopodobną liczbę zadziałań układu do wystąpienia pierwszej usterki. Miara SIL i prawdopodobieństwa usterki zostały przedstawione w tabeli 6.

Tabela 6

Miary SIL względem PFD i konsekwencji

SIL	*PFD (THR)*	*Konsekwencje*
1	$10^{-6} \leq \text{PFD} < 10^{-5}$	niewielkie urazy i uszkodzenia
2	$10^{-7} \leq \text{PFD} < 10^{-6}$	poważne urazy, jednostkowa śmiertelność
3	$10^{-8} \leq \text{PFD} < 10^{-7}$	niejednostkowe zagrożenie życia
4	$10^{-9} \leq \text{PFD} < 10^{-8}$	liczna śmiertelność

Systemy bezpieczeństwa muszą być implementowane, gdy ryzyko związane z zagrożeniami jest niedopuszczalne. Istnieją techniki szacowania poziomu SIL do istniejącego ryzyka. Do najpopularniejszych należy metoda bazująca na konsekwencjach oraz na macierzy ryzyka [39]. Zilustrowane one zostały w tabeli 6 oraz w tabeli 7.

Tabela 7

Szacowanie SIL na podstawie macierzy ryzyka

	Klasa prawdopodobieństwa		
Klasa konsekwencji	*okazjonalne*	*prawdopodobne*	*częste*
katastrofalne	3	3	4
krytyczne	2	3	3
marginalne	1	2	3
nieznaczne	brak	1	2

W ogólnej masie większość aplikacji przemysłowych nie stanowi systemów krytycznych względem bezpieczeństwa funkcjonalnego. Jednak, gdy z analizy ryzyka wynika konieczność stosowania środków zabezpieczających, nigdy nie należy ignorować takich zagrożeń. Właściwa ocena zagrożeń i wdrożenie środków zaradczych jest kluczowe we współczesnych ISP [358].

4.2.3. Nowe techniki, nowe zagrożenia

W 3.5.5 przedstawiono zagadnienia integracji ISP z intersieciami. Z migracją dziedziny zastosowań ISP poza systemy lokalne nierozerwalnie wiążą się dzisiaj nowe technologie internetowe. Typowym przykładem jest wspominany wcześniej Internet rzeczy (IoT, ang. Internet of Things). Wiąże się z tym również bardzo poważny problem bezpieczeństwa zarówno dostępowego, jak i funkcjonalnego. Dzisiejsze rozwiązania IoT, niezależnie czy mające konotacje z ISP czy nie, są pełne dziur bezpieczeństwa [148]. Jest to poniekąd oczywiste, gdyż technologie IoT są aktualnie rozwijane, i główny nurt badawczy jest skierowany na tenże rozwój samych technologii, a nie ich zabezpieczeń. Nie jest to dobry stan, gdyż na etapie pierwszych wdrożeń lub otwierania nowych obszarów zastosowań systemy są niezwykle podatne na zagrożenia. Technologie IoT mają bardzo szerokie spektrum zastosowań, od podsystemów ISP przez aplikacje samochodowe do systemów bardzo osobistych, jak np. zbieranie informacji medycznych. Zagrożenia są zatem również bardzo różnorodne, dużo szersze niż tylko dla dziedziny ISP. Typowymi zagrożeniami w przypadku ISP są:

- podatność na kradzież informacji procesowej i technologicznej
  Wiąże się z problemem nieautoryzowanego dostępu do informacji utajnionych procesu i urządzeń IoT. Może skutkować szpiegostwem gospodarczym lub działaniami związanymi z nieuczciwą konkurencją.
- podatność na przejęcie kontroli
  Dotyczy przełączenia abonenta w obszar działania innego systemu i kontrolowania jego pracy niezależnie od macierzystego systemu.
- podatność na zakłócenie poprawnej pracy
  Umożliwia zdalne podłączenie i zmianę parametrów pracy urządzeń.

W obu ostatnich przypadkach celem mogą być działania zmierzające do osiągania lokalnych celów sprzecznych z celami biznesowymi przedsiębiorstwa. Podatności takie sprzyjają również działaniom sabotażowym, kryminogennym i terrorystycznym.

Problem z zapewnieniem bezpieczeństwa w systemach wykorzystujących abonentów IoT wynika z faktu, że urządzenia te są z założenia wyposażone w komputery o małej mocy obliczeniowej i niewielkie zasoby. Uniemożliwia to implementację znanych mechanizmów bezpieczeństwa. Dobrym rozwiązaniem jest implementacja tylko tych mechanizmów, które wiążą się z ochroną konkretnych zagrożeń wynikających ze specyficzności danego urządzenia Do tego zakres implementacji usług powinien być ograniczony ściśle do usługi, jaką oferuje dane urządzenie. Dla przykładu, napęd czy termopara nie musi obsługiwać całego stosu TCP/IP i wszystkich portów, układy telemetrii mogą w ogóle nie potrzebować transportu

TCP, można wbudowywać proste, ale precyzyjnie ukierunkowane układy typu firewall [128] itp.

Istnieją normy określające bądź związane z bezpieczeństwem systemów IoT. Jest to norma IEC62443 [M17], dotycząca standardów bezpieczeństwa w systemach sterowania i automatyzacji oraz normy związane z ogólnym bezpieczeństwem systemów cyfrowych, jak np. IEEE P1363 [M61] (kryptografia kluczy), P1619 [M89] (szyfrowanie), 802.1AE [M62] (dostęp do medium), które mogą być również stosowane do IoT. Ponadto, nad rozwojem i poprawnością wdrażania standardów bezpieczeństwa czuwa obecnie kilka organizacji i stowarzyszeń. Do znaczących należą:

- ISO: The International Standards Organization – istnieje specjalna grupa robocza zajmująca się IoT,
- IIC: The Industrial Internet Consortium – zajmuje się zapewnieniem kompatybilności rozwiązań,
- OIC: The Open Interconnect Consortium – grupuje producentów urządzeń i wytwórców technologii.

Wszystkie te organizacje mają na celu rozwój idei IoT, ale wszystkie wspierają również bezpieczeństwo takich rozwiązań.

4.2.4. Dostępność

ISP w zastosowaniach krytycznych muszą charakteryzować się wysoką dostępnością, czyli zdolnością systemu do wykonania wymaganej funkcji w danych warunkach i w danym momencie czasu lub w określonym przedziale czasu, przy założeniu że dostępne są wymagane do działania zasoby i dane zewnętrzne. Miarę dostępności zdefiniowano w następnym podrozdziale.

W ISP zarówno dostępność komputerów, jak i dostępność sieci jest kluczowym zagadnieniem. Jednym ze sposobów zwiększenia dostępności jest wykorzystanie redundancji środków transmisji w sieci, elementów węzłów systemu lub całych węzłów. Przez dodanie nadmiarowości sprzętu i oprogramowania system staje się bardziej odporny na usterki. Oznacza to, że w przypadku wystąpienia błędów w systemie, system nie zawodzi względem realizacji wymaganej funkcji, gdyż jest w stanie wyeliminować zdiagnozowane defekty. W przypadku awarii danego zasobu system przełącza się na kolejny dostępny. Najistotniejszym parametrem pracy układów redundantnych, dla zapewnienia dostępności systemu, jest czas przełączania (ang. switchover) z zachowaniem spójności stanu. Jeżeli czas ten jest mniejszy lub równy okresowi akwizycji i aktualizacji informacji na danym elemencie systemu oraz gdy po przełączeniu stan aktywnego elementu jest taki sam jak stan elementu uszkodzonego, to mówi się o pracy bezuderzeniowej. Z punktu widzenia procesu zmiana elementu na rezerwowy

jest wówczas niezauważalna. W przypadku gdy czas ten jest większy, może dojść do zaburzenia sterowania (lub regulacji) lub ogólnie do negatywnego oddziaływania takiego przełączania na proces.

Zwielokrotnione elementy systemu mogą pracować w trybie „gorącej rezerwy" (ang. hot standby), zimnej rezerwy (ang. cold standby) lub rezerwy ciepłej (ang. warm standby). W trybie gorącej rezerwy zwielokrotnione środki sprzętowo-programowe stanowią element systemu i są gotowe do przejęcia zadań i udostępnienia usług ze zwłoką wynikającą z cyklu diagnostycznego i synchronizacyjnego. W praktyce jest to działanie bezuderzeniowe. Rezerwa zimna oznacza utrzymywanie zapasu komponentów systemu na wewnętrznych stanach magazynowych utrzymania ruchu. W przypadku awarii istnieje możliwość wymiany uszkodzonego elementu systemu. Jest to tryb zdecydowanie oddziałujący na pracę układu, a wykonanie stosownej procedury w trybie bezuderzeniowym jest w praktyce mało realne. Tryb ciepłej rezerwy oznacza utrzymywanie zwielokrotnionych elementów w trybie gotowości do pracy, aczkolwiek bez integracji z działającym systemem. Oznacza to, że elementy są zabudowane na obiekcie i niejednokrotnie gotowe do pracy, ale przełączenie następuje w wyniku zadziałania czynników zewnętrznych, np. serwisu, utrzymania ruchu itp.

W klasycznym podejściu typu „standby" dane środki sprzętowo-programowe oczekują bezczynnie na awarię środków podstawowych i przejmują ich funkcję, gdy ta awaria wystąpi. Aby to było możliwe, muszą działać specjalne mechanizmy utrzymujące gotowość urządzeń do natychmiastowego uruchomienia. Są one wbudowane w redundantne elementy i działają w czasie rzeczywistym systemu. Mechanizmy te obsługują synchronizację stanu, detekcję awarii oraz wypracowują decyzję, który z elementów jest aktywny w systemie. Istnieje również podejście typu „duplex", bazujące na równoległym (jednoczesnym) działaniu zwielokrotnionych elementów bez określania elementu aktywnego. Wybór elementu, z którego korzystają inne elementy systemu, dokonywany jest przez te elementy.

Redundancja dotyczy także infrastruktury sieci i medium [228]. Zwielokrotnianie całej sieci nie jest zbyt efektywnym sposobem redundancji sieci, choćby ze względów ekonomicznych. Jest jednak stosowane. Do detekcji awarii danej sieci używane są wskaźniki stopy błędów bądź parametry czasowe transmisji. Lepszym, prostszym i tańszym podejściem jest dublowanie medium na bazie architektury dwukierunkowego pierścienia. Umożliwia ona zbudowanie systemu tolerującego awarię medium z przełączeniem bezuderzeniowym [202], [201], [78]. Jeszcze lepszym rozwiązaniem jest sieć typu „mesh", ale zarządzanie i obsługa przełączania w czasie rzeczywistym jest bardzo trudna do realizacji praktycznej. Przegląd metod redundancji dla Ethernetu czasu rzeczywistego można znaleźć w [390].

Istnieje możliwość uruchomienia zadań redundancji na różnych sieciach i z użyciem różnych protokołów. Uzyskuje się wówczas zwiększoną odporność na zaburzenia elektromagne-

tyczne [224] oraz możliwość zwiększenia wydajności połączenia, w sytuacji gdy sieci pracują poprawnie [49], [227].

4.2.5. Miary

Do mierzenia niezawodności ISP zostały zdefiniowane miary statystyczne [39], [207] wyrażające się wskaźnikami dotyczącymi:

- bezawaryjności (ang. failsafe) – wskaźnik opisujący, ile średnio czasu system pracuje bez awarii. Wskaźnik znany jest pod nazwą średniego czasu bezawaryjnej pracy MTBF (ang. Mean Time Between Failures),
- rzetelności (ang. reliability) – jest to wskaźnik określający prawdopodobieństwo, że system poprawnie wykona usługi w danych warunkach i w danym czasie liczonym od ostatniego czasu, kiedy zostały one poprawnie wykonane. Wskaźnik ten nazywa się powszechnie MTTF (ang. Mean Time To Failure),
- konserwacji (ang. maintainability) – jest to wskaźnik określający prawdopodobieństwo, że system poprawnie wykona usługi w danych warunkach i w danym czasie liczonym od czasu wystąpienia awarii. Powszechnie używa się nazwy MTTR (ang. Mean Time To Repair),
- postoju (ang. downtime) – wskaźnik opisujący średni czas postoju systemu, czyli czasu, w którym nie jest wykorzystywany do realizacji zadań produkcyjnych. Znany jest jako MDT (ang. Mean Down Time).

  $$MDT = MTTR + T_{ADM}$$

  gdzie T_{ADM} dotyczy wszelkich czynności administracyjnych związanych z przywróceniem systemu do stanu sprzed wystąpienia awarii,
- zbędnych zatrzymań (ang. nuisance trip) – wskaźnik określający średni czas nieuzasadnionych postojów. Znany jest jako $MTBF^{S}$ ($MTBF^{spurious}$),
- dostępności (ang. availability) – jest to wskaźnik określający prawdopodobieństwo, że system poprawnie wykona usługi w danych warunkach i w danym czasie. Wskaźnik wyrażany jest zależnością:

  $$A = \frac{MTTF}{MTTF + MTTR}$$

- awarii (ang. failure) – wskaźnik określający prawdopodobieństwo wystąpienia awarii przy obsłudze żądania znany jako PFD (ang. Probability of Failure on Demand).

  $$PFD = \frac{MDT}{MTBF + MDT}$$

- odwrotnością tego wskaźnika jest RRF (ang. Risk Reduction Factor), który jest wygodniejszy w użyciu ze względu na przeważnie małe wartości PFD.

$$RRF = \frac{1}{PFD}$$

- śmiertelności (ang. fatality) – wskaźnik dotyczący liczby ofiar śmiertelnych na milion osobogodzin pracy systemu. Znany jest jako FAR (ang. Fatal Accident Rate). Rozpatrywany czas dotyczy okresu narażenia na niebezpieczeństwo ludzi, a nie czasu pracy systemu.

4.2.6. Normalizacja

Podstawowym standardem dla bezpieczeństwa funkcjonalnego związanego z systemami rozproszonymi jest seria norm IEC61508 (ang. Functional Safety Guide) [M37]. Definiuje ona możliwe struktury sprzętu i oprogramowania oraz proces bezpiecznego projektowania. Normalizację ochrony struktur krytycznych obejmuje standard dotyczący wymagań względem bezpieczeństwa cyfrowego (ang. cyber security) IEC62351 [M49], IEC61334 [M33], IEC62056 [M47] oraz EN13757 [M5], [352], [116].

Dla systemów rozproszonych znormalizowaniu podlegają także protokoły sieciowe. Wszystkie proponowane rozwiązania związane z bezpieczeństwem dla sieci przemysłowych są zebrane w części trzeciej normy IEC61784 (ang. Functional Safety for Fieldbus) [M41]. Dla wszystkich sieci z normy sprecyzowane są potencjalne błędy działania jako źródła możliwych usterek i pokazane jest, jak zdefiniowane rozwiązania bezpieczeństwa zapewniają zachowanie integralności systemu. Typowymi metodami jest dodawanie dodatkowych danych w poszczególnych warstwach stosu (np. w postaci preambuł, sum kontrolnych, statusów jakości itp.) i redundancja [221], [228]. Środki programowe są zazwyczaj implementowane na szczycie stosów kanałów komunikacyjnych (zasada „black channel") [125]. Ustandaryzowane (tzn. niezależne od danej aplikacji) metody redundancji medium są zebrane w normie IEC62439 [M50] (ang. High Availability Networks).

5. PODSUMOWANIE

Komputeryzacja dotyczy prawie każdej współczesnej dziedziny życia zawodowego. Zakłady produkcyjne, aby mogły być konkurencyjne na rynku, muszą nie tylko unowocześniać swoje procesy, jak i aplikować całkiem nowe rozwiązania, ale również dynamicznie reagować na zapotrzebowanie i wymagania rynku. Współcześnie, komputeryzacja jest jedyną drogą w tych poczynaniach. Książka pokazuje wgląd w zestaw różnorodnych środków służących budowie systemów informatycznych na potrzeby obsługi procesów technologicznych, jak i współpracy z systemami nadrzędnymi. Nie stanowi jednak podręcznika obsługi urządzeń czy programów. Wskazuje ścieżki i możliwości, jakie w dziedzinie istnieją. Autor ma nadzieję, że choć trochę skłoni to rutynowych projektantów do odejścia od podejścia produktowego, a osoby zaczynające swoją przygodę z informatyką przemysłową ukierunkuje na kreatywną analizę problemów skutkującą właściwym doborem i tworzeniem produktów, a nie ich adaptacją.

Istotne jest, aby czytelnik wyniósł z lektury tej książki zrozumienie i wgląd w pojęcia, które są niezależne od komercyjnych technologii dostępnych na rynku, i między którymi z oczywistych względów istnieje rywalizacja. System klasy ISP można zbudować wykorzystując różne modele, podejścia, techniki i technologie. Każde rozwiązanie może być dobre z określonego punktu widzenia. Kluczowy jest dobór odpowiedniego rozwiązania do konkretnych wymagań.

Wykorzystywane pojęcie ISP jest pewnym skrótem myślowym odnoszącym się do szerokiego zakresu pojęciowego systemów informatyki przemysłowej. Ogólne zadanie, jakie jest stawiane przy komputerowym wspomaganiu procesów produkcji, to obsługa informacji pozyskiwanych z procesu i odpowiednie oddziaływanie na ten proces. Przez obsługę ogólnie rozumie się przetwarzanie, kodowanie, składowanie i przekazywanie informacji. Do zagadnień obsługi procesów produkcyjnych dochodzi bardzo wszechstronna interakcja z procesami biznesowymi. W efekcie tworzy się bardzo obszerna dziedzina zastosowań informatyki. Główne problemy z nią związane to zapewnianie odpowiedniej charakterystyki czasowej pracy, zapewnianie przepływu informacji w środowiskach heterogenicznych, obsługa rozproszenia urządzeń, procesów i funkcjonalności oraz wielokryterialne zapewnianie bezpieczeń-

stwa. Tak jak w każdym innym zastosowaniu informatyki, zawsze stoi za tym jakaś wiedza informatyczna i nieinformatyczna. Tworzenie ISP jest przez to działaniem interdyscyplinarnym. Należy zauważyć, że informatyka w swych zastosowaniach jest usługodawcza i w każdym przypadku wykraczającym poza systemy służące komputerom samym w sobie, aplikacje wymagają zastosowania technologii informatycznych na rzecz jakiejś dziedziny nieinformatycznej. Tworząc systemy księgowe, twórcy muszą poruszać się na krawędzi zagadnień ekonomiczno-finansowych, tworząc gry pojawia się styk z fizyką światła, ruchu, a nawet historią czy sztuką, tworząc rozwiązania webowe należy poznać dziedzinę klienta, aby uniknąć tworzenia bezsensownych, sztampowych produktów o ładnym interfejsie i niemal zerowej użyteczności. Tak samo jest z informatyką przemysłową. Jak zostało napisane we wstępie do stworzenia ISP, niezbędny jest zespół inżynierów i techników o różnym pochodzeniu zawodowym i ich współpraca. W systemach przemysłowych istnieje obecnie potrzeba rozwoju informatycznego. Dotyczy to zarówno konstrukcji systemów, jak i narzędzi.

Niniejsza książka nie wyczerpuje zagadnień ISP. Stanowi wstęp do tematyki, gdzie starano się o obiektywny wybór wątków. Planowane są kolejne pozycje z serii podstaw informatyki przemysłowej, dotyczące technologii sieciowych oraz sterowników klasy PLC.

Za wszelkie uwagi dotyczące zamieszczonych treści autor będzie wdzięczny, szczególnie jeśli będą one krytyczne i wzbudzające dyskusje. Uwagi proszę kierować na adres piotr.gaj@polsl.pl.

6. BIBLIOGRAFIA

Źródła literaturowe podzielono na trzy kategorie. Wynika to z ich charakteru. Podstawę pojęciową, źródła badań, twierdzeń i opinii stanowią książki i artykuły zamieszczone w 6.1, natomiast normy, raporty, dokumentacje firmowe są zgrupowane w 6.2.

Niniejszy spis może być przydatny do dalszego zgłębiania wiedzy. W poszukiwaniu źródeł warto zaglądać na źródła online zawierające bazy wiedzy, listy dyskusyjne oraz artykuły naukowe. Przykładami mogą być: [W25], [W4], [W36], [W11], [W5]. Współcześnie nie sposób uniknąć tego typu źródeł elektronicznych. Nie stanowią one kanonicznej bazy wiedzy, ale mogą być pomocne przy wyszukiwaniu potrzebnych informacji i uczestniczeniu w życiu społeczności zajmującej się daną tematyką. Zamieszczono je w podrozdziale 6.3.

6.1. Źródła bibliograficzne

[1] Abeni L., Manica N., Palopoli L.: Efficient and robust probabilistic guarantees for real-time tasks. Journal of Systems and Software, Vol. 85, No. 5, 2012, pp. 1147÷1156.

[2] Adamczewski P.: Słownik informatyczny. Helion 2005, ss. 1÷288.

[3] Adamczyk P.: Modemy GSM od Astraada. Biuletyn Automatyki, Z. 81, Nr 3, 2014, ss. 18÷19.

[4] Ahmed K., Gregory M.: Integrating wireless sensor networks with cloud computing. [in:] Mobile Ad-hoc and Sensor Networks (MSN), 2011 Seventh International Conference on, Dec 2011, pp. 364÷366.

[5] Akerberg J., Björkman M.: Exploring network security in PROFIsafe [in:] Computer Safety, Reliability, and Security, vol. 5775 of Lecture Notes in Computer Science Springer, Berlin, Heidelberg 2009, pp. 67÷80.

[6] Akillioglu H., Onori M.: Evolvable production systems and impacts on production planning. [in:] Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on, May 2011, pp. 1÷6.

[7] Ali S., Akbar S., Pedram V.: A remote and virtual PLC laboratory via smartphones. [in:] E-Learning and E-Teaching (ICELET), 2013 Fourth International Conference on, Feb 2013, pp. 63÷68.

[8] Ali-Yahiya T.: Understanding LTE and its performance. SpringerLink: Bücher, Springer, New York 2011.

[9] Alur R., Hristu-Varsakelis D., Arzen K., Levine W., Baillieul J., Henzinger T.: Handbook of networked and embedded control systems. Control Engineering, Birkhäuser, Boston 2007.

[10] Anderson R. J.: Security engineering: a guide to building dependable distributed systems, 1st ed. John Wiley & Sons, New York, NY, USA 2001.

[11] Araujo J., Mazo M., Anta A., Tabuada P., Johansson K.: System architectures, protocols and algorithms for aperiodic wireless control systems. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 1, Feb 2014, pp. 175÷184.
[12] Awrejcewicz J., Wodzicki W.: Podstawy automatyki: teoria i przykłady. Wyd. Politechniki Łódzkiej, Łódź 2001.
[13] Barnes M.: Practical variable speed drives and power electronics. Electronics & Electrical, Elsevier Newnes June 2003.
[14] Barni M., Tondi B.: Lecture notes on information theory and coding, Universita degli Studi di Siena Facolta di Ingegneria, Siena, Italy 2012.
[15] Baxter N. L., Jesus H. D.: Remote machinery monitoring – a developing industry. Sound and Vibration, Vol. 42, No. 5, 2008, pp. 20÷24.
[16] Bayrak G., Renzhin D., Vogel-Heuser B.: Integration of control loops in an UML based engineering environment for PLC. [in:] Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on, Sept 2011, pp. 1÷8.
[17] Beck K.: Implementation patterns. Addison-Wesley signature series, Pearson Education, India 2007.
[18] Bell M.: Service-oriented modeling (SOA): service analysis, design, and architecture. NetLibrary, Inc, Wiley 2008.
[19] Bemporad A., Heemels M., Vejdemo-Johansson M.: Networked control systems. Lecture Notes in Control and Information Sciences, Springer, London 2010.
[20] Bertocco M., Gamba G., Sona A., Vitturi S.: Performance measurements of CSMA/CA-based wireless sensor networks for industrial applications. [in:] Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE, May 2007, pp. 1÷6.
[21] Bertolotti I., Manduchi G.: Real-time embedded systems: open-source operating systems perspective. Embedded Systems, Taylor & Francis 2012.
[22] Bi Z., Xu L. D., Wang C.: Internet of things for enterprise systems of modern manufacturing. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 2, May 2014, pp. 1537÷1546.
[23] Bigewski Z., Cupek R., Kwiecień A.: Stosowanie procedur języka C w celu zwiększenia częstotliwości dostępu do sieci komunikacyjnej abonentów systemu rozproszonego. Studia Informatica, Z. Vol. 24, nr 3, 2003, ss. 255÷269.
[24] Binns C.: Introduction to nanoscience and nanotechnology. Wiley Survival Guides in Engineering and Science, Wiley 2010.
[25] Black G., Vyatkin V.: Intelligent component-based automation of baggage handling systems with IEC 61499. Automation Science and Engineering, IEEE Transactions on, Vol. 7, No. 2, April 2010, pp. 337÷351.
[26] Bojic I., Granjal J., Monteiro E., Katusic D., Skocir P., Kusek M., Jezic G.: Communication and security in machine-to-machine systems [in:] Wireless Networking for Moving Objects, vol. 8611 of Lecture Notes in Computer Science Springer 2014, pp. 255÷281.
[27] Bolkowski S.: Teoria obwodów elektrycznych. Podręczniki Akademickie: Elektrotechnika, WNT, Warszawa 2008.
[28] Bolton W.: Programmable logic controllers. Programmable Logic Controllers Series, Elsevier Science 2011.
[29] Borrione D.: Advances in design methods from modeling languages for embedded systems and SoC's: selected contributions on specification, design, and verification from FDL 2009. Lecture Notes in Electrical Engineering, Springer, Netherlands 2010.
[30] Boswarthick D., Elloumi O., Hersent O.: M2M communications: a systems approach. Wiley 2012.
[31] Boucher T.: Computer automation in manufacturing: an introduction. Springer, US 2012.
[32] Boulanger J.: Safety management of software-based equipment. FOCUS Series, Wiley 2013.
[33] Bradford R., Gracki K.: Podstawy sieci komputerowych. WKŁ, Warszawa 2009.

[34] Bregulla M., Cupek R., Fojcik M.: Profinet CBA via internet [in:] Contemporary Aspects of Computer Networks, vol. II WKŁ, Warszawa 2008, pp. 223÷230.
[35] Breivold H., Jansen A., Sandstrom K., Crnkovic I.: Virtualize for architecture sustainability in industrial automation. [in:] Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on, Dec 2013, pp. 409÷415.
[36] Broel-Plater B.: Układy wykorzystujące sterowniki PLC: projektowanie algorytmów sterowania. PWN, Warszawa 2009.
[37] Brumbach M., Clade J.: Electronic variable speed drives. Cengage Learning 2015.
[38] Buchanan B.: The handbook of data communications and networks., vol. 1 Springer, US 2010.
[39] Buja G., Menis R.: Dependability and functional safety: applications in industrial electronics systems. Industrial Electronics Magazine, IEEE, Vol. 6, No. 3, Sept 2012, pp. 4÷12.
[40] Burns A., Wellings A.: Real-time systems and programming languages: ada 95, real-time java, and real-time posix. International computer science series, Addison-Wesley 2001.
[41] Buttazzo G.: Hard real-time computing systems: predictable scheduling algorithms and applications. Real-Time Systems Series, Springer 2011.
[42] Calero C. M., Piattini M.: Green in software engineering. Prentice Hall, India 2015.
[43] Candido G., Colombo A., Barata J., Jammes F.: Service-oriented infrastructure to support the deployment of evolvable production systems. Industrial Informatics, IEEE Transactions on, Vol. 7, No. 4, Nov 2011, pp. 759÷767.
[44] Candido G., Jammes F., de Oliveira J., Colombo A.: SOA at device level in the industrial domain: assessment of OPC UA and DPWS specifications. [in:] Industrial Informatics (INDIN), 2010 8th IEEE International Conference on, July 2010, pp. 598÷603.
[45] Carcano A., Coletta A., Guglielmi M., Masera M., Fovino I., Trombetta A.: A multidimensional critical state analysis for detecting intrusions in SCADA systems. Industrial Informatics, IEEE Transactions on, Vol. 7, No. 2, May 2011, pp. 179÷186.
[46] Cardey S.: Modelling language. Natural Language Processing, John Benjamins Publishing Company 2013.
[47] Cavallo A., Setola R., Vasca F.: Using MATLAB, SIMULINK and control system toolbox: a practical approach. The MATLAB curriculum series, Prentice Hall 1996.
[48] Cechich A., Piattini M., Vallecillo A.: Component-based software quality: methods and techniques. Lecture Notes in Computer Science, Springer 2003.
[49] Cena G., Valenzano A., Vitturi S.: Hybrid wired/wireless networks for real-time communications. Industrial Electronics Magazine, IEEE, Vol. 2, No. 1, March 2008, pp. 8÷20.
[50] Cengic G., Akesson K.: Definition of the execution model used in the fuber IEC 61499 runtime environment. [in:] Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on, July 2008, pp. 301÷306.
[51] Cengic G., Akesson K.: On formal analysis of IEC 61499 applications, part a: modeling. Industrial Informatics, IEEE Transactions on, Vol. 6, No. 2, May 2010, pp. 136÷144.
[52] Chang-Cheng Z., Jian-Ming X., Yao J., Jie M., Lin-Tao X.: Design of servo drive slaves based on EtherCAT. [in:] Control and Decision Conference (CCDC), 2015 27th Chinese, May 2015, pp. 5999÷6004.
[53] Charoy A., Pochanke Z.: Ekrany, filtry, kable i przewody ekranowane. Nr t. 3 [w] Kompatybilność elektromagnetyczna: zakłócenia w urządzeniach elektronicznych. Zasady i porady instalacyjne, WNT, Warszawa 2000.
[54] Cheminod M., Bertolotti I., Durante L., Maggi P., Pozza D., Sisto R., Valenzano A.: Detecting chains of vulnerabilities in industrial networks. Industrial Informatics, IEEE Transactions on, Vol. 5, No. 2, May 2009, pp. 181÷193.
[55] Cheminod M., Durante L., Valenzano A.: Review of security issues in industrial networks. Industrial Informatics, IEEE Transactions on, Vol. 9, No. 1, Feb 2013, pp. 277÷293.
[56] Chen D., Nixon M., Han S., Mok A., Zhu X.: WirelessHART and IEEE 802.15.4e. [in:] Industrial Technology (ICIT), 2014 IEEE International Conference on, Feb 2014, pp. 760÷765.

[57] Chen J., Li L., Wang L.: The application of Profibus technology in the Fengchan river project's electronic control system reform. [in:] Intelligent Networks and Intelligent Systems (ICINIS), 2012 Fifth International Conference on, Nov 2012, pp. 130÷133.
[58] Chen T., Abu-Nimeh S.: Lessons from stuxnet. Computer, Vol. 44, No. 4, April 2011, pp. 91÷93.
[59] Chi Q., Yan H., Zhang C., Pang Z., Xu L. D.: A reconfigurable smart sensor interface for industrial WSN in IoT environment. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 2, May 2014, pp. 1417÷1425.
[60] Chuanying Y., He L., Zhihong L.: Implementation of migrations from Class OPC to OPC UA for data acquisition system. [in:] System Science and Engineering (ICSSE), 2012 International Conference on, June 2012, pp. 588÷592.
[61] Cirka G.: Application of foundation – fieldbus communications for level measurement. [in:] Control and Automation, 2003. ICCA '03. Proceedings. 4th International Conference on, June 2003, pp. 68÷72.
[62] Collins K.: PLC programming for industrial automation. Exposure 2007.
[63] Comer D.: Computer networks and internets. Prentice Hall 2014.
[64] Comer D., Grudziński G., Schubert A.: Sieci komputerowe i intersieci. WNT 2003.
[65] Costache C., Sandu F., Balan T., Nedelcu A., Covei A.: Business integration of industrial communications with cloud computing. [in:] Communications (COMM), 2014 10th International Conference on, May 2014, pp. 1÷4.
[66] Coulouris G., Dollimore J., Kindberg T., Płoski Z.: Systemy rozproszone: podstawy i projektowanie. WNT, Warszawa 1998.
[67] Cseh C., Jasperneite J.: Emerging data transfer technologies for factory communication. [in:] Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE, Aug 1998, vol. 4, pp. 2122÷2126 vol.4.
[68] Cucej Z., Gleich D., Kaiser M., Planinsic P.: Industrial networks. [in:] Electronics in Marine, 2004. Proceedings Elmar 2004. 46th International Symposium, June 2004, pp. 59÷66.
[69] Cucinotta T., Mancina A., Anastasi G., Lipari G., Mangeruca L., Checcozzo R., Rusina F.: A real-time service-oriented architecture for industrial automation. Industrial Informatics, IEEE Transactions on, Vol. 5, No. 3, Aug 2009, pp. 267÷277.
[70] Cui W., Meng X., Liu S.: A service-oriented architecture of virtual enterprise for manufacturing industry. [in:] Computer Supported Cooperative Work in Design (CSCWD), 2010 14th International Conference on, April 2010, pp. 373÷377.
[71] Cupek R., Fojcik M., Sande O.: Object oriented vertical communication in distributed industrial systems [in:] Computer Networks, vol. 39 of Communications in Computer and Information Science Springer, Berlin, Heidelberg 2009, pp. 72÷78.
[72] Cupek R., Jestratjew A.: Zastosowanie algorytmu EDF do szeregowania zadań cyklicznych realizowanych przez sterownik swobodnie programowalny. [w:] Gaj P., Kwiecień A. (red.): X Konferencja Systemy Czasu Rzeczywistego, Ustroń, 15-18 września 2003. Materiały konferencyjne., 2003, ss. 131÷142.
[73] Czachórski T.: Analityczne modele kolejkowe w ocenie efektywności pracy systemów i sieci kolejkowych. Pro Dialog, Z. 16, 2003, ss. 90÷111.
[74] Czerski Z.: Komunikacje z przemiennikami częstotliwości Astraada DRV po protokole CANopen. Biuletyn Automatyki, Z. 81, Nr 3, 2014, ss. 32÷33.
[75] Czybik B., Hausmann S., Heiss S., Jasperneite J.: Performance evaluation of MAC algorithms for real-time Ethernet communication systems. [in:] Industrial Informatics (INDIN), 2013 11st IEEE International Conference on, July 2013, pp. 676÷681.
[76] Day G. W.: Sustainability. The Institute, The IEEE news source, Dec 2012, pp. 11÷11.
[77] Dębowski A.: Automatyka podstawy teorii. WNT 2012.

[78] De Dominicis C. M., Ferrari P., Flammini A., Rinaldi S., Quarantelli M.: On the use of IEEE 1588 in existing IEC 61850-based SASs: current behavior and future challenges. Instrumentation and Measurement, IEEE Transactions on, Vol. 60, No. 9, Sept 2011, pp. 3070÷3081.
[79] De Pellegrini F., Miorandi D., Vitturi S., Zanella A.: On the use of wireless networks at low level of factory automation systems. Industrial Informatics, IEEE Transactions on, Vol. 2, No. 2, May 2006, pp. 129÷143.
[80] Decotignie J.-D.: The many faces of industrial Ethernet [past and present]. Industrial Electronics Magazine, IEEE, Vol. 3, No. 1, March 2009, pp. 8÷19.
[81] Diaz J., Garcia D., Kim K., Lee C.-G., Lo Bello L., Lopez J., Min S. L., Mirabella O.: Stochastic analysis of periodic real-time systems. [in:] Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE, 2002, pp. 289÷300.
[82] Diaz-Cacho M., Delgado E., Falcon P., Barreiro A.: IoT integration on industrial environments. [in:] Factory Communication Systems (WFCS), 2015 IEEE World Conference on, May 2015, pp. 1÷7.
[83] Dietrich D., Neumann P., Schweinzer H.: Fieldbus technology: systems integration, networking, and engineering. [in:] Proceedings of the Fieldbus Conference FeT'99 in Magdeburg, Federal Republic of Germany, September 23-24,1999, Vienna, 2012, Springer.
[84] Dobbs R.: Electromagnetic waves. Student Physics Series, Springer, Netherlands 2013.
[85] Drath R., Horch A.: Industrie 4.0: hit or hype? [industry forum]. Industrial Electronics Magazine, IEEE, Vol. 8, No. 2, June 2014, pp. 56÷58.
[86] Dubinin V., Vyatkin V.: On definition of a formal model for IEC 61499 function blocks. EURASIP J. Embedded Syst., Vol. 2008, Apr 2008, pp. 7:1÷7:10.
[87] Dubinin V., Vyatkin V., Hanisch H.-M.: Modelling and verification of IEC 61499 applications using prolog. [in:] Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on, Sept 2006, pp. 774÷781.
[88] Dugalic B., Mishev A.: ISO software quality standards and certification. [in:] Budimac Z., Ivanovic M., Radovanovic M. (eds.): Local Proceedings of the Fifth Balkan Conference in Informatics (BCI Local), 2012, vol. 920 of CEUR Workshop Proceedings, CEUR-WS.org, pp. 113÷116.
[89] Dunning G.: Introduction to programmable logic controllers. Cengage Learning 2005.
[90] Durkop L., Jasperneite J., Fay A.: An analysis of real-time Ethernets with regard to their automatic configuration. [in:] Factory Communication Systems (WFCS), 2015 IEEE World Conference on, May 2015, pp. 1÷8.
[91] Durlik I.: Inżynieria zarządzania. Strategia i projektowanie systemów produkcyjnych, cz. 1 i 2. Agencja Wydawnicza Placet Warszawa 1996.
[92] Dzung D., Naedele M., von Hoff T., Crevatin M.: Security for industrial communication systems. Proceedings of the IEEE, Vol. 93, No. 6, June 2005, pp. 1152÷1177.
[93] Elattar M., Jasperneite J.: Using LTE as an access network for internet-based cyber-physical systems. [in:] Factory Communication Systems (WFCS), 2015 IEEE World Conference on, May 2015, pp. 1÷7.
[94] Elbert B., Martyna B.: Client/server computing: architecture, applications, and distributed systems management. Artech House computer science library, Artech House 1994.
[95] Eljasz D.: Analiza parametrów czasowych w systemach pomiarowo-sterujących z wykorzystaniem teorii masowej obsługi i metod szeregowania zadań. Pomiary Automatyka Kontrola, Z. 11(11), 2010, ss. 1342–1344.
[96] Erl T.: Service-oriented architecture: concepts, technology, and design. Pearson Education, India 2005.
[97] Estevez E., Marcos M.: Model-based validation of industrial control systems. Industrial Informatics, IEEE Transactions on, Vol. 8, No. 2, May 2012, pp. 302÷310.

[98] Eurich M., Boutellier R.: Middleware integration platforms: a new challenge to business models of ICT companies: unleashing the business potential of horizontalization. [in:] e-Business (ICE-B), Proceedings of the 2010 International Conference on, July 2010, pp. 1÷6.
[99] Fang S., Xu L. D., Zhu Y., Ahati J., Pei H., Yan J., Liu Z.: An integrated system for regional environmental monitoring and management based on internet of things. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 2, May 2014, pp. 1596÷1605.
[100] Fang-ying C., Jian-feng L., Hao Z.: Factory planning and digital factory. [in:] Audio Language and Image Processing (ICALIP), 2010 International Conference on, Nov 2010, pp. 499÷502.
[101] Felser M.: The fieldbus standard: history and structure. [in:] Technology Leadership Day 2002, Organised by MICROSWISS Network, HTA Luzern, Oct 2002.
[102] Felser M.: Real-time Ethernet – industry prospective. Proceedings of the IEEE, Vol. 93, No. 6, June 2005, pp. 1118÷1129.
[103] Felser M.: Real time Ethernet: standardization and implementations. [in:] Industrial Electronics (ISIE), 2010 IEEE International Symposium on, July 2010, pp. 3766÷3771.
[104] Felser M.: Profibus manual: a collection of information explaining Profibus networks. Epubli GmbH, Berlin Aug 2011.
[105] Felser M., Jasperneite J., Gaj P.: Guest editorial special section on distributed computer systems in industry. Industrial Informatics, IEEE Transactions on, Vol. 9, No. 1, Feb 2013, pp. 181÷181.
[106] Felser M., Sauter T.: The fieldbus war: history or short break between battles? [in:] Factory Communication Systems, 2002. 4^{th} IEEE International Workshop on, 2002, pp. 73÷80.
[107] Felser M., Sauter T.: Standardization of industrial Ethernet – the next battlefield? [in:] Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on, Sept 2004, pp. 413÷420.
[108] Ferenc M.: Podstawy automatyki. Skrypty Uczelniane Politechniki Śląskiej, Dział Wydawnictw Politechniki Śląskiej, Gliwice 1987.
[109] Ferrari P., Flammini A., Vitturi S.: Response times evaluation of Profinet networks. [in:] Industrial Electronics, 2005. ISIE 2005. Proceedings of the IEEE International Symposium on, June 2005, vol. 4, pp. 1371÷1376.
[110] Fiset J.-Y.: Human-machine interface design for process control applications. International Society of Automation, USA Oct 2009.
[111] Fisher A., Jacobson C. A., Lee E. A., Murray R. M., Sangiovanni-Vincentelli A., Scholte E.: Industrial cyber-physical systems – iCyPhy [in:] Complex Systems Design & Management, Springer International Publishing 2014, pp. 21÷37.
[112] Forouzan B., Mosharraf F.: Foundations of computer science. Introduction to CS Series, Cengage Learning 2008.
[113] Fowler M.: UML distilled: a brief guide to the standard object modeling language. Addison-Wesley object technology series, Addison-Wesley 2004.
[114] Frey G.: Software quality in logic controller programming. [in:] Systems, Man and Cybernetics, 2002 IEEE International Conference on, Oct 2002, vol. 1, pp. 515÷520.
[115] Frey G., Litz L.: Formal methods in PLC programming. [in:] Systems, Man, and Cybernetics, 2000 IEEE International Conference on, 2000, vol. 4, pp. 2431÷2436 vol.4.
[116] Fries S., Hof H., Seewald M.: Enhancing IEC 62351 to improve security for energy automation in smart grid environments. [in:] Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on, May 2010, pp. 135÷142.
[117] Fummi F., Martini S., Monguzzi M., Perbellini G., Poncino M.: Modeling and analysis of heterogeneous industrial networks architectures. [in:] Design, Automation and Test in Europe Conference and Exhibition, Proceedings, Feb 2004, vol. 3, pp. 342÷343.
[118] Gaj P.: Dobór protokołów dla interfejsów komunikacyjnych urządzeń współpracujących z przemysłowymi systemami kontrolno-nadzorczymi. Studia Informatica, Z. Vol. 23, nr 3, 2002, ss. 255÷272.

[119] Gaj P.: Zastosowanie protokołu TCP/IP do transmisji informacji dla potrzeb przemysłowych systemów kontrolno-nadzorczych. Rozprawa doktorska, Politechnika Śląska. Wydział Automatyki, Elektroniki i Informatyki, Gliwice 2003, ss. 1÷197 (Promotor: prof. dr hab. inż. Józef Ober).
[120] Gaj P.: Przyjazny monitoring – zdalnie, bezprzewodowo i w internecie. Napędy i Sterowanie, Racibórz, Z. 3, Nov 2009, ss. 114÷116.
[121] Gaj P.: Pessimistic useful efficiency of EPL network cycle [in:] Computer Networks, vol. 79 of Communications in Computer and Information Science Springer, Berlin, Heidelberg 2010, pp. 297÷305.
[122] Gaj P.: Przemysłowy Ethernet – szybko i wydajnie? Napędy i Sterowanie, Racibórz, Nov 2010, ss. 78÷84.
[123] Gaj P.: The concept of a multi-network approach for a dynamic distribution of application relationships [in:] Computer Networks, vol. 160 of Communications in Computer and Information Science Springer, Berlin, Heidelberg 2011, pp. 328÷337.
[124] Gaj P.: Aspekty bezpieczeństwa w informatycznych systemach przemysłowych. Napędy i Sterowanie, Nr 4, kwiecień 2013, ss. 80÷90.
[125] Gaj P., Jasperneite J., Felser M.: Computer communication within industrial distributed environment – a survey. Industrial Informatics, IEEE Transactions on, Vol. 9, No. 1, Feb 2013, pp. 182÷189.
[126] Gaj P., Kwiecień B.: Useful efficiency in cyclic transactions of Profinet IO. Studia Informatica, Vol. 31, No. 1, 2010.
[127] Gaj P., Malinowski A., Sauter T., Valenzano A.: Guest editorial: distributed data processing in industrial applications. Industrial Informatics, IEEE Transactions on, Vol. 11, No. 3, June 2015, pp. 737÷740.
[128] Gaj P., Ober J.: Firewall++ do zastosowań w systemach przemysłowych. Studia Informatica, Z. 24, Nr 3, 2003, ss. 207÷220.
[129] Gaj P., Ober J.: Problemy z wykorzystaniem sieci Ethernet w aplikacjach przemysłowych. Studia Informatica, Z. Vol. 24, nr 3, 2003, ss. 149÷159.
[130] Gaj P., Skrzewski M., Stój J., Flak J.: Virtualization as a way to distribute pc-based functionalities. IEEE Transactions on Industrial Informatics, Vol. 11, No. 3, June 2015, pp. 763÷770.
[131] Gamma E., Walczak T.: Wzorce projektowe: elementy oprogramowania obiektowego wielokrotnego użytku. Helion 2010.
[132] Gammack J., Hobbs V., Pigott D.: The book of informatics. Cengage Learning, Australia 2011.
[133] Garcia Valls M., Lopez I., Villar L.: iLAND: an enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems. Industrial Informatics, IEEE Transactions on, Vol. 9, No. 1, Feb 2013, pp. 228÷236.
[134] Gawali D., Sharma V.: FPGA based Micro-PLC design approach. [in:] Advances in Computing, Control, Telecommunication Technologies, 2009. ACT '09. International Conference on, Dec 2009, pp. 660÷663.
[135] Gawin B.: Systemy informatyczne w zarządzaniu procesami workflow, 1 ed. PWN, Warszawa 2015.
[136] Gerber C., Hanisch H.-M., Ebbinghaus S.: From IEC 61131 to IEC 61499 for distributed systems: a case study. EURASIP J. Embedded Syst., Vol. 2008, Apr 2008, pp. 4:1÷4:8.
[137] Gessner D., Barranco M., Proenza J.: Design and verification of a media redundancy management driver for a CAN star topology. Industrial Informatics, IEEE Transactions on, Vol. 9, No. 1, Feb 2013, pp. 237÷245.
[138] Ghanaim A., Frey G.: Modeling and control of closed-loop networked PLC-systems. [in:] American Control Conference (ACC), 2011, June 2011, pp. 502÷508.

[139] Ghena B., Beyer W., Hillaker A., Pevarnek J., Halderman J. A.: Green lights forever: analyzing the security of traffic infrastructure. [in:] 8th USENIX Workshop on Offensive Technologies (WOOT 14), San Diego, CA, Aug 2014, USENIX Association.
[140] Giirbea A., Nechifor S., Sisak F., Perniu L.: Design and implementation of an OLE for process control unified architecture aggregating server for a group of flexible manufacturing systems. Software, IET, Vol. 5, No. 4, Aug 2011, pp. 406÷414.
[141] Givehchi O., Imtiaz J., Trsek H., Jasperneite J.: Control-as-a-service from the cloud: a case study for using virtualized PLCs. [in:] Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on, May 2014, pp. 1÷4.
[142] Givehchi O., Trsek H., Jasperneite J.: Cloud computing for industrial automation systems – a comprehensive overview. [in:] Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on, Sept 2013, pp. 1÷4.
[143] Güngör C. V., Hancke G. P.: Industrial Wireless Sensor Networks. Applications, Protocols, and Standards. CRC Press, Boston 2013.
[144] Goźlińska E.: Maszyny elektryczne. WSiP, Warszawa 2013.
[145] Goodliffe P.: Code craft: the practice of writing excellent code. Safari Books Online, No Starch Press 2007.
[146] Gotfryd M.: Podstawy telekomunikacji: pytania, zadania, problemy: materiały pomocnicze. Materiały Pomocnicze, Oficyna Wydawnicza Politechniki Rzeszowskiej 2010.
[147] Gozalvez J., Sepulcre M., Palazon J. A.: On the feasibility to deploy mobile industrial applications using wireless communications. Computers in Industry, Vol. 65, No. 8, 2014, pp. 1136÷1146.
[148] Grau A.: Can you trust your fridge? Spectrum, IEEE, Vol. 52, No. 3, March 2015, pp. 50÷56.
[149] Gruhn P., Cheddie H. L.: Safety instrumented systems – design, analysis, and justification, 2 ed. International Society of Automation 2005.
[150] Guarese G., Sieben F., Webber T., Dillenburg M., Marcon C.: Exploiting modbus protocol in wired and wireless multilevel communication architecture. [in:] Computing System Engineering (SBESC), 2012 Brazilian Symposium on, Nov 2012, pp. 13÷18.
[151] Gungor V., Hancke G.: Industrial wireless sensor networks: challenges, design principles, and technical approaches. Industrial Electronics, IEEE Transactions on, Vol. 56, No. 10, Oct 2009, pp. 4258÷4265.
[152] Gupta A.: Industrial safety and environment. Laxmi Publications(P) Limited 2006.
[153] Hajduk Z., Trybus B., Sadolewski J.: Architecture of fpga embedded multiprocessor programmable controller. IEEE Transactions on Industrial Electronics, Vol. 62, No. 5, May 2015, pp. 2952÷2961.
[154] Hall C., Niedziałek P.: Techniczne podstawy systemów klient-serwer. WNT, Warszawa 1996.
[155] Hao X., Wu L.: Performance evaluation of industrial Ethernet and its modeling. [in:] Information Acquisition, 2004. Proceedings. International Conference on, June 2004, pp. 527÷531.
[156] Hassler S.: You in your internet of things [spectral lines]. Spectrum, IEEE, Vol. 52, No. 4, April 2015, pp. 8÷8.
[157] He Q.-F., Zeng Q.-J., Tang X.-M.: Research and implement on industry control networks based on embedded SERCOS-iii protocol. [in:] Electronics, Communications and Control (ICECC), 2011 International Conference on, Sept 2011, pp. 3868÷3872.
[158] He W., Xu L. D.: Integration of distributed enterprise applications: a survey. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 1, Feb 2014, pp. 35÷42.
[159] Hegazy T., Hefeeda M.: Industrial automation as a cloud service. IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 10, Oct 2015, pp. 2750÷2763.
[160] Held G.: Ethernet networks: design, implementation, operation and management, 4 ed. Wiley professional computing, Wiley 2003.

[161] Henßen R., Schleipen M.: Interoperability between OPC UA and automationml. Procedia CIRP, Vol. 25, 2014, pp. 297÷304 (8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution).
[162] Hernandez Lopez G., Ravize Guizar A.: Flexible manufacturing system with CC-Link network descentralized. [in:] Autonomous Decentralized Systems (ISADS), 2015 IEEE Twelfth International Symposium on, March 2015, pp. 78÷86.
[163] Higgins N., Vyatkin V., Nair N., Schwarz K.: Distributed power system automation with IEC 61850, IEC 61499, and intelligent control. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, Vol. 41, No. 1, Jan 2011, pp. 81÷92.
[164] Hirsch M., Gerber C., Hanisch H.-M., Vyatkin V.: Design and implementation of heterogeneous distributed controllers according to the IEC 61499 standard – a case study. [in:] Industrial Informatics, 2007 5th IEEE International Conference on, June 2007, vol. 2, pp. 829÷834.
[165] Hirsch M., Vyatkin V., Hanisch H.-M.: IEC 61499 function blocks for distributed networked embedded applications. [in:] Industrial Informatics, 2006 IEEE International Conference on, Aug 2006, pp. 670÷675.
[166] Hnatkowska B., Huzar Z.: Inżynieria oprogramowania: metody wytwarzania i wybrane zastosowania. PWN, Warszawa 2008.
[167] Honczarenko J.: Obrabiarki sterowane numerycznie. WNT 2009.
[168] Hong S. H., Song S. M.: Transmission of a scheduled message using a foundation fieldbus protocol. Instrumentation and Measurement, IEEE Transactions on, Vol. 57, No. 2, Feb 2008, pp. 268÷275.
[169] Hongyu S., Yong F., Na C.: A distributed digital motion control system based sercos. [in:] Electrical and Control Engineering (ICECE), 2010 International Conference on, June 2010, pp. 48÷52.
[170] Hoon P. S., Huan Y. R., Berge J., Sim B.: Foundation trade; fieldbus high speed Ethernet (hse) implementation. [in:] Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on, 2002, pp. 777÷782.
[171] Hu F.: Cyber-physical systems: integrated computing and engineering design. CRC Press 2013.
[172] Huang Y., Zhang Z., Zhu P.: The design of an industrial remote control network gateway based on P2P VPN. [in:] Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on, Aug 2012, vol. 2, pp. 140÷143.
[173] Hughes A., Drury B.: Electric motors and drives: fundamentals, types and applications. Elsevier Science 2013.
[174] Hussain T., Frey G.: Developing IEC 61499 compliant distributed systems with network enabled controllers. [in:] Robotics, Automation and Mechatronics, 2004 IEEE Conference on, Dec 2004, vol. 1, pp. 507÷512 vol.1.
[175] Hussain T., Frey G.: Migration of a PLC controller to an IEC 61499 compliant distributed control system: hands-on experiences. [in:] Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, April 2005, pp. 3984÷3989.
[176] Iwanitz F., Lange J.: OPC fundamentals, implementation and application. Laxmi Publications Pvt Limited 2010.
[177] Izydorczyk J., Płonka G., Tyma G.: Teoria sygnałów. Wstęp. Wydanie II. Helion, Gliwice 2006.
[178] Jamroż L., Raszka J.: Szeregowanie zadań cyklicznych z wykorzystaniem algorytmów priorytetowych. Czasopismo Techniczne. Nauki Podstawowe, Z. R. 108, z. 1-NP, 2011, ss. 141÷150.
[179] Jasperneite J., Imtiaz J., Schumacher M., Weber K.: A proposal for a generic real-time Ethernet system. Industrial Informatics, IEEE Transactions on, Vol. 5, No. 2, May 2009, pp. 75÷85.
[180] Jasperneite J., Schumacher M., Weber K.: Limits of increasing the performance of industrial Ethernet protocols. [in:] Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, Sept 2007, pp. 17÷24.

[181] Jaszkiewicz A.: Inżynieria oprogramowania. Helion, Gliwice 1997, s. 276.
[182] Jazdi N.: Cyber physical systems in the context of Industry 4.0. [in:] Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on, May 2014, pp. 1÷4.
[183] Jemioło T., Dawidczyk A.: Wprowadzenie do metodologii badań bezpieczeństwa. Akademia Obrony Narodowej 2008.
[184] Jenkins C.: The P-NET: european fieldbus standard [in:] Fieldbus Technology, Springer, Berlin, Heidelberg 2003, pp. 451÷465.
[185] Jestratjew A.: Zastosowanie wielozadaniowości do poprawy parametrów czasowych wykonania aplikacji w węźle rozproszonego systemu czasu rzeczywistego. Rozprawa doktorska, Politechnika Śląska. Wydział Automatyki, Elektroniki i Informatyki, Gliwice 2011, ss. 177 (Promotor: prof. dr hab. inż. Andrzej Kwiecień).
[186] Jie P., Li L.: Industrial control system security. [in:] Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2011 International Conference on, Aug 2011, vol. 2, pp. 156÷158.
[187] John K., Tiegelkamp M.: IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools. Springer, Berlin, Heidelberg 2013.
[188] Jones C.: Programmable Logic Controllers: The Complete Guide to the Technology. Patrick-Turner 1998.
[189] Josuttis N.: SOA in Practice: The Art of Distributed System Design. O'Reilly Media 2007.
[190] Jovic F.: Process Control Systems: Principles of design, operation and interfacing. Springer, Netherlands 2012.
[191] Jue Z., Shun Y.: Design of modbus-profibus fieldbus bridge based on the STM32 and VPC3 + C. [in:] Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on, June 2012, pp. 411÷414.
[192] Kacprzak S.: Programowanie sterownikow PLC zgodnie z normą IEC61131-3 w praktyce. BTC 2011.
[193] Kandel A., Langholz G.: Fuzzy Control Systems. Taylor & Francis 1993.
[194] Kaner C., Bond W.: Software engineering metrics: What do they measure and how do we know ? Direct, Vol. 8, 2004, pp. 1÷12.
[195] Kang W., Kapitanova K., Son S. H.: Rdds: a real-time data distribution service for cyber-physical systems. Industrial Informatics, IEEE Transactions on, Vol. 8, No. 2, May 2012, pp. 393÷405.
[196] Kasprzyk J.: Sterowniki PLC. Uniwersytet Rzeszowski. Katedra Mechatroniki i Automatyki 2013.
[197] Kernighan B., Pike R.: The Practice of Programming. Addison-Wesley professional computing series, Addison-Wesley 1999.
[198] Khalgui M., Carpanzano E., Hanisch H.-M.: An optimised simulation of component-based embedded systems in manufacturing industry. International Journal of Simulation and Process Modelling, Vol. 4, No. 2, 2008, pp. 148÷162.
[199] Kim M. H., Lee S., Lee K. C.: Kalman predictive redundancy system for fault tolerance of safety-critical systems. Industrial Informatics, IEEE Transactions on, Vol. 6, No. 1, Feb 2010, pp. 46÷53.
[200] Kino S.: Application and benefits of an open devicenet control system in the forest products industry. [in:] Pulp and Paper, 1999. Industry Technical Conference Record of 1999 Annual, June 1999, pp. 196÷203.
[201] Kirrmann H., Hansson M., Muri P.: Iec 62439 prp: Bumpless recovery for highly available, hard real-time industrial networks. [in:] Emerging Technologies and Factory Automation (ETFA), 2007 IEEE Conference on, Sept 2007, pp. 1396÷1399.

[202] Kirrmann H., Weber K., Kleineberg O., Weibel H.: Hsr: Zero recovery time and low-cost redundancy for industrial Ethernet (high availability seamless redundancy, IEC 62439-3). [in:] Emerging Technologies Factory Automation (ETFA), 2009 IEEE Conference on, Sept 2009, pp. 1÷4.
[203] Kjellsson J., Vallestad A., Steigmann R., Dzung D.: Integration of a wireless i/o interface for profibus and profinet for factory automation. Industrial Electronics, IEEE Transactions on, Vol. 56, No. 10, Oct 2009, pp. 4279÷4287.
[204] Kletz T. (Ed.) What Went Wrong? – Case Histories of Process Plant Disasters and How They Could Have Been Avoided, fifth edition ed. Butterworth-Heinemann, Boston 2009.
[205] Knezic M., Dokic B., Ivanovic Z.: Performance analysis of the Ethernet powerlink pollresponse chaining mechanism. [in:] Factory Communication Systems (WFCS), 2015 IEEE World Conference on, May 2015, pp. 1÷4.
[206] Kopetz H.: Real-Time Systems: Design Principles for Distributed Embedded Applications, 1st ed. Kluwer Academic Publishers, Norwell, MA, USA 1997.
[207] Kopetz H.: The real-time environment [in:] Real-Time Systems Real-Time Systems Series Springer, USA 2011, pp. 1÷28.
[208] Kopetz H.: Real-Time Systems: design principles for distributed embedded applications. Real-Time Systems Series, Springer 2011.
[209] Korniak J.: The gmpls controlled optical networks as industry communication platform. Industrial Informatics, IEEE Transactions on, Vol. 7, No. 4, Nov 2011, pp. 671÷678.
[210] Korpysz K., Salat R., Obstawski P.: Wstep do programowania sterownikow PLC. WKŁ, Warszawa 2009.
[211] Kramer U.: Continuous testing as a strategy of improving the PLC software development cycles. [in:] Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on, 2001, vol. 2, pp. 781÷786.
[212] Krzysztof S.: Automatyczne programowanie sterownika PLC z użyciem czasowej maszyny stanowej. [w:] Andrzej K. (red.): Systemy Czasu Rzeczywistego. Kierunki badań i rozwoju, SCR 2005 Poland, Warszawa, 2015, WKŁ, ss. 251÷263.
[213] Kufel M., Żurek T., Wydawnictwo U. M. C.-S. L.: Eksploracja danych. Seria Informatyczna, Wyd. Uniwersytetu Marii Curie-Skłodowskiej 2012.
[214] Kumr S., Sharma L., Khanna Y., Chattri A.: Analysing an industrial automation pyramid and providing service oriented architecture. International Journal of Engineering Trends and Technology (IJETT), Vol. V3(5), Sep-Oct 2012, pp. 586÷594.
[215] Kundu S.: Fundamentals of Computer Networks. PHI Learning 2008.
[216] Kuraś M.: System informacyjny a system informatyczny – co oprócz nazwy różni te dwa obiekty? Zeszyty Naukowe – Uniwersytet Ekonomiczny w Krakowie, Nr 770, 2009, ss. 259÷275.
[217] Kushner D.: The real story of stuxnet. Spectrum, IEEE, Vol. 50, No. 3, March 2013, pp. 48÷53.
[218] Kwaśniewski J.: Sterowniki PLC w praktyce inżynierskiej. BTC 2008.
[219] Kwiatkowski W.: Podstawy teorii sterowania. BEL Studio 2007.
[220] Kwiecień A.: Analiza przepływu informacji w komputerowych sieciach przemysłowych. Wyd. II poprawione. WPKJS, Gliwice 2013.
[221] Kwiecień A., Gaj P.: Bezpieczeństwo w sieciach przemysłowych. [w:] Materiały konferencyjne BISK'02, Gliwice, 2002, WPKJS.
[222] Kwiecień A., Gaj P.: Stefan Węgrzyn: in memoriam: praca zbiorowa. Wyd. Politechniki Śląskiej, Gliwice 2012.
[223] Kwiecień A., Jestratjew A., Gaj P.: Podstawowe problemy tworzenia oprogramowania dla systemów związanych z bezpieczeństwem. [w:] Konferencja naukowa, Metody i narzędzia wytwarzania oprogramowania. Szklarska Poręba, 14-16 maja 2007, 2007, ss. 459÷469.

[224] Kwiecień A., Maćkowski M., Stój J., Sidzina M.: Influence of electromagnetic disturbances on multi-network interface node [in:] Computer Networks, vol. 431 of Communications in Computer and Information Science Springer 2014, pp. 298÷307.
[225] Kwiecień A., Mrówka Z., Gaj P.: O pewnej implementacji interfejsu sieci typu fip. Zeszyty Naukowe Politechniki Śląskiej s. Informatyka, Z. 34, 1998, ss. 515÷528.
[226] Kwiecień A., Sidzina M.: The method of reducing the cycle of programmable logic controller (plc) vulnerable â€śto avalanche of eventsâ€ť [in:] Computer Networks, vol. 160 of Communications in Computer and Information Science Springer, Berlin, Heidelberg 2011, pp. 379÷385.
[227] Kwiecień A., Sidzina M., Maćkowski M.: The concept of using multi-protocol nodes in real-time distributed systems for increasing communication reliability [in:] Computer Networks, vol. 370 of Communications in Computer and Information Science Springer, Berlin, Heidelberg 2013, pp. 177÷188.
[228] Kwiecień A., Stój J.: The cost of redundancy in distributed real-time systems in steady state [in:] Computer Networks, vol. 79 of Communications in Computer and Information Science Springer, Berlin, Heidelberg 2010, pp. 106÷120.
[229] Kwiecień R.: Komputerowe systemy automatyki przemysłowej. Helion, Gliwice 2013.
[230] Kyusakov R., Eliasson J., Delsing J., van Deventer J., Gustafsson J.: Integration of wireless sensor and actuator nodes with it infrastructure using service-oriented architecture. Industrial Informatics, IEEE Transactions on, Vol. 9, No. 1, Feb 2013, pp. 43÷51.
[231] Lachiewicz S., Matejun M.: Współczesne koncepcje zarządzania produkcją, jakością i logistyką. Monografie – Politechnika Łódzka, Wyd. Politechniki Łódzkiej 2010.
[232] Langner R.: Stuxnet: Dissecting a cyberwarfare weapon. Security Privacy, IEEE, Vol. 9, No. 3, May 2011, pp. 49÷51.
[233] Laughton M., Warne D.: Electrical Engineer's Reference Book. Elsevier Science 2002.
[234] Lee I., Leung J., Son S.: Handbook of Real-Time and Embedded Systems. Chapman & Hall/CRC Computer and Information Science Series, CRC Press 2007.
[235] Lee J., Bagheri B., Kao H.-A.: A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufacturing Letters, Vol. 3, 2015, pp. 18÷23.
[236] Lee K. C., Lee S., Lee M. H.: Qos-based remote control of networked control systems via profibus token passing protocol. Industrial Informatics, IEEE Transactions on, Vol. 1, No. 3, Aug 2005, pp. 183÷191.
[237] Lee Y. H., Min S. H., Hong S. H.: Mathematical analysis of prioritized data transmission in the foundation fieldbus. [in:] Industrial Informatics, 2008. INDIN 2008. 6^{th} IEEE International Conference on, July 2008, pp. 1000÷1005.
[238] Lehnhoff S., Rohjans S., Uslar M., Mahnke W.: Opc unified architecture: a service-oriented architecture for smart grids. [in:] Software Engineering for the Smart Grid (SE4SG), 2012 International Workshop on, June 2012, pp. 1÷7.
[239] Lehoczky J.: Real-time queueing theory. [in:] Real-Time Systems Symposium, 1996., 17^{th} IEEE, Dec 1996, pp. 186÷195.
[240] Leon A.: Enterprise Resource Planning. Tata McGraw-Hill 2008.
[241] Li S., Wang Y., Yang T., Chen B., Yang H.: A novel digital factory technology in complex production application. [in:] Digital Manufacturing and Automation (ICDMA), 2010 International Conference on, Dec 2010, vol. 1, pp. 83÷87.
[242] Li T., Wang H., Wang P., Kim Y.: A deterministic scheduling mechanism for industrial wireless networks. [in:] Electronic Measurement Instruments, 2009. ICEMI '09. 9^{th} International Conference on, Aug 2009, pp. 4÷746÷4÷750.
[243] Lian F.-L., Moyne J., Tilbury D.: Network protocols for networked control systems [in:] Handbook of Networked and Embedded Control Systems, Control Engineering Birkhäuser, Boston 2005, pp. 651÷675.
[244] Limaye M.: Software Testing. McGraw-Hill Education, India 2009.

[245] Lin S.-Y., Ho C.-Y., Tzou Y.-Y.: Distributed motion control using real-time network communication techniques. [in:] Power Electronics and Motion Control Conference, 2000. Proceedings. IPEMC 2000. The Third International, 2000, vol. 2, pp. 843÷847 vol.2.
[246] Liptak B., Eren H.: Instrument Engineers' Handbook, Volume 3: Process Software and Digital Networks, Fourth Edition. No. t. 3, CRC Press 2011.
[247] Liu X., Ma L., Liu Y.: A middleware-based implementation for data integration of remote devices. [in:] Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing (SNPD), 2012 13th ACIS International Conference on, Aug 2012, pp. 219÷224.
[248] Lohse N., Ratchev S., Barata J.: Evolvable assembly systems – on the role of design frameworks and supporting ontologies. [in:] Industrial Electronics, 2006 IEEE International Symposium on, July 2006, vol. 4, pp. 3375÷3380.
[249] Louden K.: Programming Languages: Principles and Practices. Advanced Topics Series, Cengage Learning 2011.
[250] M. P., J. W.: Teoria sygnałów, 3 ed. Wyd. Politechniki Śląskiej, Gliwice 2007.
[251] Ma Z.: Database Modeling for Industrial Data Management: Emerging Technologies and Applications: Emerging Technologies and Applications. IGI Global research collection, Idea Group 2005.
[252] Mackay S., Wright E., Reynders D., Park J.: Practical Industrial Data Networks: Design, Installation and Troubleshooting. Practical professional books from Elsevier, Elsevier Science 2004.
[253] Mahalik N.: Fieldbus Technology: Industrial Network Standards for Real-Time Distributed Control. Springer, Berlin, Heidelberg 2013.
[254] Mahnke W., Leitner S., Damm M.: OPC Unified Architecture. SpringerLink: Springer e-Books, Springer 2009.
[255] Mai S., Vu V. T., Yi M.-J.: An opc ua client development for monitoring and control applications. [in:] Strategic Technology (IFOST), 2011 6^{th} International Forum on, Aug 2011, vol. 2, pp. 700÷705.
[256] Martin R., Gonera P.: Czysty kod: podręcznik dobrego programisty. Helion, Gliwice 2010.
[257] Martin R., Szczepaniak M., Martin M.: Agile: programowanie zwinne: zasady, wzorce i praktyki zwinnego wytwarzania oprogramowania w C#. Helion, Gliwice 2008.
[258] Mathur A. P.: Foundations of Software Testing, 1 ed. Pearson Education, India April 2008.
[259] Małysiak H.: Teoria automatów cyfrowych: laboratorium: praca zbiorowa. Wyd. Politechniki Śląskiej, Gliwice 2003.
[260] McGuiggan P.: GPRS in Practice: a Companion to the Specifications. Wiley 2005.
[261] Mendes A., Ferreira L., Tovar E.: Fieldbus networks: real-time from the perspective of the application tasks. [in:] Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on, 2000, pp. 275÷282.
[262] Meyer H., Fuchs F., Thiel K.: Manufacturing Execution Systems (MES): Optimal Design, Planning, and Deployment: Optimal Design, Planning, and Deployment. McGraw-Hill Education 2009.
[263] Meyer-Grafe K.: Interbus in safety critical applications. [in:] Control Conference (ECC), 1999 European, Aug 1999, pp. 194÷196.
[264] Mielczarek W.: Tłumienie zakłóceń i ochrona informacji w systemach pomiarowych. Skrypty Uczelniane – Politechnika Śląska, Wyd. Politechniki Śląskiej, Gliwice 1995.
[265] Mink J.: The tether-free world and wireless technology. [in:] Antennas and Propagation Society International Symposium, 2000. IEEE, July 2000, vol. 1, p. 58.
[266] Missal D., Hirsch M., Hanisch H.-M.: Hierarchical distributed controllers – design and verification. [in:] Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, Sept 2007, pp. 657÷664.

[267] Mock M., Couturier S.: Middleware – integration of small devices. [in:] Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on, Sept 2005, vol. 1, pp. 807÷814.
[268] Nagrath I.: Control Systems Engineering. New Age International 2006.
[269] Neves P., Barata J.: Evolvable production systems. [in:] Assembly and Manufacturing, 2009. ISAM 2009. IEEE International Symposium on, Nov 2009, pp. 189÷195.
[270] Nielsen O.: A real time, object oriented fieldbus management system. [in:] Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on, 2000, pp. 335÷340.
[271] Ning H., Hu S.: Internet of things: An emerging industrial or a new major? [in:] Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing, Oct 2011, pp. 178÷183.
[272] Nordin N., Dressler F.: Effects and implications of beacon collisions in co-located ieee 802.15.4 networks. [in:] Vehicular Technology Conference (VTC Fall), 2012 IEEE, Sept 2012, pp. 1÷5.
[273] Noura H., Theilliol D., Ponsart J., Chamseddine A.: Fault-tolerant Control Systems: Design and Practical Applications. Advances in Industrial Control, Springer 2009.
[274] O'Leary D.: Enterprise Resource Planning Systems: Systems, Life Cycle, Electronic Commerce, and Risk. Cambridge University Press 2000.
[275] Orfanus D., Indergaard R., Prytz G., Wien T.: Ethercat-based platform for distributed control in high-performance industrial applications. [in:] Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on, Sept 2013, pp. 1÷8.
[276] Ou H., Zou T.: The application of digital factory in domestic chemical industry. [in:] Control and Decision Conference (CCDC), 2015 27th Chinese, May 2015, pp. 4305÷4308.
[277] Panjaitan S., Frey G.: Combination of uml modeling and the IEC 61499 function block concept for the development of distributed automation systems. [in:] Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on, Sept 2006, pp. 766÷773.
[278] Parks R., Rogers E.: Vulnerability assessment for critical infrastructure control systems. Security Privacy, IEEE, Vol. 6, No. 6, Nov 2008, pp. 37÷43.
[279] Paul B.: Industrial Electronics and Control. PHI Learning 2014.
[280] Penumuchu C.: Simple Real-time Operating System: a Kernel Inside View for a Beginner. Trafford Publishing 2008.
[281] Pereira C., Neumann P.: Industrial communication protocols [in:] Springer Handbook of Automation, Springer, Berlin, Heidelberg 2009, pp. 981÷999.
[282] Pigan R., Metter M.: Automating with PROFINET. Automating with PROFINET: Industrial Communication Based on Industrial Ethernet, PUBLICIS Kommunikations Agentur 2006.
[283] Piggin R.: Developments in real-time control with ethernet. [in:] Technology and Innovation Conference, 2006. ITIC 2006. International, Nov 2006, pp. 2161÷2168.
[284] Piggin R.: Ethernet/ip – control in real time. Computing Control Engineering Journal, Vol. 17, No. 6, Dec 2006, pp. 28÷31.
[285] Piotrowski M.: Procesy biznesowe w praktyce. Projektowanie, testowanie i optymalizacja. Helion, Gliwice 2013.
[286] Ploennigs J., Neugebauer M., Kabitzsch K.: Diagnosis and consulting for control network performance engineering of csma-based networks. Industrial Informatics, IEEE Transactions on, Vol. 4, No. 2, May 2008, pp. 71÷79.
[287] Polsonetti C.: Industrial Ethernet protocol wars: Fieldbus revisited [in:] Fieldbus Technology, Springer, Berlin, Heidelberg 2003, pp. 39÷56.
[288] Pop M., Weber K.: The Rapid Way to PROFINET. PROFIBUS International, Germany 2004.
[289] Popovic D., Bhatkar V.: Distributed Computer Control Systems in Industrial Automation. Electrical and Computer Engineering, Taylor & Francis 1990.
[290] Popp M.: Industrial communication with PROFINET. PROFIBUS International, Germany 2015.

[291] Prasad J., Jayaswal M., Priye V.: Instrumentation and Process Control. I.K. International Publishing House 2009.
[292] Pratl G., Dietrich D., Hancke G. P., Penzhorn W. T.: A new model for autonomous, networked control systems. Industrial Informatics, IEEE Transactions on, Vol. 3, No. 1, Feb 2007, pp. 21÷32.
[293] Prytz G.: A performance analysis of ethercat and profinet irt. [in:] Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, Sept 2008, pp. 408÷415.
[294] Puntambekar A.: Formal Languages And Automata Theory. Technical Publications 2009.
[295] Pytel K., Osetek S.: Systemy operacyjne i sieci komputerowe. Część 1. TI Technik Informatyk, WSiP 2010.
[296] Radasky W.: Fear of frying electromagnetic weapons threaten our data networks. here's how to stop them. Spectrum, IEEE, Vol. 51, No. 9, Sept 2014, pp. 46÷51.
[297] Rahimi S., Zargham M.: Security analysis of vpn configurations in industrial control environments [in:] Critical Infrastructure Protection V, vol. 367 of IFIP Advances in Information and Communication Technology Springer, Berlin, Heidelberg 2011, pp. 73÷88.
[298] Rajput R.: Robotics And Industrial Automation. S. Chand Limited 2008.
[299] Rehg J., Sartori G.: Programmable Logic Controllers. Pearson Prentice Hall, India 2009.
[300] Renjie H., Feng L., Dongbo P.: Research on opc ua security. [in:] Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on, June 2010, pp. 1439÷1444.
[301] Ribeiro L., Barata J., Ferreira J.: Emergent diagnosis for evolvable production systems. [in:] Industrial Electronics (ISIE), 2010 IEEE International Symposium on, July 2010, pp. 2647÷2652.
[302] Ribeiro L., Barata J., Pimentao J.: Where evolvable production systems meet complexity science. [in:] Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on, May 2011, pp. 1÷6.
[303] Richards G.: Stages of automation – control theatre. Engineering Technology, Vol. 3, No. 21, December 2008, pp. 38÷41.
[304] Rigatos G.: Modelling and Control for Intelligent Industrial Systems: Adaptive Algorithms in Robotics and Industrial Engineering. Intelligent Systems Reference Library, Springer, Berlin, Heidelberg 2011.
[305] Rockoff L.: Język SQL: przyjazny podręcznik. Helion, Gliwice 2014.
[306] Rodak A.: Usprawnij zarządzanie produkcją dzięki wonderware mesd operations. Biuletyn Automatyki, Z. 81, Nr 3, 2014, ss. 28÷29.
[307] Rosen M., Lublinsky B., Smith K., Balcer M.: Applied SOA: Service-Oriented Architecture and Design Strategies. Wiley 2012.
[308] Rostafiński W.: Niedostrzegalne światy. Polska Fundacja Kulturalna 1989.
[309] Rostan M., Stubbs J., Dzilno D.: Ethercat enabled advanced control architecture. [in:] Advanced Semiconductor Manufacturing Conference (ASMC), 2010 IEEE/SEMI, July 2010, pp. 39÷44.
[310] Rotem-Gal-Oz A., Lachowski L.: Wzorce SOA: najlepsze podejście do wytwarzania oprogramowania! Helion, Gliwice 2013.
[311] Sadeghi A.-R., Wachsmann C., Waidner M.: Security and privacy challenges in industrial internet of things. [in:] Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE, June 2015, pp. 1÷6.
[312] Saifullah A., Xu Y., Lu C., Chen Y.: Real-time scheduling for wirelesshart networks. [in:] Real-Time Systems Symposium (RTSS), 2010 IEEE 31st, Nov 2010, pp. 150÷159.
[313] Samaras I., Gialelis J., Hassapis G.: A service oriented-based system for real time industrial applications. [in:] Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on, Sept 2010, pp. 1÷8.

[314] Sammarco J.: Programmable electronic and hardwired emergency shutdown systems: a quantified safety analysis. Industry Applications, IEEE Transactions on, Vol. 43, No. 4, July-Aug 2007, pp. 1061÷1068.
[315] Sato Y.: Throwing a bridge between risk assessment and functional safety. [in:] SICE, 2007 Annual Conference, Sept 2007, pp. 2484÷2488.
[316] Sato Y.: Engineering experience in foundation fieldbus technology. [in:] ICCAS-SICE, 2009, Aug 2009, pp. 2087÷2090.
[317] Sauer O.: Information technology for the factory of the future – state of the art and need for action. Procedia CIRP, Vol. 25, 2014, pp. 293÷296 (8^{th} International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4^{th} Industrial Revolution).
[318] Sauer O., Jasperneite J.: Adaptive information technology in manufacturing. [in:] Proceedings of 44^{th} CIRP Conference on Manufacturing Systems, Madison in Wisconsin, USA, June 2011, pp. 1÷5.
[319] Sauter T.: The three generations of field-level networks – evolution and compatibility issues. Industrial Electronics, IEEE Transactions on, Vol. 57, No. 11, Nov 2010, pp. 3585÷3595.
[320] Sauter T., Jasperneite J., Lo Bello L.: Towards new hybrid networks for industrial automation. [in:] Emerging Technologies Factory Automation (ETFA), ETFA IEEE Conference on on, Sept 2009, pp. 1÷8.
[321] Sauter T., Lobashov M.: How to access factory floor information using internet technologies and gateways. Industrial Informatics, IEEE Transactions on, Vol. 7, No. 4, Nov 2011, pp. 699÷712.
[322] Scheifele S., Friedrich J., Lechler A., Verl A.: Flexible, self-configuring control system for a modular production system. Procedia Technology, Vol. 15, 2014, pp. 398÷405 (2^{nd} International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering).
[323] Schemm E.: Sercos to link with Ethernet for its third generation. Computing Control Engineering Journal, Vol. 15, No. 2, April 2004, pp. 30÷33.
[324] Schleipen M., Okon M.: The caex tool suite – user assistance for the use of standardized plant engineering data exchange. [in:] Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on, Sept 2010, pp. 1÷7.
[325] Schmidt K.: Robust priority assignments for extending existing controller area network applications. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 1, Feb 2014, pp. 578÷585.
[326] Scott K.: The SQL Programming Language. Jones & Bartlett Learning 2010.
[327] Seal A.: Practical Process Control. Elsevier Science 1998.
[328] Sehgal P. P. S., Dwivedi A., Chopra W. C. A.: Special i/o modules in PLC system. International Refereed Journal of Engineering and Science (IRJES), Vol. 1, Nov 2012, pp. 20÷27.
[329] Semere D., Barata J., Onori M.: Evolvable assembly systems: Developments and advances. [in:] Assembly and Manufacturing, 2007. ISAM '07. IEEE International Symposium on, July 2007, pp. 282÷287.
[330] Seno L., Vitturi S., Zunino C.: Real time Ethernet networks evaluation using performance indicators. [in:] Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on, Sept 2009, pp. 1÷8.
[331] Shalloway A., Trott J.: Design Patterns Explained: a New Perspective on Object-Oriented Design. Software Patterns Series, Pearson Education, India 2004.
[332] Sharma K.: Overview of Industrial Process Automation. Elsevier insights, Elsevier 2011.
[333] Sheng Z., Mahapatra C., Zhu C., Leung V.: Recent advances in industrial wireless sensor networks toward efficient management in iot. Access, IEEE, Vol. 3, 2015, pp. 622÷637.

[334] Shrouf F., Ordieres J., Miragliotta G.: Smart factories in industry 4.0: a review of the concept and of energy management approached in production based on the internet of things paradigm. [in:] Industrial Engineering and Engineering Management (IEEM), 2014 IEEE International Conference on, Dec 2014, pp. 697÷701.
[335] Simoes P., Cruz T., Proença J., Monteiro E.: Specialized honeypots for SCADA systems [in:] Cyber Security: Analytics, Technology and Automation, vol. 78 of Intelligent Systems, Control and Automation: Science and Engineering Springer 2015, pp. 251÷269.
[336] Singh S.: Computer-Aided Process Control. Prentice Hall, India 2004.
[337] Smith C.: Control of Batch Processes. Wiley 2014.
[338] Solnik W., Zajda Z.: Komputerowe sieci przemysłowe Profibus DP i MPI. Oficyna Wydawnicza Politechniki Wrocławskiej 2007.
[339] Son M., Yi M.-J.: A study on opc specifications: Perspective and challenges. [in:] Strategic Technology (IFOST), 2010 International Forum on, Oct 2010, pp. 193÷197.
[340] Song J., Han S., Mok A., Chen D., Lucas M., Nixon M.: Wirelesshart: Applying wireless technology in real-time industrial process control. [in:] Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE, April 2008, pp. 377÷386.
[341] Song X., Tan S., Ding J.: A monitoring system for PLC controlled manufacturing system based on fieldbus [in:] Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management, vol. 207 of IFIP International Federation for Information Processing Springer, USA 2006, pp. 576÷581.
[342] Spitzer D.: Variable Speed Drives: Principles and Applications for Energy Cost Savings. Momentum Press 2012.
[343] Stabryła A.: Analiza i projektowanie systemów zarządzania przedsiębiorstwem. Encyklopedia Zarządzania, Mfiles.pl 2010.
[344] Stamm M., Neitzert T.: Key perfromance indicators (kpi) for the implementation of lean methodologies in a manufacture-to-order small and medium enterprise. [in:] The 3rd World Conference on Production and Operations Management. August 5-8, 2008, Tokyo, Japan, 2008, Production and Operations Management Society (POMS), pp. 355÷368.
[345] Stanislowski D., Vilajosana X., Wang Q., Watteyne T., Pister K.: Adaptive synchronization in IEEE802.15.4e networks. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 1, Feb 2014, pp. 795÷802.
[346] Suchorzewska A.: Ochrona prawna systemów informatycznych wobec zagrożenia cyberterroryzmem. Monografie, Oficyna a Wolters Kulwer business, Warszawa 2010.
[347] Suh S., Tanik U., Carbone J., Eroglu A.: Applied Cyber-Physical Systems. Springer 2013.
[348] Sul S.: Control of Electric Machine Drive Systems. IEEE Press Series on Power Engineering, Wiley 2011.
[349] Szabatin J.: Podstawy teorii sygnałów, 5 ed. WKŁ, Warszawa styczeń 2016.
[350] Szymczyk P.: Systemy operacyjne czasu rzeczywistego. AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Kraków 2003.
[351] Telsang M.: Industrial Engineering And Production Management. S. Chand & Compagny Limited 2006.
[352] Ten C.-W., Manimaran G., Liu C.-C.: Cybersecurity for critical infrastructures: Attack and defense modeling. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, Vol. 40, No. 4, July 2010, pp. 853÷865.
[353] Teumim D.: Industrial Network Security. International Society of Automation 2010.
[354] Thomson J.: High Integrity Systems and Safety Management in Hazardous Industries. Elsevier Science 2015.
[355] Thramboulidis K.: Using uml in control and automation: a model driven approach. [in:] Industrial Informatics, 2004. INDIN '04. 2004 2nd IEEE International Conference on, June 2004, pp. 587÷593.

[356] Tovar E., Vasques F.: Pre-run-time schedulability analysis of p-net fieldbus networks. [in:] Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE, Aug 1998, vol. 1, pp. 236÷241 vol.1.
[357] Tovar E., Vasques F.: Scheduling real-time communications with p-net. [in:] Real-Time Systems (Digest No. 1998/306), IEE Colloquium on, Apr 1998, pp. 9/1÷9/5.
[358] Tritschler M.: Industrial control systems security: Security assessments and risk reduction programmes. [in:] Cyber Security for Industrial Control Systems, IET Seminar on, Feb 2014, pp. 1÷22.
[359] Trkaj K.: Users introduce component based automation solutions. Computing and Control Engineering, Vol. 15, Dec 2004, pp. 32÷37(5).
[360] Trsek H., Schwalowsky S., Czybik B., Jasperneite J.: Implementation of an advanced ieee 802.11 wlan ap for real-time wireless communications. [in:] Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on, Sept 2011, pp. 1÷4.
[361] Trzcieliński S., Pawłowski E., Adamczyk M.: Procesowa orientacja przedsiębiorstwa, 1 ed. Wyd. Politechniki Poznańskiej, Poznań 2013.
[362] Tzafestas S., Pal J.: Real Time Microcomputer Control of Industrial Processes. Intelligent Systems, Control and Automation: Science and Engineering, Springer, Netherlands 2012.
[363] Łuczycka D., Pentoś K.: Automatyzacja i robotyzacja procesów produkcyjnych. ZIP Zarządzanie i Inżynieria Produkcji, Oficyna Wydawnicza ATUT – Wrocławskie Wydawnictwo Oświatowe, Wrocław 2012.
[364] Łukasik Z., Seta Z.: Programowalne sterowniki PLC w systemach sterowania przemysłowego. Wyd. Politechniki Radomskiej 2001.
[365] Ungurean I., Gaitan N.-C., Gaitan V.: An iot architecture for things from industrial environment. [in:] Communications (COMM), 2014 10th International Conference on, May 2014, pp. 1÷4.
[366] Usman M., Abbas N.: On the application of iot (internet of things) for securing industrial threats. [in:] Frontiers of Information Technology (FIT), 2014 12th International Conference on, Dec 2014, pp. 37÷40.
[367] Valenzano A.: Industrial cybersecurity: Improving security through access control policy models. Industrial Electronics Magazine, IEEE, Vol. 8, No. 2, June 2014, pp. 6÷17.
[368] Vogel-heuser B., Kegel G., Wucherer K.: Global information architecture for industrial automation. atp edition-Automatisierungstechnische Praxis, Vol. 51, No. 01-02, 2009, pp. 108÷115.
[369] Vyatkin V., Hirsch M., Hanisch H.-M.: Systematic design and implementation of distributed controllers in industrial automation. [in:] Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on, Sept 2006, pp. 633÷640.
[370] Wagner F., Bohl J., Frey G.: An IEC 61499 interpretation and implementation focused on usability. [in:] Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, Sept 2008, pp. 184÷191.
[371] Wan J., Cai H., Zhou K.: Industrie 4.0: Enabling technologies. [in:] Intelligent Computing and Internet of Things (ICIT), 2014 International Conference on, Jan 2015, pp. 135÷140.
[372] Wang L., Tan K. C.: Modern Industrial Automation Software Design. John Wiley & Sons, Inc. 2006.
[373] Wang S., Ledley R.: Computer Architecture and Security: Fundamentals of Designing Secure Computer Systems. Information security series, Wiley 2012.
[374] Wang T., Wang L., Liu Q.: A three-ply reconfigurable cnc system based on fpga and field-bus. The International Journal of Advanced Manufacturing Technology, Vol. 57, No. 5-8, 2011, pp. 671÷682.
[375] Wang Y., Vuran M. C., Goddard S.: Cyber-physical systems in industrial process control. SIGBED Rev., Vol. 5, No. 1, Jan 2008, pp. 12:1÷12:2.

[376] Wehrmeister M., Pereira C., Rammig F.: Aspect-oriented model-driven engineering for embedded systems applied to automation systems. Industrial Informatics, IEEE Transactions on, Vol. 9, No. 4, Nov 2013, pp. 2373÷2386.
[377] Węgrzyn S.: Podstawy informatyki. Biblioteka Informatyki, PWN, Warszawa 1982.
[378] Węgrzyn S., Znamirowski L.: Zarys nanonauki i informatycznych molekularnych nanotechnologii. Wyd. Politechniki Śląskiej, Gliwice 2008.
[379] Whitt M.: Successful Instrumentation and Control Systems Design. International Society of Automation 2004.
[380] Wideł S., Flak J., Gaj P.: Interpretation of dual peak time signal measured in network systems [in:] Computer Networks, vol. 79 of Communications in Computer and Information Science Springer, Berlin, Heidelberg 2010, pp. 141÷152.
[381] Wideł S., Flak J., Gaj P.: Time domain measurement representation in computer system diagnostics and performance analysis. e-Informatica Software Engineering Journal, Vol. 7, 2013, pp. 53÷64.
[382] Wideł S., Flak J., Jestratjew A., Gaj P.: Analysis of real-time systems using convolution of probability mass function. [in:] Emerging Technologies Factory Automation (ETFA), 2012 IEEE 17th Conference on, Sept 2012, pp. 1÷4.
[383] Świder K., Dec G., Trybus B.: Inżynieria systemów informatycznych: podstawy i praktyka budowy systemów oprogramowania. Materiały Pomocnicze – Oficyna Wydawnicza Politechniki Rzeszowskiej, OW Press 2004.
[384] Wilamowski B., Irwin J.: Fundamentals of Industrial Electronics. Electrical engineering handbook series, CRC Press 2011.
[385] Wilamowski B., Irwin J.: Industrial Communication Systems. Electrical engineering handbook series, CRC Press 2011.
[386] Wilamowski B., Irwin J.: The Industrial Electronics Handbook, Second Edition – Five Volume Set. Electrical Engineering Handbook, Taylor & Francis 2011.
[387] Wilamowski B., Irwin J.: The Industrial Electronics Handbook, Second Edition – Five Volume Set., vol. Volume 2: Industrial Communication Systems of Electrical Engineering Handbook Taylor & Francis 2011.
[388] Willig A.: Recent and emerging topics in wireless industrial communications: a selection. Industrial Informatics, IEEE Transactions on, Vol. 4, No. 2, May 2008, pp. 102÷124.
[389] Willig A., Matheus K., Wolisz A.: Wireless technology in industrial networks. Proceedings of the IEEE, Vol. 93, No. 6, June 2005, pp. 1130÷1151.
[390] Wisniewski L., Hameed M., Schriegel S., Jasperneite J.: A survey of Ethernet redundancy methods for real-time Ethernet networks and its possible improvements. [in:] Fieldbuses and Networks in Industrial and Embedded Systems, ser. In: 8th International Conference on Fieldbuses networks in Industrial; Embedded Systems (FET'2009), H. S. H. Juanole, Guy, Ed. IFAC, May 2009, vol. 8, pp. 163÷170.
[391] Świstek M.: Proficy historian – bezpieczna i niezawodna przemysłowa baza danych. Napędy i Sterowanie, Vol. 4, kwiecień 2012, pp. 39÷39.
[392] Wu Y.-j., Chung J.-G.: Efficient controller area network data compression for automobile applications. Frontiers of Information Technology & Electronic Engineering, Vol. 16, No. 1, 2015, pp. 70÷78.
[393] Xiong G., Xiong G.-Y., Litokorpi A., Nyberg T.: Middleware-based solution for enterprise information integration. [in:] Emerging Technologies and Factory Automation, 2001. Proceedings. 8th IEEE International Conference on, Oct 2001, vol. 2, pp. 687÷690.
[394] Xu H., Gao Y., Liu K., Zhu B., Zhang C.: Research on cross-communication based on real-time Ethernet powerlink. [in:] Control and Decision Conference (2014 CCDC), The 26th Chinese, May 2014, pp. 2577÷2581.
[395] Xu L. D., He W., Li S.: Internet of things in industries: a survey. Industrial Informatics, IEEE Transactions on, Vol. 10, No. 4, Nov 2014, pp. 2233÷2243.

[396] Yan W., Lifang W., Chenglin L., Fang L.: In-vehicle flexray bus monitoring system. [in:] Transportation Electrification Asia-Pacific (ITEC Asia-Pacific), 2014 IEEE Conference and Expo, Aug 2014, pp. 1÷5.
[397] Yang T., Peng C., Yue D., Fei M.: New study of controller design for networked control systems. Control Theory Applications, IET, Vol. 4, No. 7, July 2010, pp. 1109÷1121.
[398] Yoo S.-E., Chong P. K., Kim D., Doh Y., Pham M.-L., Choi E., Huh J.: Guaranteeing real-time services for industrial wireless sensor networks with ieee 802.15.4. Industrial Electronics, IEEE Transactions on, Vol. 57, No. 11, Nov 2010, pp. 3868÷3876.
[399] Yoon G., Kwon D. H., Kwon S. C., Park Y. O., Lee Y. J.: Ring topology-based redundancy Ethernet for industrial network. [in:] SICE-ICASE, 2006. International Joint Conference, Oct 2006, pp. 1404÷1407.
[400] Younis M., Frey G.: Formalization of PLC programs to sustain reliability. [in:] Robotics, Automation and Mechatronics, 2004 IEEE Conference on, Dec 2004, vol. 2, pp. 613÷618.
[401] Younis M., Frey G.: Visualization of PLC programs using XML. [in:] American Control Conference, 2004. Proceedings of the 2004, June 2004, vol. 4, pp. 3082÷3087.
[402] Younis M., Frey G.: Formalization and visualization of non-binary PLC programs. [in:] Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on, Dec 2005, pp. 8367÷8372.
[403] Younis M., Frey G.: A formal method based re-implementation concept for PLC programs and its application. [in:] Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on, Sept 2006, pp. 1340÷1347.
[404] Younis M., Frey G.: Uml-based approach for the re-engineering of PLC programs. [in:] IEEE Industrial Electronics, IECON 2006 – 32nd Annual Conference on, Nov 2006, pp. 3691÷3696.
[405] Younis M., Frey G.: Software quality measures to determine the diagnosability of PLC applications. [in:] Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, Sept 2007, pp. 368÷375.
[406] Zezulka F., Beran J.: Virtual automation networks – architectural principles and the current state of development. [in:] Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE, Nov 2008, pp. 1545÷1550.
[407] Zhang M., Lu Y., Xia T.: The design and implementation of virtual machine system in embedded softplc system. [in:] Computer Sciences and Applications (CSA), 2013 International Conference on, Dec 2013, pp. 775÷778.
[408] Zhang S., Zhang G., Yan A., Xiang Z., Ma T.: A highly reliable link scheduling strategy for wirelesshart networks. [in:] Advanced Technologies for Communications (ATC), 2013 International Conference on, Oct 2013, pp. 39÷43.
[409] Zhou L., Chao H.-C.: Multimedia traffic security architecture for the internet of things. Network, IEEE, Vol. 25, No. 3, May 2011, pp. 35÷40.
[410] Zhou Y., Wang Z., Wang T.-r., Yu H.-b.: Real-time communication using foundation fieldbus [in:] Fieldbus Technology, Springer, Berlin, Heidelberg 2003, pp. 311÷331.
[411] Zikopoulos P., Eaton C.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Education 2011.
[412] Zuehlke D.: Smartfactory – from vision to reality in factory technologies. [in:] International Federation of Automatic Control (IFAC), Proceedings of the 17th World Congress, Seoul, Jul 2008, vol. 17, p. 14101–14108.
[413] Zurawski R.: The Industrial Information Technology Handbook. Industrial Electronics, CRC Press 2004.
[414] Zurawski R.: Embedded Systems Handbook, Second Edition: Embedded Systems Design and Verification. Embedded systems handbook, CRC Press 2009.
[415] Zurawski R.: Industrial Communication Technology Handbook, Second Edition. Industrial Information Technology, Taylor & Francis 2014.

6.2. Raporty, dokumentacje i standardy

Referencje do tego typu źródeł oznaczono przez M. Pozycje są posortowane alfabetycznie. Daty publikacji są podane w zakresach lub jako najnowsze znane autorowi.

[M1] Booklet “The P-NET Fieldbus for Process Automation”, ref. 590 004 02, The International P-NET User Organization, 1996

[M2] Broszura: PROFIBUS Technologie i Aplikacje, PROFIBUS Nutzerorganisation e.V., PNO 11/02, Listopad 2004

[M3] Broszura: Redundant networks for industry, Siemens AG 2012

[M4] Dai W., Shih A., Pang C., Park J., Huang D., Gommans D.: Function Block (FB) Workbench (FBench) Developer’s Guide, Supervised by Valeriy Vyatkin, ©University of Auckland, 2006-2007

[M5] DIN EN 13757-1..3: Communication systems for meters – Part 1..3, DIN 2014

[M6] EPSG Working Draft Proposal 304, openSAFETY, Safety Profile Specification, Version 1.4.0, Ethernet POWERLINK Standardisation Group 2013

[M7] FDI Technical Specification: Profiles; version 1.0.3, FDI-2060-2064; 1 listopada 2014

[M8] FDI Technical Specification; version 1.0.3, FDI-2021-2027, 2035; 1 listopada 2014

[M9] FDT®2.0 Technical Specification Version 1.00; FDT Group AISBL, BP 20, 1370 Jodoigne, Belgia 2012

[M10] Haugan B., Hegvik G. K., Glomnes L. M., Larsen-Vonstett O., Johnsrud, I.: Could have stopped the water supply from mobile, 2011. Oryg.: Kunne stoppet vanntilforselen med mobilen, dostęp: sierpień 2016.

[M11] Hodson H.: Hackers accessed city infrastructure via SCADA FBI, Information Age 2011. dostęp: sierpień 2016.

[M12] IEC 61588:2009: Precision clock synchronization protocol for networked measurement and control systems, IEC 2009

[M13] IEC 62657-2:2013: Industrial communication networks – Wireless communication networks – Part 2: Coexistence management, IEC 2014

[M14] IEC 62769-x:2015 Field device integration (FDI), 11 części, IEC 2015

[M15] IEC TR 61131-4:2004: Programmable controllers – Part 4: User guidelines, IEC 2004

[M16] IEC TR 61131-8:2003: Programmable controllers – Part 8: Guidelines for the application and implementation of programming languages, IEC 2003

[M17] IEC TS 62443-1..3:2009: Industrial communication networks – Network and system security – Part 1..3

[M18] IEC TS 62657-1:2014: Industrial communication networks – Wireless communication networks – Part 1: Wireless communication requirements and spectrum considerations, IEC 2014

[M19] IEC/PN 61131-1:2004: Sterowniki programowalne – Część 1: Postanowienia ogólne, PKN 2004

[M20] IEC/PN 61131-2:2008: Sterowniki programowalne – Część 2: Wymagania i badania dotyczące sprzętu; PKN 2008

[M21] IEC/PN 61131-3:2013-10: Sterowniki programowalne – Część 3: Języki programowania, PKN 2013

[M22] IEC/PN 61131-5:2002: Sterowniki programowalne – Część 5: Komunikacja, PN-EN 2002

[M23] IEC/PN 61131-6:2013-07 Sterowniki programowalne – Część 6: Bezpieczeństwo funkcjonalne, PKN 2013

[M24] IEC/PN 61131-7:2004: Sterowniki programowalne – Część 7: Programowanie rozmyte, PKN 2004

[M25] IEC/PN 61131-9:2014-05: Sterowniki programowalne – Część 9: Interfejs komunikacji cyfrowej punkt-punkt do małych czujników i elementów wykonawczych (SDCI), PKN 2014
[M26] IEC/PN 61131-x: Sterowniki programowalne, IEC 2013
[M27] IEC/PN 61158-1:2014-12: Przemysłowe sieci komunikacyjne – Specyfikacje magistrali miejscowej – Część 1: Przegląd i wytyczne do serii IEC 61158 i IEC 61784, PKN Grudzień 2014
[M28] IEC/PN 61158-2:2014-12: Przemysłowe sieci komunikacyjne v Specyfikacje magistrali miejscowej – Część 2: Specyfikacja i definicja usług warstwy fizycznej, PKN Grudzień 2014
[M29] IEC/PN 61158-3-1..24:2008-2015: Przemysłowe sieci komunikacyjne – Specyfikacje magistrali miejscowej – Część 3-1..24: Definicja usług warstwy sprzężenia danych – Elementy typu 1..24, PKN 2008, 2014
[M30] IEC/PN 61158-4-1..24:2008-2015: Przemysłowe sieci komunikacyjne – Specyfikacje magistrali miejscowej – Część 4-1..24: Specyfikacja protokołu warstwy sprzężenia danych – Elementy typu 1..24, PKN 2008, 2014
[M31] IEC/PN 61158-5-1..24:2008-2015: Przemysłowe sieci komunikacyjne – Specyfikacje magistrali miejscowej – Część 5-1..24: Definicja usług warstwy zastosowania – Elementy typu 1..24, PKN 2008, 2014
[M32] IEC/PN 61158-6-1..24:2008-2015: Przemysłowe sieci komunikacyjne – Specyfikacje magistrali miejscowej – Część 6-1..24: Specyfikacja protokołu warstwy zastosowania – Elementy typu 1..24, PKN 2008, 2014
[M33] IEC/PN 61334-3-21,22:2002: Automatyzacja sieci rozdzielczej z użyciem łączności wykorzystującej tę sieć – Część 3-21, 22: Wymagania sieci dotyczące przesyłu sygnałów, IEC 2002
[M34] IEC/PN 61499-1:2013-07 – Bloki funkcyjne – Część 1: Architektura, PKN 2013
[M35] IEC/PN 61499-2:2013-07 – Bloki funkcyjne – Część 2: Wymagania dotyczące narzędzi programowych, PN-EN 2013
[M36] IEC/PN 61499-4:2013-10 – Bloki funkcyjne – Część 4: Reguły do profili zgodności, PKN 2013
[M37] IEC/PN 61508-x: Bezpieczeństwo funkcjonalne elektrycznych/elektronicznych/ programowalnych elektronicznych systemów związanych z bezpieczeństwem, PKN2010
[M38] IEC/PN 61511-x: Bezpieczeństwo funkcjonalne – Przyrządowe systemy bezpieczeństwa do sektora przemysłu procesowego, IEC 2007
[M39] IEC/PN 61784-1:2015-01: Przemysłowe sieci komunikacyjne – Profile – Część 1: Profile magistrali miejscowej, PKN Styczeń 2015
[M40] IEC/PN 61784-2:2015-01: Przemysłowe sieci komunikacyjne – Profile – Część 2: Profile dodatkowe magistrali miejscowej do sieci czasu rzeczywistego opartych na ISO/IEC 8802-3, PKN Styczeń 2015
[M41] IEC/PN 61784-3-1..18:2010: Przemysłowe sieci komunikacyjne – Profile – Część 3-1..18: Magistrale miejscowe bezpieczne funkcjonalnie – Specyfikacje uzupełniające do CPF 1..18, PKN 2010, 2011
[M42] IEC/PN 61784-5-1..19:2014: Przemysłowe sieci komunikacyjne – Profile – Część 5-1..19: Instalowanie sieci miejscowych – Profile instalacyjne do CPF 1..19, PKN 2012, 2014
[M43] IEC/PN 61804-2: Bloki funkcyjne (FB) do sterowania procesami – Część 2: Specyfikacja koncepcji FB, IEC 2008
[M44] IEC/PN 61804-3: Bloki funkcyjne (FB) do sterowania procesami – Część 3: Język Opisu Urządzenia Elektronicznego (EDDL), IEC 2011
[M45] IEC/PN 61918:2014-06: Przemysłowe sieci komunikacyjne – Instalowanie sieci komunikacyjnych w obiektach przemysłowych, IEC 2014
[M46] IEC/PN 61935-1,2:2011: Wymagania dotyczące sprawdzania symetrycznych i współosiowych kablowych linii informatycznych – cześć 1 i 2, IEC 2011
[M47] IEC/PN 62056-x: Wymiana danych w pomiarach energii elektrycznej, IEC 2014

[M48] IEC/PN 62061:2008 Bezpieczeństwo maszyn – Bezpieczeństwo funkcjonalne elektrycznych, elektronicznych i elektronicznych programowalnych systemów sterowania związanych z bezpieczeństwem, IEC 2008
[M49] IEC/PN 62351-x: Zarządzanie systemem elektroenergetycznym i związana z tym wymiana informacji - Ochrona danych komunikacji, IEC 2004
[M50] IEC/PN 62439-1:2010: Przemysłowe sieci komunikacyjne – Sieci automatyki o wysokiej dostępności, IEC 2010
[M51] IEC/PN 62453-x: Specyfikacja interfejsu obiektowych urządzeń narzędziowych, IEC 2010
[M52] IEC/PN 62541-x: Specyfikacja zunifikowanej architektury OPC, IEC 2011
[M53] IEC/PN 62734:2015-03: Przemysłowe sieci komunikacyjne – Sieci komunikacyjne bezprzewodowe i profile komunikacyjne – ISA 100.11a, IEC 2015
[M54] IEC/PN-EN 61000-4-6:2014-04: Kompatybilność elektromagnetyczna (EMC) – Część 4-6: Metody badań i pomiarów – Odporność na zaburzenia przewodzone, indukowane przez pola o częstotliwości radiowej, PKN 2014
[M55] IEC/PN-EN 61000-6-1:2008:Kompatybilność elektromagnetyczna (EMC) – Część 6-1: Normy ogólne – Odporność w środowiskach: mieszkalnym, handlowym i lekko uprzemysłowionym, PKN 2008
[M56] IEC/PN-EN 61000-6-2:2008: Kompatybilność elektromagnetyczna (EMC) – Część 6-2: Normy ogólne – Odporność w środowiskach przemysłowych, PKN 2008
[M57] IEC/PN-EN 61000-6-3:2008: Kompatybilność elektromagnetyczna (EMC) – Część 6-3: Normy ogólne – Norma emisji w środowiskach: mieszkalnym, handlowym i lekko uprzemysłowionym, PKN 2008
[M58] IEC/PN-EN 61000-6-4:2008: Kompatybilność elektromagnetyczna (EMC) – Część 6-4: Normy ogólne – Norma emisji w środowiskach przemysłowych, PKN 2008
[M59] IEC/PN-EN ISO 9000:2006: Systemy zarządzania jakością – Podstawy i terminologia, PKN 2006
[M60] IEEE Standard for Floating-Point Arithmetic, *IEEE Std 754-2008*, pp.1-70, Aug 29, 2008, doi: 10.1109/IEEESTD.2008.4610935
[M61] IEEE Standard for Identity-Based Cryptographic Techniques using Pairings," IEEE P1363.3/D9, May 2013, pp.1,136, Nov. 15 2013
[M62] IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security," IEEE Std 802.1AE-2006, pp.1,150, Aug. 18 2006
[M63] Industrial Automation Guide 2015, Y205-EN2-08+IndAutoGuide2015, Omron 2015
[M64] Interbus Basics, INTERBUS Club Deutschland e.V., TNR 5122378/28.03.01-00 INTERBUS Club – International Marketing Services
[M65] IO-Link Smart Sensor Profile, Draft Specification, Version 1.0.2, IO-Link Consortium July 2011
[M66] IoT@Work public deliverable, IoT@Work/WP2/D2.5/1.1, 04-07-2013
[M67] ISO/IEC 10967-1:2012 Information technology – Language independent arithmetic, ISO 2012
[M68] ISO/IEC 11801:2002+AMD1:2008+AMD2:2010 Consolidated Version: Information technology – Generic cabling for customer premises
[M69] ISO/IEC 24773:2008: Software engineering – Certification of software engineering professionals – Comparison framework, IEC 2008
[M70] ISO/IEC 25010:2011: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, IEC 2011
[M71] ISO/IEC 27033-5:2013 Information technology – Security techniques – Network security – Part 5: Securing communications across networks using Virtual Private Networks (VPNs), IEC 2013

[M72] ISO/IEC TR 24732:2009: Information technology – Programming languages, their environments and system software interfaces – Extension for the programming language C to support decimal floating-point arithmetic, IEC 2009
[M73] ISO/IEC TR 9126-x:2003: Software engineering – Product quality, ISO 2003
[M74] ISO/IEC/IEEE 60559:2011: Information technology – Microprocessor Systems – Floating-Point arithmetic
[M75] ISO/IEC/IEEE 8802-3:2014: Standard for Ethernet, IEC 2014
[M76] ITL Education Solutions Limited: Introduction to Database Systems., 1 ed. Pearson Education, India 2010
[M77] Kirrmann H.: Fault tolerant computing in industrial automation. Technical report, 2005.
[M78] Łucki Z.: Proszę... nie mówmy „technologia" na technikę! Biuletyn Informacyjny Pracowników Akademii Górniczo-Hutniczej [http://www.uci.agh.edu.pl/bip/63/11_63.htm]
[M79] Nakashima E.: Water-pump failure in illinois wasn't cyberattack after all, Nov 2011. dostęp: sierpień 2016.
[M80] PN-EN 60529:2003: Stopnie ochrony zapewnianej przez obudowy (Kod IP), PKN 2003
[M81] PROFIBUS Zalecenia odnośnie montażu i okablowania instalcji sieciowych, Wersja 1.0.6, Maj 2008
[M82] PROFINET – Wskazówki odnośnie instalacji, podłączenia i montaż, Wersja 1.0, PNO 2009
[M83] PROFINET Design Guideline Version 1.04, Order No.: 8.062, PROFIBUS Nutzerorganisation e.V., November 2010
[M84] Raport techniczny: Poradnik automatyka cz.1. – jak wybrać panel operatorski, Biuletyn Automatyki, Nr 3(81), Astor, 2014.
[M85] Raport techniczny: Wonderware 2 2014 R2 – nowy wymiar ergonomii systemów HMI/SCADA, Nr 4(82), Astor, 2014.
[M86] Redundant networks for industry, November Edition 2012, Order No. 6ZB5530-1BL02-0BA0, Siemens AG 2012
[M87] Schiffer V.: The common industrial protocol (CIP™) and the family of cip networks. Technical report, 1099 Highland Drive, Suite A, Ann Arbor, MI 48108-5002 Rockwell Automation USA.
[M88] Sprunt, Brinkley., Sha, Lui., & Lehoczky, John. (1989). Scheduling Sporadic and Aperiodic Events in a Hard Real-Time System (CMU/SEI-89-TR-011). Retrieved August 25, 2015, from the Software Engineering Institute, Carnegie Mellon University
[M89] Standard for Wide-Block Encryption for Shared Storage Media, IEEE Unapproved Draft Std P1619.2/D11, Aug 2009, vol., no., pp., 2009
[M90] Sternstein A.: Hackers manipulated railway computers, 2012. dostęp: sierpień 2016.
[M91] TCP/IP Ethernet Communications for PACSystems User's Manual, GFK-2224F, GE Fanuc Intelligent Platforms, Programmable Control Products, November 2007
[M92] Technical Short Description, IEC 61491, EN 61491 SERCOS interface. The internationally-standardized digital interface for communication between controllers and drives in numerically controlled machines, Interest Group SERCOS interface e.V.
[M93] The Foundation Fieldbus Primer Prepared by Fieldbus inc., Revision 1.1, Released June 24, 2001
[M94] The Industrial Control Systems Cyber Emergency Response Team (ICS-CERT). Introduction to recommended practices. Technical report. dostęp: sierpień 2016
[M95] Weiss J.: Water system hack – the system is broken, 2011. dostęp: sierpień 2016.
[M96] White Paper: Elliott Middleton, Product Manager, Historian & Clients, Invensys Operations Management, Myths about Historians, Rel. 12/12 PN WW-4087, Invensys Systems, Inc. 2012
[M97] Whitepaper AutomationML. Part 1 – Architecture and general requirements, AutomationML consortium, Version 2.0, Maj 2010
[M98] Xinghu Z.: Latvenergo rigas hes-2 hacked, 2011. dostęp: sierpień 2016.

6.3. Źródła web

W przestrzeni Internetu istnieje wiele stron i portali informacyjnych, które dostarczają interesujących treści, mimo że nie stanowią źródeł bibliograficznych. Podobnie z dokumentacjami firmowymi, raportami, instrukcjami itp. Poniżej zmieszczono listę polecanych źródeł. Stanowią one dobre uzupełnienie bibliografii. Dostęp był wykonywany w sierpniu 2015. Nie zamieszczono pełnych ścieżek URL do materiałów, gdyż z natury swojej mają one charakter tymczasowy. Referencje do tego typu źródeł oznaczono przez W. Pozycje są posortowane alfabetycznie według adresów www.

[W1] http://control.com – strona społeczności związanej z systemami sterowania
[W2] http://fb61499.com – strona wspierająca rozwiązania wg IEC61499
[W3] http://homepages.engineering.auckland.ac.nz/~vyatkin/fbench – strona projektu Fbench
[W4] http://ieeexplore.ieee.org – strona baz wiedzy IEEE
[W5] http://link.springer.com – strona przykładowego wydawnictwa
[W6] http://nuttx.org – strona wsparcia systemu NuttX
[W7] http://plcedge.com – strona dla początkujących programistów PLC
[W8] http://profibus.felser.ch – strona prowadzona przez prof. Maxa Felsera (BFH Bern) z użytecznymi informacjami o sieci Profibus
[W9] http://seg.ece.upatras.gr/Corfu/dev/faq.htm – strona projektu CORFU
[W10] http://smartmanufacturingcoalition.org – strona projektu SMLC (ang. Smart Manufacturing Leadership Coalition)
[W11] http://support.automation.siemens.com – strona wsparcia przykładowego producenta
[W12] http://www.apics.org – słownik pojęć technicznych online
[W13] http://www.automationml.org – strona poświęcona technologii AutomationML
[W14] http://www.cclinkamerica.org – strona wsparcia sieci CC-Link prowadzona przez stowarzyszenie CLPA (ang. CC-Link Partner Association)
[W15] http://www.eddl.org – strona poświęcona językowi EDDL
[W16] http://www.ethercat.org – strona wsparcia sieci EtherCAT prowadzona przez organizację ETG (ang. EtherCAT Technology Group)
[W17] http://www.ethercat.org/en/safety.html - strona o technologii Safety over Ethercat
[W18] http://www.falowniki.edu.pl – strona poświęcona przekształtnikom częstotliwości
[W19] http://www.fdi-cooperation.com – strona projektu FDI
[W20] http://www.fordiac.org – strona projektu 4DIAC
[W21] http://www.freertos.org – strona wsparcia systemu FreeRTOS
[W22] http://www.holobloc.com – strona projektu FBDK
[W23] http://www.iec.ch – strona organizacji IEC (ang. International Electrotechnical Commission)
[W24] http://www.iec61499.org – strona wspierająca rozwiązania wg IEC61499
[W25] http://www.ieee.org – strona Instytutu IEEE (ang. Institute of Electrical and Electronics Engineers)
[W26] http://www.iot-at-work.eu – strona projektu IoT@Work
[W27] http://spectrum.ieee.org – „Interactive: The Top Programming Languages 2016” – przegląd popularności języków programowania dla roku 2016 opublikowany online w ramach magazymu IEEE Spectrum
[W28] http://www.isagraf.com – strona projektu ISaGRAF
[W29] http://istqb.org – strona międzynarodowego komitetu kwalifikacji dotyczacych certyfikacji oprogramowania

[W30] http://www.mesa.org – strona międzynarodowej organizacji Manufacturing Enterprise Solutions Association (MESA)
[W31] http://www.namur.de – strona organizacji NAMUR
[W32] http://www.plcopen.org – strona organizacji PLCOpen wspierającej otwarte rozwiązania dla PLC, w tym standaryzację według IEC61131
[W33] http://www.plcs.net – strona wsparcia dla programistów PLC
[W34] http://www.plctalk.net – forum dyskusyjne o sterownikach
[W35] http://www.p-net.org – strona o sieci P-NET
[W36] http://www.sciencedirect.com – baza artykułów naukowych
[W37] http://www.sixsigmadaily.com – strona informacyjna dla idei „sześć sigma”

7. DODATKI

W niniejszym rozdziale zawarto materiały dodatkowe, stanowiące ilustracje i uzupełnienie treści oraz spisy rysunków, zdjęć i odsyłaczy do kluczowych pojęć.

7.1. Przykłady ISP

Dla ilustracji zagadnień opisanych w książce, w poniższych podrozdziałach przedstawiono kilka przykładów systemów ISP zrealizowanych w praktyce. Szczegóły opisu obiektów i implementacji nie są prezentowane ze względu na ochronę rozwiązań i poufność użytkowników. Zatem, nie wszystkie elementy konfiguracji, które są opisane w niniejszej książce, są tu ujawnione. Prezentowany opis ma raczej charakter konceptualny niż dokumentacyjny. Celem pokazania przykładów jest ilustracja różnorodności problemów projektowych i aplikacyjnych oraz struktur i topologii systemów, dla typowych zastosowań. Zastosowania i aplikacje mniej typowe generują bardziej złożone i różnorodne w swej konstrukcji systemy. Prezentowanych rozwiązań nie należy traktować jako wzorcowych. Nie są one idealne i stanowią kompromis między aspektami wymagań procesu, klienta, ekonomicznych, elastyczności i rozwoju (1.2.3). Autor książki jest współprojektantem i współwykonawcą wszystkich prezentowanych przykładów systemów.

7.1.1. Rozproszony system telemetryczny

Pierwszy przykład dotyczy systemu, gdzie zadania sterowania są trywialne, a główny problem leży w rozproszeniu terytorialnym abonentów i braku infrastruktury komunikacyjnej. Celem działania systemu jest zbieranie informacji o pracy obiektów zlokalizowanych w różnych miejscach obszaru miejskiego, terenów wiejskich oraz leśnych.

❖ **Założenia**

Założenia do systemu dotyczą trzech obszarów działania: lokalnego na obiektach rozproszonych, systemowej lokalnej wizualizacji w centrali nadzorczej oraz zdalnej wizualizacji

w przestrzeni publicznej Internetu. Podstawowe wymagania dotyczą następujących funkcji ISP:

- Akwizycja informacji z kilkudziesięciu obiektów rozproszonych. Obiekty są różnego rodzaju i mają różne wymagania względem algorytmów sterujących. Algorytmy te nie są złożone.
- Każdy obiekt może mieć inny zestaw opisujących go danych i podlegających akwizycji.
- Centralna prezentacja danych odbywa się na komputerze nadzorczym w formie graficznej.
- Konieczne jest powiadamianie o stanach awaryjnych obiektów na wybranych telefonach komórkowych. Konieczne jest również odpytywanie o stan obiektu z telefonu. Telefony są starego typu (bez OS), a powiadamianie odbywa się przez SMS.
- Konieczna jest bieżąca i historyczna prezentacja stanu oraz wybranych danych i zdarzeń ze wszystkich obiektów przez przeglądarkę internetową.
- Obiekty nie mają ani dostępu do sieci lokalnych, ani rozległych, ani żadnych przewodowych. Istnieje dostęp do sieci komórkowych, choć nie dla każdego obiektu.

❖ **Rozwiązanie**

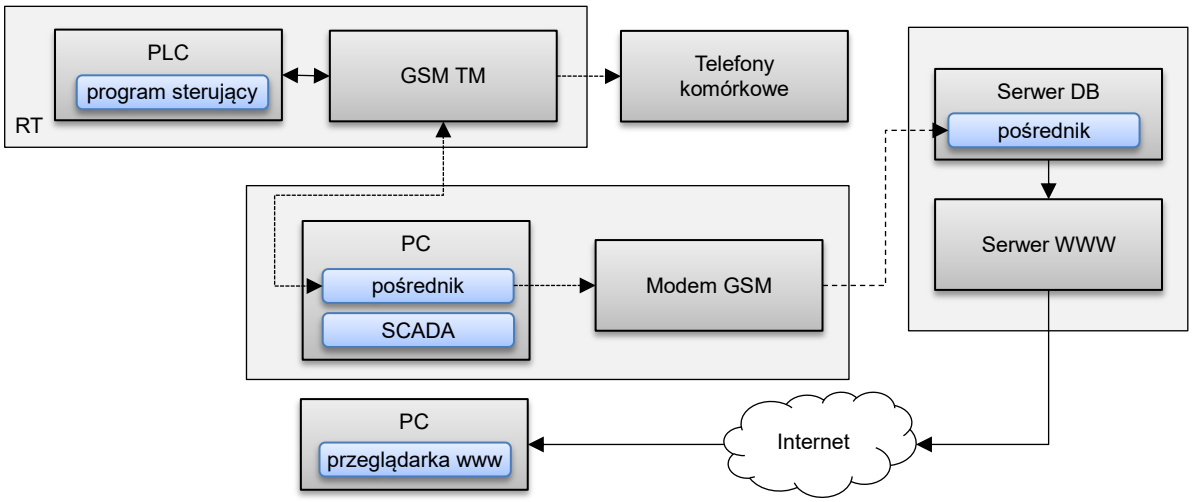
System został zrealizowany z użyciem modemów bezprzewodowych dla sieci komórkowych i radiolinii. Wykorzystano:

- sterowniki PLC klasy kompaktowej do realizacji sterowań lokalnych na obiektach,
- dedykowane sterowniki telemetryczne (GSM TM) dla obsługi powiadamiania SMS oraz transmisji danych do lokalnego systemu wizualizacji,
- system klasy SCADA do wizualizacji obiektów w ramach systemu lokalnego,
- modem GPRS do realizacji połączenia z serwerem www,
- radiomodem do połączeń z obiektami poza zasięgiem komórkowym,
- program pośrednika (ang. middleware) do integracji obiegów informacyjnych w sieci komórkowej z systemem SCADA oraz serwerem,
- bazę danych do zbierania historii,
- serwer www i skrypty PHP do celów publikacji na stronach webowych.

❖ **Schemat systemu**

Na rysunku 135 przedstawiono ogólny schemat konceptualny systemu. Występują tu cztery obszary działania. Pierwszy to źródła informacji, czyli obiekty rozproszone, obsługiwane przez PLC oraz połączone z nimi sterowniki dedykowane do funkcji telemetrycznych.

W każdym obiekcie PLC zapewnia realizację zadań sterowania i regulacji dla urządzeń obiektowych. Sterownik telemetryczny zapewnia powiadamianie przez SMS oraz dostarczanie danych do dalszego przetwarzania w ramach funkcji wizualizacji. Z tego punktu widzenia jest to podsystem lokalny ze scentralizowanym przetwarzaniem i interfejsem integracyjnym. Obszar drugi to budynek nastawni, gdzie pracuje nadzorczy komputer PC z oprogramowaniem integracyjnym oraz systemem SCADA. Dodatkowo, ze względu na brak sieci przewodowych, znajduje się tam modem GPRS umożliwiający transmisje stanu obiektów rozproszonych do bazy danych, pracującej na zewnętrznym serwerze dzierżawionym. Trzeci obszar to serwerownia z serwerem bazy danych i serwerem www wraz z oprogramowaniem integracyjnym. Obszar czwarty to przestrzeń publiczna, gdzie pracują sieci oraz zdalne stacje wizualizacyjne zrealizowane z użyciem usług webowych. W obszarze tym pracują również radiomodemy niepokazane na schemacie. Radiomodemy[108] zestawiają połączenie pomiędzy PLC obiektu znajdującego się poza zasięgiem komórkowym a najbliższą stacją mającą zasięg stabilny.



Rys. 135. Schemat rozproszonego systemu telemetrycznego
Fig. 135. Schema of a distributed telemetric system

❖ **Wady i zalety**

Do zalet rozwiązania można zaliczyć:

- dużą niezależność i separację funkcjonalną podsystemów,
- mobilność rozwiązania,
- niski koszt wdrożenia.

Stwierdzona wada to względnie wysoki koszt utrzymania połączeń sieciowych.

[108] Nie są przedstawione na rysunku.

7.1.2. System obsługi linii produkcyjnych

Przykładem aplikacji sterowania oraz oprogramowania typu SCADA i MES może być system uruchomiony na liniach produkcyjnych pewnego produktu. Zintegrowane są cztery warstwy oprogramowania w przedsiębiorstwie. System sterowania, nadzór i wizualizacja, wspomaganie zarządzania produkcją oraz integracja z systemem zarządzania przedsiębiorstwem.

❖ **Założenia**

Na wstępie sformułowano następujące założenia i wymagania względem funkcjonowania systemu.

- Linia produkcyjna składa się z kilku sekcji odpowiedzialnych za przygotowanie surowca, wytworzenie produktu, testy i odbiór.
- System sterowania obsługuje elementy czujnikowe i wykonawcze linii, działając zgodnie z aktualnie wybraną recepturą. Sterowania i regulacje procesem są realizowane tylko i wyłącznie przez system RT.
- Program nadzorczo-wizualizacyjny pracuje na IPC przy stanowisku operatora przy każdej linii i stanowi integralną część ISP oraz jest połączony ze zdalną bazą danych. SCADA służy w tym przypadku tylko jako lokalny interfejs nadzorczy linii dla operatora. Wszelkie sterowania i regulacje wykonywane pozornie z jego poziomu są w rzeczywistości przekazywane do innych węzłów ISP i przez nie wykonywane w reżimie HRTS.
- Operator ma możliwość podglądu stanu procesu, parametryzacji, ręcznego oddziaływania na proces przez interfejs programu nadzorczego oraz wydruku odpowiednich dokumentów chrakteryzujących uzyskany produkt.
- Istnieją dedykowane stanowiska zarządzające, gdzie kierownicy mogą śledzić jakość wytwarzania i statystyki produkcji.
- Część informacji z systemu trafia do zakładowego systemu wspomagania zarządzania klasy ERP.

❖ **Rozwiązanie**

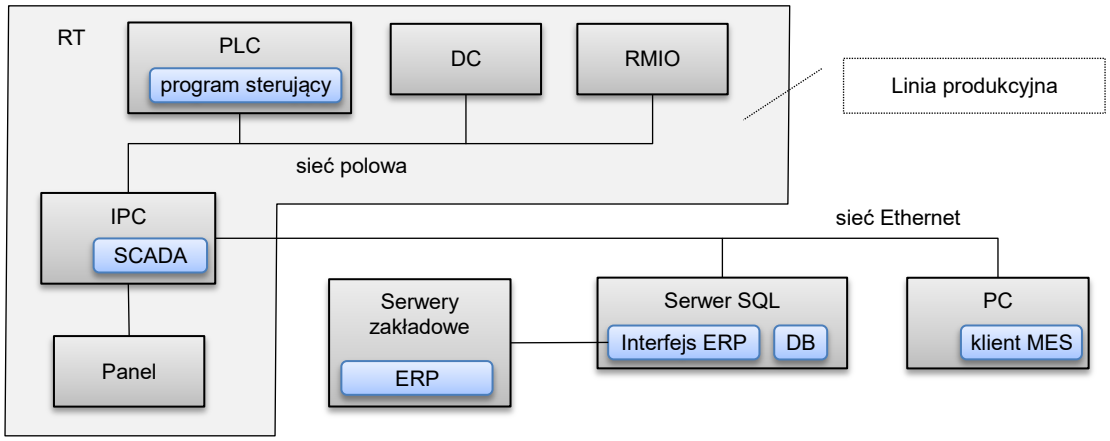
Do realizacji ISP wykorzystano:

- sterownik PLC wraz z układami zdalnych IO i obsługą sieci polowej,
- sterowniki napędów (DC),
- komputer klasy IPC do uruchomienia systemu SCADA,
- panel ekranowy dla komputera IPC,

- sieć polową dla połączenia PLC z układami IO, sterownikami napędów, komputerem oraz innymi węzłami,
- sieć Ethernet dla połączeń z bazą danych,
- oprogramowanie klasy SCADA,
- komputer serwerowy, serwerowy system operacyjny oraz oprogramowanie serwera SQL wraz z procedurami obsługi funkcjonalności MES oraz interfejsem danych dla zakładowego systemu ERP,
- oprogramowanie klasy MES.

❖ Schemat systemu

Na rysunku 136 przedstawiono ogólną ideę ISP obsługującego linię technologiczną. PLC, DC, RMIO oraz niepokazane na schemacie elementy diagnostyczne wraz z siecią polową, które pracują w reżimie czasu rzeczywistego i realizują sterowanie. IPC i panel stanowią lokalny interfejs dla operatora. Razem tworzą ISP obsługujący daną linię produkcyjną.



Rys. 136. Schemat idei współpracy elementów systemu
Fig. 136. Schema of cooperation of system elements

Każda linia produkcyjna jest wyposażona w taki system. Serwer baz danych obsługuje wszystkie linie. Oprogramowanie MES jest uruchamiane na wybranych komputerach PC działu produkcyjnego. Z poziomu klienta MES można uzyskiwać dostęp do analiz i raportów dla dowolnej linii produkcyjnej. Bezpieczeństwo dostępu jest zapewniane przez autoryzację użytkowników na poziomie systemów operacyjnych stacji roboczych oraz serwera, a także na poziomie autoryzacji serwera i baz danych SQL. Nad standardowymi mechanizmami są uruchomione dodatkowe procedury związane z lokalnymi politykami zakładu. ERP uzyskuje dostęp do danych przez specjalnie przygotowywane widoki na poziomie serwera.

❖ **Wady i zalety**

Do zalet rozwiązania można zaliczyć:

- dobre rozdzielenie kompetencji użytkowników,
- odseparowanie funkcjonalności warstw.

Stwierdzona wada to brak zabezpieczenia przed utratą komunikacji z serwerem.

7.1.3. Rozproszony system obsługi maszyn

Celem systemu jest śledzenie różnorodnych parametrów produkcji prowadzonej na kilku halach produkcyjnych z użyciem parku maszyn dwojakiego typu. Monitoring pracy odbywa się lokalnie przy każdej maszynie, jak i globalnie w skali całej produkcji. Jest to zatem system składający się z rozproszonych lokalnych systemów sterowania maszynami, systemu lokalnego klasy HMI oraz nadrzędnego klasy MES.

❖ ***Założenia***

Istnieje kilka kluczowych założeń i wymagań względem działania systemu.

- System sterowania odpowiada za obsługę *n* maszyn wymagających utworzenia lokalnego systemu sterowania oraz *m* maszyn posiadających swój sterownik dedykowany realizujący podstawowe funkcje maszyny.
- W obu przypadkach system sterowania współpracuje z elementami zadajników operatora (przyciski, stacyjki) oraz komputerem z oprogramowaniem HMI.
- Sterowanie maszyną jest samodzielne i do pracy nie potrzebuje systemu wizualizacji ani systemu zarządzania. Jednak do parametryzacji produkcji niezbędny jest interfejs HMI.
- W bazie danych zbierane są dane dotyczące zdarzeń z przebiegu sterowania maszyn oraz ich obsługi. Dane są zbierane na bieżąco. Na ich podstawie dokonywane są analizy i generowane raporty. System MES i oprogramowanie HMI mogą korzystać z zapisanych danych, odwołując się zarówno do danych bieżących, jak i historycznych (okresu czasu), produktu, typu zdarzeń itp.
- Warunki środowiskowe pracy maszyn nie są trudne i niewiele odbiegają od biurowych. Istnieje zwiększone zagrożenie zaburzeń EMC oraz wibracje.
- Systemy działają niezależnie przy braku komunikacji. Niedopuszczona jest utrata danych z przebiegu produkcji.
- Rozległość systemu nie przekracza 100 m w jednej hali.

❖ **Rozwiązanie**

Do stworzenia systemu wykorzystano następujące elementy:

- sterownik PLC przy każdej maszynie do obsługi zadań sterowania z wbudowaną obsługą protokołu sieci przemysłowej na łączu szeregowym,
- komputer klasy IPC przy każdej maszynie do uruchomienia oprogramowania HMI oraz lokalnych baz danych SQL,
- komputer serwerowy dla centralnego serwera SQL służącego do gromadzenia danych o całej produkcji i obsługi zapytań od klientów MES,
- sieć polową dla połączenia sterownika z IPC,
- sieć Ethernet do zbierania danych z maszyn wraz z przełącznikami dla każdej hali,
- stacje robocze PC do uruchamiania programów klienckich systemu MES,
- oprogramowanie sterujące dla PLC,
- oprogramowanie HMI dla IPC,
- oprogramowanie pośredników dla wymiany danych pomiędzy częścią RT a non-RT,
- oprogramowanie serwerów SQL dla maszyn oraz dla repozytorium centralnego,
- oprogramowanie klasy MES do obsługi analiz i raportów na stacjach PC.

❖ **Schemat systemu**

Na rysunku 137 przedstawiono ogólny schemat zrealizowanego systemu z uwypukleniem przepływów informacji i związanych z nim obiegów informacyjnych. Przekazywanie informacji między komponentami maszyny a sterownikiem odbywa się w czasie rzeczywistym. Między sterownikiem a pośrednikiem komunikacja jest realizowana siecią zdeterminowaną czasowo, choć sam pośrednik już nie gwarantuje obsługi uzyskanych danych w określonym czasie. Gwarantuje natomiast zachowanie wszystkich danych i ich kolejności. Podobnie następuje przekazanie danych w środowisku IPC. Pomiędzy bazami lokalnymi a bazą centralną działa replikacja utrzymująca spójność czasową baz z dokładnością do cyklu replikacji. Ewentualne problemy z dostępem do maszyn są rozwiązywane po wznowieniu komunikacji. W czasie jej braku dane są składowane tylko lokalnie.

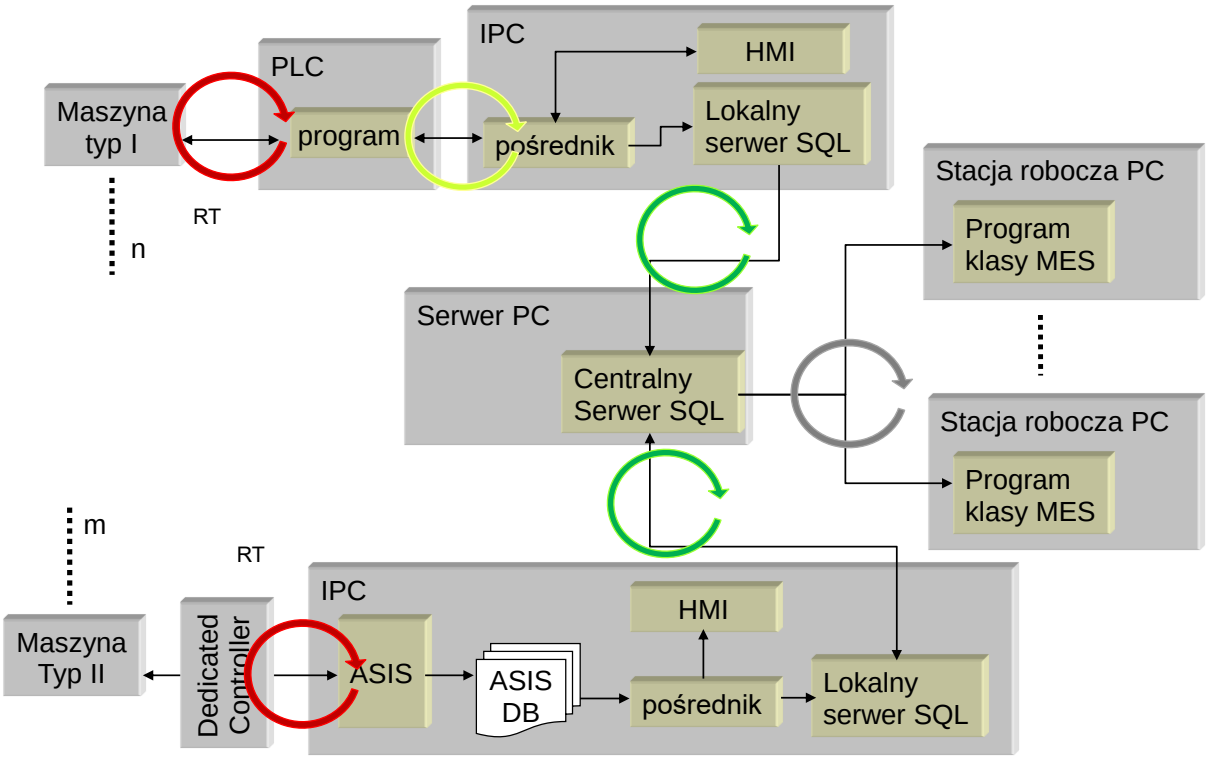
Obsługa części MES odbywa się w sieci zakładowej i nie ma żadnych ograniczeń czasowych. Występujący na rysunku skrót ASIS oznacza oprogramowanie i bazę danych specyficzne dla danej maszyny (ang. Application Specific Integrated Software).

❖ **Wady i zalety**

Do zalet rozwiązania można zaliczyć:

- duże bezpieczeństwo i niezawodność pozyskiwania danych,
- niewielkie opóźnienia czasu w warstwie MES (do kilku minut).

Stwierdzona wada to podatność na nieautoryzowany dostęp fizyczny do IPC.



Rys. 137. Schemat idei systemu i przepływu danych
Fig. 137. Schema of system and data flow

7.1.4. System sterowania i monitorowania

Prezentowany przypadek ilustruje zastosowanie różnych urządzeń i kanałów komunikacji dla systemu sterowania i lokalnego nadzoru.

❖ **Założenia**

Przedmiotem aplikacji jest obiekt, gdzie cześć wykonawcza musi pozostać oddalona od części sterującej. Obiekt jest rozproszony i znajduje się w różnych budynkach. Zadaniem jest skonstruować system sterowania i regulacji obiektu, wizualizacji HMI oraz zarządzania zdalnego z możliwością zdalnego sterowania.

- Obiekt składa się z kilku rozproszonych urządzeń wykonawczych, sensorowych i napędów. Odległości między elementami dochodzą do kilkuset metrów.

- Regulacja odbywa się w dwóch trybach: automatycznym i manualnym. W trybie automatycznym włączony jest regulator PID, w trybie ręcznym urządzenia wykonawcze i przekształtniki pracują względem nastaw dokonanych przez operatora. Możliwe polecenia sterowania ręcznego to:
- załączenie lub wyłączenie urządzeń wykonawczych,
- nastawa wartości zadanej dla regulacji PID,
- zmiana trybu pracy regulatora,
- ustawianie parametrów pracy dla trybu manualnego,
- ISP musi realizować również śledzenie pracy urządzeń, aby zapewnić bezuderzeniowe przejście pomiędzy poszczególnymi trybami pracy,
- połączenia kablowe między budynkami są dostępne częściowo, dla niektórych występuje brak jakiegokolwiek połączenia kablowego.

❖ **Realizacja**

Do realizacji wykorzystano:

- Regulatory prędkości obrotowej pracujące przy obiekcie.
- Zdalną stację wizualizacyjną na bazie komputera klasy PC i oprogramowania klasy SCADA.
- Sterownik PLC z panelem HMI pracujący w oddaleniu od obiektu. Zastosowano sterownik z dwoma modułami rozszerzeń. Jego konfiguracja lokalna to:
- 10 wyjść dyskretnych,
- 24 wejść dyskretnych,
- 6 wejść analogowych,
- 3 wyjścia analogowe,
- 2 porty szeregowe.
- Sieć WiFi na bazie AP z antenami kierunkowymi oraz sieć komórkowa dla dostępu do kilku pomiarów znajdujących się poza zasięgiem innej komunikacji.

Poniżej zamieszczono wybrane zmienne obiektowe i wewnętrze prezentowane w kolejności wynikającej z ich przynależności funkcjonalnej.

Tabela 8

Przykładowe zmienne obiektowe

Rodzaj	*Typ*	*Nazwa*	*Opis*	*Zakres*
I	Int	aiiT1	Temperatura T1 na zasilaniu	0-150 st
I	Int	aiiT2	Temperatura T2 na powrocie X	0-150 st
I	Int	aiiT3	Temperatura T3 na powrocie Y	0-150 st
I	Int	aiiCP1	Ciśnienie P1 na zasilaniu	0-1,6MP
I	Int	aiiCP2	Ciśnienie P2 na powrocie	0-1,6MP
Wew.	Int	iDP	Różnica ciśnień	0-200kP
O	Int	aoiObrP1	Obroty pompy P1	0-100%
O	Int	aoiObrP2	Obroty pompy P2	0-100%
O	Bool	obPracaP1_roz	Sygnał załączenia P1	0,1
O	Bool	obPracaP2_roz	Sygnał załączenia P2	0,1
I	Bool	ibWyłP2	Wyłącznik P1	0,1
I	Bool	ibWyłP1	Wyłącznik P2	0,1
I	Bool	ibAwaria_P1	Awaria zasilania P1	0,1
I	Bool	ibAwaria_P2	Awaria zasilania P2	0,1
O	Bool	ibAwaria_P1	Sygnał z termika P1	0,1
O	Bool	ibAwaria_P2	Sygnał z termika P2	0,1
I	Bool	ibPlus	Przycisk zwiększający obroty	0,1
I	Bool	ibMinus	Przycisk zmniejszający obroty	0,1
Wew.	Dint	lCzasP1	Licznik czasu pracy P1	0-2147483647 (w zakresie nieujemnym)
Wew.	Dint	lCzasP2	Licznik czasu pracy P2	0-2147483647 (w zakresie nieujemnym)
…				

❖ **Schemat systemu**

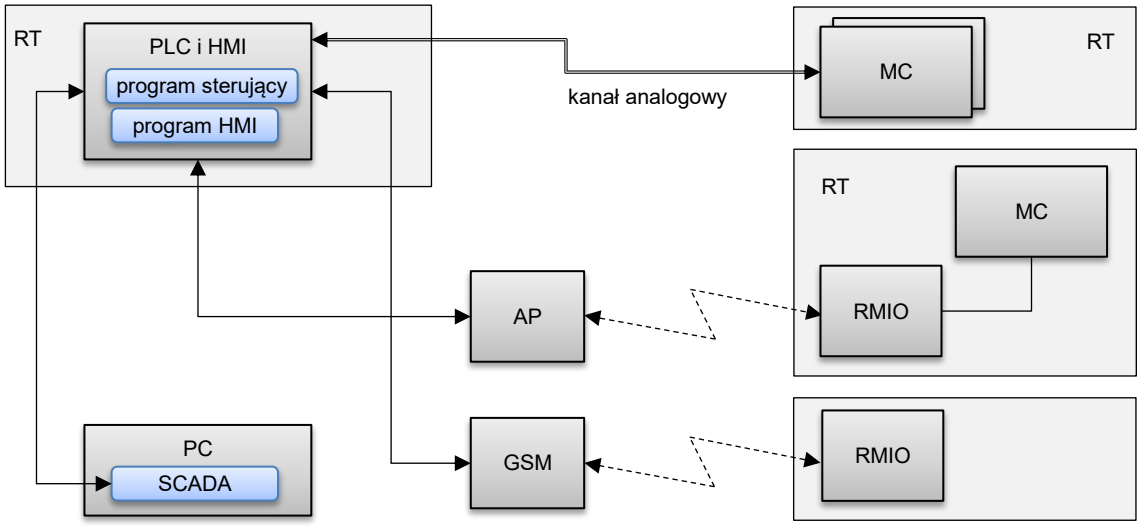
Na rysunku 138 przedstawiono ogólną koncepcję systemu. Układy IO sterownika są połączone z urządzeniami AKP, z których urządzenie pobiera dane oraz przez które steruje obiektem. Połączenia są kablowe wewnątrz budynków oraz bezprzewodowe pomiędzy nimi. Wszelkie połączenia z ograniczeniami czasowymi są dostępne w tym samym budynku co PLC. Dla pozostałych nie ma wymagań RT. Na sterowniku pracuje program sterujący oraz oprogramowany jest zintegrowany panel graficzny HMI. Zastosowaniem tego interfejsu jest szybka wizualna kontrola pracy urządzeń.

❖ **Wady i zalety**

Do zalet rozwiązania można zaliczyć:

- dużą niezawodność działania,
- niewielkie opóźnienia.

Stwierdzona wada to podatność na zaburzenia komunikacji radiowej.



Rys. 138. Schemat systemu dla różnych kanałów komunikacji
Fig. 138. Schema of system for various communication channels

7.1.5. Rozproszony system obsługi maszyn dziewiarskich

Prezentowany przypadek stanowi przykład połączenia systemu czasu rzeczywistego z narzędziami wizualizacji, przygotowania produkcji i wspomagania projektowania wyrobów.

❖ **Założenia**

Aplikacja dotyczy systemu obsługującego park maszyn dziewiarskich zarówno na poziomie sterowania, jak i nadzoru, przygotowania produkcji oraz projektowania wyrobów.

- System produkcyjny składa się z zestawu maszyn dziewiarskich oraz stanowiska nadzoru i przygotowania produkcji.
- Przy każdej maszynie znajduje się lokalne urządzenie umożliwiające sterowanie jej pracą.
- Operator maszyny dysponuje lokalnym panelem HMI do szybkiej diagnostyki, śledzenia etapów pracy maszyny oraz bieżącego stanu produkcji, a także wprowadzania własnej identyfikacji (wprowadzenia kodu identyfikacyjnego).
- Wymagane jest działanie programu narzędziowego umożliwiającego stworzenie programu pracy maszyny przez projektanta-technologa i przesłanie tego programu zdalnie do konkretnej maszyny (program dla programu sterującego).
- Istnieją trzy typy maszyn różniących się konstrukcją mechaniczną, elektryczną. Główna część rozkazów programowania maszyn jest taka sama. Nowsze maszyny mają dodatkowe funkcje, np. wykonanie ażuru.
- Wymagane jest monitorowanie pracy wszystkich maszyn na stanowisku nadzorczym.

Poniżej zamieszczono kilka wybranych uściśleń założeń dla ilustracji często spotykanej specyficzności stosowanej terminologii. Zachowano oryginalne skróty, nazwy i kolokwializmy występujące w specyfikacji.

- Program sterujący maszyną składa się z podprogramów (np.: rękaw, przód, tył). Istnieje możliwość przejścia do konkretnego podprogramu z konsoli operatorskiej.
- Sygnał „przesuw pręta" (I1A1) zdejmuje sygnał „poszerzanie wodzikiem" (Q13B5) i odlicza poszerzenia.
- Sygnały od Q25B1 do Q30B6 są kasowane przez „obrót wału" (I3A3).
- Sygnał „przesuw pręta" (I1A1) odlicza jedno poszerzenie.
- Sygnał „obrót wału" (I3A3) odlicza jeden rządek.
- Istnieje możliwość uproszczenia funkcji przesuwu górnej i dolnej szyny ażuru o podanie jednorazowo liczby igieł do przesuwu.

❖ **Rozwiązanie**

Do stworzenia rozwiązania wykonano ISP obsługujący maszyny, oprogramowanie nadzorcze i narzędziowe. W skład rozwiązania wchodzi:

- Sterownik klasy PLC z modułami IO oraz koprocesorem komunikacyjnym dla każdej maszyny.
- Konsola operatorska z wyświetlaczem LCD.
- Klawiatura numeryczna wraz z klawiszami funkcyjnymi.
- Oprogramowanie klasy SCADA zintegrowane z dedykowanym programem narzędziowym klasy CAD.
- Program narzędziowy klasy CAD do projektowania wyrobów i generacji programów dla sterowników maszyn.
- Przyjęto 20% rezerwę na liczbę wejść/wyjść dla każdej maszyny.

Poniżej zamieszczono fragment specyfikacji obwodów wejść i wyjść dla sygnałów dyskretnych.

Tabela 9

Przykładowe sygnały obiektowe

Rodzaj	*Opis sygnału*
I	Sygnał obrotu wału (sygnał z krzywki)
I	Przesuw pręta
I	Praca ręka / automat – przełącznik
I	Blokada bezpieczeństwa
I	Blokada urządzenia przekładającego

cd. tabeli 9

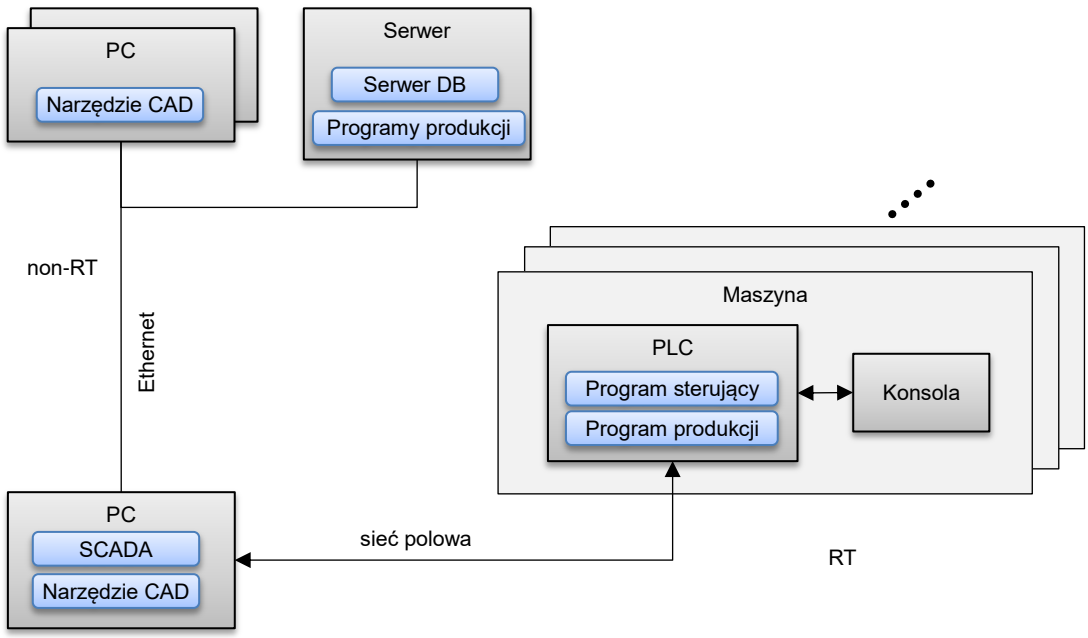
Rodzaj	*Opis sygnału*
I	Blokada pola środkowego
O	Zatrzymanie maszyny
O	Urządzenie przekładające w górę
O	Urządzenie przekładające w dół
O	Załączenie/wyłączenie wodzika 1 (1 zał./ 0 wył.)
O	Załączenie/wyłączenie poszerzania wodzikiem
O	Przestaw wału krzywkowego
O	Transport grzebieni do przekładania
O	Zwężanie przekładaczami
O	Poszerzanie przekładaczami
O	Ruch rozpieraczy
O	Przekładanie ażuru
O	Blokada krzywki ażurującej
O	Ruch krzywki ażurującej
O	Ruch śruby ażuru

Rodzaje:

- I – wejście (ang. input),
- O – wyjście (ang. output).

❖ Schemat systemu

Na rysunku 139 przedstawiono ogólną koncepcje systemu.



Rys. 139. Ogólny schemat ISP dla maszyn dziewiarskich
Fig. 139. General ICS schema of knitting machines

Każda maszyna jest wyposażona w sterownik oraz konsolę. Sterownik przez układy IO i dzięki programowi steruje pracą układów elektromechanicznych maszyny. Konsola służy operatorowi i jest podłączona do sterownika niezależną siecią przemysłową na bazie łącza szeregowego. Wszystkie sterowniki są spięte jednym segmentem sieci przemysłowej. W sieci tej pracuje również komputer klasy PC, na którym system SCADA zbiera informacje o pracy maszyn i je prezentuje pracownikom działu utrzymania produkcji. Ponadto, w ramach zarządzania produkcją, ze stanowiska PC można przesłać na wybraną maszynę program określający, co i jak ma ona produkować. W przestrzeni sieci zakładowej pracuje również serwer produktów i stanowiska ich przygotowania.

❖ **Wady i zalety**

Do zalet rozwiązania można zaliczyć:

- Rozdzielenie kompetencji i funkcjonalności.
- Duża odporność EMC części związanej z maszynami.

Stwierdzona wada to niewielka szybkość przesyłania programów na maszynę.

7.1.6. Przykłady kanałów komunikacyjnych

Poniższy przykład nie ilustruje ISP, lecz przykłady kanałów komunikacyjnych wykorzystywanych w różnego rodzaju rozwiązaniach ISP. Komunikacja jest jednym z ważniejszych aspektów funkcjonowania ISP, a jej konstrukcja może zostać wykonana na wiele sposobów. Poniższe przykłady nie wyczerpują możliwości, a wręcz stanowią tylko maleńką namiastkę możliwych rozwiązań. Autor książki realizował wszystkie prezentowane przykłady zarówno w warunkach laboratoryjnych, jak i aplikacyjnych.

❖ **Kanały proste dla systemów lokalnych**

Idea ogólna polega na łączeniu urządzeń przy użyci sieci lokalnej.

Węzeł ISP ↔ LAN ↔ Węzeł/Węzły ISP

Siecią LAN może być zarówno sieć polowa, inna przemysłowa sieć lokalna, jak i łącze szeregowe. W poniższych przykładach pokazano kanał, a nie sieć. Zatem w danym przypadku urządzeń może być wiele.

- PLC (Modbus RTU) ↔RS485 ↔ PLC (Modbus RTU)
  Bardzo klasyczne i proste połączenie wielopunktowym łączem szeregowym. Umożliwia połączenie fizyczne wielu urządzeń w topologii magistrali, np. PLC, i uruchomienie wielu protokołów przemysłowych, jak np. Modbus, Profibus, SUKOnet K, MPI itp.

- PLC (CAN) ↔ CAN ↔ IPC (CAN/CANOpen, SCADA)
  Kanał rzadziej spotykany ze względu na mniej popularne wykorzystanie sieci CAN w ISP. Umożliwia wielopunktowe połączenie urządzeń w topologii magistrali. W przykładzie jednym z nich jest komputer IPC z kartą sieci CANOpen i system SCADA, a drugim sterownik PLC z wbudowaną obsługą sieci CAN.
- RMIO (Profibus) ↔ RS485 ↔ PLC (Profibus)
  Typowy i popularny przypadek podobny do pierwszego. Urządzenia są łączone w topologii magistrali przez sieć Profibus. Urządzenie RMIO funkcjonuje jako wyspa IO. PLC pracuje jako stacja typu master, a RMIO jako slave.
- Miernik → Pętla 4-20mA → RMIO (WorldFip) ↔ PC (WorldFip, SCADA)
  Przykład transmisji danych IO przez łącze analogowe do zdalnego modułu IO, a następnie siecią polową WorldFip do karty sieciowej komputera PC i dalej do systemu SCADA przez sterownik (ang. driver) sieci.
- PLC (ModbusTCP) ↔ Ethernet ↔ PC (ModbusTCP, PDS, DB)
  Wykorzystanie sieci Ethernet do łączenia sterowników i aplikacji na komputerze klasy PC. W przykładzie pokazano protokół ModbusTCP, serwer danych oraz bazę danych.
- PLC (Profinet) ↔ Ethernet ↔ RMIO (Profinet)
  Przypadek podobny do trzeciego, z tym że połączenie jest realizowane z wykorzystaniem sieci Ethernet i protokołu Profinet IO. Obsługa Profinet jest w tym przypadku wbudowana w urządzenia.

❖ **Kanały proste dla systemów zdalnych**

Idea jest podobna jak powyżej, z tym że z racji rozproszenia w skład łącza mogą wchodzić różne sieci, protokoły i urządzenia.

- PLC ↔ RS232 ↔ Modem ↔ linia telekomunikacyjna komutowana ↔ Modem ↔ RS232/RS485 ↔ PLC
  Ilustracja problemu zdalnego dostępu przez łącza komutowane. Modem ma łącze RS232 wymuszające komunikację punkt-punkt ze sterownikiem, i tym samym skonfigurowanie jego portu szeregowego na ten standard. Drugi modem i sterownik wymagają dopasowania różnych standardów łączy szeregowych.
- PLC (Modbus) ↔ RS485 ↔ Modem ↔ linia telekomunikacyjna stała ↔ Modem ↔ USB ↔ PC (Modbus, SCADA)
  Przypadek dotyczy łączenia zdalnego sterownika z aplikacją na komputerze PC. Połączenie jest zrealizowane jako transparentne. Dzięki temu aplikacja PC, działająca w roli ma-

stera może z punktu widzenia aplikacji i transportu komunikować się bezpośrednio ze sterownikiem. Oczywiście opóźnienia w takim kanale są większe, niż gdyby to była sieć lokalna.

- PC ↔ USB ↔ Bluetooth ↔ RS485 ↔ PLC
  Połączenie bezprzewodowe urządzeń z wykorzystaniem łączy szeregowych i Bluetooth. Mogą być używane inne standardy, np. WiFi lub Zigbee. Niestety, charakterystyka czasowa transmisji dla użytego protokołu względem aplikacji węzłów może być zaburzona przez kanał bezprzewodowy.
- PLC ↔ Profinet ↔ Router ↔ VPN (Internet) ↔ Router ↔ Profinet ↔ PC
  Jest to przykład zdalnego dołączania węzła do systemu lokalnego przez Internet. VPN zapewnia bezpieczeństwo transmisji oraz możliwość zarządzania ruchem.

❖ **Kanały złożone dla systemów lokalnych**

Komplikacja połączeń lokalnych zwykle ma miejsce w związku z integracją różnych protokołów.

Węzeł ISP ↔ LAN ↔ Adaptacja ↔ LAN ↔ Węzeł/Węzły ISP

Sieci LAN mogą stanowić różne sieci przemysłowe i łącza szeregowe. Pojęcie konwertera jest skrótem myślowym opisanym w 3.3.2.

- Licznik (Mbus) ↔ MBus ↔ Konwerter ↔ Ethernet ↔ PC (MBus, OPC, SCADA)
  Przykład odczytu licznika znajdującego się w systemie lokalnym, gdzie siecią komunikacyjną jest Ethernet. Stąd wymagane przejście ze standardu MBus, którym dysponuje licznik na sieć Ethernet, która stanowi sieć systemową ISP. W tym przypadku przejście jest transparentne, ale można użyć gatewaya zmieniającego protokół, np. na ModbusTCP itp.
- Enkoder (Modbus RTU) ↔ RS485 ↔ Konwerter ↔ USB ↔ PC (Modbus RTU)
  Częsty przypadek konieczności podłączenia urządzenia obiektowego do aplikacji na PC. W przykładzie zastosowano transparentne przejście z medium RS485 na USB.
- PLC ↔ EtherNet/IP ↔ Konwerter ↔ Profibus↔ PLC
  Przykład dostosowania komunikacji z użyciem sieci polowej i sieci Ethernet. Konwerter jest rodzajem gatewaya.
- PLC ↔ RS232 (Modbus TCP) ↔ Konwerter ↔ Ethernet (Modbus TCP) ↔ SCADA
  Dołączenie sterownika z portem szeregowym do sieci Ethernet, z którą pracuje ISP i np. stacja SCADA. Dostęp do sterownika jest realizowany jako połączenie transparentne dla protokołu Modbus TCP.

- RMIO ↔ Profinet ↔ Proxy ↔ Profibus ↔ PLC
  Przykład integracji starszych i nowszych rozwiązań komunikacyjnych. Pokazano przejście pomiędzy węzłem działającym w sieci Profibus (sieć polowa) a węzłem działającym w sieci Profinet (sieć RTE). Translacja protokołów dokonywana jest przez tzw. urządzenie proxy, które jest rodzajem gatewaya.
- PLC ↔ EPL ↔ Gateway ↔ CAN ↔ PLC
  Podobne przejście jak powyżej. Pokazano połączenie sterownika z interfejsem EPL (sieć RTE) ze sterownikiem z interfejsem CAN. Translacja protokołów dokonywana jest przez specjalny gateway dostępowy EPL.

❖ **Kanały złożone dla systemów zdalnych**

Złożone kanały zdalne występują, gdy istnieje potrzeba dopasowania technik i środków komunikacyjnych do konkretnych istniejących potrzeb. Należy unikać kanałów złożonych, gdyż wzrost złożoności zwiększa awaryjność połączenia. Pokazane elementy infrastruktury takich kanałów są mocno uproszczone. Szczegóły zależą od konkretnej infrastruktury.

- PLC ↔ Jump Server ↔ LAN ↔ Router ↔ Internet ↔ Router ↔ LAN ↔ PC (klient)
  Przykład kanału zdalnego dostępu do ISP przez serwer pośredniczący. Funkcje związane z takim dostępem mogą być związane ze zdalną obsługą systemu, diagnostyką, serwisem itp.
- PLC ↔ RS485(Modbus) ↔ RS232 ↔ Modem GPRS ↔ GSM ↔ Internet ↔ LAN ↔ PC (PDS) ↔ IPC (DDE) ↔SCADA ↔ SQL DataBase ↔ PHP ↔Serwer WWW ↔ Internet ↔ Klient WWW
  Jest to kanał do zdalnego zbierania danych i ich publikacji. Węzły ISP są rozproszone, a dostęp do nich uzyskuje się przez sieci komórkowe oraz Internet. Lokalnie w ISP wykorzystywana jest sieć lokalna, serwer danych oraz aplikacje do lokalnej prezentacji danych. Do publikacji wykorzystany jest serwer bazy danych i serwer www.
- IO → modem SMS → PC (DataServer) → Modem GPRS ↔ GSM → Internet → LAN ↔ PC (DataServer) ↔ SQL DataBase ↔ Serwer WWW ↔ Internet ↔ Klient WWW
  Przykład podobny do powyższego, tylko w nieco innej topologii połączeń.

Poza pokazanymi przykładami można tworzyć dowolnie inne kanały, pamiętając o zachowaniu wymaganej charakterystyki czasowej przekazywania danych i dopasowaniu kodowania, transportu i adresacji (zob. 3.3). Przykłady charakterystyki dla połączenia z wykorzystaniem różnych sieci pokazano w [125] oraz [123]. Przykładowe charakterystyki połączeń dla kanału zdalnego zaprezentowano w [119].

7.2. Spis rysunków

7.3. Spis tabel

7.4. Indeks terminów

Poniżej zamieszczono wykaz pojęć, które są z jakiegoś punktu widzenia specyficzne i warte uwagi, wraz z odwołaniami do stron, gdzie zostały zdefiniowane lub występują.

7.5. Objaśnienia

Poniżej zamieszczono opis używanych symboli oraz uwagi do wykorzystywanych treści.

7.5.1. Używane symbole

Symbol używany dla zwrócenia uwagi na rzeczy ważne, choć pozornie poboczne względem poruszanego tematu.

Kwestie dyskusyjne. Prezentowane opinie mogą być subiektywne i skłaniać do dyskusji.

7.5.2. Uwagi

Wszelkie nazwy, loga i znaki towarowe są własnością firm, do których należą.

Użyte symbole graficzne mogą przypominać rzeczywiste urządzenia dla lepszego zobrazowania przekazu. Jednak ich podobieństwo nie ma związku z kontekstem użycia.

Prezentowane zdjęcia są wykonane na różnych obiektach w kraju i za granicą. Wszystkie informacje mogące stanowić dane poufne, wrażliwe i objęte ochroną są ukryte lub nie są prezentowane na zdjęciach.

Autorem zdjęć jest autor książki, chyba że wskazano inaczej.

Dobór platform sterownikowych do demonstracji kodu jest przypadkowy i związany z pokazaniem różnych podejść. Nie odzwierciedla żadnych preferencji.

Rynek urządzeń i oprogramowania jest bardzo dynamiczny. Przedstawione przykłady klasyfikacji, przynależności i funkcjonalności mogą się zmieniać wraz z rozwojem produktów.

Autor oraz Wydawnictwo dołożyli wszelkich starań, aby zamieszczone w niniejszej książce informacje były rzetelne i zgodne z bieżącym stanem wiedzy. Nie biorą jednak żadnej odpowiedzialności za ewentualne wykorzystanie zawartych informacji oraz tego skutki. W związku z tym nie ponoszą również żadnej odpowiedzialności za ewentualne naruszenie tym praw autorskich lub patentowych.

WYBRANE ZAGADNIENIA PROJEKTOWANIA SYSTEMÓW INFORMATYKI PRZEMYSŁOWEJ

STRESZCZENIE

Niniejsza monografia jest poświęcona zagadnieniom projektowania systemów informatyki przemysłowej oraz systematyzacji i analizie pojęć związanych z tym zagadnieniem. Zasadnicza zawartość książki podzielona jest na trzy części: modele, elementy systemów oraz bezpieczeństwo.

W części pierwszej określono standardowe pojęcia, modele systemów i ich elementów składowych oraz abstrakcje konstrukcyjno-funkcjonalne wykorzystywane przy tworzeniu systemów informatyki przemysłowej. Znajomość abstrakcyjnych środków opisu i umiejętność ich użycia umożliwiają prowadzenie prac projektowych niezależnie od konkretnych rozwiązań sprzętowych i technologii. Dlatego, oprócz przedstawienia podejść na bazie istniejących norm, zaproponowano wgląd w zagadnienie z użyciem pojęć szerszych i bardziej uniwersalnych niż powszechnie stosowane. Podjęto dyskusje wykorzystania znanych wzorców projektowych oraz adaptacji najnowszych idei konstruowania systemów rozproszonych. Poruszono problem testowania i zaproponowano stosowne wytyczne. Rozważania zilustrowano zarówno wynikami badań, jak i odniesieniami do praktyki, w tym pracami naukowymi oraz doświadczeniem aplikacyjnym autora książki.

W części drugiej przedstawiono elementy sprzętowe i programowe umożliwiające konstruowanie systemów. Zawarto opis dziedzin zastosowań, typowych zadań oraz miejsca i roli danego elementu w omawianej klasie systemów. Opisy nie dotyczą konkretnych produktów, tylko ich klas, co uniezależnia proces poznawczy od producenta i zmusza czytelnika do rozważania struktury systemu, przepływów i powiązań informacyjnych oraz modeli danych i ich przetwarzania, a nie do dopasowywania konkretnych komponentów. Poruszono także zagadnienia dotyczące komunikacji w systemach, zarówno w kontekście lokalnym, jak i intersieciowym. W opisie uwzględniono istotne kwestie wykorzystania mediów sieciowych w kontekście stosowania komputerowych sieci przemysłowych. Opisano również zagadnienia integracji międzysystemowej oraz współdziałania elementów heterogenicznych na różnych poziomach abstrakcji.

Część trzecia poświęcona jest szeroko pojętym zagadnieniom związanym z projektowaniem bezpiecznych systemów informatyki przemysłowej oraz z bezpieczeństwem ich eksploatacji. Przedstawiono zagadnienia dotyczące konstrukcji systemów względem bezpieczeństwa dostępowego i funkcjonalnego oraz zagadnienia związane z niezawodnością działania takich systemów i ich elementów. Opisano standardowe pojęcia związane z bezpieczeństwem, stosowane miary i normy, typowe zagrożenia i ryzyka spotykane w omawianej dziedzinie, a także zagrożenia wynikające z nowych technologii. Zaproponowano wytyczne projektowe zwiększające poziom bezpieczeństwa już od etapu planowania aż po realizacje i wdrożenie.

Książka zawiera również przykłady rzeczywistych rozwiązań, ilustrujące poruszane kwestie oraz bogaty zestaw materiałów źródłowych. Poruszane tematy są dyskutowane w kontekście dokonanych oraz bieżących badań prowadzonych w ośrodkach krajowych oraz zagranicznych, w tym badań, projektów, aplikacji i wdrożeń autora.

SELECTED ISSUES OF INDUSTRIAL COMPUTER SYSTEMS DESIGN

ABSTRACT

This monograph is devoted to the design of the industrial computer systems and analysis of the major terms related to such process. The main body of the book is divided into three parts: models, systems elements, and security and safety.

In the first part, the standard terms are defined, the system models and its constituent elements are presented, as well as some abstract models are suggested, all of which concern the structure and the functionality of the industrial system. The knowledge of abstract descriptions and the skill to use them allow to make a design work independently from the given hardware and software solutions and technologies. Thus, besides showing the approaches based on existing standards, an insight into the issue is proposed which uses broader and much more universal terms than the ones usually employed. The utilization of the well-known design patterns is discussed, as well as adaptation of brand new ideas of constructing distributed systems. The issues of system testing are taken into account and appropriate guidelines are delivered. The presented considerations are illustrated by research results and by references to practice, including the scientific work and applications experience of the author.

In the second part of the book, the hardware and software elements which allow to build up the system are presented. The description of the usage domain, typical tasks, and a place and a role of the given element in the considered system kind is included. The descriptions do not refer to specific products but to their classes. This makes the cognitive process independent from producers, and forces the reader to consider the system structure, data flows and models, applications relations, and ways of processing, but not only to fit the specific components. The issues of distributed system communication are also taken into consideration, both in local and remote context. The relevant questions of media usage are deliberated within the context of industrial networks. The intersystem integration issues are also described, as well as collaboration of heterogeneous elements on the various levels of abstraction.

The third part of the book is devoted to broadly defined problems connected with the process of designing industrial informatics systems in a safe way, as well as to the security of the

systems' operation. The issues of the system construction concerning security and safety, as well as issues related to the system and the dependency of its elements are shown. The standard terms of security and safety, measure and norms, typical threats and risks used in the considered domain, as well as threats coming from the new technologies are described. The design guidelines are proposed which can increase the security and safety level, starting from the planning, through implementation and commissioning, to deployment process.

The book also contains examples of real solutions, which highlight the considered issues, as well as a rich set of bibliography and sources. The considered topics are discussed in relation to both past and current research lead by domestic and foreign entities, including research, projects, applications and implementations being made by the author.