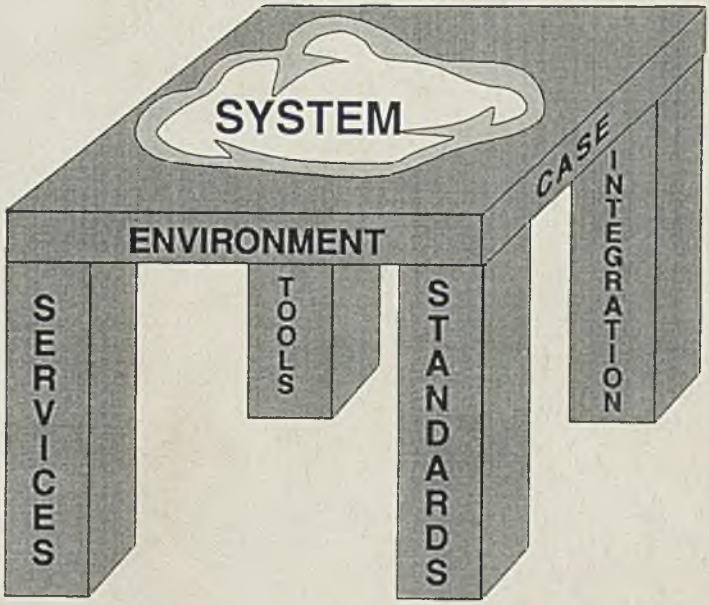
Zdzisław Szyjewski

Komputerowe wspomaganie tworzenia systemów informatycznych



Stół montażowy dla tworzenia systemu informatycznego

Zdzisław Szyjewski

Komputerowe wspomaganie realizacji systemów informatycznych

Szczecin 1994

© Copyright by Zdzisław Szyjewski

SPIS TREŚCI

Wstęp 7

1. Tworzenie systemów informatycznych zarządzania 19

1.1. Struktura systemu informatycznego 19
1.2. Metody i techniki tworzenia systemów informatycznych 24

1.2.1. Metody tworzenia sytemów informatycznych 24
1.2.2. Techniki heurystyczne 27
1.2.3. Techniki specyfikacji 29
1.2.4. Techniki szczegółowe 31

1.3. Jakość systemów informatycznych 33
1.4. Przesłanki standaryzacji 39
1.5. Metodologie tworzenia systemów informatycznych 41

1.5.1. Podstawy metodyczne tworzenia systemów informatycznych 41
1.5.2. Strukturalizacja procesu tworzenia systemu informatycznego 48
1.5.2.1. Podejście do procesu tworzenia 48
1.5.2.2. Analiza i projektowanie strukturalne 52

2. Metodyka punktów węzłowych 57
2.1. Ogólne założenia metodyki 57
2.2. Punkty węzłowe procesu tworzenia systemu informatycznego ... 58
2.3. Tablica zależności czasowych realizacji punktów węzłowych 82
2.4. Powiązania sieciowe punktów węzłowych 88

2.5. Propozycja podziału procesu tworzenia systemu na etapy 92
2.5.1. Założenia metodologiczne 92
2.5.2. Podział procesu tworzenia systemu na etapy 93
2.6. Propozycja podziału cyklu życia systemu 95

3. Normy i standardy w informatyce 99

3.1. Przesłanki standaryzacji 99
3.1.1. Strategie standaryzacji 99
3.1.2. Światowe przedsięwzięcia standaryzacyjne w informatyce . 103
3.2. Zakres standaryzacji 108
3.3. Systems Application Architecture — SAA 116
3.3.1. Założenia i cechy SAA 116
3.3.2. Standard dialogu 119
3.3.3. Standard programowania 120
3.3.4. Standard wspomagania komunikacji 122
3.3.5. Standard aplikacji 122
3.3.6. Zalety SAA 124
3.3.7. Kierunki rozwoju 126
3.4. Walidacja systemów informatycznych 127
3.4.1. Problemy jakości oprogramowania 127
3.4.2. Znaczenie walidacji 129
3.4.3. Warunki prawidłowej walidacji 132
3.4.3.1. Specyfikacja wymagań dla systemu 132
3.4.3.2. Walidacja systemu informatycznego 134
3.5. Zastosowanie norm 136

4. Komputerowe narzędzia wspomagania tworzenia systemu informatycznego 139

4.1. Przesłanki komputerowego wspomagania 139
4.2. Klasyfikacja narzędzi wspomagania 143
4.2.1. Składnica informacji (repository) 144
4.2.2. Pielęgnacja systemu (maintenance) 148
4.2.3. Wspomaganie cyklu życia systemu 149

4.3. Kierunki rozwoju narzędzi CASE 151
4.3.1. Rozwój oprogramowania narzędziowego 151
4.3.2. Systemy zarządzania bazą danych i języki 4GL 152
4.3.3. Środowisko COHESION 157
4.3.3.1. Założenia koncepcji środowiska COHESION 157
4.3.3.2. Narzędzia wspomagania firmy Digital 160
4.4. Nowa jakość systemów informatycznych — systemy otwarte 170

Wnioski 177

Literatura 181

Spis rysunków i tabel 190

Wstęp

Rozwój informatyki objawiający się coraz powszechniejszym stosowaniem informatycznych środków technicznych i programowych w codziennej działalności, na coraz większej ilości stanowisk pracy, stawia przed informatykami nowe wyzwania. W zaistniałej sytuacji oczywisty jest postulat wzrostu produktywności zespołów wykonujących aplikacje informatyczne. Coraz większy nacisk kładziony jest na wysoką jakość wykonywanych produktów. Stały, szybki rozwój sprzętu wymusza elastyczność rozwiązań oraz możliwość łatwego przeniesienia ze starszego sprzętu na nowszy. Powstające produkty często muszą współpracować z powstałymi wcześniej aplikacjami, co wymaga odpowiednich mechanizmów integracji realizowanej na różnych płaszczyznach.

Komputerowe wspomaganie inżynierii programowania (CASE — Computer Aided Software Engineering), w oczekiwaniu wielu informatyków miało być tym co pozwoli sprostać wzrastającym oczekiwaniom. Powstające aplikacje powinny nadążać za wzrastającymi potrzebami użytkowników, które charakteryzują się nie tylko wzrastającą szybko ilością powstających aplikacji ale równie szybko wzrasta złożoność i wymagania jakościowe nowo tworzonych systemów.

Nowe aplikacje powstają w wykształtowanym już środowisku informatycznym, co zmusza autorów do integracji nowych rozwiązań z już użytkowanymi systemami. Aplikacje nie są już prostymi systemami ewidencyjnymi ale są to często bardzo złożone systemy informatyczne wymagające dużej wiedzy i nowoczesnych metod wytwarzania.

Wysokie wymagania jakościowe w połączeniu z koniecznością opracowywania nowych systemów w krótszym czasie spowodowały, że przydatność dotychczasowych metod wytwarzania systemów informatycznych była

coraz mniejsza. Konieczne stało się zautomatyzowanie prac nad systemem. Inżynieria programowania w połączeniu z komputerowym wspomaganiem gwarantowała spełnienie wysokich wymagań jakościowych oraz umożliwia wzrost produktywności tworzenia systemów informatycznych.

Pierwsze doświadczenia ze stosowania CASE publikowane na konferencjach (146) wskazywały na wyraźny wzrost produktywnosci i jakości opracowywanych systemów. Stosowanie CASE stało się modą, w dobrym tego słowa znaczeniu, przy opracowywaniu systemów informatycznych. Potrzeba zmiany technologii wytwarzania systemów była oczywista i koncepcja CASE trafiła na podatny grunt. W Polsce również zaistniały warunki do stosowania nowoczesnych narzędzi wytwarzania. Klasa tworzonych aplikacji przeniosła się z prostych systemów ewidencyjnych, dla których komputer klasy PC był wystarczający, na rzecz bardziej złożonych systemów, gdzie dotychczasowe metody wytwarzania były niewystarczające.

Nowoczesna technologia wytwarzania systemów informatycznych posiada liczne uwarunkowania. Efektywne jej stosowanie wymaga nowego środowiska pracy, innych metod wytwarzania i nowej organizacji prac. Komputerowe wspomaganie nie może by złotym środkiem na wszystkie dotychczasowe bolączki procesu wytwarzania. Duże oczekiwania w zderzeniu z praktyką tworzenia systemów informatycznych powodują, że wyniki są poniżej oczekiwań. Następuje rozczarowanie wynikające z braku spodziewanych efektów w porównaniu z rozbudzonymi oczekiwaniami. Czy tak musi być?

Stosowanie CASE wymaga odpowiednio uzbrojonego stanowiska pracy programisty. Dotychczasowe atrybuty pracy programisty w postaci inwencji i proceduralnego języka programowania, wspomagane długopisem i kartką papieru, ewentualnie szablonem wspomagającym kreślenie schematów to zbyt mało aby osiągać spodziewane efekty. Przedstawiony wizerunek stanowiska nie jest zbyt przejaskrawiony ale oddaje praktykę przeciętnego stanowiska pracy programisty. Dodatkowe, podnoszące jakość tego stanowiska elementy, są wynikiem inwencji programisty, jego doświadczeń fachowych i zaradności. Użytkowane narzędzia informatyczne są często własnej produkcji lub zostały zdobyte w sposób przypadkowy nie zawsze w sposób całkiem legalny.

Nowoczesne stanowisko pracy programisty to stacja robocza o stosunkowo dużych zasobach, często podłączona do sieci komputerowej, która gwarantuje legalny dostęp do dużych baz danych, szybkich procesorów i umożliwiająca łatwe komunikowanie się z innymi członkami zespołu pracującego nad tym samym projektem. Na ograniczenia sprzętowe nakładają sie ograniczenia programowe. które sprowadzają sie do konieczności wyposażenia stacji roboczej w odpowiednio sprawne narzędzia wytwarzania i komunikacji. Czas mozol-

nego kreślenia schematów i pisania kodu źródłowego programu to historia informatyki. Praca indywidualna musi zostać ograniczona ramami pracy w zespole realizującym projekt według założonej strategii postępowania.

Wysokie koszty stanowiska pracy programisty to tylko część trudności jakie związane są z wytworzeniem odpowiednich warunków pracy. Dobór oprogramowania wspomagającego i narzędziowego, jego spójność, to elementarne wymogi stawiane przed oprogramowaniem w jakie powinno być wyposażone stanowisko pracy. Ograniczeniem z którym w tym przypadku mamy do czynienia jest oprócz bardzo wysokich kosztów, embargo związane z licencjonowaniem nowoczesnej technologii i stosunkowo trudny dostęp do informacji o dostępnych na rynku narzędziach programowych. Często dobór jest przypadkowy, nie poprzedzony pełną analizą wyboru, zgodnie z potrzebami. Istotne dla dokonywanego wyboru narzędzi są również ograniczenia finansowe.

Zastosowanie nowoczesnego narzędzia w nieprzygotowanym do tego środowisku nie może przynieść spodziewanych wyników. Na środowisko składają się nie tylko powszechnie znane trudności telekomunikacyjne ale również brak odpowiedniego środowiska standardów w różnych dziedzinach życia. Nowa technologia wymaga nowej wiedzy, której brak jest szczególnie dotkliwy. Wiedza teoretyczna, a conajwyżej taką można zdobyć na naszych Uczelniach, okazuje się niewystarczająca w zderzeniu z praktyką.

Nieliczne doświadczenia krajowe stosowania CASE wskazują na potencjalne możliwości i liczne ograniczenia, które powodują, że osiągnięte wyniki są nieadekwatne do oczekiwań (por. (90)). Przykładem, z którym pośrednio styka się każdy podatnik, jest historia opracowania systemu POLTAX, gdzie nakłady na nowoczesną technologię nie przyniosły efektu w spodziewanym czasie, co ma nie tylko wymiar finansowy ale tworzy niedobrą atmosferę wokół stosowania informatyki w odbiorze powszechnym. Brak pełnej analizy tego przypadku nie pozwala na profesjonalną ocenę, ale można się spodziewać, że większość przyczyn niepowodzeń leży w otoczeniu, czyli braku odpowiedniego środowiska stosowania komputerowego wspomagania.

Systemy otwarte (Open Systems), to klasa sprzętu i oprogramowania narzędziowego, które mają gwarantować uniezależnienie się od jednego dostawcy. Systemy otwarte pozwalają na swobodny dobór różnorodnych elementów sprzętu i oprogramowania zgodnie z potrzebami użytkownika tak aby dokonywanie zmian i modyfikacji systemów było proste i szybkie w realizacji. Przed oprogramowaniem wspomagającym stawiane są wymagania odnośnie wspomagania:

— prowadzenia eksploatacji na różnych platformach,

— prowadzenia rozwoju i eksploatacji na różnych platformach,
— integracji narzędzi dla róznych źródeł pochodzenia.

Systemy otwarte tworzą środowisko, w którym tworzenie systemów informatycznych wyzwolone jest z licznych ograniczeń sprzętowych i programowych ale nie rozwiązują wszystkich problemów, które legły u podstaw niezadowalających wyników stosowania CASE. Tendencja zmierzająca do uniwersalności rozwiązań to kierunek, który powinien zagwarantować tworzenie odpowiedniego środowiska, gdzie stosowanie technologii CASE przyniesie spodziewane wyniki.

Na środowisko, w którym stosowanie technologii CASE przyniesie spodziewane wyniki, składają się cztery fundamentalne elementy:

— **konsultacje z zakresu wiedzy o metodologii wytwarzania systemu informatycznego,**
— **standardy rozwiązań w eksploatacji i rozwoju systemu,**
— **środki integracji sposób prezentacji, przechowywania i kontroli danych,**
— **kompletna rodzina narzędzi wspomagania prac wytwarzania systemu.**

Spójne środowisko powinno gwarantować, że stosowanie CASE przyniesie spodziewane wyniki i nie spowoduje:

— braku integracji pomiędzy używanymi narzędziami wytwarzania,
— braku możliwości wymiany danych pomiędzy stosowanymi narzędziami różnych dostawców,
— stosowane narzędzia nie pokrywają całego cyklu wytwarzania systemu,
— stosowane narzędzia gwarantują otwartość tylko na ograniczoną grupę dostawców,
— narzędzia oferują prawnie zastrzeżone rozwiązania,
— narzędzia nie gwarantują automatycznej kontroli realizowanego procesu,
— narzędzia są adoptowane przez zespoły nieautorskie dla określonych metod i procedur wytwarzania.

Wymienione wcześniej cztery fundamentalne elementy środowiska mają wspólnie zagwarantować, że wymienione zagrożenia nie wystąpią i tworzenie systemu będzie odbywało się szybko i sprawnie a wytworzony produkt osiągnie oczekiwaną jakość.

Konsultacje

Przystępując do wytwarzania systemu informatycznego według technologii CASE, należy zagwarantować sobie:

— odpowiedni cykl szkoleń,
— serwis gwarancyjny i usługi serwisowe na stosowane narzędzia,
— stały nadzór konsultanta w całym cyklu wytwarzania.

Nadzór konsultanta ma mieć charakter techniczny i ekonomiczny. Konsultant powinien przygotowywać decyzje odnośnie odpowiedniego cyklu szkoleń zespołu wykonawców, zakupu koniecznych narzędzi programowych, tak aby realizacja projektu odbywała się sprawnie przy najmniejszych niezbędnych kosztach. Do zadań konsultanta powinno należeć stałe nadzorowanie realizacji projektu zgodnie z przyjętą metodologią wytwarzania. Nie może on decydować o przyjmowanych rozwiązaniach ale wskazywać zagrożenia jednych decyzji a zalety innych rozwiązań.

Przez odpowiedni cykl szkoleń należy rozumieć szkolenia, które nie tylko zagwarantują biegłość w stosowaniu narzędzi ale również zagwarantują wiedzę ogólną pozwalającą na prawidłową strategię rozwoju i stosowania nowoczesnej technologii wytwarzania. Szkolenia te powinny dotyczyć osób zgodnie z ich zakresem odpowiedzialności w przyszłej realizacji projektu.

Istotnym problemem jest zapewnienie zespołowi wykonawczemu wspomagania merytorycznego w całym cyklu wytwarzania systemu. W trakcie realizacji projektu często występują problemy cząstkowe, szczegółowe, które mogą wpływać na tempo realizacji. Zapewnienie zespołowi wykonawców stałego doradztwa w przypadku napotkania trudności w stosowaniu narzędzi wyraźnie przyspiesza realizację i pozwala zespołowi koncentrować uwagę na istocie rozwiązywanego problemu a nie skupianie się na poznawaniu tajników stosowanych narzędzi. Praktyka pokazuje, że dobre opanowanie narzędzi wymaga bardzo wiele czasu i nie gwarantuje to, że w trakcie realizacji nie powstaną sytuacje, które nie mogą być rozwiązane własnymi silami. Wykupienie abonamentu typu „hot-line" lub inne tego typu rozwiązanie jest najlepsze z punktu widzenia ekonomicznego i sprawności realizacji projektu. Oszczędności polegające na nie wykupieniu usług „hot-line" często są pozorne i mogą doprowadzić do poważnych zahamowań w procesie realizacyjnym gdy napotkamy na trudny do usunięcia błąd lub brakuje nam wiedzy z szczegółowych zastosowad narzędzi wykorzystywanych w pracy.

Poprawna realizacja projektu wymaga przyjęcia określonej metodyki postępowania i konsekwentnego stosowania jej w całym cyklu realizacyjnym. Próby odstępstwa od przyjętej metodyki lub uproszczenia w dowolnej fazie realizacji projektu powodują niespójność i trudności realizacyjne w dalszych fazach wykonania. Konsekwencja strategi realizacji może być osiągnięta wówczas gdy istnieje niezależny, niezaangażowany w proces wykonawczy,

czynnik kontrolny, który nie dopuści do uproszczeń i odstępstw od metodyki postępowania. Zewnętrzny konsultant, mający okresowy kontakt z zespołem może spełnić taką rolę.

Formalizm takiego postępowania może wydawać się dziwny i często napotyka na brak zrozumienia w zespole wykonawczym. Ważna jest jednak konsekwencja w postępowaniu okupiona nawet ograniczeniem inwencji wykonwców. Chłodne i realizowane co jakiś czas nadzorowanie prac nad projektem przez czynnik zewnętrzny dobrze wpływa na proces realizacji projektu. Decyzje nie są podejmowane emocjonalnie i subiektywnie ale jest szansa na spokojne i przemyślane postępowanie. Dystans do rozwiązań szczegółowych pozwala na strategiczne spojrzenie na proces realizacyjny z punktu widzenia całego cyklu realizacyjnego a nie konkretnego rozwiązania szczegółowego.

Standardy

Systemy otwarte, bardzo ostatnio popularne, zakładają standaryzację rozwiązań i wynikające z tego uniezależnienie się od jednego dostawcy. Większość producentów sprzętu komputerowego i oprogramowania oferuje otwartość swoich rozwiązań, co często nie ma pokrycia w rzeczywistości. Brak otwartości rozwiązań prowadzi najczęściej do uzależnienia od jednego dostawcy a z tego wynika istotne ograniczenie możliwości wyboru stosowanych w cyklu realizacyjnym narzędzi i metod.

Systemy otwarte są domeną dużych firm komputerowych, które są szczególnie zainteresowane tą koncepcją. Prowadzenie prac nad systemami otwartymi wymaga dużych nakładów, na co stać tylko duże firmy oraz odpowiednio rozbudowanej oferty sprzętowej i programowej. Firmy, nie chcąc utracić swoich klientów, proponują odmienne nieco podejścia do otwartości systemu. Różnice te polegają na tym, że promowana jest głównie własna oferta sprzętowa i programowa, a produkty innych firm stanowią tylko uzupełnienie oferty lub gwarantowane są przejścia między różnymi środowiskami. Firma Hewlett Packard wychodząc naprzeciw zapotrzebowaniu swoich klientów, rozpoczęła prace i dostarcza elementy systemów otwartych, tak aby klienci mogli płynnie wprowadzać nowe technologie informatyczne nie rezygnując z dotychczasowych osiągnięć i przyzwyczajeń. W opracowaniu (83) firma ta definiuje podstawowe elementy platformy systemów otwartych, wymieniając w klasie:

— systemów sieciowych — OSI, ARPA, SNA Novell itp.;
— baz danych — ORACLE, INGRES, INFORMIX itp.;
— interfejsu użytkownika — X-Windows, OSF/MOTIF;
— interfejsu systemów operacyjnych — POSIX.

Z wykazu produktów widać, że nie są to oryginalne produkty firmy HP, a mające największą popularność wśród użytkowników, produkty wielu różnych firm softwarowych.

Firma Digital Equipment Corporation proponuje własną strategię integracji zastosowań pracujących w środowiskach sprzętowo-programowych różnych firm komputerowych. Strategia ta wspomagająca koncepcję systemów otwartych to NAS(Network Application Support) (101). NAS jest zbiorem produktów i narzędzi programowych wspomagających proces tworzenia systemu informatycznego. Założeniem koncepcji NAS jest integracja różnorodnych elementów. NAS nie jest technologią zamkniętą, statyczną ale uwzględnia dynamikę rozwoju sprzętu i oprogramowania, akceptując różne rodzaje sprzętu komputerowego i różne systemy operacyjne z komunikacją w niejednorodnym środowisku. Umożliwia to rozbudowę aplikacji o elementy sprzętowe i programowe firmy Digital ale również innych producentów, nie zamykając możliwości rozbudowy tylko do produktów jednej firmy.

Przedstawione rozwiązania firm Hewlett Packard i Digital Equipment Corporation, propagują systemy otwarte jako rozwiązanie uniwersalne, akceptujące różnorodne środowiska sprzętowe i programowe, nie ograniczając się do jednego producenta. Pozwala to na swobodę wyboru twórców systemów informatycznych w dobieraniu sprzętu i oprogramowania dla realizacji własnej koncepcji rozwiązania. Nieco inne założenie stanowi podstawę podobnych działań firmy IBM.

IBM, jako niekwestionowany lider światowy w produkcji sprzętu i oprogramowania, działania integracyjne ogranicza do własnych produktów, które wypełniają całą paletę możliwości sprzętowych i programowych. Założenie otwartości systemu i przenośności oprogramowania pozostaje celem tych działań, podobnie jak w przedstawionych wcześniej koncepcjach, ale oferta narzędzi wspomagających nie wychodzi poza środowisko produktów IBM.

Korporacja IBM posiadająca w swej ofercie różnorodne, w sensie mocy obliczeniowej i rozwiązań logicznych, propozycje sprzętowe i oprogramowania operacyjnego, była zmuszona do uporządkowania produktów oferowanych klientom. Stałą dewizą IBM jest przenoszalność systemów użytkowych, z modeli niższych na modele wyższe. W celu zagwarantowania tego wymogu oraz dla podniesienia konkurencyjności swojej oferty IBM zaproponował standardową Architekturę Systemów Aplikacyjnych SAA (Systems Application Architecture) (121). Jest to koncepcja standardowej platformy konstrukcyjnej oraz zestaw standardowych narzędzi wytwarzania systemu informatycznego, której celem jest:

— łatwość przenoszenia aplikacji w różne środowiska,
— spinanie różnych dziedzin działalności przez aplikacje,
— dostęp użytkownika do aplikacji ma być prosty i jednolity.

Biorąc pod uwagę wysokie koszty standaryzacji i przedsięwzięć zmierzających do powstawania systemów otwartych, firmy komputerowe zjednoczyły się w Open Software Foundation (OSF), której celem jest opracowywanie i wdrażanie standardów. Przykładem tych działań może być opracowanie i próby wdrożenia standardu dla systemu operacyjnego. Jako wersję bazową przyjęto koncepcje systemu operacyjnego UNIX wraz z jego licznymi implementacjami. POSIX (Portable Operating System Interface for Computer Environments) jest koncepcją łączącą w sobie zbiór standardów, projektowanych dla zapewnienia przenośności aplikacji.

Stosowanie standardów umożliwia faktyczne wykorzystanie narzędzi będących autorstwa różnych dostawców. Standaryzacja stosowanych narzędzi daje szansę na budowę produktu, który też będzie miał cechy produktu standardowego czyli będzie spełniał cechy jakościowe przypisywane systemom otwartym. Uniezależnienie się od jednego dostawcy pozwala nie tylko na dokonanie optymalnego wyboru narzędzi w sensie zgodności z potrzebami procesu wytwarzania ale również z punktu widzenia kosztów.

Integracja

Procesy integracyjne w realizacji projektu są realizowane na wielu płaszczyznach. Integracja realizowana jest wielowymiarowo w zakresie integracji:
— prezentacji,
— sterowania,
— danych.

Integracja prezentacji polega na wyborze standardowego interfejsu komunikacyjnego, który pozwala programiście i użytkownikowi na korzystanie z różnorodnych narzędzi bez konieczności zmian przyzwyczajeń i nawyków. Podobny obraz ekranu, takie samo wykorzystanie klawiszy funkcyjnych, kolorów jednoznaczna sygnalizacja, to zalety integracji prezentacji. Przykładem takiego standardu integracyjnego może być standard komunikacyjny Windows, zdobywający popularność w licznych nowych aplikacjach.

Jednorodne środowisko prezentacji nie ogranicza się do obrazu ekranu monitora ale dotyczy również zarządzania ekranem i zbiorami. Programiści i użytkownicy nie muszą uczyć się nowych zasad komunikacji w sytuacji gdy wykorzystywane są różnorodne narzędzia wspomagania prac. Stosowanie tych

samych mechanizmów zarządzania, znacznie upraszcza proces przyzwyczajania się użytkownika do nowej aplikacji.

W procesie realizacyjnym programista wykorzystuje różnorodne narzędzia pracy w zależności od tego w jakiej fazie procesu znajduje się. Narzędzia te, jak wspomnieliśmy wcześniej mogą być od różnych dostawców, powinny pozwalać na swobodne przekazywanie wyników jednej czynności, do dalszej obróbki przez inne narzędzia. Możliwość skoncentrowania się na rozwiązywanym problemie nie może by zakłócona sprawami zgodności wyników pośrednich powstających w procesie realizacji. Integracja sterowania przepływem wyników pośrednich pomiędzy narzędziami wykorzystywanymi przez realizatorów jest wymogiem gwarantującym sprawną realizację projektu.

Integracja prezentacji i sterowania musi być uzupełniona integracją danych powstających w trakcie prac wykonawczych. Integracja sterowania gwarantuje dynamiczne możliwości łączenia i przekazywania wyników pośrednich. Integracja danych musi gwarantować spójność w dłuższej perspektywie czasowej. Kolejne czynności realizacyjne mogą być wykonywane według odmiennej technologi ale będą operowały na tych samych danych. Integracja ta musi być zagwarantowana nie tylko na poziomie narzędzi CASE ale również przy tradycyjnych metodach realizacji.

Dane łączą kolejne fazy realizacji w całym cyklu życia systemu, stąd spójność ich jest szczególnie istotna. Gwarancją tej integracji jest składnica danych (repository), do której odwołują się wczystkie narzędzia i wszyscy wykonawcy w całym procesie realizacyjnym. Spójność danych, ich integracja polega na jednorodności struktur, sposobie zapamiętania i przechowywania oraz dostępu do danych.

Narzędzia wspomagania

Typowy proces realizacji systemu informatycznego dzielony jest na kilka etapów realizacji. Etapy te składają się na cykl wytwarzania systemu i każdy z nich posiada własną grupę narzędzi wspomagania. Najczęściej stosowany podział na etapy uwzględnia fazy:

— definicję wymagań,
— analizę problemu,
— specyfikację problemu,
— projektowanie systemu,
— programowanie systemu (implementacja),
— testowanie systemu,

— pielęgnacja (maintenance).

Dla sprawnej realizacji systemu potrzebne są narzędzia wspomagające konkretne czynności, konieczne do wykonania w kolejnych fazach realizacji oraz druga grupa narzędzi, które pozwalają na kontrolę procesu realizacji.

Można wyróżnić kilka kategorii narzędzi wspomagania wykorzystywanych w procesie realizacji, takich jak:

— śledzenie procesu realizacji,
— planowanie prac wykonawczych,
— zarządzanie projektem,
— wymiarowanie prac,
— dokumentowanie prac.

Srodowisko pracy zespołu realizacji systemu tworzą narzędzia wchodzące w skład różnych wymienionych kategorii. Dobór odpowiedniego zestawu narzędzi zależy od przyzwyczajeń członków zespołu realizowanego zadania, możliwości finansowych i specyfiki realizowanego projektu. Dla sprawności realizacji ważne jest dopasowanie narzędzi do przyzwyczajeń członków zespołu oraz pokrycie narzędziami wspomagania całego cyklu życia systemu a niezbędne jest wspomaganie najbardziej pracochłonnych prac.

Wybór, odpowiedniego dla konkretnego przypadku, narzędzia jest niezwykle istotny dla powodzenia całego przedsięwzięcia. Każde silne narzędzie wspomagające jest skuteczne wtedy, gdy je umiejętnie stosujemy. Nie będziemy stosować kafara do wbijania gwoździ a zwykłego młotka przy budowie mostu. Ale tak jest tylko wtedy, gdy rozumiemy metody i procedury, których stosowanie umożliwia właściwy sposób wykorzystania narzędzi. Stosowanie bez zrozumienia może być zarówno bezproduktywne, jak i niebezpieczne. Wielu programistów próbuje stosować skomplikowane narzędzia CASE z pobieżną tylko znajomością inżynierii oprogramowania, co prowadzi do rozczarowań z otrzymanych efektów.

Krytyczna analiza praktyki wprowadzania nowoczesnych metod wytwarzania systemów informatycznych pozwala na próbę zdefiniowania modelu wdrażania nowej technologii wytwarzania. Nie może to być jednorazowy krótki proces ale cykl systematycznych działań prowadzonych przez kierownictwo i członków zespołu.

Proponowany model postępowania przedstawiony zostanie w postaci kolejnych kroków składających się na realizowane systematycznie etapy działań.

— *Ocena stanu obecnego.* Pierwszym krokiem cyklu powinna być krytyczna ocena stosowanych metod i procedur postępowania. Ocena powinna dotyczyć zarówno danych jakościowych, jak i ilościowych. Ocena taka ma

na celu nie tylko zbilansowanie wad i zalet dotychczas stosowanych metod i procedur ale daje materiał dla podejmowania decyzji dotyczących cyklu szkoleń, rozwiązań organizacyjnych oraz pozwala na szacunek efektów uzyskiwanych w trakcie kolejnych faz wdrażania nowej technologii.

— *Szkolenie.* Prawidłowo zorganizowany cykl szkoleń nie może być traktowany jako premia udzielana pracownikom w okresie gdy nie ma pilnych do realizacji zadań, ale powinien to być przemyślany cykl dobrze dobranych kursów i staży w zależności od przewidywanych funkcji w ramach przyszłych działań. Szkolenia dają pracownikom sygnał, że kierownictwo uważa nowoczesne metody wytwarzania za rzecz ważną i gotowe jest poświęcić odpowiednie środki na jej wdrożenie. Daje wszystkim spójne źródło informacji, które mogą być w dalszej pracy efektywnie wykorzystane.

— *Wybór narzędzi i procedur postępowania.* Proces wyboru jest bardzo kłopotliwy z uwagi na ogromny zbiór metod, procedur i narzędzi. Różnice pomiędzy metodami, narzędziami są trudno uchwytne, co dodatkowo utrudnia wybór. Realizacja poprzednich kroków może być pomocna przy wyborze ale istotne są kryteria zastosowane na tym etapie wdrożenia. Ważne są kryteria techniczne wyboru ale jeszcze ważniejsze są cele ogólne kierownictwa, a do tych należą jakość i wydajność.

— *Instalacja narzędzi.* Prawidłowo zrealizowane poprzednie kroki postępowania powinny ustrzec przed rozczarowaniem i nieporozumieniem wynikającym z faktu, że próbujemy stosować narzędzia do nieodpowiednich problemów lub przez nieodpowiednich ludzi. Instalacja nie może być jednorazowym aktem ale powinna być realizowana sukcesywnie, w miarę postępu prac, należy instalować kolejne narzędzia. Unikniemy sytuacji kiedy zakupione narzędzie, nie jest stosowane i odstawione na półkę. Proces instalacji musi być odpowiednio zorganizowany i przygotowany, tak aby proces wdrażania przebiegał sprawnie.

— *Podsumowanie i kontrola wdrożenia.* W trakcie procesu wdrażania należy pamiętać, że jest to proces długofalowy i powinien stale podlegać analizie i ocenie. Pierwszy krok procesu daje materiał porównawczy, czy realizowany proces przynosi spodziewane efekty i pozwala na bieżące korygowanie procesu wdrażania nowoczesnej technologii.

Prawidłowa realizacja procesu wdrożenia nowoczesnej technologii wytwarzania systemu informatycznego jest niezwykle trudna do przeprowadzenia sprawnie i bezkolizyjnie. Pomocne w tym może być zatrudnienie do pomocy w realizacji tego procesu doświadczonego konsultanta zewnętrznego,

który okresowo mógłby bez emocji i obiektywnie oceniać postęp prac i doradzać kolejne posunięcia.

1. Tworzenie systemów informatycznych zarządzania

1.1. Struktura systemu informatycznego

Przez system informatyczny zarządzania rozumie się system automatycznego przetwarzania danych wspomagający proces zarządzania, przy czym:

— „automatyczne" oznacza przetwarzanie danych za pomocą komputera;

— „proces zarządzania" jest wieloetapowym sekwencyjnym procesem podejmowania decyzji, na który składają się: opracowanie i akceptacja programu działania, ustalenia wynikające z funkcji organizowania i decyzje związane z koordynowaniem działalności wyodrębnionych podsystemów oraz funkcji kontroli i nadzoru, umożliwiające działanie sprzężeń zwrotnych w systemie (49, s. 125).

Komputerowe wspomaganie procesu zarządzania jest w Polsce określane jako system informatyczny zarządzania (por. (64, 73)). W terminologii angielskiej używa się nazwy MIS (Management Information System), a w rosyjskiej — ASU (Awtomatizirowannyje Sistiemy Uprawlenija). Jest to pewna klasa systemów informatycznych mających na celu głównie przetwarzanie danych. Ta klasa systemów będzie podstawowym obiektem dalszych zainteresowań, co w niczym nie ogranicza ogólności rozważań, a jedynie pozwala na lepsze ukierunkowanie przykładów. Z definicji tej klasy systemów wynika ich różnorodność. W literaturze można znaleźć różnorodne klasyfikacje i typologie (38), które dzielą systemy w zależności od wspomaganego szczebla zarządzania lub złożoności systemu.

W zależności od kryterium klasyfikacji mówi się o systemach obiektowych, krajowych, resortowych, regionalnych, a według innego kryterium są to systemy cząstkowe (wycinkowe), dziedzinowe, kompleksowe. Niezależnie od sposobu klasyfikowania, pewne cechy systemów informatycznych zarządzania są wspólne, wynikają bowiem ze struktury systemu informatycznego — tymi cechami należy się zająć bliżej. Należy pamiętać, że w zależności od konkretnego przypadku różnie mogą być rozłożone akcenty na poszczególne elementy.

Według innej definicji system informatyczny zarządzania jest zbiorem elementów i relacji występujących między nimi (72). W praktyce liczba elementów i relacji jest tak duża, że rozpatrywanie ich w jednym „płaskim" układzie nie tylko nie daje zadowalającej interpretacji, ale tworzy często wypaczony, sprzeczny z rzeczywistością obraz. Słuszniej i wygodniej jest traktować system informatyczny nie jako zjawisko „płaskie", ale jako „przestrzenne", i to w układzie wielu współrzędnych. Należy wówczas rozpatrywać wiele struktur systemu w wielu różnych odrębnych (chociaż wzajemnie powiązanych) przekrojach z różnych punktów widzenia, w zależności od różnych, odrębnych kryteriów analizy i dekompozycji. Stosując tego typu podejście, można w zależności od potrzeby wyodrębnić wiele możliwych i celowych przekrojów.

Duża różnorodność poszczególnych składników systemu informatycznego skłania do rozpatrywania systemu jako struktury zbudowanej z podstruktur, z których każda będzie się charakteryzowała jednorodnością wchodzących w jej skład elementów. Istnieje potrzeba wyodrębnienia takich podstruktur według zdefiniowanych kryteriów:

— cele, funkcje i zadania systemu;
— informacje i algorytmy ich przekształcania;
— środki techniczne przetwarzania danych;
— przestrzenne rozmieszczenie wymienionych składników.

Uwzględniając powyższe kryteria, w informatycznym systemie zarządzania można wyodrębnić następujące substruktury:

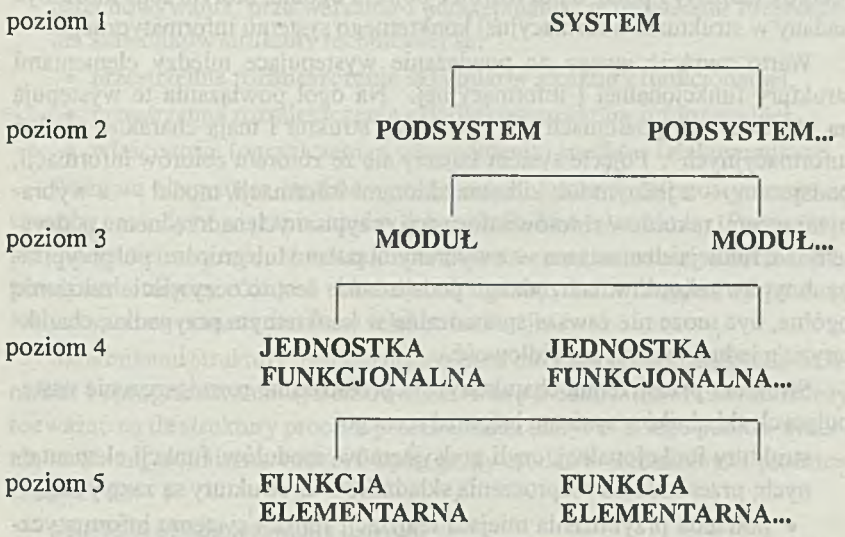
— funkcjonalną;
— informacyjną;
— przestrzenną;
— techniczną[1]

Powyższy podział struktury systemu informatycznego zarządzania ma nie tylko charakter poznawczy, lecz również istotne znaczenie dla tworzenia sys-

[1] Szczegółowy opis poszczególnych struktur i konsekwentną realizację według nich całego procesu tworzenia systemu można znaleźć w pracy (123).

temu, pozwala bowiem uporządkować i sprowadzić proces tworzenia do problemu modelowania poszczególnych substruktur.

Struktura funkcjonalna jest zbiorem celów i funkcji systemu oraz wzajemnych ich współzależności. Wiąże ona system informatyczny z systemem zarządzania w układzie pionowym — podrzędnie w stosunku do tego ostatniego. Praktycznie struktura funkcjonalna jest zbiorem elementarnych zadań wykonywanych przez system informatyczny na rzecz systemu zarządzania. Struktura ta ma postać dendrytu, którego węzły oznaczają składniki, a krawędzie — relacje występujące między nimi. Liczba szczebli w tej strukturze zależy od stopnia złożoności systemu; może się zmieniać w różnych systemach. W praktyce najczęściej na poszczególnych szczeblach hierarchii wyodrębnia się nastepujące, uogólnione składniki struktury funkcjonalnej systemu informatycznego:



W tej uogólnionej strukturze funkcjonalnej systemu wprowadzono takie pojęcia, jak: podsystem, moduł, jednostka funkcjonalna, funkcja elementarna. Przyporządkowując powyższym składnikom konkretne zadania i funkcje, określając powiązania miedzy nimi, nadając im odpowiednie nazwy, można się podjąć zdefiniowania poszczególnych składników struktury funkcjonalnej systemu oraz sklasyfikowania relacji występujących pomiędzy składnikami.

Waga struktury informacyjnej systemu polega na tym, że w systemach informatycznych informacje są podstawowym tworzywem, przy czym muszą one

być odpowiednio ustrukturalizowane i uporządkowane, stosownie do funkcji i zadań systemu, charakteru podejmowanych decyzji, struktury procesów technologicznych itd. Inaczej mówiąc, muszą one być ułożone w odpowiednią strukturę, której podstawowymi składnikami są jednostki informacji i występujące między nimi relacje oraz procedury przekształcania informacji[2].Można powiedzieć, że na strukturę informacyjną systemu informatycznego składają się:

— zbiory informacji wejściowych;
— zbiory informacji wyjściowych;
— zbiory informacji stałych (przechowywanych);
— algorytmy przekształcania informacji.

Jest rzeczą oczywistą, że każdy z wymienionych zbiorów danych ma swoją wewnętrzną strukturę, w której wyróżnia się takie jednostki informacji, jak rekordy (zapisy) i pola. Są to również pojęcia ogólne, którym sens może być nadany w strukturze informacyjnej konkretnego systemu informatycznego.

Warto zwrócić uwagę na powiązanie występujące między elementami struktury funkcjonalnej i informacyjnej. Na ogół powiązania te występują na identycznych poziomach hierarchii tych struktur i mają charakter zasileń informacyjnych[3]. Pojęcie system kojarzy się ze zbiorem zbiorów informacji, podsystem — z jednym lub kilkoma zbiorami informacji, moduł — z wybranymi typami rekordów zbiorów informacji przypisanych nadrzędnemu podsystemowi, funkcja elementarna — z wybranymi polami lub grupami pól przypisanych typów rekordów nadrzędnego podsystemu. Jest to oczywiście założenie ogólne, być może nie zawsze sprawdzalne w konkretnym przypadku, charakteryzuje jednak pewną prawidłowość.

Struktura przestrzenna charakteryzuje przestrzenne rozmieszczenie następujących składników systemu informatycznego:

— struktury funkcjonalnej, czyli podsystemów, modułów, funkcji elementarnych; przesłankami rozproszenia składników tej struktury są zazwyczaj:
  - potrzeba przybliżenia miejsca realizacji funkcji systemu informatycznego do miejsca realizacji funkcji systemu zarządzania,
  - rozmieszczenie składników struktury informacyjnej,
  - wymagania zastosowanej technologii przetwarzania;

[2] Szczegółowe informacje o stosowanych strukturach i modelach danych w systemach informatycznych przedstawiono, między innymi, w pracach (140, 138).

[3] Cecha ta znalazła zastosowanie w metodach projektowania systemów informatycznych Jacksona i Warniera. Opis tych metod zawiera praca (129).

— struktury informacyjnej, czyli zbiorów informacji wejściowych, zbiorów informacji wyjściowych, zbiorów informacji stałych oraz procedur przetwarzania, czyli oprogramowania użytkownika; przesłankami rozproszenia składników struktury informacyjnej są zazwyczaj:
  - potrzeba lokalizacji zbiorów informacji wejściowych w pobliżu źródeł powstawania informacji,
  - potrzeba lokalizacji zbiorów informacji wyjściowych w miejscu ich użytkowania, a więc realizacji funkcji systemu zarządzania,
  - potrzeba lokalizacji zbiorów informacji stałych w pobliżu powstawania zbiorów informacji wejściowych aktualizujących oraz w pobliżu wspomaganych przez nie składników struktury funkcjonalnej (podsystemów);
— struktury technicznej, czyli środków technicznych do zbierania danych, ich przechowywania, przetwarzania i udostępniania; przesłankami rozproszenia składników struktury technicznej są:
  - przestrzenne rozmieszczenie składników struktury funkcjonalnej,
  - przestrzenne rozmieszczenie składników struktury informacyjnej,
  - właściwości (ograniczenia i udogodnienia) środków telekomunikacji.

Ważnym elementem struktury przestrzennej systemu informatycznego są sposoby i zasady powiązania między sobą składników tej struktury. Przeważnie przyjmuje się założenie, że o strukturze przestrzennej mówi się wówczas, gdy powiązania między różnymi składnikami struktury informacyjnej systemu są realizowane za pomocą środków telekomunikacji.

Składnikami struktury technicznej systemu informatycznego są środki techniczne i oprogramowanie systemowe. Strukturę środków technicznych należy rozważać na tle struktury procesu przetwarzania danych. Z tego punktu widzenia strukturę techniczną tworzyć będą grupy środków technicznych przeznaczonych do:
— zbierania i kompletowania danych;
— zapisu danych na maszynowych nośnikach danych;
— przechowywania zbiorów danych;
— przekształcania danych;
— udostępniania danych;
— przesyłania danych.

Oprogramowanie systemowe to przede wszystkim systemy operacyjne i systemy zarządzania danymi.

1.2. Metody i techniki tworzenia systemów informatycznych

1.2.1. Metody tworzenia systemów informatycznych

Tworzenie systemu informatycznego jest procesem twórczym, który może być wspomagany różnorodnymi technikami. Istnieje prawidłowość, która polega na tym, że im obiekt procesu tworzenia jest definiowany za pomocą bardziej ogólnych pojęć, tym więcej różnych metod i technik można wykorzystać do wspomagania procesu.

W celu jednoznacznego określenia pojęć **metoda** i **technika** przyjmuje się za T. Kotarbińskim (75), że **metoda** to sposób systematycznie stosowany, a za O. Lange (77), że **technika** to zespół środków oraz sposób ich stosowania. Mówiąc inaczej,

— **technika** tworzenia systemu jest to sposób wykorzystania narzędzi (aparat pojęciowy, oprogramowanie itp.) do rozwiązania problemów projektowych;
— **metoda** tworzenia systemu jest to sposób posługiwania się i wykorzystania technik w rozwiązywaniu problemów projektowych.

Materiałem wyjściowym do projektowania są zdefiniowane cele i funkcje systemu informatycznego zarządzania oraz analiza potrzeb informacyjnych. Dysponując takim materiałem wyjściowym, można powiedzieć, że projektowanie struktury funkcjonalnej sprowadza się do odpowiedzi na pytanie: jak połączyć poszczególne funkcje w większe składniki, do podsystemów włącznie? Poszukując odpowiedzi na to pytanie, projektant wchodzi w skomplikowane procesy wyszukiwania, oceny i selekcji realizowane w wielowarstwowych i pojemnych obszarach problemowych. Posłużyć się tu można dwoma metodami, a raczej sposobami podejścia:

— przez dekompozycje celów systemu na składniki (podsystemy, moduły, funkcje elementarne);
— przez syntezę funkcji elementarnych w składniki wyższego rzędu (moduły, podsystemy).

Metoda dekompozycji celów będzie miała zastosowanie w rozwiązywaniu problemów słabo ustrukturalizowanych, gdy projektant nie dysponuje wymaganiami i ograniczeniami ze strony użytkownika, które określają wartości progowe, szczegółowe i kierunkowe co do formy struktury funkcjonalnej. War-

tości progowe to granice określające akceptowane i nieakceptowane formy. Wartości szczegółowe określają wąską klasę rezultatów, do których należą formy akceptowane. Wartości kierunkowe określają maksymalne lub minimalne charakterystyki akceptowanych form.

Metoda syntezy funkcji elementarnych może mieć zastosowanie w rozwiązywaniu problemów o różnym stopniu wstępnego ustrukturalizowania pod warunkiem, że w etapach poprzedzających projektowanie struktury funkcjonalnej zostaną szczegółowo zdefiniowane funkcje elementarne. Ogólnie rzecz ujmując, polega ona na łączeniu funkcji elementarnych w składniki wyższego rzędu za pomocą określonego kryterium syntezy.

W procesach dekompozycji celów lub syntezy funkcji elementarnych można się posłużyć dwoma rodzajami kryteriów:

— dekompozycją lub syntezą opartą na uwzględnieniu cech;
— dekompozycją lub syntezą opartą na uwzględnieniu współzależności.

Dekompozycja lub synteza oparta na uwzględnieniu cech zakłada, że najważniejszym kryterium jest podobieństwo. Projektowanie polega wówczas na dzieleniu celów systemu lub łączeniu funkcji elementarnych w składniki według reguły: dziel lub łącz te elementy, które odnoszą się do podobnych obiektów lub czynności (zasoby, procesy).

Dekompozycja lub synteza oparta na uwzględnieniu współzależności (relacji) zakłada występowanie relacji między składnikami. Projektowanie polega wówczas na dekompozycji celów lub syntezie funkcji elementarnych według kryterium intensywności występowania relacji. Regułę postępowania można sformułować następująco: łącz lub dziel tak, aby związki między wydzielonymi podsystemami były możliwie małe, a wewnątrz podsystemów możliwie liczne. Posługując się jedną z tych metod, uwzględnia się również pozostałe zdefiniowane kryteria[4].

Intensywne prace nad metodami tworzenia systemów informatycznych prowadzi wiele ośrodków badawczych na całym świecie (16). Większość obecnie opracowywanych metod obejmuje cztery podstawowe składniki:

— sposób postępowania w procesie budowy i rozwoju systemu, który zawiera strukturę procesu (podział na etapy), opis czynności na każdym etapie (podetapie) ze wskazaniem sposobu ich wykonania i punkty kontrolne związane z decyzjami zleceniodawcy (użytkownika);
— model systemu informacyjnego, który dostarcza zestawu pojęć do formalnego wyrażania struktury i sposobu funkcjonowania fragmentu rzeczywi-

[4] Bogaty przegląd metod projektowania, uwzględniający przedstawiony sposób podejścia, można znaleźć w pracy (133). Inne, nowsze koncepcje zostały opisane w pracy (91).

stości obsługiwanej przez system informacyjny oraz projektowanej struktury i funkcjonowania samego systemu informacyjnego;
— język opisu systemu informacyjnego, który umożliwia sformalizowany opis pojęć występujących w modelu systemu informacyjnego możliwych do zinterpretowania komputerowo;
— narzędzia komputerowego wspomagania budowy i rozwoju systemów informacyjnych; wspomaganie to może obejmować skomputeryzowany opis projektowanego systemu (dokumentowanie prac), zautomatyzowanie czynności projektowych i programowych na poszczególnych etapach budowy systemu, zarządzanie pracami nad systemem oraz przydzielonymi zasobami.

Znaczącą rolę w teorii i praktyce tworzenia systemów informatycznych odgrywa model obiekt — atrybut — związek. W oryginalnej (15) bądź zmodyfikowanej postaci jest on podstawą licznych szczegółowych metod wykorzystywanych przede wszystkim na etapie analizy i projektowania systemów. Wydaje się, że zadecydowała o tym łatwość i przystępność, co umożliwia przeprowadzenie pełnego i spójnego opisu i modelowania danej dziedziny przedmiotowej. Stosowany formalizm jest niezależny od technologii informatycznej i gwarantuje dogodny sposób komunikowania się członków zespołu wykonawczego. Reprezentatywne dla tego podejścia są metody MERISE i REMORA (85)[5].

Interesującą metodą tworzenia systemów informatycznych, zdobywającą należną jej popularność, jest metoda prototypowania, która dla efektywnego stosowania wymaga jednak silnego wspomagania narzędziami komputerowymi (88, 35).

Przedstawione metody tworzenia systemu informatycznego mają zastosowanie na różnych etapach prac. Jedne z nich obejmują dłuższy okres z cyklu życia systemu[6], inne krótszy, w zależności od uniwersalności metody. Techniki, z racji swej większej szczegółowości, mają zastosowanie w jeszcze krótszym okresie cyklu życia, są bardziej specjalistyczne. Różnorodność technik stosowanych w procesie tworzenia systemu informatycznego jest ogromna. Brak jednorodnych kryterium klasyfikacji utrudnia dobre, skrótowe ich przedstawienie. Proponowane zestawienie jest jednym z możliwych i sprowadza się

[5] Inne podejście, które wykorzystuje ten model, jest podstawą oryginalnej metody globalnej, opracowanej przez zespół przywołany już w pracy (123), por. (105).

[6] Cykl życia systemu (life cycle) to okres od zdefiniowania zadania projektowego, przez realizację i wdrożenie, aż po eksploatację użytkową i pielęgnację systemu. Okres ten jest bardzo różnorodnie dzielony na etapy realizacji. Problem będzie dokładniej omówiony w dalszej części pracy.

do wyboru niektórych, najpopularniejszych stosowanych w procesie tworzenia systemu. Techniki tworzenia systemów powinny być:
— „technologicznym" rozwinięciem wcześniej przedstawionych metod;
— w sposób zadowalający zdefiniowane, nadające się do zastosowania w pracach nad systemem informatycznym.

Techniki te podzielono na trzy następujące grupy:
— heurystyczne, czyli techniki twórczego rozwiązywania problemów projektowych;
— specyfikacji;
— szczegółowe.

1.2.2. Techniki heurystyczne

Techniki heurystyczne dzieli się zazwyczaj na cztery następujące grupy:
— techniki odroczonego wartościowania,
— techniki transpozycji,
— techniki sugerowania,
— techniki złożone.

Klasycznym przykładem technik odroczonego wartościowania są różnorodne wersje burzy mózgów. Technika burzy mózgów należy do podstawowych metod poszukiwania pomysłów i może być wykorzystana do usprawniania dowolnej dziedziny działalności ludzkiej. Jej podstawą jest założenie, że grupowa praca interdyscyplinarnych zespołów twórczych zapewnia większą efektywność niż twórcze możliwości jednostki. Istotnym elementem tej techniki jest zasada rozdzielenia fazy poszukiwań rozwiązania od fazy oceny rozwiązań. Zakłada się bowiem, że zbyt wczesne ocenianie często powoduje zaprzestanie poszukiwań po zgłoszeniu pierwszej propozycji lub odrzucenie propozycji nowych i oryginalnych tylko dlatego, że nie mieszczą się w znanych, utartych schematach.

Przykładem techniki transpozycji może być technika analizy funkcji. Analiza funkcji jest badaniem szczegółowych i ogólnych zasad zależności zachodzących w systemach. Ma na celu raczej ułatwienie zrozumienia tych zasad i zależności niż wskazanie konkretnego rozwiązania.

Przykładem technik sugerowania może być krytyczna ocena i analiza Polaya. Krytyczna ocena i analiza jest uniwersalną techniką heurystyczną, najczęściej stosowaną do poszukiwania, oceny i optymalizacji rozwiązań or-

ganizacyjnych. Może jednak być z powodzeniem wykorzystana do procesów wytwórczych, informacyjnych, struktur, funcjonowania jednostek gospodarczych itp. Pozwala na wprowadzanie nowych rozwiązań, eliminację zbędnych czynności, uproszczenie sposobu wykonywania czynności, zmianę kolejności, miejsca.

Technika Polaya jest zbiorem warunków koniecznych i dostatecznych, które powinny być spełnione przez każdego, kto zamierza efektywnie poszukiwać rozwiązań. Technikę tę cechuje niezwykle wysoki stopień ogólności, co zwiększa możliwości jej zastosowania. Podstawowe wskazanie jest bardzo proste i sprowadza się do tego, aby odgadywać i sprawdzać. Należy postępować racjonalnie, oszczędnie i wytrwale, zachowując następujące reguły preferencji:

— łatwe ma pierwszeństwo przed trudnym;
— lepiej znane ma pierwszeństwo przed mniej znanym;
— to, co ma więcej punktów wspólnych z zadaniem, ma pierwszeństwo przed tym, co ma mniej takich punktów;
— całość ma pierwszeństwo przed częściami;
— zadania wcześniej rozwiązane mają pierwszeństwo przed innymi.

Przykładami technik złożonych są: technika morfologiczna, synektyka, ARIZ, ANKOT.

Technika morfologiczna, często nazywana metodą produkowania wynalazków, może być stosowana do analizy i projektowania rozwiązań w dowolnej sferze ludzkiej działalności. Pozwala na ujawnianie możliwych rozwiązań, to znaczy: sposobów rozwiązania, nowych, nieznanych rozwiązań, rozwiązań racjonalnych, oryginalnych, nierealnych, niemożliwych do realizacji, czy potencjalnych kierunków rozwiązań. Najogólniej technika morfologiczna polega na systematycznym przeszukiwaniu interesującej nas dziedziny, aby jasno i jednoznacznie sformułować zadanie, a następnie podać wszystkie możliwości jego realizacji.

Uwzględniając fakt, że najciekawsze i oryginalne wynalazki, oraz nowe teorie powstają przez obalenie uprzednio istniejących prawd, uznanych za niepodważalne, synektyka opiera się na zasadach negacji, inwersji lub podważania najbardziej utartych prawd, reguł i dogmatów. Kruszenie czegoś, co powszechnie jest uważane za monolit, oraz łączenie rzeczy, cech, faktów, które pozornie się ze sobą nie łączą — to podstawowe kanony synektyki.

Myślą przewodnią techniki ARIZ jest dążenie do uzyskania idealnego wyniku. Przeszkodą w osiągnięciu pożądanego ideału są sprzeczności między cechami wyimaginowanego rozwiązania a rzeczywistością. Przezwyciężanie tych sprzeczności nie pozwala wprawdzie na pełne osiągnięcie ideału, lecz

prowadzi do rozwiązania, które jest doskonalsze od obecnego. Wprowadza się pojęcie maszyny idealnej, traktowanej jako wzorzec, do którego powinno zmierzać rozwiązanie. Pojęcie maszyny idealnej uogólnia się na pojęcie idealnego wyniku końcowego, określonego jako rozwiązanie zadania techniczno-organizacyjnego, gdy nie istnieją żadne ograniczenia.

Technika Analizy Kontrastowej Obiektów Technicznych (ANKOT) powstała z potrzeby kontrastowania opisu obiektów w celu uzyskania informacji niezbędnych do pełnej i obiektywnej oceny. Technika może być stosowana do opisu obiektu i poszukiwania nowych rozwiązań. Jest ona stosunkowo prosta i polega na kontrastowaniu kolejnych obrazów, cech aspektów, atrybutów itp. opisujących badany obiekt. Technika ta jest szczególnie przydatna do opracowywania i oceny pomysłów uzyskanych podczas seansów burzy mózgów.

1.2.3. Techniki specyfikacji

Pełna definicja zadania projektowego jest problemem trudnym do rozwiązania, ale równie trudne jest specyfikowanie zadania projektowego, które ma być rozwiązane przez zespół w dalszej części procesu tworzenia. Są różnorodne techniki specyfikacji. W pracy będą przedstawione te, które są najpopularniejsze i mają odpowiadające im narzędzia komputerowego wspomagania, czyli:

— pakiet PSL/PSA (136),
— tablice decyzyjne (109).

Zasadniczym celem pakietu PSL/PSA jest wspomaganie projektowania struktury informacyjnej systemu informatycznego. Przede wszystkim wspomagane są: opis i analiza systemu, projektowanie struktury danych, ocena zaproponowanych rozwiązań.

Technika specyfikacji modułu wykorzystująca dane została stworzona dla Projektu ISDOS i jest znana jako Język Formułowania Problemu (Problem Statement Language, PSL). Stworzony pakiet programowy PSL/PSA wspomaga specyfikację danych oraz ich przepływ między modułami. Daje on następujące możliwości:

— opis systemu informacyjnego;
— rejestrację, przechowywanie i udostępnianie tego opisu;
— aktualizację opisu;
— drukowanie dokumentacji.

PSL jest językiem komputerowo-przetwarzalnym. Do opisu pojęć i związków między nimi istnieje ograniczona liczba zarezerwowanych słów. Wszystkie związki w opisie mogą być precyzyjnie zdefiniowane. Każde pojęcie jest wprowadzane do bazy danych PSA wraz z kontekstem, w którym było użyte. Ponowne użycie tego samego pojęcia w sprzecznym kontekście jest sygnalizowane przez oprogramowanie. W ten sposób zapewniona jest spójność opisu. Zdania PSL stanowią wytyczne do tego, co należy podać w opisie, aby dokumentacja była zupełna.

Składnia języka PSL jest zbliżona do języka potocznego, co powoduje, że opis wyrażony tym językiem jest łatwy do zrozumienia. Język PSL umożliwia opis systemu wychodząc od ogólnych informacji i schodząc w miarę zdobywania danych do szczegółów. Możliwe jest również podejście odwrotne.

Zbieranie informacji może być realizowane metodami tradycyjnymi, a PSL tylko wskazywać, które dane mają być zbierane. Wyrażanie zebranych informacji w języku PSL polega na identyfikacji obiektów i pojęć oraz nadaniu im nazw. Określa się związki między obiektami i opisuje istotne własności każdego obiektu. Wprowadzanie danych do bazy danych jest połączone z kontrolowaniem poprawności zapisów, ich spójności i precyzji. Wydruki kontrolne i diagnostyczne są redagowane współbieżnie. Możliwe jest generowanie zestawień analitycznych z założonej bazy danych.

Tablica decyzyjna jest specyficznym sposobem zapisywania i analizowania złożonych algorytmów, w których bada się określony zestaw warunków i w zależności od ich aktualnego stanu wykonuje się jeden ze zdefiniowanych zestawów czynności. Tablica decyzyjna umożliwia zwięzłe i przejrzyste opisanie algorytmu przez pokazanie wszystkich branych pod uwagę warunków, rozważanych kombinacji tych warunków, wszystkich przewidywanych czynności i wszystkich zestawów czynności powiązanych z zestawami warunków. Tablice decyzyjne mogą być używane na różnych etapach procesu tworzenia systemu. Za ich pomocą można specyfikować sytuacje decyzyjne na etapie analizy oraz opisywać algorytmy zamiast sieci działań. Jest ona logicznie podzielona na cztery podstawowe pola, które można schematycznie przedstawić następująco:

Odcinek warunków	Pozycja warunków
Odcinek czynności	Pozycje czynności

Odcinek warunków zawiera specyfikację wszystkich warunków branych pod uwagę w opisywanym problemie. Każdy warunek jest wyrażeniem logicznym typu IF, reprezentującym wartość prawda lub fałsz. Warunki powinny być zapisane w kolejności priorytetów ich testowania. Warunki mniej istotne lub zależne od innych powinny być zapisane po warunkach bardziej znaczących. Do tablicy wpisuje się tylko treść warunku.

Pozycja warunków zawiera stabelaryzowaną specyfikację wszystkich zestawów rozważanych warunków w opisywanym problemie. Dla każdej rozważanej w algorytmie kombinacji warunków należy wpisać w pozycji warunków tablicy decyzyjnej odpowiadający tej kombinacji zestaw kodów, które określają wymagany stan warunków[7].

Odcinek czynności zawiera specyfikację wszystkich czynności wykonywanych w opisanym problemie, która może być bezpośrednim zapisem czynności w tablicy lub wywołaniem procedury zdefiniowanej niezależnie od tablicy decyzyjnej. Może to być również wywołanie innej tablicy decyzyjnej. Czynności należy podawać w takiej kolejności, w jakiej mają być wykonywane.

Pozycja czynności zawiera stabelaryzowaną specyfikację wszystkich czynności odpowiadających poszczególnym zestawom warunków wyspecyfikowanych w pozycji warunków. Każdej specyfikacji zestawu warunków w pozycji warunków musi odpowiadać tylko jedna specyfikacja w pozycji czynności. Czynności wskazane do wykonania w zestawie są wykonywane w kolejności ich zapisania w odcinku czynności.

1.2.4. Techniki szczegółowe

Techniki szczegółowe to duża grupa technik mających zastosowanie w różnych fazach procesu tworzenia systemu informatycznego. Przedstawione zestawienie dotyczy głównie technik wykorzystywanych we wstępnych fazach prac nad systemem. Inne techniki związane z procesem szczegółowego projektowania, programowania i dokumentowania prac wymagałyby dłuższego opisu. O niektórych z nich będzie mowa w dalszej części.

Skrót HIPO pochodzi od pełnej nazwy tej techniki, która jednocześnie oddaje jej istotę — Hierarchy plus Input — Process — Output (Hierarchia plus Wejście — Proces — Wyjście). Jest to technika dokumentowania ukierunko-

[7] Wpisanie Y oznacza, że warunek musi być spełniony (prawda), a N oznacza, że warunek nie może być spełniony (fałsz), jeśli reguła ma być wybrana. Wpisanie — oznacza, że warunek nie ma znaczenia.

wana na prezentowanie struktury systemu za pomocą zestawu schematów[8]. Schematy te umożliwiają pełne odzwierciedlenie systemu na różnych poziomach szczegółowości, aż do poziomu programów (52).

Technika HIPO może być zastosowana, zgodnie ze swym pierwotnym przeznaczeniem, do dokumentowania istniejących lub nowo tworzonych systemów, lub do projektowania systemu. System opisuje się za pomocą trzech rodzajów schematów, rozpoczynając od schematu najbardziej ogólnego do schematów szczegółowych:

— schemat zawartości systemu, który obrazuje hierarchiczny podział systemu na składniki funkcjonalne niższego rzędu; stanowi on jednocześnie identyfikację schematów funkcjonalnych, które należy wykonać w dalszych pracach;
— schematy funkcjonalne ogólne, które służą do opisu ogólnych funkcji systemu, zlokalizowanych na najwyższych poziomach struktury hierarchicznej systemu; opisują one wejście, procesy i wyjścia w sposób ogólny (na poziomie zbiorów), zawierają jednocześnie odesłanie do schematów szczegółowych;
— schematy funkcjonalne szczegółowe służą do opisu funkcji systemu zlokalizowanych na najniższych poziomach struktury hierarchicznej systemu; pokazują szczegółowe dane wykorzystywane w realizacji danej funkcji, ciąg operacji przekształcających dane wejściowe oraz dane wyjściowe stanowiące wynik danego procesu; zawierają również odsyłacze do tych składników dokumentacji, które nie mogą być opisane za pomocą techniki HIPO, jak projekty tabulogramów, zbiorów, dokumentów itp.

Na opis systemu składa się więc pakiet schematów funkcjonalnych ogólnych i szczegółowych dla wszystkich wymienionych w schemacie zawartości systemu poziomów struktury hierarchicznej systemu.

Inną techniką stosowaną w procesie tworzenia systemu jest wywiad, bogate źródło wiadomości o badanym systemie, zdobywane w trakcie bezpośrednich rozmów z osobami, które pracują w jednostkach będących obiektem badania. Technika wywiadu jest szczególnie przydatna do wyjaśnienia zagadnień niezrozumiałych, które są wynikiem mylnego interpretowania przez użytkownika obowiązujących regulaminów i instrukcji. W czasie wywiadu badający ma doskonałą okazję do wyjaśnienia swojej działalności badawczo-analitycznej[9].

[8] Schematy są podstawową cechą tej techniki, stąd wynika jej inne usytuowanie niż tablic decyzyjnych, gdzie tablica jest tylko formą wyrażenia procesu postępowania.

[9] Niezwykle ciekawe spostrzeżenia związane z praktyką wywiadów można znaleźć w pracy (71). Autor zwraca uwagę na precyzyjne przygotowanie scenariusza i określenie celów wywiadu oraz opisuje technikę prowadzenia i opracowywania materiału.

Inną, podobną do wywiadu techniką zbierania informacji jest ankieta. Ankieta jest praktycznym środkiem otrzymania w krótkim czasie konkretnych informacji z różnych jednostek organizacyjnych badanego obiektu. Technika ta wymaga opracowania zbioru pytań, które dotyczą określonego zagadnienia. Stosuje się ją wówczas, gdy można liczyć na współpracę respondentów.

Obserwacja jest techniką polegającą na dokładnym śledzeniu czynności wykonywanych przez osoby biorące udział w procesie funkcjonowania systemu.

Dużą grupę stosowanych technik szczegółowych tworzą różnorodne schematy i wykresy, które służą do zobrazowania hierarchii podporządkowania oraz podziału uprawnień i funkcji między komórkami w aparacie zarządzania. Dzięki schematom można uzyskać autoryzowane wzorce, na których podstawie łatwo jest ustalić obszary władzy, sfery odpowiedzialności i kanały informacyjne analizowanego obiektu.

Szczególną formą schematów jest tablica krzyżowa, czyli dwuwymiarowa tablica, której podstawą są dwie zmienne. Za pomocą tych tablic można uchwycić wzajemne związki między wyszczególnionymi zmiennymi. Istnienie związku wyjaśnia znak X, który jest wstawiany w miejscu przecięcia się wierszy z kolumnami.

1.3. Jakość systemów informatycznych

Wybór odpowiedniej metody, techniki tworzenia systemu ma nie tylko wpływ na sprawność procesu realizacji, ale również na jakość wytworzonego produktu. Produktem będącym obiektem oceny przez użytkownika jest system informatyczny wspomagający proces zarządzania. System ten powinien spełniać oczekiwania użytkownika na zrealizowanie funkcji zdefiniowanych na początku procesu. Obiektem oceny jest oprogramowanie systemu, które realizuje rozwiązania projektowe i jest ostatecznym wynikiem długiego cyklu wytwarzania. Mówiąc o jakości systemu, ma się w zasadzie na myśli cechy jakościowe systemu oprogramowania powstałego w procesie tworzenia systemu informatycznego.

Wytworzone oprogramowanie realizuje koncepcję i rozwiązania zdefiniowane we wcześniejszych fazach procesu tworzenia. Aktualność tych

rozwiązań, po niezbędnym czasie na wytworzenie oprogramowania, w dużej mierze zależy od inwencji autorów oraz zastosowanych metod i technik pracy. Oczywiste jest jednak, że system oprogramowania będzie funkcjonował pewien czas, po którym konieczne będą zmiany i modyfikacje, które najczęściej wynikają ze zmian w obiekcie użytkowania systemu lub z nowych koncepcji użytkownika. Konieczny będzie więc etap pielęgnacji systemu informatycznego [10], tak aby cały czas spełniał zmieniające się oczekiwania użytkownika. W związku z tym należy inaczej spojrzeć na sam proces i stosowane metody, techniki pracy. Bardzo ważna staje się jakość systemu informatycznego, a w zasadzie jakość systemu oprogramowania, które decydująco wpływa na jakość całego systemu.

Użytkownika interesuje tylko oprogramowanie pracujące. Elegancja programu, metody wytwarzania są mniej interesujące z punktu widzenia użytkowego. Podstawową cechą programu musi być jego funkcjonowanie zgodnie z założeniami i celami wytworzenia.

W procesie tworzenia systemu oprogramowania istotny jest czas i koszt realizacji oprogramowania. Minimalizowanie czasu i kosztów realizacji oraz dostarczenie narzędzi ułatwiających kontrolę i sterowanie procesem tworzenia, to wymogi stawiane metodom programowania przez programistów i kierujących zespołami. Inne elementy na tym etapie procesu tworzenia są drugorzędne.

Podstawowy atrybut jakości oprogramowania sprowadza się do tego, że program jest pracujący. Inne atrybuty jakości, takie jak:

— solidność,
— przenośność,
— podatność na przyszłe zmiany,
— łatwość zrozumienia przez innych,
— ekonomiczność w działaniu,
— skuteczność,
— odpowiednie reagowanie w sytuacjach nietypowych,

to cechy, których istota ujawni się w czasie użytkowania.

Problemy jakości oprogramowania, na które dotychczas nie zwracano zbyt dużej uwagi, stały się przedmiotem badań. Jakość wytwarzanego oprogramowania zależała od doświadczeń i umiejętności programistów. Gdy koszt sprzętu komputerowego maleje, a koszty wytworzenia oprogramowania rosną, zwłaszcza relatywnie wysokie są koszty pielęgnacji programu, sprawa jakości

[10] Angielski termin maintenance w polskiej literaturze jest tłumaczony jako pielęgnacja lub konserwacja. Chodzi o proces mający na celu utrzymanie systemu oprogramowania w stanie użyteczności.

nabrała odpowiedniego wymiaru. Nowe metody wytwarzania oprogramowania kładą nacisk na tworzenie programów wysokiej jakości.

Jakimi metodami osiągnięto wyższą jakość oprogramowania? Jedną z dróg jest wspomaganie procesu tworzenia programu. Metody programowania powinny uwzględniać:

— projektowanie programu;
— pisanie programu w najkrótszym czasie;
— gruntowne testowanie;
— usprawnienie wytwarzania i kontrolę procesu tworzenia.

Realizacja tych postulatów napotyka jednak na następujące trudności:

— praktycznie niemożliwe jest poprawne projektowanie za pierwszym podejściem;
— zbyt skomplikowane jest jednoczesne projektowanie, kodowanie i optymalizowanie;
— pomyłki powstają na wszystkich etapach realizacji;
— elementy kontroli wpływają na spadek tempa wytwarzania.

Metody programowania powinny więc godzić z sobą przeciwstawne wymagania, znajdować kompromisowe wyjście. Należy akceptować współbieżny proces kodowania i projektowania. Dzielić pracę na zespoły, eliminować przyczyny powstawania pomyłek, szybko je wykrywać i usuwać. Można mnożyć wymagania i ograniczenia procesu tworzenia oprogramowania oraz żądania stawiane metodom programowania. Część tych postulatów jest prosta w realizacji, inne wymagają gruntownej zmiany stylu pracy. Główne wytyczne do stylu programowania można znaleźć w pracach (94, 25, 74, 139).

Jakość programu komputerowego nie jest prosta do zdefiniowania. W odróżnieniu od produktów materialnych trudno jest ocenić jakość produktu konceptualnego. Można ją określić jako stopień ideału lub wielkość odchylenia od wzorca idealnego. Wówczas pojawia się problem określenia ideału. Ponadto istnieją różnorodne kryteria oceny jakości programu w zależności od konkretnej sytuacji i stawianych wymagań.

W związku z trudnościami z jednoznacznym zdefiniowaniem jakości programu można mówić o atrybutach jakości. Atrybuty jakości programu będą określone na dwóch poziomach: „wysokim", czyli takim, który bezpośrednio określa relacje użyteczności i utrzymania programu w aktualnym stanie, oraz „prymitywów", czyli podstawowych elementów, których kombinacja daje atrybut „wysokiego" poziomu (78).

Atrybutami „wysokiego" poziomu, które opisują jakość programu, są:

Poprawność. Jest to podstawowy atrybut o największym znaczeniu. Program musi pracować zgodnie z założeniami. Gdy program jest niepoprawny, wszystkie inne atrybuty są nieistotne. Poprawność nie dotyczy kodu źródłowego, ale pełnej zgodności ze specyfikacją przy założeniu, że specyfikacja dobrze odwzorowuje rzeczywiste wymagania użytkownika.

Poza poprawnością występują dwie główne grupy atrybutów:

— wewnętrzne, określające jakość programu z punktu widzenia budowy kodu źródłowego;

— kontekstowe, określające jakość procesu tworzenia oprogramowania i jego środowisko.

Atrybuty wewnętrzne dla użytkownika, to:

Niezawodność, która zależy od liczby błędów w programie. Program nie jest błędny, gdy reaguje prawidłowo na błędne dane lub ich kombinacje albo reaguje poprawnie na błędy obsługujących system ludzi, błędne parametry itp.

Kompletność. Wszystkie sytuacje, które mogą wystąpić, muszą być obsłużone przez program.

Używalność. Program musi być zrozumiały i łatwy w użyciu. Należy pamiętać, że program jest przeznaczony dla ludzi bez specjalnego przygotowania informatycznego.

Efektywość. Wyniki pracy programu muszą być dostarczone w odpowiednim czasie, zgodnie z potrzebami użytkownika. Czas otrzymania wyników będzie zależał od środowiska sprzętowego i programowego oraz od klasy systemu. W systemach czasu rzeczywistego czas jest bardzo istotnym kryterium i od niego zależy wiele innych decyzji w systemie.

Atrybuty wewnętrzne dla zarządzania przetwarzaniem danych to:

Możliwość testowania. Projekt musi pozwalać na wykonanie odpowiedniego testowania. Można to osiągnąć przez modularną strukturę wydzielającą indywidualne komponenty, moduły programowe. Każdy z nich powinien mieć jeden punkt wejścia i wyjścia[11].

Możliwość modyfikowania. Projekt powinien mieć charakter otwarty, tak aby istniała możliwość łatwego dołożenia nowych funkcji i dokonania zmian w już realizowanych bez potrzeby wstrzymania eksploatacji.

Możliwość przenoszenia. Powinna istnieć możliwość wykonania programu na różnym sprzęcie, co oznacza łatwość instalowania programu w nowym środowisku sprzętowym i programowym. Przy minimalnym koszcie pro-

[11] Problematyka ta znalazła odzwierciedlenie w literaturze w pojęciu programowanie modularne (89).

gram powinien dać się adaptować do warunków innego, podobnego obiektu. Osiąga się to przez uniezależnienie programu od struktur danych i wykonywanych operacji, używanie standardowych procedur i języków programowania „wysokiego" poziomu.

Atrybuty wewnętrzne dla pielęgnacji programu, to:

Podatność na pielęgnację. Poprawność programu jest cechą oczywistą i musi być utrzymana w całym okresie eksploatacji. Wymaga to logicznej, czytelnej struktury programu napisanego w prosty sposób, bez specjalnych komplikacji jego rozwiązań. Program musi mieć czytelną i poprawną dokumentację.

Zgodność. Program powinien być pisany na standardową konfigurację sprzętową i wykorzystywać podstawowy aparat instrukcyjny języków programowania. Powinien umożliwiać łatwe zrozumienie i wprowadzanie zmian przez innych.

Atrybuty wewnętrzne dla operowania programem, to:

Używanie pamięci operacyjnej. Pamięć powinna być wykorzystywana ekonomicznie bez specjalnych „sztuczek" programowych.

Używanie urządzeń zewnętrznych. Urządzenia powinny być przydzielane do programu dynamicznie tylko wtedy, gdy zachodzi potrzeba wykonania transmisji. Wszelkie błędy transmisji powinny być obsługiwane przez program w sposób standardowy.

Bezpieczeństwo. Program powinien być zabezpieczony przed złamaniem poufności, czyli mieć zastrzeżony dostęp itp.[12]

Łatwość użytkowania. Operowanie programem powinno być zawsze jednakowe i możliwe do wykonania bez specjalnego instruktażu. Należy unikać decyzji w trakcie biegu programu. Operowanie zbiorami danych powinno być możliwe wyłącznie z konsoli operatorskiej.

Wprawdzie jakość programu jest cechą trudną do jednoznacznego zdefiniowania, ale biorąc pod uwagę konkretny projekt, można określić pewien standard cech jakościowych produktu końcowego, który będzie wymagany od zespołu wykonawców. Mogą to być atrybuty odnoszące się tylko do wybranych części projektu lub mieć charakter ogólnych wytycznych.

Jeśli definiuje się jakieś kryterium jakościowe, to powinno się mieć metody pozwalające na stwierdzenie, czy program spełnia je, czy nie. Znane są trzy proste techniki sprawdzania jakości programu.

[12] Problematyka ta jest szczególnie istotna w sytuacji, gdy program pracuje w sieciach komputerowych. Wskazówki jak osiągnąć w programie tę cechę można znaleźć w pracy (53).

Jedna z nich polega na tym, że program źródłowy jest sprawdzany przez innego członka zespołu zanim zostanie poddany dalszej obróbce. Jest to dość prymitywna metoda i nie gwarantuje pełnego sukcesu. Pozwala jednak na nowe spojrzenie na program przez osobę postronną, nie związaną emocjonalnie z programem, co może wnieść wiele nowego. Rozwiązanie to jest bardzo tanie i możliwe jedynie w małych organizacjach.

Inna, bardziej złożona metoda polega na strukturalnym przejściu przez program. W metodzie tej nie porównuje się programu z własną wizją, ale wykonuje go logicznie, zgodnie z zapisem kodu źródłowego. Postępowanie takie nosi nazwę „suchego przebiegu", gdyż polega na wykonaniu programu na danych testowych, ale bez użycia komputera, tylko zgodnie z logiczną realizacją instrukcji programowych. Metoda ta może być stosowana indywidualnie lub zespołowo.

Metoda inspekcji, lansowana przez IBM, jest wprowadzeniem do rozwoju inżynierii oprogramowania. Określone są pewne punkty kontrolne, które powinny być osiągnięte w całym cyklu projektowania. Metoda ta ma charakter porównania. Zespół ekspertów ocenia, czy określony etap prac został wykonany zgodnie z wzorcem, czy należy go powtórzyć. Spełnienie wymagań danego etapu i akceptacja przez ekspertów pozwala na przejście do następnego kroku postępowania projektowego.

Proces inspekcji dzieli się na pięć kroków, w których kolejno cały zespół oceniający oraz poszczególni członkowie zapoznają się z wynikami prac poddawanych ocenie. W zależności od liczby i wagi wykrytych błędów, zespół oceniający decyduje, czy poprawienie zauważonych w trakcie inspekcji błędów pozwala na przejście do dalszych prac, czy ponawiany jest proces inspekcji. Zakłada się, że metoda inspekcji pozwala na eliminację około 70 % typowych błędów projektowania i kodowania.

Przedstawione w zarysie sposoby osiągnięcia niektórych cech jakościowych nie rozwiązują podstawowego problemu, który polega na zagwarantowaniu odpowiedniego poziomu jakości całemu systemowi informatycznemu.

1.4. Przesłanki standaryzacji

Według jednej z najprostszych definicji jakość jest to spełnienie oczekiwań użytkownika. Problem projektowy polega na tym, czy oczekiwania użytkownika są prawidłowo interpretowane przez wykonawców. W procesie wytwarzania systemu często mówi się o specyfikacji i sprawdzaniu wyników prac w sensie zgodności ze specyfikacją. Zakłada się, że specyfikacja jest:

— jednoznaczna,
— zupełna,
— możliwa do zweryfikowania,
— spójna,
— możliwa do zmodyfikowania,
— łatwa do prześledzenia,
— możliwa do wykorzystania w fazie tworzenia, eksploatacji i pielęgnacji oprogramowania (66).

W celu osiągnięcia takich cech specyfikacji nie można zdać się tylko na rzetelność autorów, ale należy wprowadzić mechanizmy porządkujące proces specyfikowania. Kierując się tą przesłanką, opracowano dokument *Software Requirements Specification* (3), który jest standardem specyfikacji. Podano w nim wymagania dotyczące zawartości i treści, które w formie standardu są jednym z istotnych elementów technologii wytwarzania oprogramowania i koniecznym warunkiem do zapewnienia należytej jakości.

Standaryzacja procesu programowania jest jednym ze środków osiągnięcia wysokiej jakości. Bardzo trudno to osiągnąć z wielu powodów. Nie można sobie wyobrazić standardu programowania, który pasowałby do wszystkich przypadków programowania. Różnorodność procesu programowania jest tak duża, że praktycznie jest to niemożliwe. Nie oznacza to jednak zaniechania prac nad standaryzacją procesu programowania.

Standardy mogą być ograniczone przez sprzęt, oprogramowanie operacyjne, język programowania, lub inne specyficzne uwarunkowania, ale ich przydatność, jest zawsze nieoceniona. Programowanie nie jest osobistą działalnością, ale zespołową, stąd konieczność standaryzacji, tak aby prace mogły być bez większych problemów przekazywane różnym członkom zespołów wykonawców. Standaryzacja powinna dotyczyć podstawowych elementów programowania, istotnych z punktu widzenia pracy zespołowej. Pozostałe elementy podlegają indywidualnej inwencji twórców programów.

Standardy programowe powinny być elementem dyscyplinującym i porządkującym proces programowania, ale zbytnio nie utrudniającym wykonawstwa. Praktyka dowodzi, że standardy wspomagające proces tworzenia oprogramowania będą przyjęte i stosowane przez programistów, a inne, utrudniające tworzenie programów — odrzucone.

Osiągnięcie odpowiednich cech jakościowych i niezawodności jest coraz trudniejsze w miarę wzrostu liczby zastosowań. Problemy integracji systemów informatycznych współpracujących z sobą występują na wielu różnorodnych poziomach. Potęgowo rosnące trudności z integracją mogą spowodować, że korzyści, które powinny wystąpić w związku z odpowiednio szerokim stosowaniem środków informatyki, są praktycznie nieosiągalne. Przyczyny są różne, ale najczęściej sprowadzają się do tego, że tworząc kolejne zastosowania, nie zwrócono uwagi na problem integracji i nie przygotowano odpowiednich środków łączności z innymi systemami informatycznymi (131).

Problem integracji zastosowań informatycznych występuje na całym świecie i pochłania bardzo duże środki (137). Rozwiązanie go jest znacznie prostsze, gdy w procesie tworzenia systemu stosuje się standardowe rozwiązania. Inwencja i wiedza twórców systemów informatycznych oraz odpowiedni poziom początkowych prac projektowych tylko częściowo gwarantuje istnienie elementów integrujących różne zastosowania. Zasadniczym problemem jest znajomość i akceptowanie standardów postępowania oraz normatywów obowiązujących w różnych fazach realizacji systemu. Zastosowanie istniejących standardów i normatywów gwarantuje, że powstałe systemy dadzą się zintegrować, tworząc nową jakość stosowania informatyki.

1.5. Metodologie tworzenia systemów informatycznych

1.5.1. Podstawy metodyczne tworzenia systemów informatycznych

W. M. Turski we wprowadzeniu do pracy[13] na temat metodologii programowania, pisze: „Metodologia (...) zawiera trzy omalże nierozdzielne pierwiastki: naukę, rzemiosło i folklor (mitologię)" (139, s. 1). Trafność takiego określenia złożoności procesu tworzenia systemu informatycznego wynika głównie z tego, że zwraca uwagę na wiele różnorodnych składników tego procesu. Dodając do tego cechy osobowe autorów, ich nawyki i przyzwyczajenia, uzyskuje się pełny obraz złożoności procesu. Kładąc nacisk na inne składniki procesu, otrzymuje się bardzo różnorodne podejścia do wytwarzania systemu informatycznego. Te różne podejścia są nazywane metodykami szczegółowymi lub metodykami i w większości koncentrują się na różnych rozwiązaniach tego elementu, który W. M. Turski nazywa rzemiosłem.

S. Wrycza w pracy (148) wyróżnia następujące składniki metodyki tworzenia systemu informatycznego:

— formalizmy, modele opisu rzeczywistości, czyli dziedziny przedmiotowej, jej statyki i dynamiki;
— strukturalizacja procesu tworzenia w postaci odpowiedniej procedury etapów, podetapów i poszczególnych zadań;
— szczegółowe metody i techniki tworzenia, dokumentowania wraz z odpowiednią symboliką graficzną;
— narzędzia wspomaganego komputerem tworzenia, w tym prototypowanie;
— specyfikacja wymagań merytorycznych wobec poszczególnych twórców interdyscyplinarnego zespołu wykonawców;
— kryteria oceny jakości projektu i systemu oraz mechanizmy jej kontroli;
— zasady planowania i sterowania rozwojem systemu.

Wymienione składniki tworzą środowisko pracy projektanta systemu informatycznego. Powiązania poszczególnych składników środowiska przedstawia rysunek 1.1. Istotnym elementem tego środowiska jest warsztat pracy projek-

[13] Praca ta może być doskonałym przewodnikiem ideowym o tworzeniu systemów informatycznych.

tanta, który wspomaga jego wiedzę i doświadczenie różnorodnymi narzędziami wytwarzania.

Dotychczas nie wypracowano jednorodnych, powszechnie uznanych zasad wytwarzania systemu, ale są różne metodyki, które kładą nacisk na inne elementy procesu. Jedną z przyczyn takiego stanu jest stosunkowo krótki okres doświadczeń, czyli nowość dziedziny, oraz jej nieustrukturalizowany charakter. Do tego dochodzi dynamiczny rozwój poszczególnych składników środowiska, co ma istotny wpływ na proces wytwarzania (133).

Niezadowalające wyniki zmuszają do poszukiwań nowych sposobów wytwarzania, które nie zawsze idą we właściwym kierunku. Często bowiem wprowadzają oryginalne rozwiązania, które sprowadzają się do nowej grafiki i nazewnictwa. Dziedzina ta dostarcza coraz nowszych sposobów tworzenia systemu informatycznego. Dzięki temu rozwijane jest środowisko i przybywa doświadczeń, co zaowocuje nowszymi, lepszymi jakościowo systemami.

Powstające i funkcjonujące metodyki można sklasyfikować według określonych kryteriów oceny[14] Nie wnikając w szczegóły klasyfikacyjne, można zdefiniować wymagania stawiane przed każdą metodyką. Analiza metodyk prowadzi do wniosku, że powinny one spełniać następujące wymagania:

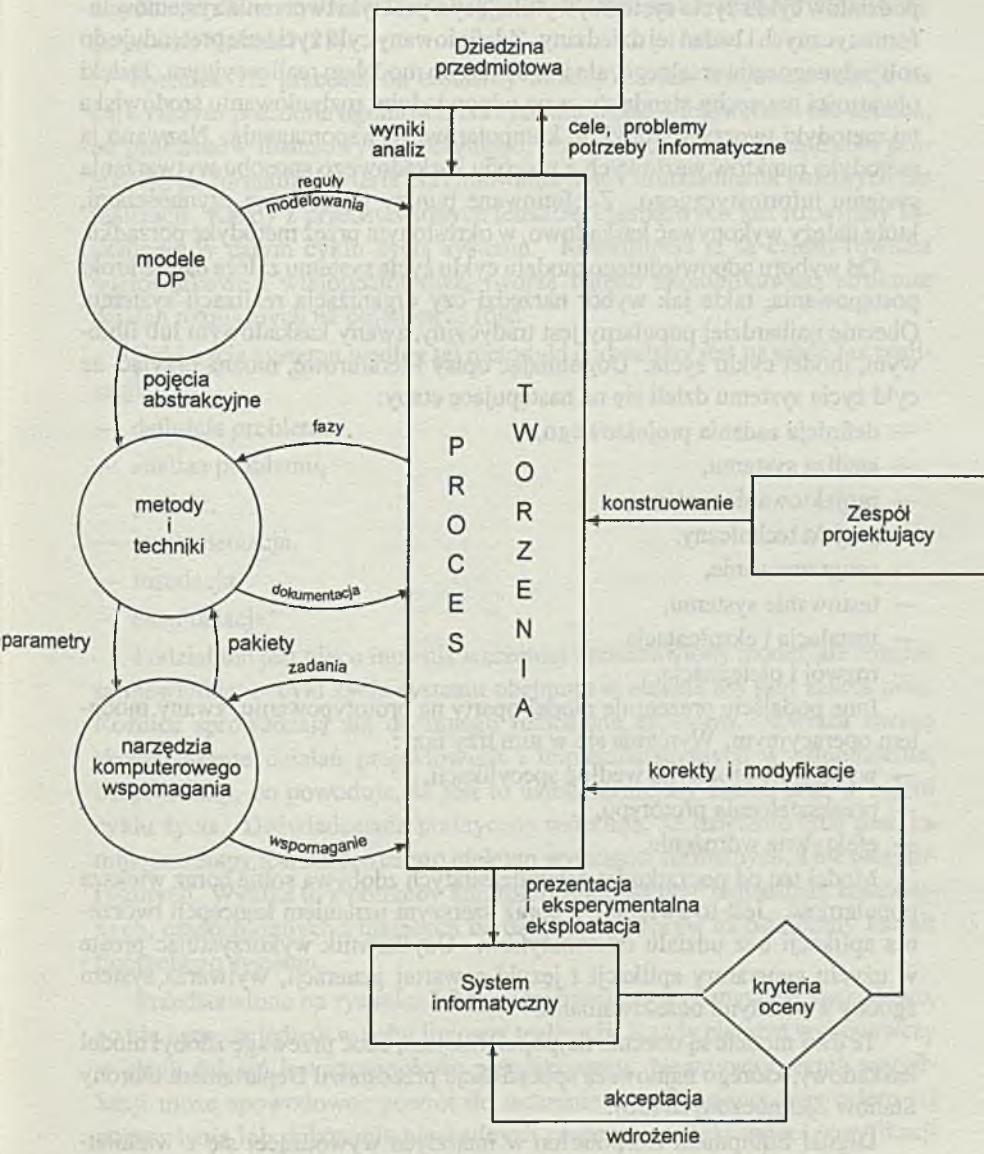
— obejmować cały cykl życia systemu — od zdefiniowania problemu aż po pielęgnację eksploatowanego systemu z jednoczesnym mechanizmem płynnego przejścia przez poszczególne fazy realizacji;
— proces tworzenia systemu powinien być wspomagany różnorodnymi metodami, technikami i narzędziami komputerowymi;
— mieć łatwy mechanizm komunikowania się członków zespołu wykonawczego;
— operować standardami, czyli mieć łatwość uczenia się i stosowania w różnych środowiskach.

Z przedstawionej listy wymagań wyłania się metodyka obejmująca pełny cykl życia systemu, operująca na odpowiednim zestawie standardów i wspomagana komputerowymi narzędziami wytwarzania (145).

Cykl życia systemu, jego różne definicje i podziały, to element, który leży u podstaw definiowania różnych metodyk tworzenia sytemów informatycznych[15]. Analiza literatury i praktyczne doświadczenia wynikające z realizacji różnych systemów informatycznych prowadzą do wniosku, że brakuje jednoznacznej, akceptowanej przez wykonawców metodyki.

[14] Próbę takiej klasyfikacji można znaleźć w pracy (106).

[15] Najbardziej reprezentatywne definicje różnych cykli życia systemu przedstawiają prace (103, 128, 40, 87, 79, 30).



Rys. 1.1. Środowisko pracy twórcy systemu informatycznego (148)

W dalszej części zostanie dokładniej przedstawiony jeden z możliwych podziałów cyklu życia systemu, wynikający z praktyki tworzenia systemów informatycznych i badań tej dziedziny. Zdefiniowany cykl życia nie pretenduje do roli jedynego uniwersalnego, ale jest otwartym modelem realizacyjnym. Dzięki otwartości ma cechy standardu, a po odpowiednim rozbudowaniu środowiska tej metodyki tworzy warunki do komputerowego wspomagania. Nazwano ją metodyką punktów węzłowych z powodu kaskadowego sposobu wytwarzania systemu informatycznego. Zdefiniowane punkty węzłowe są czynnościami, które należy wykonywać kaskadowo, w określonym przez metodykę porządku.

Od wyboru odpowiedniego modelu cyklu życia systemu zależą dalsze kroki postępowania, takie jak wybór narzędzi czy organizacja realizacji systemu. Obecnie najbardziej popularny jest tradycyjny, zwany kaskadowym lub liniowym, model cyklu życia. Uogólniając opisy literaturowe, można przyjąć, że cykl życia systemu dzieli się na następujące etapy:

— definicja zadania projektowego,
— analiza systemu,
— projektowanie systemu,
— projekt techniczny,
— programowanie,
— testowanie systemu,
— instalacja i eksploatacja,
— rozwój i pielęgnacja.

Inne podejście prezentuje model oparty na prototypowaniu, zwany modelem operacyjnym. Wyróżnia się w nim trzy fazy:

— wykonanie prototypu według specyfikacji,
— przekształcenia prototypu,
— efektywne wdrożenie.

Model ten od początku lat osiemdziesiątych zdobywa sobie coraz większą popularność. Jest to związane z coraz szerszym uznaniem koncepcji tworzenia aplikacji bez udziału informatyków. Użytkownik wykorzystując proste w użyciu generatory aplikacji i języki czwartej generacji, wytwarza system zgodny z własnymi oczekiwaniami.

Te dwa modele są obecnie najpopularniejsze, choć przewagę zdobył model kaskadowy, którego najnowszą specyfikację przedstawił Departament Obrony Stanów Zjednoczonych (30).

Digital Equipment Corporation w metodyce wywodzącej się z wieloletnich badań i doświadczeń praktycznych proponuje podejście kaskadowe, ale w ujęciu wielowarstwowym. Liniowo rozwijane działania przebiegają na wielu

różnych poziomach, wzajemnie się uzupełniając i kontrolując realizację celu wspólnego dla wszystkich ścieżek działań, którym jest system informatyczny „wysokiej" jakości (28).

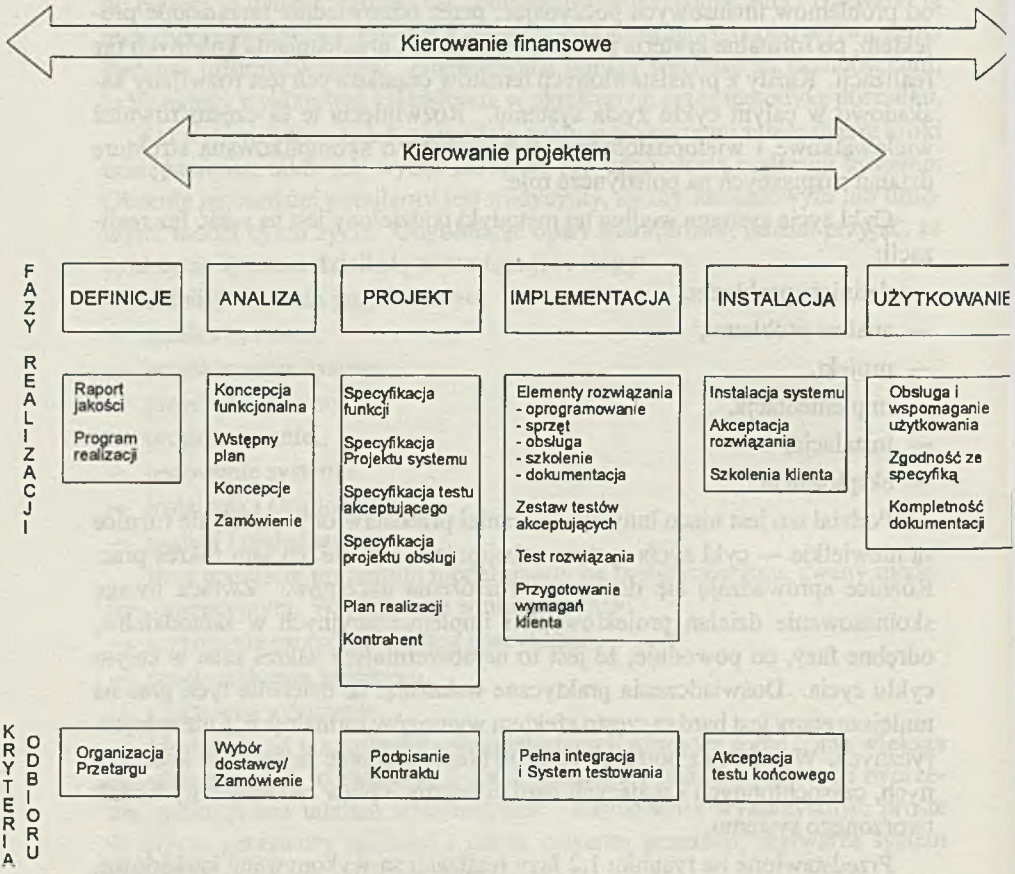
Rysunek 1.2 przedstawia elementy metodyki i ich wzajemne relacje na najwyższym poziomie ogólności. Na rysunku widać wielowątkowość działań, od problemów finansowych poczynając, przez odpowiednie zarządzanie projektem, po formalne kryteria przyjmowania prac i uruchamiania kolejnych faz realizacji. Każdy z przedstawionych tematów cząstkowych jest rozwijany kaskadowo w całym cyklu życia systemu. Rozwinięcia te są często również wielowątkowe i wielopoziomowe, tworzą bardzo skomplikowaną strukturę działań rozpisanych na pojedyncze role.

Cykl życia systemu według tej metodyki podzielony jest na sześć faz realizacji:

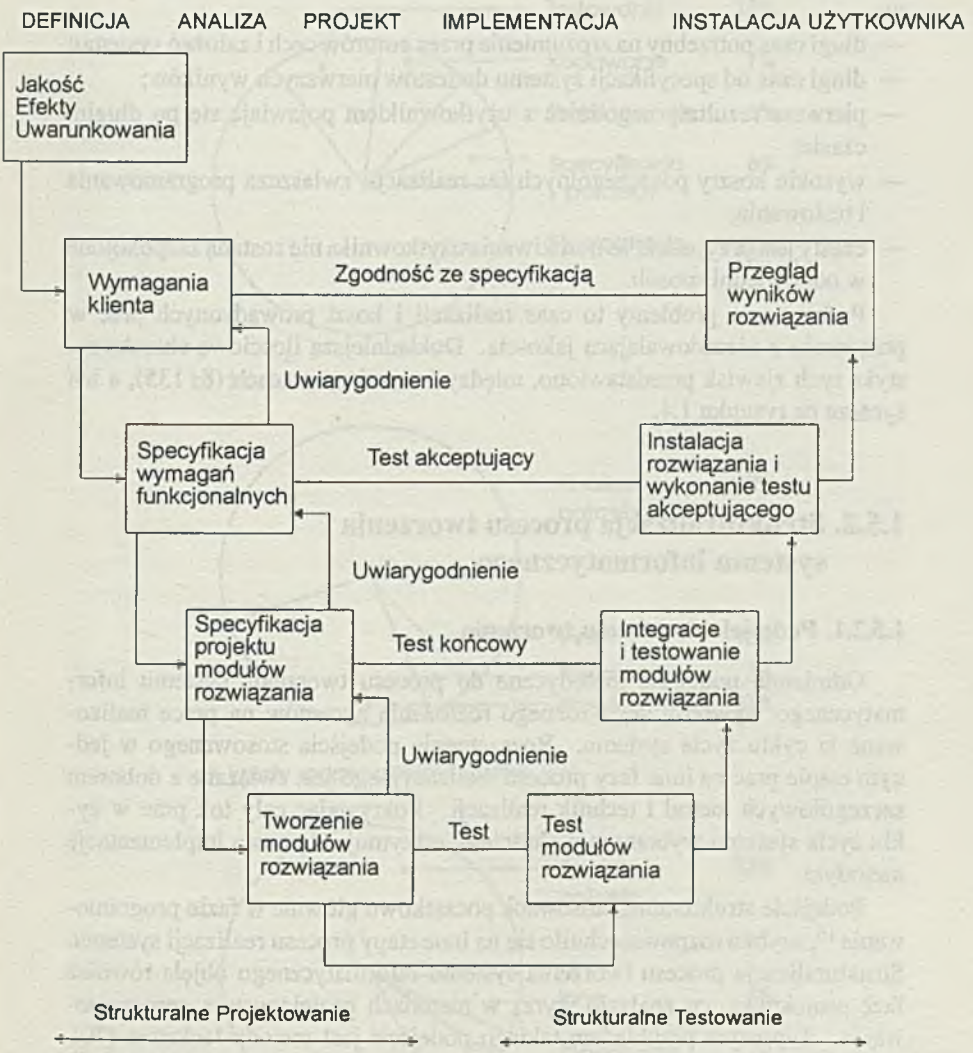
— definicja problemu,
— analiza problemu,
— projekt,
— implementacja,
— instalacja,
— eksploatacja.

Podział ten jest nieco inny niż wcześniej przedstawiony model, ale różnice są niewielkie — cykl życia systemu obejmuje w efekcie ten sam zakres prac. Różnice sprowadzają się do innego rozłożenia akcentów. Zwraca uwagę skomasowanie działań projektowych i implementacyjnych w samodzielne, odrębne fazy, co powoduje, że jest to najobszerniejszy zakres prac w całym cyklu życia. Doświadczenia praktyczne wskazują, że dzielenie tych prac na mniejsze etapy jest bardzo często efektem wymogów formalnych, a nie merytorycznych. Wynika to z potrzeby kontroli pośredniej prac najbardziej kosztownych, czasochłonnych i mających bardzo istotny wpływ na ostateczny kształt tworzonego systemu.

Przedstawione na rysunku 1.2 fazy realizacji są wykonywane kaskadowo, co nie oznacza jednak w pełni liniowej realizacji. Każdy element wykonawczy w ciągu działań jest szczegółowo weryfikowany. Negatywny wynik weryfikacji może spowodować powrót do wcześniej zrealizowanej fazy celem jej poprawienia lub dokonania niezbędnych uzupełnień. Taki model weryfikacji w połączeniu z odpowiednią dokumentacją tego procesu obowiązuje w całym cyklu życia systemu. Przedstawia to graficznie rysunek 1.3.



Rys. 1.2. Cykl życia systemu według metodyki firmy Digital (28)



Rys. 1.3. Realizacja systemu według metodyki firmy Digital (28)

Metodyki wywodzące się z różnych cykli życia systemu mają wspólne niedogodności, co zmusza do stałego ich doskonalenia. Podstawowe problemy to:

— długi czas potrzebny na zrozumienie przez autorów cech i założeń systemu;
— długi czas od specyfikacji systemu do testów pierwszych wyników;
— pierwsze rezultaty uzgodnień z użytkownikiem pojawiają się po długim czasie;
— wysokie koszty poszczególnych faz realizacji, zwłaszcza programowania i testowania;
— częsty jest przypadek, że oczekiwania użytkownika nie zostaną zaspokojone w odpowiedni sposób.

Podstawowe problemy to czas realizacji i koszt prowadzonych prac w połączeniu z niezadowalającą jakością. Dokładniejszą ilościową charakterystykę tych zjawisk przedstawiono, między innymi, w pracach (8, 135), a ich syntezę na rysunku 1.4.

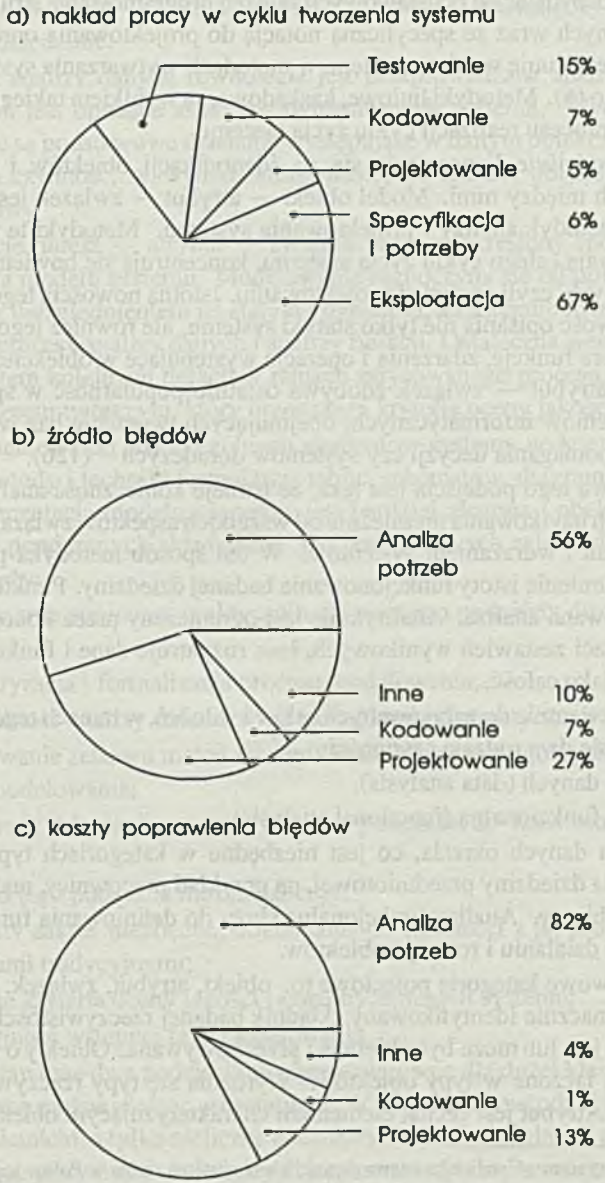
1.5.2. Strukturalizacja procesu tworzenia systemu informatycznego

1.5.2.1. Podejście do procesu tworzenia

Odmienne podejścia metodyczne do procesu tworzenia systemu informatycznego wywodzą się z różnego rozłożenia akcentów na prace realizowane w cyklu życia systemu. Rozszerzenie podejścia stosowanego w jednym etapie prac na inne fazy procesu realizacyjnego jest związane z doborem szczegółowych metod i technik realizacji. Pokrywając cały tok prac w cyklu życia systemu wybranym podejściem, otrzymuje się nową implementację metodyki.

Podejście strukturalne, stosowane początkowo głównie w fazie programowania [16], szybko rozpowszechniło się na inne etapy procesu realizacji systemu. Strukturalizacja procesu tworzenia systemu informatycznego objęła również fazę projektową, co znalazło wyraz w metodach projektowania oprogramowania. Typowym przykładem takiego podejścia jest metoda Jacksona (70),

[16] Podstawy teoretyczne programowania strukturalnego stworzył E. Dijkstra w latach 1968–1972 w licznych publikacjach, z których najważniejsze to prace (29, 22). Praktyczne wdrożenie programowania strukturalnego rozpoczęło się od pozytywnych doświadczeń firmy IBM, opisanych w pracach (5, 92).



Rys. 1.4. Koszty i czas realizacji prowadzonych prac (148)

która rozpoczynając od podstawowych założeń programowania strukturalnego, zastosowanych wraz ze specyficzną notacją do projektowania oprogramowania (127), ewoluuje w miarę postępu w metodach wytwarzania systemu informatycznego (6). Metodyki liniowe, kaskadowe, są wynikiem takiego podejścia do całego procesu realizacji cyklu życia systemu.

Inne podejście koncentruje się na identyfikacji obiektów i relacji zachodzących między nimi. Model obiekt — atrybut — związek jest podstawą licznych metodyk analizy i projektowania systemu. Metodyki te najczęściej nie pokrywają całego cyklu życia systemu, koncentrują się bowiem na fazach początkowych, czyli analizie i projektowaniu. Istotną nowością tego podejścia jest możliwość opisania nie tylko statyki systemu, ale również jego dynamiki. Opis zawiera funkcje, zdarzenia i operacje występujące w obiekcie. Podejście obiekt — atrybut — związek zdobywa ostatnio popularność w specyficznej klasie systemów informatycznych, obejmujących tworzenie baz wiedzy, systemów wspomagania decyzji czy systemów doradczych[17] (126).

Podstawą tego podejścia jest teza, że istnieje konieczność analizy danych i potrzeb ich użytkowania niezależnie od wszelkich aspektów związanych z projektowaniem i wdrażaniem systemów. W ten sposób metodyka pozwala na pełne zrozumienie istoty funkcjonowania badanej dziedziny. Punktem wyjścia jest opracowana analiza. Analityk nie jest ograniczany przez koncentrowanie się na postaci zestawień wynikowych, lecz rozpatruje dane i funkcje badanej dziedziny jako całość.

Konsekwentnie do zarysowanych celów i założeń, w ramach tego podejścia wykonuje się dwa rodzaje czynności:

— analizę danych (data analysis);
— analizę funkcjonalną (functional analysis).

Analiza danych określa, co jest niezbędne w kategoriach typów rzeczy dla istnienia dziedziny przedmiotowej, na przykład pracownicy, materiały, dostawcy, odbiorcy. Analiza funkcjonalna służy do definiowania funkcji, które decydują o działaniu i rozwoju obiektów.

Podstawowe kategorie pojęciowe to: obiekt, atrybut, związek. Obiektem jest jednoznacznie identyfikowany składnik badanej rzeczywistości, o którym informacja jest lub może być zbierana i przechowywana. Obiekty o wspólnych cechach są łączone w typy obiektów. Wyróżnia się typy rzeczywiste i abstrakcyjne. Atrybut jest cechą, elementem charakteryzującym obiekty i relacje

[17] Expert system — angielska nazwa pewnej klasy systemu, która w Polsce jest tłumaczona jako system ekspertowy lub ekspercki. Tutaj przyjmuje się tłumaczenie lansowane przez Polskie Towarzystwo Informatyczne (patrz (9)) — system doradczy.

w badanej dziedzinie. Związek lub relacja to powiązanie między obiektami w badanej dziedzinie.

Oprócz analizy danych równolegle jest przeprowadzana analiza funkcji, której celem jest opisanie aspektów dynamicznych systemu. Wyodrębniane i opisywane są podstawowe działania występujące w danym obiekcie. Funkcja oznacza tu czynność, w wyniku której następuje zmiana dotychczasowego stanu na nowy.

Podejście obiekt — atrybut — związek narzuca określony sposób dokumentowania modelu systemu. Modelowanie rozpoczyna się od opisu badanej dziedziny z uwzględnieniem jej statyki i dynamiki, a następnie równolegle realizuje się procesy analizy danych i analizy funkcji. Ostateczna wersja modelu jest rezultatem kolejnych iteracji w ramach zarysowanego procesu. Działanie to jest procesem twórczym, który przestrzega kryteria oceny jakości systemu.

Dokumentowanie poszczególnych elementów systemu wykorzystuje różnorodne metody i techniki formularzy, tablic, schematów, diagramów, macierzy. Dokumentacja modelu procesów jest bardziej złożona i obejmuje wiele wzajemnie powiązanych składników. Do podstawowych zalet opisanego podejścia należy:

— opracowanie spójnego, praktycznie użytecznego podejścia do modelowania;
— strukturyzacja i formalizacja procesu modelowania;
— niezależność analizy dziedziny od technologii informatycznej;
— opracowanie zestawu metod i technik realizacji poszczególnych faz w procesie modelowania;
— możliwość adaptacji prognostycznego podejścia do konstruowania systemów.

Do wad tego podejścia można zaliczyć:

— obszerny zakres niezbędnej dokumentacji i trudności z jej prowadzeniem metodami tradycyjnymi;
— rozmyte kryteria oceny jakości tworzonych modeli systemu;
— intuicyjność wstępnej fazy modelowania.

Przedstawione dwa podejścia są reprezentatywne dla dużej klasy metodyk, których opisy można znaleźć w literaturze. W większości metodyki te mają charakter akademicki, a tylko nieliczne doczekały się profesjonalnych zastosowań, wzbogacając pierwowzór w kolejnych implementacjach. Powstające w środowisku akademickim metodyki są raczej koncepcjami o wyraźnych podstawach teoretycznych albo co najwyżej są wdrażane eksperymentalnie. W pracy (148)

można znaleźć ciekawe porównawcze zestawienie piętnastu metodyk z punktu widzenia kilku kryteriów.

1.5.2.2. Analiza i projektowanie strukturalne

Podejście strukturalne, które pozwoliło osiągnąć pozytywne efekty po zastosowaniu w pracach programowych, szybko zostało adaptowane jako metoda postępowania w innych fazach tworzenia systemu informatycznego, zwłaszcza do analizy i projektowania. Strukturalna analiza systemów integruje projekt danych z projektem algorytmów (procesów). Podejście to zostało wypracowane w organizacji Yourdon Inc. przez Yourdona, De Marco (23) oraz Gane'a i Sarsona (45).

Inna implementacja podejścia strukturalnego to metodyka SADT (Structured Analysis and Design Technique). Autorem tej metodyki jest D. T. Rossa, który rozwijał ją twórczo w ramach prac firmy SofTech (114).

Podstawą analizy strukturalnej jest konstrukcja graficznego, logicznego modelu systemu. Model ten wraz z ustalonymi celami systemu i jego ograniczeniami tworzy właściwą definicję potrzeb przyszłego użytkownika. Podstawową techniką opisu związków między danymi i procesami w analizie strukturalnej są diagramy przepływu danych. Przez analizę kolejnych procesów i zbiorów danych oraz dostosowanie przepływów danych do tych zmian następuje uszczegółowienie modelu i coraz precyzyjniejszy opis dynamiki systemu.

Diagramy przepływu danych umożliwiają jedynie nazwanie procesów, zbiorów danych i przepływów danych. W pracach analitycznych konieczna jest głębsza charakterystyka poszczególnych elementów danych. Do tego celu wykorzystywane są słowniki danych, w których jest opisywana struktura danych. Do opisu wzajemnych związków logicznych w procesach wykorzystywane są takie techniki decyzyjne, jak: tablice decyzyjne, język strukturalny, drzewa decyzyjne.

Metodyka SADT składa się z dwu zasadniczych części:

— diagramowego języka analizy strukturalnej, wykorzystującego bloki i łuki;
— techniki projektowania polegającej na uporządkowanym sposobie działania z efektywnym wykorzystaniem języka.

Metodyka SADT jest szczególnie efektywna we wczesnych i końcowych fazach cyklu życia systemu, takich jak: analiza, definicja potrzeb, specyfikacja funkcjonalna, konfigurowanie sprzętu, oprogramowanie systemu, procedury operacyjne, programowanie szkolenia użytkownika, kierowanie i ocena projektu. Rzadziej jest natomiast stosowana do projektowania technicznego.

Metodyka została stworzona przede wszystkim do realizacji analizy, czyli definiowania potrzeb inicjujących cykl tworzenia systemu. W ramach prac analitycznych precyzyjnie definiowane są potrzeby użytkownika i cele, jakie ma realizować tworzony system. Należy więc odpowiedzieć na pytanie: dlaczego system informatyczny jest niezbędny i jakie jego cechy będą realizowały stawiane przed nim zadania. Przedstawiany jest również pierwszy kształt systemu, czyli sposób jego konstrukcji.

Definiowanie potrzeb, według D. T. Rossa (114), powinno obejmować następujące trzy dziedziny:

— analizę kontekstu, czyli przyczyn, dla których system ma być stworzony;
— specyfikację funkcjonalną;
— ograniczenia projektu.

Do realizacji tych celów służą techniki graficzne prezentacji. Język graficzny to ograniczony zestaw podstawowych symboli, z których analityk może tworzyć dowolne uporządkowane struktury. Są to bloki i łuki. Blok reprezentuje część całości, a łuk stanowi połączenie między częściami. Diagramy są wzbogacane opisem w języku naturalnym, co zwiększa przejrzystość i czytelność opisu. Model systemu jest zorganizowaną kolekcją diagramów z odpowiednim opisem.

Język analizy strukturalnej umożliwia graficzne wyrażenie potrzeby przetwarzania danych. Podstawową funkcją języka jest powiązanie, strukturyzacja i komunikowanie się wyrażone naturalnym dla użytkownika językiem. Analiza strukturalna umożliwia strukturalną dekompozycję, która pozwala na późniejszą realizację strukturalnej syntezy.

Podstawą dobrego rozumienia problemu i przebiegu strumieni komunikowania jest w tej metodyce hierarchiczna dekompozycja. Najważniejszym elementem realizacji projektu jest łączenie części w całość. Ważny jest również punkt widzenia użytkownika i jego sposób ustalania celów. Dokonuje się zstępującej dekompozycji przez kolejne warstwy, a pełny zbiór diagramów tworzy projekt systemu. Diagramy są cały czas korygowane i uzgadniane z przyszłym użytkownikiem.

Określa się siedem podstawowych zasad metodyki SADT:

— złożone problemy są najlepiej rozwiązywane przez tworzenie modelu, który wyraża ten problem;
— analiza problemu winna być zstępująca, modularna, hierarchiczna i strukturalna;
— model jest opisywany za pomocą techniki diagramowej,
— w modelu muszą być reprezentowane zarówno rzeczy, jak i zdarzenia;

— należy rozróżniać różne poziomy szczegółowości modeli;
— metody analizy powinny wspierać uporządkowaną pracę zespołową;
— wszystkie decyzje i komentarze muszą mieć postać pisaną i być dostępne dla wszystkich członków zespołu.

Analiza wielu metodyk w zderzeniu z doświadczeniami praktyki wytwarzania różnych systemów informatycznych skłania do kaskadowego modelu realizacji, wzbogaconego o elementy wspomagania procesu tworzenia przez techniki tworzenia systemów informatycznych. Wymienione techniki wspomagają poszczególne fazy realizacji systemu i mogą być stosowane przez twórców w zależności od specyfiki rozwiązywanego problemu i upodobań autorów. Część z tych technik ma odpowiednie narzędzia komputerowego wspomagania, dzięki temu są szczególnie atrakcyjne. Techniki te wspomagają określone czynności, które należy wykonać w procesie tworzenia systemu informatycznego, a zwłaszcza:

a) modelowanie przedsiębiorstwa i jego działań, opis hierarchicznie zdekomponowanych funkcji i procesów, ogólnego przepływu danych między zdarzeniami i procesami, składników systemu i związków pomiędzy nimi;
b) architekturę programów;
c) szczegółową logikę programów;
d) tworzenie struktur zbiorów;
e) tworzenie struktur baz danych.

Uwzględniając tę klasyfikację, S. Wrycza (147) zestawia najpopularniejsze techniki i pokazuje jak wspomagają różne czynności wykonywane w procesie tworzenia systemu informatycznego.

Z tabeli 1.1 wynika, że wymienione techniki mogą być stosowane w różnych fazach realizacji systemu informatycznego. Wybór odpowiedniej techniki zależy od specyfiki rozwiązywanego problemu i tylko częściowo od przyjętej metodyki postępowania. W przedstawionej dalej metodyce punktów węzłowych również mogą mieć zastosowanie.

Tabela 1.1

Porównanie technik tworzenia systemów informatycznych (147)

Lp.	Nazwa techniki	Kategoria czynności
1.	Diagramy przepływu danych (Youdona/DeMarco lub Gane'a Sarsona)	a
2.	Diagramy dekompozycji funkcjonalnej	a
3.	Schematy struktury	a b
4.	Diagramy HIPO	a b c
5.	Diagramy Warniera/Orra	a b c d
6.	Diagramy Jacksona	b c d
7.	Schematy blokowe	c
8.	Język strukturalny i pseudokod	c
9.	Diagramy Nassi/Schneidermana	c
10.	Diagramy działań	a b c d
11.	Drzewa decyzyjne	c
12.	Tablice decyzyjne	c
13.	Diagramy struktur danych	e
14.	Diagramy obiekt-atrybut-związek	e
15.	Diagramy nawigacji danych	b e
16.	Diagramy HOS	a b c d

Tabela 1.1

1.	Diagramy przepływu danych
2.	Słowniki danych
3.	Schematy blokowe
4.	Drzewa decyzyjne
5.	Tablice decyzyjne
6.	Diagramy stanów obiektu
7.	Diagramy hierarchii obiektów
8.	Diagramy współpracy obiektów
9.	Diagramy przypadków użycia
10.	Diagramy czynności
11.	Drzewa decyzyjne
12.	Tablice decyzyjne
13.	Diagramy stanów danych
14.	Diagramy obiektów
15.	Diagramy powiązań danych
16.	Diagramy HOS

2. Metodyka punktów węzłowych

2.1. Ogólne założenia metodyki

Wytworzenie systemu informatycznego zarządzania jest procesem twórczym. Wymaga inwencji od autorów, którzy nadają dziełu, czyli, systemowi osobiste piętno. Oprócz inwencji wymaga się od nich głębokiej znajomości informatycznego rzemiosła. Autorzy systemu muszą wykonać wiele różnorodnych czynności, które decydują o powodzeniu całego działania. Zakres wykonywanych prac oraz ich waga dla całego projektu będą zależały od konkretnie wytwarzanego projektu. Czynności realizowanego procesu są wzajemnie powiązane w ciągi wyznaczające następstwo wykonywania. Zakres prac oraz wzajemne uwarunkowania konkretnych czynności to jeden z elementów informatycznego rzemiosła.

Autor systemu musi zatem wiedzieć, **co?** i w jakiej kolejności musi wykonać. Musi więc znać pełną listę czynności, które należy wykonać, oraz wzajemne związki między tymi czynnościami. Odrębnym problemem jest, **jak?** będą te czynności wykonywane. Odpowiedni dobór metod i technik wykonania czy twórcza implementacja metod wnosi osobiste piętno autora.

Przystępując do prac nad systemem informatycznym zarządzania, należy mieć pełną świadomość **co?** trzeba wykonać i w jakiej kolejności czynności te powinny być wykonywane. Mogą wystąpić pewne niewielkie odchylenia od ogólnego modelu procesu projektowania systemu, zmienne też będzie akcentowanie oraz szczegółowość realizowanych czynności w zależności od konkretnego typu systemu i warunków realizacyjnych. W szczególnych przypadkach można zrezygnować z wykonania pewnych czynności lub wykonać je

w nieco zmienionym porządku, ale zawsze musi to być świadome, przemyślane działanie projektanta, znającego konsekwencje wynikające z tej modyfikacji.

W dalszych rozważaniach zostanie zarysowany typowy model procesu tworzenia systemu informatycznego zarządzania. Model ten ma na celu zdefiniowanie pełnej listy czynności, które należy wykonać w całym cyklu tworzenia systemu informatycznego, oraz ich powiązanie relacjami porządkującymi w sensie następstwa. Dla konkretnego systemu informatycznego tak zdefiniowana lista może być nadmiarowa, ale projektantowi pozostawia się wybór odpowiedniego sposobu realizacji i decyzje o pominięciu, szczątkowym wykonaniu lub dokładnym realizowaniu wymienionych i powiązanych relacjami następstwa konkretnych, zdefiniowanych czynności.

Jeszcze raz należy podkreślić, że cała uwaga koncentruje się tu nie na tym, jak należy wykonywać konkretne prace składające się na wytworzenie systemu informatycznego, ale na tym, jakie czynności i w jakiej kolejności powinny być wykonane w całym procesie tworzenia systemu informatycznego. Przez projekt systemu rozumie się cały cykl życia systemu — od momentu powstania potrzeby rozwiązania problemu za pomocą systemu informatycznego, aż do eksploatacji użytkowej wykonanego systemu.

2.2. Punkty węzłowe procesu tworzenia systemu informatycznego

Doświadczenia praktyczne zdobyte w wyniku projektowania, programowania i wdrażania wielu różnorodnych systemów informatycznych zarządzania oraz analiza literatury są podstawą do zdefiniowania listy czynności składających się na proces tworzenia systemu. Z doświadczenia wiadomo, że każdy konkretny system ma swoją specyfikę, ale można określić pewien standardowy układ czynności, które są najczęściej wykonywane w procesie twórczym.

Opracowana lista czynności do wykonania została uzupełniona relacjami określającymi, jakie czynności warunkują wykonanie innej czynności oraz wykonanie jakiej czynności umożliwia rozpoczęcie dalszych czynności. Czyn-

ności z listy zostały więc powiązane w sieć działań. Każda z nich została zdefiniowana co do celu i zakresu wykonywanych w jej ramach prac.

Na potrzeby opracowania zdefiniowano pojęcie **punktu węzłowego.** Punkt węzłowy to udokumentowany wynik prac wykonanych w procesie tworzenia systemu informatycznego zarządzania.

Oddzielnym problemem w procesie tworzenia systemu informatycznego jest dokumentacja. Dokumentacja to nośnik komunikowania się między zespołami wykonującymi projekt, poszczególnymi członkami zespołów oraz zapis przyjętych rozwiązań i uzasadnienie ich przyjęcia na potrzeby prowadzonych w przyszłości prac rozwojowych i pielęgnacyjnych systemu. Pomija się tu formę i szatę graficzną dokumentacji, ale zwraca uwagę na zawartość informacyjną dokumentacji prowadzonych prac. Sprawy szaty graficznej, sposobu wykonania, technik szczegółowych utrzymania dokumentacji w aktualnym stanie pozostawia się do rozstrzygnięcia projektantowi systemu. Z punktu widzenia prezentowanego modelu systemu istotna jest tylko i wyłącznie zawartość informacyjna dokumentacji, która powinna odpowiadać zakresowi prac poszczególnych punktów węzłowych. Punkt węzłowy jest więc czynnością lub grupą czynności, które muszą być wykonane w procesie tworzenia systemu informatycznego zarządzania. Wyróżnia się dwa poziomy szczegółowości punktów węzłowych.

Pierwszy poziom szczegółowości to grupa czynności powiązanych tematycznie, które są wykonywane w miarę spójnie, przy czym ich wykonanie jest zależne od realizacji innych punktów węzłowych. Czynności wchodzące w skład punktu węzłowego pierwszego poziomu warunkują wykonanie innych punktów tego samego poziomu i to najczęściej sąsiednich. Punkty węzłowe pierwszego poziomu mogą tworzyć odrębną całość dokumentacyjną i być elementem oceny przyszłego użytkownika. Ocena taka, połączona z formalnym odbiorem, stanowi punkt kontrolny decydujący o kształcie projektowanego systemu lub o zaniechaniu dalszych prac nad projektem. W niżej podanej specyfikacji punktów węzłowych są one oznaczone symbolami z zerem jako ostatni znak numeru punktu węzłowego.

Drugi poziom szczegółowości, to czynności, jakie składają się na punkt węzłowy pierwszego poziomu. Kryterium wydzielenia czynności drugiego poziomu nie jest jednoznaczne dla wszystkich punktów węzłowych pierwszego poziomu. Będą to czynności wykonywane przez jedną grupę osób w procesie projektowania lub grupa czynności, która warunkuje dalsze prace nad tym samym punktem węzłowym pierwszego poziomu. Podany podział jest propozycją wynikającą z doświadczeń praktycznych. Numeracja nadana punktom

węzłowym nie ma charakteru porządkującego, lecz jedynie służy do identyfikacji.

Porządek wykonania poszczególnych punktów węzłowych jest wyznaczony oddzielnie. Każdy punkt węzłowy jest definiowany przez szczegółowy zakres prac, jakie muszą być wykonane i udokumentowane, oraz listę punktów węzłowych, które warunkują wykonanie tych prac. Ponadto dołączana jest lista punktów węzłowych. Wyniki prac danego punktu mogą i powinny być wykorzystane w procesie dalszego tworzenia systemu. Lista poprzedników i następników, wyspecyfikowana dla każdego punktu węzłowego, wyznacza jego miejsce w sieci działań w całym procesie tworzenia systemu.

Zakres prac niektórych punktów węzłowych drugiego poziomu jest tak szeroki, że można byłoby dokonać dalszej dekompozycji tego punktu, wydzielając hierarchicznie trzeci poziom i dalsze. Decyzje tego typu pozostawia się projektantowi konkretnego systemu. Opisany model stanowi szkielet metodyki projektowania, który można w poszczególnych przypadkach modyfikować, rozszerzać, uzupełniać w tych miejscach, które są szczególnie istotne lub decydują o specyfice realizowanego systemu.

Pełna lista punktów węzłowych pierwszego i drugiego poziomu, definiująca proponowany model metodyki projektowania systemów informatycznych zarządzania, przedstawia się następująco:

010 zgłoszenie potrzeby
020 analiza problemu
  021 badanie możliwości i sposobu rozwiązania problemu
  022 analiza wariantów rozwiązań i ofert
  023 ocena opłacalności
  024 możliwości realizacji
030 zdefiniowanie zadania projektowego
  031 uwarunkowania
  032 główne funkcje systemu
  033 powiązania z innymi systemami
  034 zakres prac projektowych
  035 umowa
040 opis istniejącego stanu
  041 charakterystyka stanu organizacji
  042 charakterystyka danych (dokumentów, kartotek)
  043 strumienie informacyjne
  044 opis algorytmów

050 analiza istniejącego stanu
- 051 analiza stanu organizacji
- 052 analiza danych
- 053 analiza strumieni informacyjnych

060 koncepcja systemu informatycznego
- 061 zakres systemu (struktura funkcjonalna)
- 062 koncepcja struktury informacyjnej
- 063 koncepcja struktury technicznej
- 064 koncepcja struktury przestrzennej
- 065 obraz funkcjonowania systemu
- 066 założenia organizacyjne systemu
- 067 założenia ekonomiczne systemu

070 normatywy
- 071 normatywy komunikowania się z innymi systemami
- 072 zasady budowy nazw w systemie
- 073 normatywy wydawnictw i wprowadzania danych
- 074 normatyw dokumentowania
- 075 normatywy technologii przetwarzania
- 076 normatywy programowania
- 077 protokoły transmisji
- 078 normatywy identyfikacji zbiorów danych

080 projekt struktury informacyjnej
- 081 projekt wyjść
- 082 projekt systemu kodów
- 083 projekt wejść
- 084 struktura logiczna danych
- 085 projekt zbiorów podstawowych

090 projekt technologii przetwarzania
- 091 wybór technologii
- 092 podział systemu na jednostki technologiczne
- 093 projekt schematów przetwarzania
- 094 projekt zbiorów roboczych
- 095 projekt struktury przestrzennej
- 096 projekt struktury technicznej

100 synteza projektu
- 101 synteza struktur systemu

102 organizacja funkcjonowania systemu oraz koszty jego eksploatacji
103 projekt zmian w organizacji obiektu i wymagania wdrożeniowe
104 harmonogram i preliminarz prac programowo-wdrożeniowych
105 koszty i efekty
110 projekt oprogramowania
111 założenia ogólne oprogramowania systemu
112 założenia do programów
113 specyfikacja algorytmów
114 zdefiniowanie struktur danych
115 projekt modułów oprogramowania
116 projekt technologii produkcji oprogramowania
120 oprogramowanie — wykonanie
121 wykonanie modułów wspólnych
122 wykonanie opisu struktur danych (w języku programowania)
123 wykonanie programów
124 testowanie pojedynczych modułów i programów
125 testowanie całego systemu oprogramowania
126 skompletowanie dokumentacji programów
130 organizacyjne przygotowanie obiektu
131 pozyskanie zasobów
132 pracochłonność i koszty wdrożenia systemu
133 pracochłonność i koszty eksloatacji
134 przygotowanie kadr
135 baza normatywna systemu — opracowanie
136 podstawy prawne wdrożenia i eksploatacji
140 eksperyment na danych modelowych
141 opracowanie eksperymentu
142 wykonanie eksperymentu
143 analiza wyników eksperymentu
144 wnioski wypływające z eksperymentu
150 instalacja systemu
151 zawarcie umowy o eksploatacje
152 biblioteki programów
153 zbiory kartotekowe
154 inicjacja taśm, dysków, kartotek
155 implementacja technologii w konkretnym ośrodku

160 wdrożenie pilotowe
161 wybór metody wdrażania
162 opracowanie kryteriów oceny
163 eksploatacja pilotowa systemu
164 ocena wyników eksploatacji
165 rozpowszechnienie systemu
170 projekt eksploatacyjny
171 aktualna dokumentacja systemu
172 instrukcja wypełniania i obiegu dokumentów
173 instrukcja tworzenia maszynowych nośników
174 instrukcja kompletacji i kontroli danych wejściowych
175 instrukcja operowania systemem — harmonogram
176 instrukcja kompletacji i kontroli danych wyjściowych
177 instrukcje instalacyjne
180 eksploatacja i rozwój

Lista punktów węzłowych drugiego poziomu, składających się na punkt węzłowy pierwszego poziomu, daje obraz, jakie prace są wykonywane w ramach każdego z wyszczególnionych punktów. Brak miejsca uniemożliwia szczegółowy opis każdego punktu węzłowego z podanej listy. W dalszej części zostaną opisane punkty węzłowe pierwszego poziomu, podany podstawowy zakres prac oraz zawartość informacyjna tworzonej dokumentacji, która jest podstawą dalszych prac. Najczęściej punkty węzłowe pierwszego poziomu są wykonywane zgodnie z rosnącymi numerami tych punktów, ale o uwarunkowaniach wykonawczych każdego punktu może decydować specyfika realizowanego systemu.

Przedstawiona lista punktów węzłowych specyfikuje i grupuje czynności, które są wykonywane w procesie tworzenia systemu informatycznego. Istotną nowością jest grupa 070, mająca na celu zdefiniowanie normatywów i standardów, które powinny obowiązywać w toku dalszych prac.

010 — zgłoszenie potrzeby

Indywidualny wniosek, inicjatywa organizacji, grupy pracowników lub każdy inny impuls wskazujący na potrzebę zastosowania narzędzi informatycznych dla usprawnienia procesu zarządzania obiektem, to początek procesu projektowego. Rozpoczyna się długi i złożony proces tworzenia systemu informatycznego, który nie zawsze dojdzie aż do ostatniego punktu węzłowego — 180.

Jeżeli zgłoszona potrzeba otrzyma akceptację kierownictwa obiektu (w którym ma być wdrażany system), formalnie rozpoczyna się prace nad budową systemu informatycznego. W tym punkcie następuje wstępne zdefiniowanie celu i zakresu systemu oraz przygotowanie dokumentacji wymaganej w danej jednostce organizacyjnej, umożliwiające rozpoczęcie prac lub wygenerowanie formalnego zlecenia dla zespołu zewnętrznego ewentualnie sprecyzowanie treści ogłoszenia o przetargu na wykonanie systemu. Inicjujący rozpoczęcie prac powinien przygotować dokument zawierający następujące informacje:

— wstępne określenie celu i zakresu systemu;
— jakie komórki organizacyjne obiektu powinny współpracować przy jego realizacji;
— możliwości korzystania z pomocy ekspertów zewnętrznych;
— propozycje powołania komitetu kierującego przebiegiem prac.

Zawartość informacyjna dokumentacji to:

1. Wstępne określenie celu i zakresu:
   a) wykaz funkcji, które w ramach funkcji systemu zarządzania powinien realizować system informatyczny;
   b) ograniczenia i uwarunkowania, które zgłaszającemu potrzebę są znane, na przykład:
   — wymagana współpraca z innymi systemami,
   — dostępny lub posiadany sprzęt,
   — obowiązujące przepisy dotyczące funkcji, które będą realizowane przez system,
   — inne uwagi.
2. Wykaz komórek organizacyjnych lub osób, które powinny współpracować przy projektowaniu i wdrażaniu systemu.
3. Wykaz osób (organizacji, ekspertów z zewnątrz), które będą współpracować przy definiowaniu zadania projektowego.

Z punktu widzenia tworzenia systemu informatycznego nie jest istotne, jakie okoliczności spowodowały zgłoszenie potrzeby projektowania systemu informatycznego. Ten punkt węzłowy jest początkiem dalszych prac, można więc przyjąć, że nie występują żadne poprzedniki dla punktu 010.

Udokumentowane zgłoszenie potrzeby projektowania systemu informatycznego otwiera szeroki front różnorodnych prac przygotowawczych i szczegółowych, które wchodzą w skład dalszych punktów węzłowych. W naturalny sposób należałoby rozpocząć wykonywanie punktu węzłowego 020

(analiza problemu), ale jednocześnie można rozpocząć prace nad punktem 040 (opis istniejącego stanu).

020 — analiza problemu

Zgłoszona potrzeba wymaga przeprowadzenia wstępnych badań i analiz, których celem jest określenie możliwości realizacji i dostarczenie danych dla zdefiniowania zadania projektowego.

Prace wykonywane w ramach analizy problemu mają za zadanie dostarczenie danych do zdefiniowania zadania projektowego. Jest to analiza wstępna, pobieżna, zakończona opisem ilościowym problemów, które należy rozwiązać w następnych etapach procesu projektowania i wdrażania systemu informatycznego zarządzania. Zakres prac wykonywanych w ramach tego punktu może być różny. W przypadku, gdy zdefiniowanie zadania projektowego nie jest możliwe na podstawie wstępnej analizy, należy przeprowadzić głęboką analizę zgodnie z punktem węzłowym 040 i 050 i na tej podstawie definiować zakres systemu oraz prac projektowych.

Należy przeanalizować wszystkie dostępne sposoby realizacji i dokonać odpowiedniego wyboru. Wybór ten powinien być oparty na głębokiej analizie ofert oraz rachunku ekonomicznej opłacalności. Może się bowiem okazać, że problem, który zainicjował te prace, daje się rozwiązać w inny, lepszy sposób niż przez system informatyczny.

Zawartość informacyjna dokumentacji powinna odzwierciedlać kierunki prowadzonych analiz, materiały źródłowe oraz wnioski z uzasadnieniem podjętych decyzji czy przyjętych rozwiązań.

Zgłoszona w punkcie węzłowym 010 potrzeba inicjuje prace tego punktu węzłowego. Wyniki analizy problemu pozwalają na podjęcie wiążących decyzji co do dalszych prac nad systemem. Przede wszystkim zgromadzony jest wystarczający materiał, aby przystąpić do pełnego zdefiniowania zadania projektowego.

030 — zdefiniowanie zadania projektowego

Zdefiniowanie zadania projektowego ma na celu jasne i jednoznaczne określenie zakresu systemu i prac projektowych. Definiując zadanie projektowe, należy uwzględnić następujące kryteria:

— realne warunki wdrożenia systemu, traktowane jako kryterium nadrzędne;
— posiadane lub przewidziane środki realizacji — techniczne, kadrowe i finansowe;

— poziom nauki w dziedzinie zastosowań informatyki;
— aktualny stopień organizacyjnego przygotowania użytkownika do wdrażania informatyki;
— dopuszczalny stopień zmian w organizacji obiektu.

Prawidłowe zdefiniowanie zadania projektowego stwarza zespołowi realizatorów warunki do zaprojektowania i wdrożenia systemu jakościowo dobrego i użytecznego. Znane są przypadki, gdy wskutek niewłaściwego (niepełnego, niejednoznacznego) sformułowania zadań praca projektantów okazała się nieefektywna.

Należy uwzględnić wszystkie uwarunkowania mające wpływ na kształt systemu i proces jego realizacji. Będą to uwarunkowania organizacyjne, sprzętowe, kadrowe, finansowe itp. Ponadto należy wyspecyfikować uwarunkowania prawne i oddziaływanie otoczenia mające wpływ na projektowany system.

Główne funkcje systemu zostały ogólnikowo zdefiniowane w momencie zgłaszania potrzeby. Analiza problemu (020) pogłębiła te badania, co daje materiał do uszczegółowienia i jednoznacznego zdefiniowania zadania projektowego. W przypadku dużych przedsiębiorstw o skomplikowanej strukturze organizacyjnej, określenie głównych funkcji systemu może wymagać bardzo szczegółowych analiz i badań; jest to często samodzielne zadanie badawczo-projektowe.

Projektowany system musi być niekiedy wkomponowany w inne funkcjonujące lub przewidywane zastosowania informatyki, co zmusza do precyzyjnego określenia związków i powiązań z tymi systemami informatycznymi.

Prace projektowe to czynności mające na celu określenie ogólnych zasad realizacji projektu, odpowiedzialności, zakresu współdziałania oraz oszacowanie pracochłonności. Wszystkie te czynności powinny doprowadzić do precyzyjnego zdefiniowania zadania projektowego, które ma być realizowane w dalszych fazach. Gdy prace projektowe nie będą wykonywane przez własny zespół projektujący, zgromadzone materiały pozwalają na zawarcie umowy o realizację systemu.

Zdefiniowanie zadania projektowego w wielu przypadkach nie wymaga aż tak szczegółowych prac, zależy to od konkretnej sytuacji i doświadczenia projektującego system. Opisana procedura stanowi przewodnik postępowania, gdy brakuje doświadczeń lub problem jest bardzo duży i skomplikowany, a wartość i zakres dalszych prac wymaga optymalnych decyzji już na wstępie.

Można również przyjąć, że dopiero zdefiniowanie zadania projektowego rozpoczyna zasadnicze prace projektowe. Niezależnie od sposobu realizacji

omówionych dotychczas prac, dopiero zdefiniowanie zadania projektowego inicjuje dalsze prace projektowe. Mogą one być realizowane wielowariantowo w zależności od specyfiki obiektu. Przykładowo, w obiekcie, który jest dopiero na etapie projektu, punkty węzłowe 040 i 050 zostają pominięte z naturalnych przyczyn lub przyjmują szczątkową postać. W funkcjonujących obiektach wykonanie badań istniejącego stanu wydaje się niezbędne.

040 — opis stanu istniejącego

Opis istniejącego stanu obejmuje zbieranie, systematyzowanie oraz logiczne powiązania faktów dotyczących obszarów planowanego działania systemu i otoczenia, w którym ma funkcjonować system. W ramach opisu istniejącego stanu należy wykonać:
— charakterystykę stanu organizacji;
— charakterystykę danych (dokumentów, kartotek);
— opis strumieni informacyjnych;
— opis algorytmów.

Wykonane badania i opisy faktycznego stanu mają na celu stworzenie obrazu funkcjonowania obiektu bez systemu informatycznego. Charakterystyka stanu organizacji i opis strumieni informacyjnych nie powinny się ograniczać do analizy dokumentów, ale dążyć do odwzorowania faktycznego stanu, nawet jeśli jest on różny od postulowanego czy zadekretowanego przepisami. Pozwoli to na poznanie powiązań nieformalnych, błędnych rozwiązań organizacyjnych, które wspomagane środkami informatyki nie mogłyby niczego usprawnić z uwagi na wadliwą organizację służb, stanowisk czy przepływ informacji.

Specyfikacja faktycznie użytkowanych danych, ich obieg, wzory graficzne, poddane wnikliwej analizie na dalszych etapach prac, są podstawą projektowania danych w systemie informatycznym.

Zawartość informacyjna dokumentacji punktu węzłowego, opisu istniejącego stanu jest trudna do jednoznacznego określenia. Raczej należy przyjąć założenie, że może być ona różna w zależności, między innymi, od kwalifikacji zespołu projektantów, stopnia zdefiniowania problemów, celów systemów. Faktem jest natomiast, że opis istniejącego stanu nie powinien się rozrastać do nadmiernych rozmiarów. Projektant przed przystąpieniem do pracy winien więc mieć listę głównych problemów, które wymagają opisu. Lista ta w miarę postępu prac może być uszczegółowiona.

Prace realizowane w ramach tego punktu węzłowego nie są w zasadzie uwarunkowane wykonaniem wcześniej opisanych punktów węzłowych. Me-

teriał wcześniej zgromadzony może jedynie ukierunkować prowadzone badania i wpływać na większą lub mniejszą szczegółowość konkretnych czynności. Naturalne jest, że analiza opisanego stanu, czyli punkt węzłowy 050, może być wykonana po zakończeniu prac tego punktu.

050 — analiza istniejącego stanu

Analiza istniejącego stanu ma na celu uchwycenie zależności funkcjonalnych między funkcjami systemu i ukierunkowanie dalszych prac projektowych. Jest ona podstawą tematycznego ukierunkowania dalszych prac projektowych i zastosowanych rozwiązań projektowych. Powinna przede wszystkim uchwycić zależności funkcjonalne między podstawowymi funkcjami danego obszaru działania.

Prace analityczne powinny charakteryzować się następującymi cechami:

— funkcjonalnym podejściem do problemu; o ile opis istniejącego stanu jest wykonywany w obrębie poszczególnych dziedzin składających się na obszar działania systemu, o tyle analiza istniejącego stanu winna zajmować się problemami powiązań poziomych, a zatem zależnościami występującymi pomiędzy przedsięwzięciami składającymi się w obszarze działania systemu na funkcje wyższego rzędu;

— stopniowym przechodzeniem od problemów ogólnych do rozwiązań szczegółowych.

Przed przystąpieniem do budowy koncepcji systemu na podstawie opisu i analizy istniejącego stanu należy przeprowadzić także rozpoznanie już wdrożonych rozwiązań krajowych i zagranicznych. Analiza ta winna dostarczyć dane do uszczegółowienia definicji zadania projektowego i dalszego zakresu prac projektowych. W pierwszym przypadku, gdy są możliwości wdrożenia istniejących rozwiązań, zadanie projektowe może być rozszerzone lub zawężone do zakresu proponowanego rozwiązania.

W drugim przypadku, jeżeli projektowany system ma uwzględnić istniejące rozwiązania cząstkowe, może się zasadniczo zmienić zakres prac projektowych. Generalnie chodzi o to, aby przez adaptację istniejących rozwiązań przyspieszać proces wdrożenia informatyki w przedsiębiorstwie i obniżać związane z tym koszty.

Zawartość informacyjna dokumentacji powinna wskazywać nie tylko na różnice między wzorcem zarysowanym w definicji zadania projektowego a faktycznym stanem wcześniej zbadanym, ale również na niespójność, sugerować zmiany i modyfikacje pod kątem zastosowań informatyki. Prawidłowe wykonanie analizy jest uwarunkowane wykonaniem badań i opisu istniejącego

stanu (040) oraz dokładnie zdefiniowanym zadaniem projektowym (030). Wyniki prac wykonanych w ramach punktu węzłowego 050 są podstawą zdefiniowania koncepcji systemu informatycznego (060), który gwarantowałby realizację wcześniej określonego zadania.

060 — koncepcja systemu informatycznego

Koncepcja systemu kończy dotychczas prowadzone prace badawcze i analityczne. Zawiera więc wnioski i rozstrzygnięcia, które składają się na obraz docelowej wizji systemu. Nakreślony w koncepcji kształt systemu, po akceptacji przyszłego użytkownika, będzie wykonywany w dalszych szczegółowych krokach. Można powiedzieć, że jest to wzorzec, z którym porównanie ma decydować o odbiorze i podjęciu dalszych prac nad tworzeniem systemu informatycznego.

Przyjmując, że na system informatyczny składają się struktury:
— funkcjonalna,
— informacyjna,
— techniczna,
— przestrzenna,
w koncepcji muszą być zarysowane modele, wzorce tych struktur dla projektowanego systemu.

Struktura funkcjonalna ostatecznie i jednoznacznie określa, jakie funkcje będzie realizował projektowany system. Koncepcja struktury funkcjonalnej zawiera więc przyjęty w wyniku przeprowadzonych badań zakres systemu. Dla całego projektowanego systemu zakreśla obszar danych wynikowych, danych źródłowych, zbiorów danych oraz strumieni informacyjnych wraz ze źródłami ich powstania i miejscami wykorzystania. Dane te wynikają z konieczności obsłużenia zdefiniowanych w strukturze funkcjonalnej potrzeb. Są one podstawą określającą możliwości rozbudowy systemu. Zmodyfikowane lub zupełnie nowe funkcje mogą się pojawić w trakcie prac projektowych lub po rozpoczęciu eksploatacji, a ich wdrożenie jest często uzależnione od zakresu danych przetwarzanych przez system. Gdy te funkcje można obsłużyć danymi z eksploatowanego systemu, modyfikacja najczęściej nie jest zbyt złożona, ale potrzeba przetwarzania zupełnie nowych danych wymaga istotnych modyfikacji projektu. Dane zdefiniowane w koncepcji struktury informacyjnej, wyznaczają również zakres łatwych do wprowadzenia modyfikacji gotowego produktu.

Realizacja funkcji projektowanego systemu, czyli przetwarzanie danych, jest wykonywane za pomocą sprzętu informatycznego. Koncepcja doboru

sprzętu do realizowanych funkcji, typ sprzętu, konfiguracja oraz sposób jego pozyskania, to koncepcja struktury technicznej systemu.

Struktura przestrzenna i jej koncepcja mają dać odpowiedź na pytanie, gdzie dokładnie są zlokalizowane źródła danych wejściowych, punkty wykorzystania danych wynikowych oraz jak przebiegają kanały informacyjne w obiekcie w warunkach pracy systemu informatycznego.

Niejako podsumowaniem koncepcji struktur systemu jest obraz funkcjonowania systemu informatycznego w konkretnym obiekcie, dla którego projektuje się system.

Doprowadzenie do wdrożenia systemu informatycznego to nie tylko wykonanie konkretnych prac projektowych i programowych, ale również różnorodne prace organizacyjne i inwestycyjne w obiekcie. Założenia organizacyjne powinny rozstrzygać, czy system będzie realizowany własnymi siłami, czy zlecony innym zespołom. Czy przyszła eksploatacja będzie się odbywać w zewnętrznym ośrodku obliczeniowym, czy we własnym. Te istotne decyzje powinny wynikać z dokładnego rachunku ekonomicznego, uwzględniającego koszty prac projektowo-programowych, eksploatacji systemu, zakupu sprzętu i innych inwestycji oraz spodziewane efekty wdrożenia systemu.

Zawartość informacyjna dokumentacji koncepcji systemu powinna oprócz opisu struktur, danych do zbilansowania nakładów i efektów, dawać czytelny i kompletny materiał do podjęcia niezbędnych decyzji o prowadzeniu dalszych prac. Materiałem do opracowania koncepcji są wcześniej wykonane badania istniejącego stanu oraz inne zebrane dane. Zaakceptowana koncepcja uruchamia proces realizacji struktur systemu, działań organizacyjnych i innych zapisanych w koncepcji.

070 — normatywy

Prace projektowe i programowanie mają charakter twórczy, ale unifikacja jednych czynności jest wymuszona przez otoczenie, a innych wynika z potrzeby sprawnej realizacji systemu. Przed przystąpieniem do konkretnych prac projektowych należy wyspecyfikować te uwarunkowania i normy obowiązujące przy wykonywaniu dalszych prac. Proces normalizacji czy standaryzacji będzie przede wszystkim dotyczył:

— komunikowania się z innymi systemami;
— budowania nazewnictwa zbiorów, programów i danych w ramach całego systemu;
— postaci wydawnictw i dokumentów wyjścia;
— dokumentowania prac;

— technologii przetwarzania;
— metod i technik programowania;
— protokołów transmisji itp.

Proces standaryzacji tworzenia systemu informatycznego ma na celu wypracowanie uniwersalnych procedur postępowania, tak aby systemy powstały szybko, tanio i były niezawodne w działaniu. Normalizacja funkcji i procedur postępowania nakłada na zespół projektujący rygory, ale przynosi liczne efekty, bezpośrednio zauważalne w procesie tworzenia, a zwłaszcza w procesie modyfikacji i pielęgnacji systemu.

Przestrzeganie nałożonych norm pozwala na tworzenie systemów, które będą łatwe do analizy i zrozumiałe dla przyszłych użytkowników. System będzie pracował w jakimś środowisku programistycznym, przestrzeganie standardów ułatwia więc komunikowanie się i współpracę różnych systemów. Przyjęcie już obowiązujących norm pomaga w procesie wdrażania i adaptacji użytkownika do wymogów systemu informatycznego. Szczególnie istotne jest wypracowanie i przestrzeganie jednolitych standardów dla różnorodnych dokumentów funkcjonujących w różnych systemach informatycznych. Przykładem są standardy wydawnictw, ekranów i inne elementy komunikowania się systemu z użytkownikiem.

Normatywy się tworzy, gdy projektowany system pojawia się pierwszy raz w danym obiekcie, lub przejmuje z innych, wcześniejszych rozwiązań informatycznych i adaptuje na potrzeby projektowanego systemu.

Opracowane w ramach tego punktu normy, standardy i wzory będą obowiązywały we wszystkich dalszych pracach. Wyniki tego punktu dotyczą wszystkich dalszych punktów węzłowych, a często nawet innych systemów, które mają wspólne obiekty działania.

080 — projekt struktury informacyjnej

Ten punkt węzłowy jest rozwinięciem i uszczegółowieniem opracowanej w ramach punktu węzłowego 062 koncepcji struktury informacyjnej. Zgodnie z wymogami sztuki informatycznej dane wejściowe i wyjściowe muszą być odpowiednio opisane, a na potrzeby przetwarzania projektowane są zbiory komputerowe.

Projekt struktury informacyjnej ma na celu zaprojektowanie przepływu zasileń informacyjnych wspomagających realizacje funkcji leżących w obszarze zainteresowania systemu. Zakres prac wykonywanych w ramach projektu struktury informacyjnej obejmuje zaprojektowanie styków między użytkownikiem systemu a ośrodkiem obliczeniowym przez określenie:

— jakie informacje, w jakich układach i w jakich terminach mogą pojawić się na wyjściu systemu;
— bazy normatywnej systemu i zasad kodowania;
— jakie informacje, w jakich układach i w jakich terminach muszą pojawić się na wejściu systemu, aby zapewnić możliwość uzyskania zaprojektowanych wyjść.

W systemach informatycznych zarządzania dane odgrywają kluczową rolę. Prace projektowe wykonane w ramach tego punktu węzłowego mają decydujący wpływ na kształt i jakość projektowanego systemu. Wynikiem pracy systemu są wyjścia w najróżnorodniejszych postaciach, od sygnałów bezpośrednio sterujących przebiegiem procesów, przez zbiory na nośnikach komputerowych, ekrany monitorów, do wydawnictw w formie tabulogramów. Wyjścia zaprojektowane w ramach tego punktu węzłowego muszą zapewnić obsłużenie wszystkich funkcji zdefiniowanych w strukturze funkcjonalnej systemu.

Dla komputerowej realizacji wyjść system informatyczny musi być zasilany danymi wprowadzanymi. Dokładna specyfikacja tych danych to kolejny ważny element prac projektowych. Na potrzeby sprawnej organizacji przetwarzania danych wejściowych w dane wyjściowe przyjmuje się oznaczenia kodowe, identyfikatory, pola specjalne. Rozwiązania te powinny być wewnętrzną sprawą systemu informatycznego i nie dotyczą użytkownika. Przyjęte sposoby realizacji muszą być jednak dokładnie przemyślane i opisane ze szczególnym uwzględnieniem elementów komunikowania się z użytkownikiem systemu. Elementem kontrolnym, niezmiernie istotnym dla poprawności funkcjonowania systemu, jest sprawdzenie, czy dane wejściowe zapewniają wyprowadzenie zaprojektowanych danych wyjściowych.

Z procedury przetwarzania danych i założeń technologii obliczeń będzie wynikała potrzeba szczegółowego opisu zbiorów danych przechowywanych na nośnikach komputerowych. Elementem uzupełniającym szczegółową specyfikację danych wejściowych, wyjściowych i zbiorów jest opis struktury logicznej danych, obrazujący logiczne powiązania między danymi. Powiązania logiczne mogą być podstawowym materiałem do projektowania technologii przetwarzania danych.

Zawartość informacyjna tego punktu węzłowego to szczegółowa specyfikacja wraz ze wzorami i opisami struktury dla każdego występującego w systemie dokumentu wejściowego i wyjściowego, wydawnictwa, zbioru i wszystkich sygnałów przyjmownych przez system i generowanych z systemu.

Materiał do prac nad strukturą informacyjną jest zawarty w dotychczas przeprowadzonych badaniach, a zakres i ostateczny kształt tych prac określa

koncepcja i zdefiniowane normatywy. Wyniki prac tego punktu węzłowego są podstawą budowy technologii przetwarzania. Są też wykorzystywane w wielu innych rozwiązaniach i pracach szczegółowych nad systemem.

090 — projekt technologii przetwarzania

Kierunkowe decyzje w tej dziedzinie są podejmowane na etapie projektowania koncepcji systemu (060). Obecnie chodzi o uściślenie i ostateczny dobór metod i technik przetwarzania danych w obszarze systemu i jego wyodrębnionych składników oraz faz procesu przetwarzania danych. W pierwszym przypadku wybór dotyczy ogólnego modelu procesu technologicznego (na przykład przetwarzanie partiowe, w czasie rzeczywistym); w drugim przypadku chodzi o wybór konkretnych metod i technik technologicznej realizacji poszczególnych faz procesu przetwarzania danych (na przykład techniki ujmowania danych, techniki tworzenia maszynowych nośników danych, techniki przechowywania danych).

Projektowana technologia przetwarzania danych musi oczywiście akceptować określone w punkcie 070 normatywy. Wiele rozwiązań technologicznych jest zdeterminowanych przyjęciem za standard konkretnej wersji systemu operacyjnego czy metody przetwarzania. Prace tego punktu węzłowego polegają na wkomponowaniu projektowanego systemu w przyjęte już założenia ogólne oraz określeniu rozwiązań szczegółowej realizacji kolejnych kroków funkcjonowania systemu.

Podział systemu na jednostki technologiczne może wynikać z logicznej struktury przetwarzanych danych lub z okresów przetwarzania. Wydzielanie jednostek technologicznych, gdy został wybrany standard jakiegoś systemu operacyjnego, to prototyp biblioteki procedur systemu operacyjnego, która będzie wykonana na dalszych etapach prac.

Dla realizacji wymaganego przetwarzania danych często wykorzystywane są zbiory tymczasowe, które są istotne tylko i wyłącznie z punktu widzenia technologii obliczeń, a nie interesują użytkownika. Szczegółowy opis tych zbiorów ma znaczenie dla dalszych prac projektowych i przyszłej pielęgnacji systemu informatycznego.

Realizacja opisanej w ramach tego punktu węzłowego technologii odbywa się w określonej przestrzeni i na określonym sprzęcie komputerowym. Projekt doboru sprzętu komputerowego, gdy nie jest on zdeterminowany istniejącym stanem, oraz odpowiednie rozmieszczenie go w przestrzeni objętej systemem ma zapewnić realizację zaprojektowanej technologii. Dobór sprzętu komputerowego, realizowany w tak odległym od początku procesu projektowa-

nia punkcie węzłowym, jest sytuacją prawidłową, ale praktycznie osiąganą w wyjątkowych wypadkach. W praktyce system jest wykonywany pod określony sprzęt komputerowy, który ma użytkownik systemu. Posiadana konfiguracja sprzętu koputerowego powinna więc być określona w punkcie 070 lub 031, a w projekcie technologii należy się zająć wykorzystaniem sprzętu, jego obciążeniem lub harmonogramem pracy.

Projekt technologii przetwarzania zamyka etap projektowania struktur i jest podstawą prac programowych.

100 — synteza projektu

W procesie inwestycyjnym elementem zamykającym prace projektowe są założenia technologiczno-ekonomiczne (ZTE), które są podstawą prac wykonawczych. Wydaje się, że potrzeba takiego zbiorczego opracowania, zawierającego wszystkie najważniejsze rozstrzygnięcia projektu i uwarunkowania organizacyjno-ekonomiczne, pojawia się również w tworzeniu systemu informatycznego zarządzania. Jest to dokumentacja przeznaczona dla kierownictwa użytkownika systemu. Ma być mu przydatna w podejmowaniu decyzji o dalszych losach systemu, sposobach i tempie realizacji, itp. Akceptacja tej dokumentacji przez kierownictwo obiektu jest podstawą do kontynuowania dalszych prac nad systemem.

Wielowątkowe prace projektowe powodują, że w opracowywanych dokumentacjach i dokumentach znajdują się bardzo różne wersje, często wzajemnie sprzeczne. Po pewnym czasie brakuje jednej wzorcowej dokumentacji, która zawierałaby ostateczne rozstrzygnięcia przekazane do realizacji. Proponowany punkt węzłowy ma na celu stworzenie ostatecznej specyfikacji projektowej, która będzie realizowana w następnych fazach. Dokumentacja winna zawierać:

— syntezę struktur systemu;
— organizację funkcjonowania systemu oraz koszty jego eksploatacji;
— projekt zmian w organizacji obiektu i wymagania wdrożeniowe;
— harmonogram i preliminarz prac programowo-wdrożeniowych;
— koszty i efekty.

Ten punkt węzłowy jest syntezą wszystkich dotychczasowych prac, jego poprzednikami są więc dotychczas wykonywane punkty węzłowe. Opracowana synteza jest materiałem wyjściowym prac realizacyjnych. Dalsze prace programowe i organizacyjne przygotowanie obiektu mają za zadanie wykonanie zaleceń projektowych zapisanych w syntezie projektu systemu.

110 — projekt oprogramowania

Dotychczas wykonywane prace projektowe pozwoliły na przyjęcie ostatecznych rozwiązań. Prace programowe same w sobie są bardzo złożone i trudne w realizacji. Szczegółowe ich zaplanowanie na poziomie rozważań projektowych, określających kształt systemu informatycznego, nie jest możliwe z wielu powodów.

Przed przystąpieniem do konkretnych prac programowych należy na cały system spojrzeć „z góry". Pozwoli to na uchwycenie głównych powiązań między poszczególnymi elementami oprogramowania, określenie ważności poszczególnych programów, co ma istotny wpływ na organizację prac programowych. Prace te mają na celu rozwinięcie prac z punktu 076 i przetłumaczenie znajdujących się tam normatywów na konkretne zalecenia do programowania, oraz stwierdzenie zgodności założeń z normatywami. Wynikiem tych prac projektowych są szczegółowo opisane (najlepiej w notacji łatwo transponowanej na język programowania) algorytmy, struktury danych, założenia programowe dla poszczególnych modułów i programów oraz projekt organizacji i technologii wykonania oprogramowania całego systemu.

Projekt oprogramowania ma za zadanie tak przygotować prace, aby programowanie było realizowane sprawnie i dało niezawodny produkt. Żeby to osiągnąć, niezbędna jest weryfikacja prac projektowych. Wszelkie wątpliwości, niezgodności czy brak szczegółów muszą być zasygnalizowane przed konkretnymi pracami programowymi. W ramach prac projektowania oprogramowania ponownie wykonuje się specyfikację algorytmów i danych z punktu widzenia realizacji programowej. W przypadku braków lub nieścisłości prace te są uzupełniane wspólnie z wykonawcami projektu.

Poprzednikiem tego punktu węzłowego jest oczywiście synteza projektu, ale czasami zachodzi potrzeba sięgnięcia do wcześniejszych prac. Wynikiem projektowania oprogramowania będzie konkretny wykaz prac do wykonania, z listą programów, modułów programowych, podanie w każdym przypadku technologii wykonania wraz z harmonogramem czasowym realizacji całego systemu oprogramowania, a zwłaszcza każdego programu.

120 — oprogramowanie — wykonanie

Celem prac programowych jest wytworzenie oprogramowania użytkowego realizującego wszystkie funkcje i algorytmy zdefiniowane w projekcie systemu. Wykonane programy powinny być zapamiętane w odpowiednich komputero-

wych bibliotekach programów oraz mieć pełną, zgodną z przyjętymi normatywami (punkt 074) dokumentację.

Szczegółowy zakres prac wykonywanych w ramach tego punktu jest określony w projekcie oprogramowania. Tam też są kierunkowe decyzje co do technologii wytworzenia poszczególnych programów i modułów. Oprogramowanie może być realizowane na wiele różnych sposobów, ale nie to jest w tym miejscu przedmiotem zainteresowania. W ramach prac projektowych należy wytworzyć kod źródłowy dla wszystkich programów występujących w systemie i zapisać te wersje źródłowe w odpowiednich bibliotekach; przeprowadzić niezbędne testy gwarantujące zgodność programów ze specyfikacją, a wynikowe wersje programów również zapisać w bibliotekach. Prace programowe opierają się głównie na projekcie oprogramowania, który niezależnie od metody realizacji musi poprzedzić prace programowe.

Wynikiem punktu węzłowego 120, niezmiernie istotnego dla jakości i realizacji całego systemu informatycznego, są biblioteki programów i odpowiednie dokumentacje oprogramowania systemu, a w przypadku technologii systemu operacyjnego, również biblioteka procedur systemu operacyjnego. Gotowe oprogramowanie umożliwia rozpoczęcie prac wdrożeniowych.

130 — organizacyjne przygotowanie obiektu

Zastosowanie informatyki w zarządzaniu obiektem powoduje zmiany jakościowe w funkcjonowaniu tego obiektu. Zmiany te muszą być odpowiednio przygotowane, żeby proces wdrażania nie spowodował usterek w podstawowym funkcjonowaniu. Zakres zmian został wcześniej precyzyjnie określony w projekcie systemu. Tam też zostały zdefiniowane działania organizacyjno-przygotowawcze, jakie należy podjąć przed zastosowaniem systemu informatycznego.

Punkt ten ma na celu organizacyjne przygotowanie obiektu do wdrażania systemu. Należy dodać, że niektóre z przedsięwzięć podejmowanych w ramach tego punktu mogą być wcześniej wykonane, na etapie projektowania systemu. Do głównych przedsięwzięć organizacyjnych można zaliczyć:

— pozyskanie zasobów (kadry, sprzęt, środki finansowe);
— określenie pracochłonności i kosztów wdrożenia;
— określenie pracochłonności i kosztów eksploatacji;
— przygotowanie kadr;
— bazę normatywną systemu (opracowanie);
— podstawy prawne wdrożenia i eksploatacji.

Zarysowany w koncepcji kształt przyszłego systemu uruchomił dwa niezależne nurty działań. Jeden polega na pracach informatycznych, których końcowym efektem jest gotowy system oprogramowania. Drugi nurt to prace, których efektem jest obiekt przygotowany organizacyjnie do wdrożenia systemu informatycznego. Pozyskanie środków finansowych, zakupy inwestycyjne, szkolenie personelu, to tylko niektóre z wielu większych i mniejszych przedsięwzięć. Dla powodzenia przedsięwzięcia istotne jest, aby struktura organizacyjna, obieg dokumentów, zakresy odpowiedzialności na poszczególnych stanowiskach pracy były zgodne z koncepcją systemu. Wymaga to często wydania niezbędnych zarządzeń, regulaminów i innych rozporządzeń prawnych, które usankcjonują porządek narzucony przez system informatyczny.

Front prac dla tego punktu węzłowego został stworzony odpowiednio wcześnie. Poprzednikami są odległe punkty węzłowe, od koncepcji 060 poczynając, a na syntezie projektu 100 kończąc. Prace programowe mogą limitować szkolenie obsługi systemu i opracowanie bazy normatywnej, gdy wykorzystuje się do tego specjalnie tworzone oprogramowanie.

Przygotowany organizacyjnie obiekt umożliwia rozpoczęcie prac wdrożeniowych.

140 — eksperyment na danych modelowych

Każde większe przedsięwzięcie przed przekazaniem do użytkowania jest poddawane generalnej próbie. W warunkach eksploatacyjnych splata się tyle różnorodnych oddziaływań, że trudno jest wskazać faktyczną przyczynę jakiejś zauważonej usterki. Sprawność funkcjonalna systemu informatycznego powinna więc być wcześniej dokładnie zbadana.

Celem eksperymentów jest sprawdzenie zgodności opracowanego projektu i oprogramowania z założeniami systemu i oczekiwaniami użytkownika. Eksperyment winien obejmować cały zakres systemu, zgodność z założeniami, czytelność i kompletność dokumentacji, odporność na zakłócenia itp. Dobór danych modelowych decyduje o jakości przeprowadzanego eksperymentu. Wskazane jest przeprowadzenie kilku eksperymentów i dopiero ich pozytywne wyniki upoważniają do pilotowego wdrożenia.

Przygotowanie eksperymentu można rozpocząć bardzo wcześnie, nawet bezpośrednio po zakończeniu koncepcji systemu, ale jego wykonanie jest uwarunkowane zakończeniem prac programowych. Oprogramowanie systemu jest sprawdzane za pomocą odpowiednich testów. Eksperyment na danych modelowych ma za zadanie przedłużenie tego procesu lub jego weryfikację. Ekspery-

ment powinien przygotować przyszły użytkownik lub wykonawca oprogramowania w ścisłej współpracy z nim. Chodzi o to, żeby sprawdzić realizację odpowiednich funkcji systemu, a nie poprawność funkcjonowania poszczególnych programów.

Najważniejszym elementem jest więc odpowiednie przygotowanie eksperymentu. Musi być jasno określony cel i wynik eksperymentu. Realizacja eksperymentu to rutynowe prace zgodnie z zasadami projektu. Analiza wyników eksperymentu i ewentualne wnioski co do wprowadzenia zmian w projekcie lub oprogramowaniu, to bezpośredni wynik prac tego punktu węzłowego. Potrzeba modyfikowania projektu lub oprogramowania może więc cofnąć pracę do wcześniej zakończonych punktów węzłowych. Dopiero pozytywny wynik wszystkich przewidzianych do wykonania eksperymentów pozwala na kontynuację prac wdrożeniowych. Wynikiem tego punktu węzłowego jest sprawny i zgodny ze specyfikacją system informatyczny.

150 — instalacja systemu

Prace programowe projektowanego systemu mogą być realizowane na sprzęcie, na którym w przyszłości będzie on eksploatowany. W takim przypadku instalacja systemu to drobne zabiegi kosmetyczne. Gdy system był wykonywany na zewnątrz obiektu wdrożenia, wymagane jest zainstalowanie go w konkretnym ośrodku sprzętowym i programowym.

Celem prac instalacyjnych jest stworzenie warunków do użytkowej eksploatacji systemu. Na prace instalacyjne składają się czynności organizacyjne, polegające na:

— implementacji technologii do warunków konkretnego ośrodka obliczeniowego;
— uruchomieniu odpowiednich służb obsługi systemu, pracach programowych wykonanych na konkretnym komputerze, takich jak:
  • zapisanie bibliotek programów na nośnikach komputerowych danego komputera,
  • zapisanie danych niezbędnych do funkcjonowania systemu,
  • inicjowanie taśm magnetycznych, dysków.

Implementacja wykonanego systemu w konkretnych warunkach środowiska programowego, technologii przetwarzania i organizacji obliczeń oraz w strukturze funkcjonowania obiektu, to cel tego punktu węzłowego.

Podstawową sprawą jest przygotowanie umowy o eksploatację systemu informatycznego, gdy użytkownik nie ma własnego ośrodka obliczeniowego. Określenie zadań, odpowiedzialności, terminy i koszty, to zasadnicze sprawy

do uzgodnienia. W warunkach eksploatacji użytkowej nie ma czasu na uzgadnianie działań w sytuacjach awaryjnych. Na każdą okoliczność powinna być opracowana odpowiednia procedura postępowania, która zapewni otrzymanie wyników w przewidywanych w projekcie terminach i postaci. W momencie pertraktowania warunków finansowych umowy o eksploatację jest możliwa weryfikacja założeń ekonomicznych opracowanych w punkcie 133.

Instalacja bibliotek, zbiorów, inicjowanie nośników to czysto techniczne procedury, które nie powinny sprawiać większego problemu, jeśli sprzęt i oprogramowanie operacyjne są zgodne. Brak takiej zgodności może bardzo skomplikować i wydłużyć prace instalacyjne.

Jest oczywiste, że prace instalacyjne można rozpocząć wówczas, gdy system informatyczny jest wykonany i sprawdzony. Nie wyklucza to wcześniejszego wykonania prób, żeby niezgodności sprzętu czy oprogramowania nie pojawiły się dopiero na etapie instalacji.

Instalacja systemu w konkretnych warunkach eksploatacyjnych jest ostatnią czynnością przygotowawczą do użytkowania systemu informatycznego.

160 — wdrożenie pilotowe

Mimo dużej staranności i dokładności w wykonywaniu prac projektowo-programowych, instalacyjnych i innych działań przygotowawczych, przejście na informatyczne wspomaganie zarządzania nie powinno być natychmiastowe. Konieczny jest okres adaptacyjny, pozwalający na ostateczne sprawdzenie mechanizmów systemu i dopasowanie ich do funkcjonowania obiektu.

Zadaniem tego etapu prac jest wytestowanie systemu na danych i w rzeczywistych warunkach. Niekiedy ten etap działań na systemie nazywa sie eksploatacją próbną. Zakres podejmowanych działań obejmuje:

— wybór metody wdrażania systemu;
— opracowanie kryteriów oceny;
— eksploatację pilotową systemu;
— ocenę wyników eksploatacji;
— rozpowszechnienie systemu.

Wybór metody wdrażania systemu oraz kryteria oceny próbnej eksploatacji, decydujące o tym, kiedy tradycyjny system ma być zaniechany, są uzależnione od wpływu nowego systemu na działalność obiektu oraz stopnia skomplikowania systemu informatycznego. Harmonogram prac wdrożeniowych powinien być opracowany wcześniej.

W trakcie wdrożenia pilotowego weryfikowane są wszystkie prace wykonywane w czasie tworzenia systemu informatycznego. Prace informatyczne były już wielokrotnie weryfikowane, co oczywiście nie przesądza o ich poprawności. Jeśli jednak poprzednie weryfikacje były bardzo staranne, nie powinno być błędów. Wdrażanie pilotowe jest główną weryfikacją prac organizacyjnych przygotowania obiektu oraz współdziałania struktur obiektu w warunkach funkcjonowania systemu informatycznego.

Na etapie wdrożenia pilotowego zbiegają się wszystkie wykonywane współbieżnie ścieżki realizacji różnorodnych prac nad systemem. Pozytywny wynik eksploatacji próbnej pozwala na przejście do eksploatacji użytkowej systemu, czyli osiągnięcia finału długiego i złożonego procesu tworzenia systemu informatycznego.

170 — projekt eksploatacyjny

Opisując wcześniej wykonywane prace, wskazywano na możliwość wystąpienia cykli działań. Gdy weryfikacja pewnych czynności nie dawała pozytywnego wyniku, konieczny był powrót do wcześniejszych czynności. Modyfikacje w trakcie prac są rzeczą naturalną i dokonuje się ich prawie zawsze. Taki sposób wykonania systemu nie może ujemnie wpływać na produkt końcowy, a zwłaszcza na dokumentację.

Kiedy wszystkie rozstrzygnięcia w systemie są już ostateczne, można przystąpić do sporządzenia ostatecznej wersji dokumentacji, opisów, instrukcji i innych materiałów dokumentujących wykonaną pracę oraz umożliwiających eksploatację i rozwój systemu.

Celem projektu eksploatacyjnego systemu jest stworzenie warunków do:

— eksploatacji systemu bez udziału autorów systemu;
— bezkonfliktowego obiegu w ośrodku obliczeniowym danych źródłowych, nośników informacji, zbiorów danych, oprogramowania i bezkonfliktowego przebiegu samych procesów obliczeniowych.

Zakres projektu eksploatacyjnego może być różny w zależności od rodzaju systemu, zastosowanej technologii, rodzaju sprzętu itp., najczęściej jednak obejmuje:

— aktualną dokumentację systemu;
— instrukcje wypełniania i obiegu dokumentów;
— instrukcje tworzenia maszynowych nośników;
— instrukcje kompletacji i kontroli danych wejściowych;
— instrukcje operowania systemem — harmonogram;

— instrukcje kompletacji i kontroli danych wyjściowych;
— instrukcje instalacyjne.

Projekt eksploatacyjny i dokumentacje mogą być wykonywane wcześniej, w miarę postępu prac nad systemem. Wydzielenie specjalnego punktu węzłowego ma na celu ostateczne skompletowanie i zweryfikowanie wcześniej wykonywanych opisów i dokumentacji.

Wykonane w ramach tego punktu węzłowego dokumentacje to materiał dla zespołu zajmującego się eksploatacją systemu, jego pielęgnacją, a w miarę potrzeb powinny umożliwiać wprowadzanie modyfikacji.

180 — eksploatacja i rozwój

Po zainstalowaniu systemu w ośrodku obliczeniowym i u przyszłego użytkownika oraz pomyślnym wdrożeniu pilotowym można przystąpić do bieżącej eksploatacji systemu. Eksploatacja odbywa się zgodnie z opracowanym projektem i przygotowaną dokumentacją.

W trakcie eksploatacji wskazane jest prowadzenie zeszytu uwag, gdzie notowane są wszelkie niedogodności użytkowania ewentualnie propozycje zmian i modyfikacji rozwiązań projektowych, programowych i organizacyjnych. Niedogodności szczególnie kłopotliwe i oczywiste usterki powinny być usunięte natychmiast bez przerywania eksploatacji. Wynikające z nich zmiany powinny być udokumentowane i wdrożone.

Inne uwagi oraz propozycje zmian i usprawnień systemu należy szczegółowo analizować z punktu widzenia możliwości ich wdrożenia. Po pewnym czasie eksploatacji z reguły zachodzi potrzeba istotnych zmian w systemie, które wynikają z rozwoju obiektu, sprzętu, organizacji czy realizowanych przez system funkcji. W takiej sytuacji należy opracować nową, zmodyfikowaną wersję projektu w pełnym cyklu opisanej metodyki. Zakres prac poszczególnych punktów węzłowych będzie oczywiście polegał na twórczej modyfikacji istniejącego projektu.

2.3. Tablica zależności czasowych realizacji punktów węzłowych

Macierz powiązań buduje się w celu uchwycenia wzajemnych związków między punktami węzłowymi procesu projektowania systemów informatycznych zarządzania. Ustalane zależności są wykorzystywane na dalszych etapach prac badawczych nad metodyką projektowania oraz do organizacji procesu projektowania systemów informatycznych zarządzania.

W procesach badawczych nad metodyką projektowania systemów informatycznych ustalone związki pomiędzy punktami węzłowymi są wykorzystywane do dokonywania podziału procesu projektowania na etapy (fazy) oraz ustalenia szczegółowego zakresu prac wykonywanych w poszczególnych punktach węzłowych. Przy organizacji prac projektowych wzajemne związki służą do ustalenia harmonogramu prac, kolejności wykorzystywania poszczególnych punktów węzłowych (szeregowo, równolegle), podziału prac pomiędzy członków zespołu itp.

W tabeli 2.1 przedstawiono macierz powiązań między punktami węzłowymi procesu projektowania (technika budowy macierzy polega na przedstawieniu zależności w formie tablicy krzyżowej, jednak biorąc pod uwagę dużą liczbę punktów węzłowych, przedstawiona tablica jest bardziej czytelna). Powiązania między punktami węzłowymi podzielono na dwie klasy: powiązania kaskadowe i powiązania pętlowe. Powiązania kaskadowe występują zgodnie z kolejnością wykonywania punktów węzłowych. Powiązania pętlowe występują w kierunku odwrotnym i wynikają z konieczności ponownego wykorzystania poszczególnych punktów węzłowych. Potrzeba taka wynika ze zmian i uzupełnień do już wykonanych i zweryfikowanych prac.

Tabela 2.1

Zależności między punktami węzłowymi

Punkty inicjujące		Punkty węzłowe	Punkty inicjowane	
Powiązania			Powiązania	
kaskadowe	pętlowe		pętlowe	kaskadowe
1	2	3	4	5
		010 zgłoszenie potrzeby		020, 030
010		020 analiza problemu		030
010		021 badanie możliwości		022, 023, 024
010, 021		022 analiza wariantów		012, 030
021, 022		023 ocena opłacalności		024, 030
021, 023		024 możliwości realizacyjne		030
010, 020		030 definicja zadania		040
022, 023, 024		031 uwarunkowania		035
010, 031		032 główne funkcje		033, 035
032		033 powiązania		035, 071
031, 032		034 zakres prac projektowych		035
031, 032, 033, 034		035 umowa		040
030		040 opis istniejącego stanu		050
		041 charakterystyka stanu organizacyjnego		042, 043, 051
041		042 charakterystyka danych		043, 052
041, 042		043 strumienie informacyjne		053
041, 042, 043		044 opis algorytmów		053, 081
040		050 analiza istniejącego stanu		060
041		051 analiza stanu organizacyjnego		061, 062, 081
042		052 analiza danych		062, 082

1	2	3	4	5
043, 044		053 analiza strumieni		061, 062, 063
040, 050	080	060 koncepcja systemu		080, 090
032, 043, 053		061 zakres systemu		080, 090
051, 052		062 koncepcja struktury informacyjnej		080
053, 061, 062		063 koncepcja struktury technicznej		064, 096
051, 061, 062, 063		064 koncepcja struktury przestrzennej		066, 095
053, 061, 062, 063, 064		065 obraz funkcjonowania		080, 090
041, 051, 061, 062, 063, 065		066 założenia organizacyjne		100, 130
040, 050, 063, 064		067 założenia ekonomiczne		100
060		070 normatywy		080
060, 033		071 normatywy komunikowania		080, 090, 075, 077, 078
063		072 zasady budowy nazw		080
060		073 normatywy wejścia/ /wyjścia		080, 091
066		074 normatywy dokumentacji		126, 170
060, 071		075 normatywy technologii		077, 078, 090
072, 075		076 normatywy programowania		110, 120
071, 064, 065		077 protokoły transmisji		082, 078, 091, 121
072, 073, 075, 067		078 normatywy identyfikacji zbiorów		080, 094
060, 070		080 projekt struktury informacyjnej	060	090, 110
044, 062, 070		081 projekt wyjść		090

1	2	3	4	5
053, 062, 081		082 projekt systemu kodów		084, 085, 090, 110
042, 052, 062, 070, 082		083 projekt wejść		090, 110
081, 082, 083		084 struktura logiczna		085
081, 082, 083, 084		085 projekt zbiorów podstawowych		090
060, 070, 080		090 projekt technologii	080	100, 110, 120, 150, 160, 170
060, 070, 080		091 wybór technologii		092, 093, 095, 086, 100, 110
060, 080, 091		092 podział systemu		093, 095, 096, 110, 120
070, 080, 091, 092		093 projekt schematu		094, 095, 096, 110, 120, 150
080, 092, 093		094 projekt zbiorów roboczych		110, 120
061, 064, 080, 092		095 projekt struktury przestrzennej		096, 110, 120, 130, 150
063, 080, 091, 092, 095		096 projekt struktury technicznej		100, 110, 120, 130, 150
060, 080, 090		100 synteza projektu		110, 120, 130, 140, 150, 160
061, 080, 095, 096		101 synteza struktur systemu		102, 103, 104, 105
061, 065, 067, 092, 101		102 organizacja funkcjonowania systemu		103, 105, 130
066, 067		103 projekt zmian organizacyjnych		104, 105, 130
035, 167, 102		104 harmonogram prac		105
035, 023, 067, 102, 104		105 koszty i efekty		
100		110 projekt programowania	060, 080	120

1	2	3	4	5
093, 092, 091, 076		111 założenia ogólne oprogramowania		112, 113, 116, p120
093, 094, 080, 111, 114	113, 114	112 założenia do programowania		120
111, 080	114	113 specyfikacja algorytmów	112	121, 123, 115
085, 111, 081, 083, 084		114 definicja struktury danych	113	120
111, 114, 113, 076		115 projekt modułów		116, 121, 124, 112
111, 115, 092, 093, 091		116 projekt technologii produkcji		120
110		120 oprogramowanie — wykonanie		140
115, 116, 111, 113		121 wykonanie modułów		124
116, 124, 085		122 wykonanie opisu struktury danych		123
112, 111		123 wykonanie programów		124
123, 122, 121		124 test modułów i programów		125
124		125 test całego systemu		142
074, 124		126 skompletowanie dokumentacji programu		142
070, 082, 083, 124, 125, 100		130 organizacyjne przygotowanie obiektu		151, 160, 170
066, 095, 096, 102, 103, 104		131 pozyskanie zasobów		140, 150, 160
104		132 pracochłonność i koszty systemu		140, 150, 160
102, 110, 120		133 pracochłonność i koszt eksploatacji		180
		134 przygotowanie kadr		135, 140, 150, 160

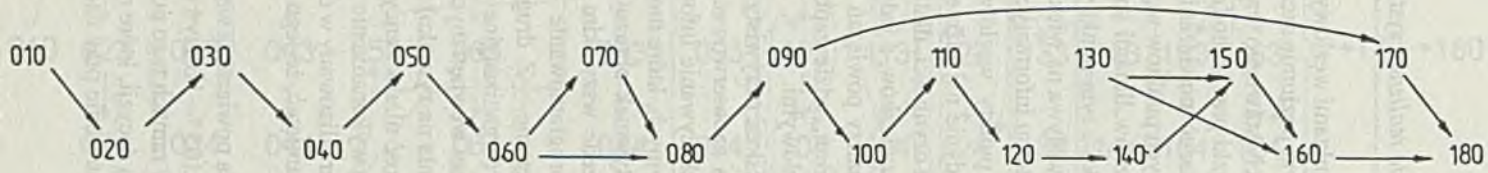
1	2	3	4	5
082		135 opracowanie bazy normatywów		160, 150, 170, 180
066, 070, 080, 102, 103, 104, 133, 134		136 podstawy prawne eksploatacji		150, 160, 180
120		140 eksperyment na modelu	080	130
066, 031, 032		141 opracowanie eksperymentu		142
141, 135, 126		142 wykonanie eksperymentu		143
142		143 analiza wyników eksperymentu		144
143		150 instalacja systemu		160
130		151 umowa o eksploatację		152, 153, 154, 155
125, 151		152 biblioteka programów		153
135, 151	154, 155	153 zbiory kartotekowe		160
131	155	154 inicjacja taśm, dysków	152, 153	
151		155 implementacja techniczna		152, 153, 154
130, 140, 150		160 wdrożenie pilotowe	080, 170	180
140, 150		161 wybór metody wdrażania		162, 163
		162 opracowanie kryteriów oceny		163, 164
130, 140, 152		163 eksploatacja pilotowa systemu		164
162, 163		164 ocena wyników eksploatacji		165
163, 164		165 rozpowszechnienie systemu		180
181, 082, 083, 085, 092, 093		170 projekt eksploatacyjny		150, 180

1	2	3	4	5
		171 aktualna dokumentacja		
082, 083, 136		172 instrukcja wypełniania dokumentacji		160, 180
082, 083		173 instrukcja tworzenia maszynowych nośników		160, 180
091, 092, 093, 094		174 instrukcja kompletacji		180
150		175 instrukcja operowania		180
081, 120		176 instrukcja kontroli		180
070, 120, 150		177 instrukcje instalacyjne		150
170		180 eksploatacja i rozwój		

2.4. Powiązania sieciowe punktów węzłowych

Na rysunkach 2.1 i 2.2 przedstawiono graficznie sieć powiązań między poszczególnymi punktami węzłowymi. Wyróżniono dwa poziomy punktów węzłowych. Pierwszy poziom to punkty węzłowe wyższej kategorii, grupujące kilka punktów węzłowych drugiego poziomu. W wykazie punkty pierwszego poziomu mają zero jako ostatnią cyfrę numeru. Zarówno jeden, jak i drugi rysunek pokazuje sekwencyjność procesu w jego początkowej fazie i coraz większą liczbę powiązań między punktami w miarę zbliżania się do końca procesu.

Rysunek 2.1 ilustruje następstwo poszczególnych grup czynności realizacyjnych na bardzo wysokim poziomie ogólności. Pokazane tam związki nie wymagają specjalnego komentarza, gdyż w połączeniu z zakresem prac, który kryje się pod określonym numerem punktu, następstwo jest oczywiste. Godne uwagi jest powiązanie końcowych punktów, które — jak to widać z rysunku — mogą być w dużej części realizowane współbieżnie. Właśnie z uwagi na możliwość równoległej realizacji i stosunkowo mało skompliko-



Rys. 2.1. Sieć głównych powiązań punktów węzłowych — poziom 1

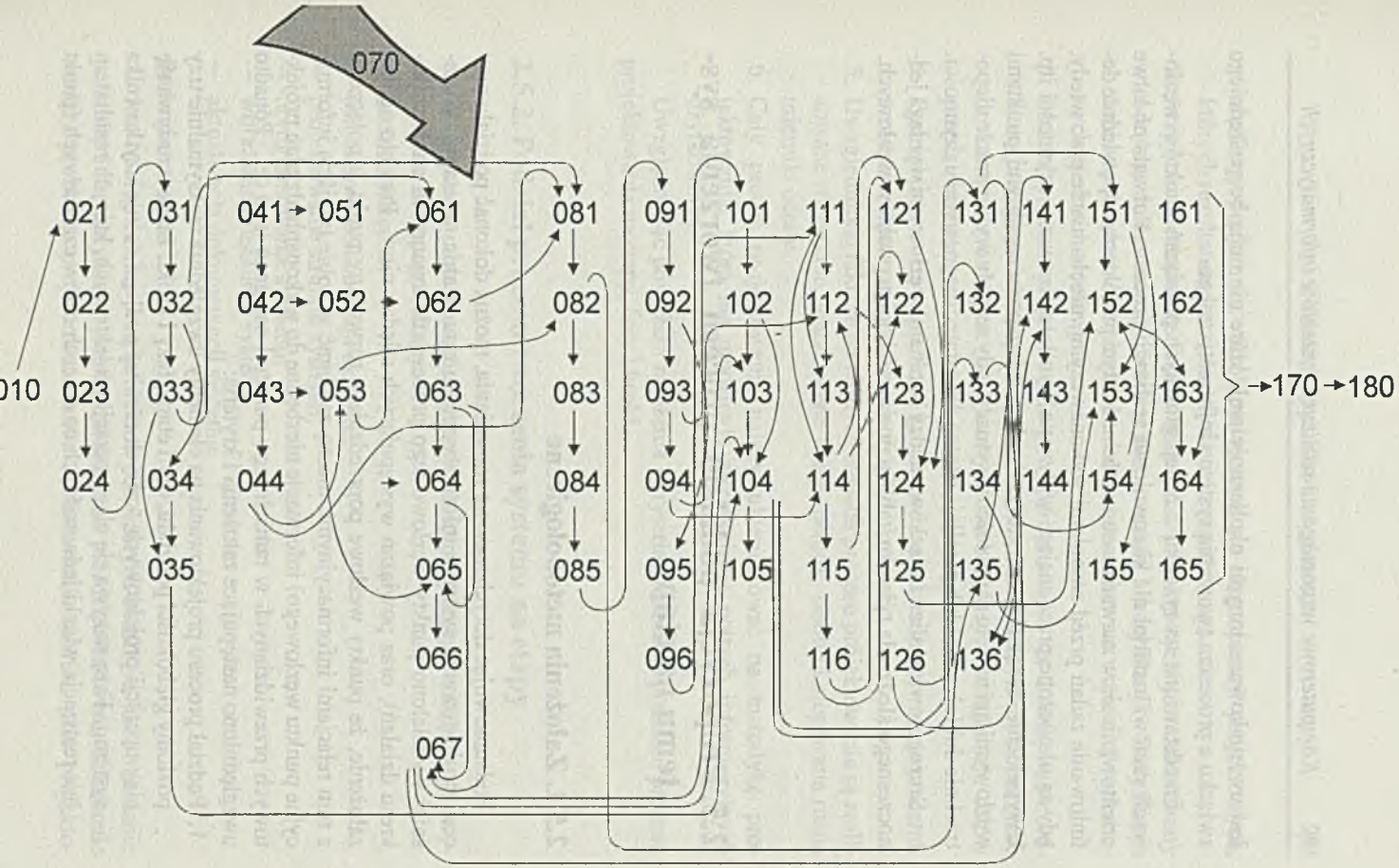
wane połączenia z pozostałymi punktami węzłowymi, na rysunku 2.2 końcowe punkty pozostawiono na pierwszym poziomie szczegółowości.

Wychodząc z zupełnie przeciwstawnego założenia, wydzielono punkt węzłowy 070 (normatywy). Związki prac wchodzących w zakres tego punktu są tak silne, że należałoby wykazać powiązania poszczególnych punktów grupy 070 z większością pozostałych punktów węzłowych drugiego poziomu. Graficzne zaznaczenie tych związków liniami spowodowałoby zmniejszenie przejrzystości rysunku. W związku z tym punkt węzłowy 070 został umieszczony wyżej z zaznaczeniem, że wpływa na większość dalszych prac realizowanych w procesie tworzenia systemu informatycznego.

Linie łączące poszczególne punkty węzłowe nie obrazują wszystkich powiązań między punktami, a jedynie najważniejsze. Rysunek ma pokazać następstwo realizacji konkretnych czynności, dlatego skorzystano z przechodniości i nie pokazano wszystkich związków, a jedynie najbliższe i szczególnie istotne połączenia. Dla pełnej analizy powiązań konieczne jest śledzenie tablicy określającej macierz powiązań, gdzie zapisane są wszystkie znaczące powiązania między punktami węzłowymi.

Analizując rysunek 2.2, widać dość oczywisty podział całego procesu tworzenia systemu na etapy. Można zaobserwować dużą liczbę wewnętrznych powiązań w ramach etapu i przekazywanie informacji z etapu na etap. Na rysunku brakuje powiązań iteracyjnych, które mogą wynikać z prac nad konkretnym punktem węzłowym. Rysunek obrazuje kolejność wykonywanych czynności w jednym cyklu. Przede wszystkim punkty węzłowe 110 (projekt oprogramowania) i 120 (oprogramowanie — wykonanie) mają liczne i różnorodne powiązania wewnętrzne. Z drugiej strony punkty węzłowe wchodzące w skład punktu 130 (organizacyjne przygotowanie obiektu) nie wykazują prawie żadnych powiązań wewnętrznych, a jedynie z innymi, często dość odległymi (w sensie położenia na rysunku) punktami węzłowymi. Taki sposób połączeń oznacza możliwość niesekwencyjnej realizacji czynności danego punktu węzłowego, którego wykonanie nic nie warunkuje. Widać więc wyraźnie, że punkt ten może być zrealizowany w dowolnym momencie, w sensie rozpoczęcia, chociaż należy pamiętać, że jego zakończenie warunkuje rozpoczęcie innych prac.

Niektóre punkty węzłowe mają powiązania wejściowe, a brak im powiązań wyjściowych (na przykład punkt 105 — koszty i efekty). Taka sytuacja oznacza, że czynności wykonywane w ramach tego punktu mają na celu przygotowanie materiałów do określonych decyzji, które nie wpływają na tok czynności realizacyjnych procesu. Zakończenie prac nad takim punktem węzłowym



Rys. 2.2. Sieć głównych powiązań punktów węzłowych — poziom 2

jest uwarunkowane innymi okolicznościami, które nie mają bezpośredniego związku z procesem tworzenia systemu informatycznego.

Przedstawiona na rysunku 2.2 sieć głównych powiązań punktów węzłowych stanowi materiał dla kierownictwa realizacji procesu. Pozwala na łatwe orientowanie się w zaawansowaniu prac nad całym projektem, odpowiednie definiowanie zadań przed zespołami wykonawczymi, uruchamianie prac wtedy, gdy są ukończone prace mające wpływ na wynik realizowanej czynności itp. Uwypuklenie związków merytorycznych między poszczególnymi punktami węzłowymi ma na celu zapobieżenie sytuacji, gdy zespoły wykonawcze niepotrzebnie dublują prace lub pomijają istotne dla ostatecznego sukcesu czynności.

Szczegółowa analiza związków między punktami węzłowymi wymaga jednoczesnego śledzenia rysunku, tablicy powiazań i opisu punktów węzłowych.

2.5. Propozycja podziału procesu tworzenia systemu na etapy

2.5.1. Założenia metodologiczne

Podsumowując dotychczasowe rozważania, można dokonać podziału procesu projektowania systemu informatycznego zarządzania na etapy. Wcześniej wydzielono punkty węzłowe tego procesu, dokonano ich analizy (zakresu działań), oraz powiązań występujących między nimi. Przyjęto ogólne założenie, że punkty węzłowe poprzedzajace rozważany punkt, a połączone z nim relacjami informacyjnymi, muszą zawierać w sobie (wyjście informacyjne punktu węzłowego) informacje niezbędne do wykonania działań projektowych przewidzianych w ramach tego punktu i osiągnąć jego cel. Ponadto uwzględniono następujące założenia i kryteria:

1. Podział procesu projektowania na elementy uwzględnia maksymalnie trzy poziomy głębokości podziału, czyli etapy, fazy i kroki. Etapem nazywa się ciąg operacji projektowych, których rezultat podlega ocenie użytkownika systemu. Fazą nazywa się ciąg operacji projektowych, których rezultatem jest realizacja wiązki jednorodnych co do cech celów cząstkowych (punkt

węzłowy poziom 1). Krokiem nazywa się ciąg operacji projektowych, których rezultatem jest osiągnięcie celu cząstkowego.

2. Przyjęto założenie, że każdy etap prac projektowych jest zakończony odrębną dokumentacją, która podlega ocenie i weryfikacji. Weryfikacja może spowodować konieczność dokonania zmian we wcześniej wykonanych punktach węzłowych.
3. Podział procesu projektowania powinien pozwalać na dokonywanie odbioru prac przez użytkownika we właściwym czasie, czyli nie dopuszczać do kontynuowania prac projektowych nie rokujących osiągnięcia celów przedsiębiorstwa.
4. Podział procesu na etapy, fazy i kroki powinien ułatwić kierowanie pracami, a zwłaszcza procesy planowania działań i kontroli.
5. Uwzględniono fakt, że na różnych etapach procesu projektowania są realizowane różne cele, angażowane różne metody, zasoby i stosowane różne mierniki oceny.
6. Cały proces projektowania należy ukierunkować na metodykę projektowania, która ma umożliwić zaspokojenie potrzeb informacyjnych użytkownika.

Uwzględniając powyższe założenia i kryteria, dokonano podziału procesu projektowania na etapy, fazy i kroki.

2.5.2. Podział procesu tworzenia systemu na etapy

W procesie projektowania systemu informatycznego zarządzania wyodrębnia się następujące etapy:

— badania wstępne;
— projektowanie koncepcji systemu;
— projektowanie struktur systemu;
— oprogramowanie systemu;
— wdrażanie i instalacja systemu;
— aktualizacja dokumentacji systemu.

Badania wstępne

Jest to ciąg działań przedprojektowych, których zadaniem jest rozeznanie potrzeby informatyzacji systemu zarządzania przedsiębiorstwem, dokonanie analizy problemu projektowania oraz zdefiniowanie zakresu prac projekto-

wych. Ogół tych działań powinien doprowadzić do zdefiniowania głównych funkcji systemu informatycznego zarządzania, dokonanie oceny opłacalności przedsięwzięcia i możliwości jego zrealizowania oraz do podpisania umowy o realizację systemu.

Projektowanie koncepcji systemu

Celem tego etapu projektowania jest zdefiniowanie szczegółowej struktury funkcjonalnej systemu, czyli dokonanie podziału głównych funkcji systemu na składniki niższego rzędu, aż do funkcji elementarnych włącznie, oraz zdefiniowanie powiązań informacyjnych między wyszczególnionymi składnikami struktury funkcjonalnej. Ponadto w ramach koncepcji systemu powinny zostać zarysowane pozostałe struktury systemu, czyli informacyjna, przestrzenna i techniczna. Koncepcję systemu powinny zamykać założenia dotyczące jego funkcjonowania i założenia organizacyjne. Drogą prowadzącą do osiągnięcia tego celu projektowania jest opis i analiza istniejącego stanu, pozwalająca na szczegółowe prześledzenie struktur i funkcjonowania tradycyjnego systemu przetwarzania danych.

Projektowanie struktur systemu

Celem tego etapu jest szczegółowe zdefiniowanie struktury informacyjnej (wraz z technologią przetwarzania), przestrzennej i technicznej systemu. Podsumowaniem zaś powinna być synteza projektu, będąca materiałem dla użytkownika systemu do podjęcia decyzji co do dalszych prac projektowych. Ważnym zadaniem jest opracowanie normatywów projektowych obowiązujacych na wszystkich dalszych etapach projektowania.

Oprogramowanie systemu

Zadaniem tego etapu jest przygotowanie oprogramowania systemu, stosownie do zaprojektowanej technologii przetwarzania, struktury informacyjnej, przestrzennej i technicznej systemu. Zakres prac obejmuje przygotowanie projektu oprogramowania, wynikającego z niego oprogramowania i jego wytestowanie.

Wdrażanie i instalacja systemu

Ostatecznym celem tego etapu jest praktyczne wdrożenie rozwiązań systemowych do działalności przedsiębiorstwa. Można tu wyróżnić wiele różnorodnych zadań: organizacyjne przygotowanie przedsiębiorstwa, eksperyment na danych modelowych, instalacje systemu na komputerze czy wdrożenie pilotowe w wybranym obszarze funkcjonowania systemu.

Aktualizacja dokumentacji systemu

Po ostatecznym wdrożeniu systemu, gdy zakończono, między innymi, wprowadzanie zmian do rozwiązań systemowych (które dyktowały doświadczenia wynikające z wdrażania systemu), należy dokonać weryfikacji aktualnego stanu dokumentacji systemu.

2.6. Propozycja podziału cyklu życia systemu

Zachowując trzyszczeblową strukturę hierarchiczną procesu tworzenia systemu informatycznego i uwzględniając specyfikę przedsiębiorstw, można wyróżnić sześć etapów projektowania systemu, które dzielą się na fazy i konkretne kroki według następującego porządku:

Etap 1 — Badania wstępne (B)

B010 — zgłoszenie potrzeby

B020 — analiza problemu
- B021 — badanie możliwości i sposobu rozwiązania problemu
- B022 — analiza wariantów rozwiązań i ofert
- B023 — ocena opłacalności
- B024 — możliwości realizacji

B030 — zdefiniowanie zadania projektowego
- B031 — uwarunkowania
- B032 — główne funkcje systemu
- B033 — powiązania z innymi systemami
- B034 — zakres prac projektowych
- B035 — umowa

Etap 2 — Koncepcja systemu (K)

K040 — opis istniejącego stanu
- K041 — charakterystyka stanu organizacji

K042 — charakterystyka danych (dokumentów, kartotek)
K043 — strumienie informacyjne
K044 — opis algorytmów

K050 — analiza istniejącego stanu
K051 — analiza stanu organizacji
K052 — analiza danych
K053 — analiza strumieni informacyjnych

K060 — koncepcja systemu informatycznego
K061 — zakres systemu (struktura funkcjonalna)
K062 — koncepcja struktury informacyjnej
K063 — koncepcja struktury technicznej
K064 — koncepcja struktury przestrzennej
K065 — obraz funkcjonowania systemu
K066 — założenia organizacyjne systemu
K067 — założenia ekonomiczne systemu

Etap 3 — Projektowanie struktur systemu (S)

S070 — normatywy
S071 — normatywy komunikowania się z innymi systemami
S072 — zasady budowy nazw w systemie
S073 — normatywy wydawnictw i wprowadzania danych
S074 — normatyw dokumentowania
S075 — normatywy technologii przetwarzania
S076 — normatywy programowania
S077 — protokoły transmisji
S078 — normatywy identyfikacji zbiorów danych

S080 — projekt struktury informacyjnej
S081 — projekt wyjść
S082 — projekt systemu kodów
S083 — projekt wejść
S084 — struktura logiczna danych
S085 — projekt zbiorów podstawowych

S090 — projekt technologii przetwarzania
S091 — wybór technologii

S092 — podział systemu na jednostki technologiczne
S093 — projekt schematów przetwarzania
S094 — projekt zbiorów roboczych
S095 — projekt struktury przestrzennej
S096 — projekt struktury technicznej

S100 — synteza projektu
S101 — synteza struktur systemu
S102 — organizacja funkcjonowania systemu oraz koszty jego eksploatacji
S103 — projekt zmian w organizacji obiektu i wymagania wdrożeniowe
S104 — harmonogram i preliminarz prac programowo-wdrożeniowych
S105 — koszty i efekty

Etap 4 — Oprogramowanie systemu (P)

P110 — projekt oprogramowania
P111 — założenia ogólne oprogramowania systemu
P112 — założenia do programów
P113 — specyfikacja algorytmów
P114 — zdefiniowanie struktur danych
P115 — projekt modułów oprogramowania
P116 — projekt technologii produkcji oprogramowania

P120 — oprogramowanie — wykonanie
P121 — wykonanie modułów wspólnych
P122 — wykonanie opisu struktur danych (w języku programowania)
P123 — wykonanie programów
P124 — testowanie pojedynczych modułów i programów
P125 — testowanie całego systemu oprogramowania
P126 — skompletowanie dokumentacji programów

Etap 5 — Instalacja i wdrożenie systemu (W)

W130 — organizacyjne przygotowanie obiektu
W131 — pozyskanie zasobów
W132 — pracochłonność i koszty wdrożenia systemu
W133 — pracochłonność i koszty eksloatacji
W134 — przygotowanie kadr

W135 — baza normatywna systemu — opracowanie
W136 — podstawy prawne wdrożenia

W140 — eksperyment na danych modelowych
W141 — opracowanie eksperymentu
W142 — wykonanie eksperymentu
W143 — analiza wyników eksperymentu
W144 — wnioski wypływające z eksperymentu

W150 — instalacja systemu
W151 — zawarcie umowy o eksploatacje
W152 — biblioteki programów
W153 — zbiory kartotekowe
W154 — inicjacja taśm, dysków, kartotek
W155 — implementacja technologii w konkretnym ośrodku

W160 — wdrożenie pilotowe
W161 — wybór metody wdrażania
W162 — opracowanie kryteriów oceny
W163 — eksploatacja pilotowa systemu
W164 — ocena wyników eksploatacji
W165 — rozpowszechnienie systemu

Etap 6 — Aktualizacja dokumentacji systemu (D)

D170 — projekt eksploatacyjny
D171 — aktualna dokumentacja systemu
D172 — instrukcja wypełniania i obiegu dokumentów
D173 — instrukcja tworzenia maszynowych nośników
D174 — instrukcja kompletacji i kontroli danych wejściowych
D175 — instrukcja operowania systemem — harmonogram
D176 — instrukcja kompletacji i kontroli danych wyjściowych
D177 — instrukcje instalacyjne

Etap 7 — Pielęgnacja systemu (Maintenance)

M180 — eksploatacja i rozwój

3. Normy i standardy w informatyce

3.1. Przesłanki standaryzacji

3.1.1. Strategie standaryzacji

W opracowanej przez Biuro Informatyki Urzędu Rady Ministrów w sierpniu 1991 roku *Propozycji strategii rządu Rzeczpospolitej Polskiej w dziedzinie informatyki* (111) wśród elementów strategii rządu związanej z umacnianiem państwa, wymienia się:

„— opracowanie bazy normatywnej, aby rejestrowane w różnych systemach wielkości mogły być ze sobą porównywane,
— szeroko rozumiana standaryzacja, począwszy od stosowanych narzędzi sprzętowych i programowych, skończywszy na np. ujednoliceniu sposobów zapisu nazw ulic w adresach".

W dokumencie[18] opracowanym na zlecenie Urzędu Rady Ministrów przez Polskie Towarzystwo Informatyczne pod tytułem *Propozycja strategii rozwoju informatyki i jej zastosowań w Rzeczypospolitej Polskiej* stwierdza się:

„Działania strategiczne powinny również objąć niezmiernie ważny problem standaryzacji sprzętu i oprogramowania informatycznego. Obecnie większość

[18] Opracowanie (110) wykonane przez członków Polskiego Towarzystwa Informatycznego zawiera opinie autorskie, które w znacznej części pokrywają się z opinią Towarzystwa, przedstawianą wielokrotnie w publicznych wystąpieniach i publikacjach. Przywołana wcześniej *Strategia...* rządu jest w dużej mierze opracowana na tej podstawie.

norm i standardów międzynarodowych nie ma swych odpowiedników w rejestrze Polskich Norm lub normy tam zawarte są przestarzałe. W związku z tym należy:

— zweryfikować polskie normy informatyczne pod względem ich aktualności i kompletności;
— szybko opracować nowe standardy i normy dotyczące sprzętu komputerowego, ośmiobitowych kodów polskich znaków, układów klawiatur, protokołów komunikacyjnych i innych norm uzupełniających;
— zacieśnić i rozszerzyć współpracę z Międzynarodową Organizacją Standardów (ISO) i narodowymi organizacjami standaryzacyjnymi w celu uwzględnienia w przyszłych międzynarodowych normach polskich poglądów i wymagań; szczególnie ważne jest podjęcie współpracy z Czecho-Słowacją i Węgrami w sprawie kodów znaków narodowych;
— zainicjować prace na temat metod atestacji jakości i bezpieczeństwa oprogramowania oraz zasad eksploatacji systemów informatycznych;
— rozpocząć prace w celu zastosowania w gospodarce narodowej klasyfikacji gospodarczych EWG;
— wymagać dostarczania pełnych polskich wersji oprogramowania od wykonawców kontraktów na informatyzację administracji państwowej".

Przedstawione strategiczne propozycje standaryzacji na poziomie kraju muszą być wspomagane działaniami standaryzacyjnymi w poszczególnych resortach, zgodnie z ogólną koncepcją i specyfiką danej dziedziny działalności. Oprócz norm światowych i narodowych funkcjonują normy branżowe i zakładowe, które są ustanawiane z inicjatywy resortów czy zakładów.

Obecnie większość zastosowań informatyki ma charakter wyspowy, bez odpowiedniej ogólnej koncepcji informatyzacji danej dziedziny. Nawet gdy taka koncepcja istnieje, to realizacja często rozmija się z początkową wizją. Przyczyny takiej sytuacji są bardzo różnorodne. Nie będą one tu przedmiotem analizy, ponieważ po stwierdzeniu tego faktu ważniejsze jest, by podjąć działania, które zmienią tę sytuację.

Potrzeba integracji systemów informatycznych współpracujących z sobą na różnych poziomach jest niewątpliwa. Potęgowo rosnące trudności z integracją coraz większej liczby stosowanych systemów informatycznych mogą spowodować, że korzyści z zastosowania nowego systemu informatycznego są praktycznie nieosiągalne. Wynika to często z tego, że tworząc kolejne zastosowanie, nie zwrócono uwagi na problem integracji i nie przygotowano odpowiednich połączeń z innymi systemami informatycznymi.

Problem integracji zastosowań informatycznych występuje na całym świecie i pochłania bardzo duże środki. Rozwiązywanie problemów integracji już

na wstępnych etapach realizacji, przez stosowanie rozwiązań standardowych, jest ogólnie znacznie tańsze, mimo że podnosi koszty projektu i komplikuje proces realizacji. Integracja wyspowych zastosowań jest naturalną koniecznością, wynikającą z rozpowszechniania się stosowania informatyki, z potrzeby stosowania informatycznych sposobów komunikowania się.

O tym, czy mechanizmy integracyjne zostaną wmontowane w tworzone zastosowania, decyduje dużo czynników, od inwencji twórców poczynając, na możliwościach technicznych i finansowych kończąc. Dodatkowym problemem jest stan informatyzacji środowiska, w którym będzie wykorzystana tworzona aplikacja. Na pewnym etapie rozwoju zostosowań okazuje się, że istnieje konieczność integracji z innymi systemami, które były tworzone obok i wcześniej nie można było przewidzieć potrzeby takiej integracji lub powstały nowe możliwości funkcjonalne w związku z dużym nasyceniem środowiska środkami informatyki. Jeśli nawet udałoby się rozwiązać problemy integracyjne danej dziedziny w kraju, to pozostaje problem integracji w ramach wymiany międzynarodowej.

Należy zwrócić uwagę na jeszcze jeden bardzo istotny problem, związany ze stosowaniem elementów integracyjnych. Z punktu widzenia konkretnego projektu, realizowanego na określone zamówienie, wprowadzanie elementów integracyjnych najczęściej powoduje podniesienie kosztów realizacji, skomplikowanie wykonawstwa, a w zamian daje niezauważalne efekty. Taki stan rzeczy wymaga albo specjalnego wyspecyfikowania problemów integracyjnych w zamówieniu, albo trzeba liczyć na wysoki profesjonalizm autorów.

Bardzo często stwierdza się, że problemy integracyjne występują u „sąsiada", który chce skorzystać z wyników danego systemu, lub w systemie nadrzędnym, który wykorzystuje wyniki pracy systemów. Ta rozbieżność interesów dodatkowo komplikuje powszechność stosowania elementów integracyjnych. Problematyka ta występuje szczególnie mocno w systemach interdyscyplinarnych, które wykorzystują wspólne dane i wyniki pracy różnorodnych systemów.

Powstaje problem znanych i powszechnie akceptowanych standardów postępowania i normatywów obowiązujących w różnych fazach realizacji systemu. Standardy te powinny być wspomagane komputerowymi narzędziami, co podniosłoby ich atrakcyjność. Można więc mówić o komputerowo-zintegrowanym wytwarzaniu systemów informatycznych, co gwarantuje, że produkt końcowy, czyli system informatyczny, będzie łatwy do integrowania na różnych poziomach.

W literaturze problem ten występuje pod nazwą Computer-Integrated Manufacturing (CIM) (137). Komputerowe wytwarzanie jest związane z integracją użytkowanych, przetwarzanych danych. Dotyczy to wszystkich funkcji wewnętrznych i zewnętrznych związanych z wytwarzaniem danych. Istota CIM tkwi w integracji inżynierii i działań organizacyjnych dla funkcji planowania i wytwarzania systemu. Wyróżnia się użytkowanie danych w ramach systemu oraz przekazywanie danych na zewnątrz za pomocą odpowiedniego pośrednictwa (interfejsu), bazy danych i sieci komputerowych.

W opracowaniu (137) proponowane są trzy podstawowe metody realizacji CIM:

— metoda pojedynczego źródła, czyli wykorzystanie zgodności pomiędzy produktami w procesie wytwarzania;
— łączenie niezgodnych systemów przez specjalne pośrednictwo;
— łączenie niezgodnych systemów przez standardowe pośrednictwo, tzw. „wtyczki".

Pierwsza metoda polega na użytkowaniu jednorodnego środowiska sprzętowo-programowego we wszystkich zastosowaniach. Uzależnia to nowe zastosowania od zgodności sprzętu i oprogramowania, co prowadzi do monopolu. Metoda ta była lansowana w Polsce, a monopolistą był jedyny producent sprzętu i oprogramowania — ELWRO. Rozwiązanie to, abstrahując od polskiego przypadku, ma liczne wady i zalety; stosuje je jednak większość dużych firm komputerowych.

Druga metoda jest związana z wytwarzaniem specjalnych procedur komunikacyjnych tylko i wyłącznie na potrzeby łączenia niestandardowych systemów lub ich elementów. Mogą to być zarówno rozwiązania sprzętowe, jak i programowe. W obu przypadkach jest to związane z dodatkowymi, dużymi kosztami na wytwarzanie, utrzymanie i pielęgnację takiego rozwiązania. Metoda ta jest więc kosztowna i kłopotliwa w użytkowaniu. Zastosowanie standardowego pośrednictwa, czyli „uniwersalnych wtyków", ma zarówno wady, jaki i zalety. Takie rozwiązanie jest ekonomiczne w sytuacji, gdy można bezkonfliktowo połączyć więcej niż dwa systemy. Pozwala na dużą swobodę projektowania przez składanie systemów z dowolnej konfiguracji standardowych części. Wadą natomiast jest konieczność dostosowania się do wymogów stawianych przez standardowe elementy i standardowe pośrednictwo. Ogranicza to nie tylko inwencję twórców, ale ma istotny wpływ na zastosowane rozwiązania szczegółowe.

Z przedstawionych trzech metod osiągania integracji godna polecenia jest trzecia metoda — stosowanie standardowego pośrednictwa. Warunkiem jej

użycia jest istnienie standardów akceptowanych przez praktykę tworzenia systemów informatycznych i odpowiednich mechanizmów, nie tylko komputerowego wspomagania.

3.1.2. Światowe przedsięwzięcia standaryzacyjne w informatyce

Konieczność uwzględnienia standardów w zastosowaniach informatycznych jest na świecie dość powszechna. Intensywne prace standaryzacyjne w różnych dziedzinach prowadzą od wielu lat zespoły skupione w specjalnie powołanych organizacjach międzynarodowych. Obszerny przegląd historyczny standardów międzynarodowych, narodowych i międzynarodowych organizacji standaryzacyjnych oraz podejmowanych przez nie przedsięwzięć można znaleźć w pracy (13).

Jako jeden z pierwszych i najważniejszych został opracowany standard komunikacyjny, niezbędny w sieciowych zastosowaniach informatycznych. W związku z powszechnym stosowaniem sieci lokalnych i rozległych oraz wzajemnym łączeniem się różnych sieci komputerowych problem protokołów komunikacyjnych stał się szczególnie istotny. Zgodnie z propozycją ISO (International Standard Organization) dużą popularność zdobył siedmiowarstwowy model protokołu, który wydziela różne poziomy łączności. Standard ten ma charakter otwarty (OSI — Open Systems Interconnection) i wydziela warstwy (67, 41):

— fizyczną (physical layer);
— liniową (link layer);
— sieciową (network layer);
— transportową (transport layer);
— sesji (session layer);
— prezentacji (presentation layer);
— aplikacji (application layer).

Standard OSI/ISO z uwagi na otwartość konstrukcji, międzynarodowy sposób powstania, powszechność i możliwości zastosowań, może być wzorcem standardu w tworzeniu zastosowań informatycznych. Wykorzystując go i uzupełniając o odpowiednie rozwiązania w poszczególnych warstwach, tworzy się nowe standardy, ukierunkowane tematycznie.

Inne z projektów nie są tak popularne, chociaż niektóre z nich są godne zainteresowania z uwagi na dziedzinę, której dotyczą. Manufacturing Automation Protocol (MAP) jest przeznaczony do zastosowań związanych z obszarem działalności przedsiębiorstwa, a Technical and Office Protocol (TOP) — z obszarem działalności biurowej.

Ponadto tworzone są standardy dla:

— programowania robotów (CLDATA, APT, IRDATA);
— przetwarzania graficznego (GKS, SET, VDAPS);
— wymiany danych w systemach CAD (IGES, SET, VDAPS);
— przetwarzania standardowych partii danych (VDAPS);
— wymiany modeli (PDES, STEP).

Opracowanie standardu jest bardzo skomplikowanym procesem, ale jeszcze trudniejsze jest wdrożenie standardu do praktyki międzynarodowej, gdyż dopiero ono wieńczy długi cykl prac standaryzacyjnych.

Oprócz międzynarodowych organizacji standaryzacyjnych powstają narodowe organizacje standaryzacyjne, które koordynują wprowadzanie standardów w danym kraju i współpracują z innymi narodowymi organizacjami. Z uwagi na bardzo wysokie koszty prac standaryzacyjnych dość powszechnie stosowany jest międzynarodowy podział pracy z uzgadnianiem wyników na szczeblu międzynarodowym.

Oto kilka przykładów międzynarodowego podziału pracy przy tworzeniu standardu oraz udział w tych pracach wysoko rozwiniętych krajów i ich narodowych organizacji standaryzacyjnych.

W Stanach Zjednoczonych Ameryki Północnej American National Standards Institution (ANSI) realizuje projekt PDDI (Product Data Definition Interface), który standaryzuje wymianę danych między projektowaniem a implementowaniem. Ponadto realizowany jest projekt AMRF (Automated Manufacturing Research Facility) standaryzujący procedurę badawczą na najniższych poziomach integracji w długim cyklu realizacyjnym.

W Japonii Japanese Industrial Standards Committee (JISC) realizuje projekty INSTAC (Information Technology Research and Standarization Center) oraz FAIS (Factory Automation Interoperable System).

W Wielkiej Brytanii British Standards Institution (BSI) wspomaga rozwój i testowanie wielu projektów, na przykład duży europejski projekt CIMAP.

We Francji Association Francaise de Normalization (AFNOR) pomaga we wdrażaniu projektu ACERCI i innych modułów CIM.

Podobnie działają organizacje normalizacyjne w Kanadzie, RFN, Szwecji i innych rozwiniętych krajach Zachodu. Na szczególną uwagę zasługuje

program ESPRIT (European Strategie Programms for Research and Development in Information Technologies), finansowany ze środków Europejskiej Wspólnoty Gospodarczej, który aktywnie wspiera badania i wdrożenia przedsięwzięć standaryzacyjnych i normalizacyjnych.

Porównując zaangażowanie finansowe i merytoryczne w proces normalizacji i standaryzacji, widać wyraźną dominację USA. Potwierdza się więc spostrzeżenie, że problematyka ta jest szczególnie ważna w krajach o rozwiniętej technologii i dużym nasyceniu środowiska sprzętem i systemami informatycznymi oraz że do prowadzenia prac standaryzacyjnych i ich wdrożenia potrzebne są duże środki finansowe.

Z europejskich przedsięwzięć standaryzacyjnych na szczególną uwagę zasługuje program ESPRIT (39) z uwagi na wielkość przeznaczonych środków finansowych (5700 milionów ECU na lata 1990-1994), rangę zaangażowanych w to przedsięwzięcie uczestników (EWG), zasięg terytorialny oraz zawartość merytoryczną i bezpośrednią zależność Polski od wdrażanych standardów. Program ESPRIT, oszacowany w budżecie EWG na poziomie 3500 osobolat, wyróżnia następujące problemy standaryzacji:

a) CIM architektury i komunikacji;
b) wytwarzanie systemów projektowania i implementowania (CASE);
c) projektowanie produktu i analiza systemu;
d) zarządzanie i kontrola procesu wytwarzania;
e) robotyka i systemy biurowe.

W dziedzinie standaryzacji architektury i komunikacji program ESPRIT zakłada głównie wspomaganie popularyzacji i adaptacji istniejących standardów i prototypów. Ponadto opracowywanie strategii wdrażania do praktyki przedsięwzięć standaryzacyjnych.

Komputerowe wspomaganie procesu projektowania (CAD) i komputerowe narzędzia tworzenia systemów informatycznych (CASE) stworzyły nową jakość pracy twórców projektów i systemów informatycznych. Stosowanie komputerowego wspomagania wymaga jednak uwzględnienia pewnych uwarunkowań i wymogów. Muszą one być opracowane i powszechnie znane. Problematyka związana z tego typu zadaniami to przedmiot prac wymienionych w punktach b) i c).

Standaryzacja procesu wytwarzania jest naturalnym wynikiem stosowania standardowych narzędzi wytwarzania. Ułatwia również proces zarządzania projektem i kontrolę postępu prac. Standaryzacja obejmuje nie tylko środki techniczne i programowe, ale także przedsięwzięcia organizacyjne. W dzie-

dzinie robotyki i systemów biurowych prace koncentrują się głównie na standaryzacji sensorów.

Krótki przegląd światowych działań standaryzacyjnych, obrazuje sytuację w tej dziedzinie. W Polsce w zasadzie nie prowadzi się takiej działalności. Z punktu widzenia instytucjonalnego działa w Warszawie Centralny Ośrodek Badawczy Normalizacji, ale brak środków finansowych i inne mechanizmy oddziaływania praktycznie uniemożliwiają jego funkcjonowanie. Udział Polski w organizacjach światowych był raczej dekoracyjny i nie wnosił nic do praktyki standaryzacji. Działania te miały najczęściej charakter inicjatyw lokalnych, branżowych lub środowiskowych.

W państwach zgrupowanych w Radzie Wzajemnej Pomocy Gospodarczej w latach siedemdziesiątych zrodziła się inicjatywa wprowadzenia standardu informatycznego na poziomie rozwiązań sprzętowych i programowych. Przyjęto zobowiązującą nazwę: Jednolity System, i skończyło się na tym. Oprócz międzynarodowych narad nie udało się nie tylko wdrożenie jakiegokolwiek standardu, ale nawet nie opracowano żadnej, mającej cechy wdrażalności koncepcji.

Monopolista na produkcję sprzętu komputerowego i oprogramowania operacyjnego nie przejawiał żadnych inicjatyw, chociaż z uwagi na pozycję na ubogim rynku miał wszelkie dane na wdrożenie standardu. Obecnie zakupy dokonywane są na rynku światowym, brak zgodności sprzętowej i programowej stał się więc faktem, mimo deklarowanych w reklamach gwarancji „kompatybilności".

W szczytowych latach centralizmu i funkcjonowania „sieci ZETO" próbowano standaryzować proces tworzenia systemu informatycznego i jego dokumentację przez lansowanie metodyki projektowania i wspomagające ją standardy dokumentacyjne[19]. Mimo stosunkowo szerokiego zastosowania, głównie dzięki ośrodkom obliczeniowym ZETO, powstawały jednak standardy branżowe, co spowodowało, że obecnie nie ma jednolitego standardu nazewnictwa, toku prac projektowania i programowania systemu informatycznego. Dokumentacje powstających systemów są nieporównywalne nie tylko z powodu różnorodnych rozwiązań graficznych i redakcyjnych, ale głównie z powodu różnej zawartości informacyjnej.

Jedną z głównych przyczyn braku wyników w nielicznych działaniach standaryzacyjnych był administracyjny, nakazowy sposób wdrażania standardów, nie poparty żadnymi innymi działaniami. Doświadczenia światowe wynikające

[19] Metodyka ta, znana pod nazwą „metodyki Zygiera", od nazwiska głównego jej propagatora, oparta jest na pracach szczecińskiego zespołu zebranych w publikacji (11).

z wdrażania standardów potwierdzają, że prace te nie mogą mieć charakteru akcyjnego, ale powinny być odpowiednio przygotowane, kompleksowe, prowadzone na różnych obszarach, wspomagane przez rozwiązania techniczne, organizacyjne, prawne i inne. W Polsce często utrudniano zamiast pomagać w praktycznym wdrażaniu jakiegoś standardu.

Przykładem standardu stosowanego w konkretnej dziedzinie działalności może być jeden z programów realizowanych w ramach ESPRIT — program DRIVE (Dedicated Road Infrastructure for Vehicle safety in Europe) (32), którego celem jest projekt i stworzenie warunków do integracji środowiska transportu w dziedzinie technologii informacyjnej i telekomunikacyjnej. Dziedziny te, usługowe z punktu widzenia procesu transportowego, są niezmiernie ważne w całym cyklu transportowym. Program zawiera nowe technologie, kierunki zmian i różne opcje strategii rozwoju transportu oraz specyfikacje protokołów i propozycje standardów.

Integracja zawsze wymaga zaawansowanej standaryzacji procesu. Opracowywane standardy dotyczą bardzo różnych dziedzin. Przykładowo, projekt V1037:STAMMI (Definition of Standards for In-Vehicle Man-Machine Interface), realizowany w ramach programu DRIVE, dotyczy standardów sposobu wyświetlania i informowania kierowcy o bezpieczeństwie jazdy[20]. Standardowy interfejs między kierowcą a pojazdem jest bardzo istotny z punktu widzenia pracującego w różnych stresowych sytuacjach, gdy po szybkim odebraniu sygnału musi podjąć odpowiednią decyzję. Standardy ergonomiczne podnoszą bezpieczeństwo jazdy. Ten przykład obrazuje różnorodność działań mających na celu opracowanie standardu, który mógłby być zaakceptowany przez większość uczestników procesu. Kolejna sprawa to wdrożenie opracowanego standardu do praktyki, gdzie pojawiają się problemy dostosowania technologii procesu do nowego standardu, szkolenia użytkowników, tworzenia środowiska dla standardu i umożliwienia współpracy z innymi standardami. Systemy informatyczne mają zastosowanie w różnych dziedzinach. Rozważania skoncentrują się jednak na problemach związanych z wspomaganiem procesu zarządzania. Z tego punktu widzenia można wyróżnić trzy dziedziny standaryzacji:

— szczegółowe standardy techniczne;
— standardy informacyjne;
— standardy metodologiczne.

Standardy techniczne to grupa rozwiązań uniwersalnych wykorzystywanych do tworzenia systemu informatycznego. Rozwiązania te mają najczęściej

[20] Por. pracę (32), s. 145.

powszechne zastosowanie w tych systemach. Chodzi tu o takie standardy, jak na przykład sposoby kodowania polskich znaków narodowych, protokoły transmisji danych.

Standardy informacyjne to określenie niezbędnej zawartości informacyjnej dokumentów oraz innych danych gromadzonych i przetwarzanych w procesie zarządzania. Przykładem jest standaryzacja dokumentacji intensyfikacji wymiany handlowej z Europą Zachodnią, gdzie prace te są bardzo zaawansowane. Problematyka standardowej zawartości informacyjnej w połączeniu ze standardową szatą graficzną, to obiekt zainteresowania projektu EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) (33).

Standardy metodologiczne to metody i zasady postępowania w procesie zarządzania; w miarę jednorodna procedura postępowania oraz porównywalne metody działania.

Wszystkie standardy będą rozważane z punktu widzenia systemów informatycznych.

3.2. Zakres standaryzacji

Efektywne stosowanie systemów informatycznych wymaga od nich elastyczności, otwartości na zmiany i modyfikacje wynikające ze zmian zachodzących w środowisku, w którym funkcjonuje dana aplikacja. Możliwość wzrastania, rozwoju, zgodnie z potrzebami użytkownika, to cechy jakościowe, które ujawniają się w fazie eksploatacji. Brak możliwości łatwego adaptowania systemu informatycznego do zmieniającego się otoczenia często powoduje, że przyjęte rozwiązania zamiast stymulować rozwój dziedziny, w której pracują, ograniczają jej rozwój.

Wdrożenie systemu informatycznego w jakiejś dziedzinie jest tak dużą zmianą jakościową, że po pewnym czasie zachodzi potrzeba modyfikacji założeń leżących u podstaw tego systemu. Mimo stosowania nowoczesnych metod wytwarzania, dużego doświadczenia i inwencji autorów, system informatyczny nie może być użytkowany w nie zmienionej postaci przez dłuższy okres. Powinien on żyć i rozwijać się wraz ze środowiskiem, w którym funkcjonuje.

Informatyka, a zwłaszcza sprzęt komputerowy, oprogramowanie operacyjne, technologia pracy, to dziedziny, które rozwijają się szczególnie dynamicznie. Eksploatowane systemy informatyczne muszą umieć współpracować z nowymi technologiami, komunikować się z innymi, nowszymi systemami użytkowymi oraz wdrażać nowinki technologiczne do własnej pracy. System użytkowy musi umieć wzrastać wraz z otoczeniem.

Cech jakościowych, które umożliwiają wzrost systemu użytkowego, nie można w pełni zdefiniować w momencie powstawania tego systemu, gdyż trudno przewidzieć wszystkie kierunki rozwoju technik komputerowych i zmian w obszarze zastosowania. Jedynym wyjściem jest zatem stosowanie rozwiązań standardowych, które akceptując pewne zasadnicze reguły, dają możliwość przyszłego rozwijania i współdziałania systemu użytkowego z innymi rozwiązaniami standardowymi (81).

Skuteczność rozwiązań standardowych zależy od powszechności stosowania i zasięgu danego standardu. Zakres stosowania standardu zależy od jego prostoty i narzędzi wspomagających użytkowanie. Efekty wynikające ze stosowania rozwiązań standardowych są dość oczywiste, ale nie można zapominać o trudnościach związanych z ich opracowaniem i wdrożeniem. Stosowanie standardów może być skuteczne tylko w dużych i silnych jednostkach, mających odpowiednią „siłę przebicia". Problem polega na wypracowaniu odpowiedniego standardu i środowiska wspomagającego wdrożenie. Przez środowisko rozumie się nie tylko narzędzia bezpośrednio ułatwiające użytkowanie standardu, ale również całe otoczenie prawne i organizacyjne. Może to być połączone z mechanizmami fiskalnymi i innymi ułatwieniami (118).

Przedstawiony punkt widzenia należy do użytkownika systemu informatycznego. Nieco inne spojrzenie na te problemy mają producenci sprzętu komputerowego oraz wytwarzający oprogramowanie operacyjne i systemy użytkowe. Producenci działający na rynku muszą mieć możliwość szybkiego reagowania na zmieniające się zapotrzebowanie klientów i rozwijającą się technologię. Klient nie może ponosić kosztów wynikających z zakupienia systemu, który starzeje się bardzo szybko[21]. Duże firmy komputerowe muszą

[21] W przypadku środków informatyki okres moralnego zużycia sprzętu jest bardzo krótki, dlatego w krajach rozwiniętych tak często odnawia się park maszynowy. W Polsce sytuacja jest odmienna. Z uwagi na ograniczone środki finansowe i przepisy dotyczące amortyzacji środków trwałych, sprzęt użytkowany jest najczęściej do zupełnego zużycia fizycznego. Polskie przedsiębiorstwa stają się klientami firm zachodnich, które oferują sprzęt „second hand". W ten sposób następuje dalsze pogłębienie luki technologicznej między naszymi przedsiębiorstwami a przemysłem krajów rozwiniętych. Środowisko informatyczne wielokrotnie występowało o zmianę okresu amortyzacji w przypadku sprzętu komputerowego (141).

więc gwarantować łatwe przenoszenie funkcjonujących aplikacji w nowe środowisko sprzętowe i programowe. Aby rozwiązać ten problem, propaguje się ostatnio systemy otwarte (open systems). Otwartość systemu cyfrowego ma gwarantować możliwość łatwego przenoszenia systemu informatycznego z jednego środowiska sprzętowo-programowego w inne. Powinna być zapewniona możliwość swobodnego komunikowania się w różnorodnym środowisku sprzętowym i programowym.

Systemy otwarte są domeną dużych firm komputerowych, szczególnie zainteresowanych tą koncepcją. Prowadzenie prac nad systemami otwartymi wymaga wysokich nakładów, na co stać tylko duże firmy, oraz odpowiednio rozbudowanej oferty sprzętowej i programowej. Firmy, nie chcąc utracić swoich klientów, proponują nieco odmienne podejścia do otwartości systemu. Różnice te polegają na tym, że promuje się głównie własną ofertę sprzętową i programową, a produkty innych firm są tylko uzupełnieniem oferty lub gwarantują przejścia między różnymi środowiskami.

Trzy różne koncepcje systemów otwartych firmy Hewlett Packard przedstawiono na rysunku 3.1. Wychodząc naprzeciw zapotrzebowaniu klientów, HP[22] zaczęła dostarczać elementy systemów otwartych, aby można było płynnie wprowadzać nowe technologie informatyczne bez rezygnowania z dotychczasowych osiągnięć i przyzwyczajeń. W opracowaniu (83) zdefiniowano podstawowe elementy platformy systemów otwartych, wymieniając w klasie:

— systemów sieciowych — OSI, ARPA, SNA Novell itp.;
— baz danych — ORACLE, INGRES, INFORMIX itp.;
— interfejsu[23] użytkownika — X-Windows, OSF/MOTIF;
— interfejsu systemów operacyjnych — POSIX.

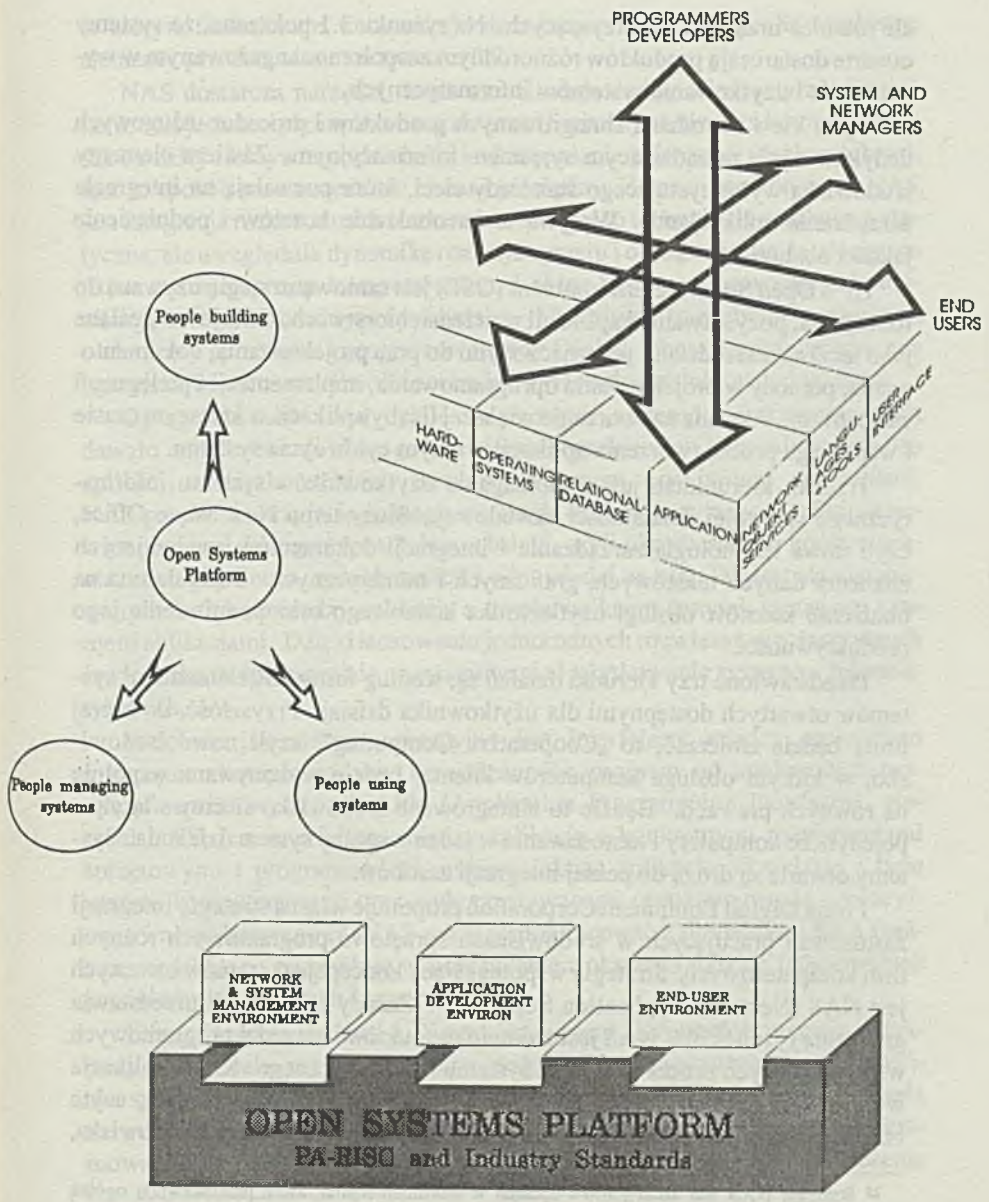
Nie są to oryginalne produkty firmy HP, lecz najbardziej popularne wśród użytkowników produkty wielu różnych firm softwerowych.

Jeśli chodzi o sprzęt, systemy otwarte są oparte na oryginalnym produkcie HP PA-RISC (Reduced Instruction Set Computing)[24]. Technologia RISC jest podstawą całej, nowej rodziny komputerów, które cieszą się coraz większym zainteresowaniem użytkowników. Technologia ta dotyczy nie tylko procesorów,

[22] HP jest często stosowanym skrótem firmy Hewlett Packard.

[23] Interface (ang.) tłumaczy się na język polski jako pośrednictwo. Ostatnio coraz częściej autorzy skłaniają się do polskiej pisowni (fonetycznego brzmienia słowa), które jest jednoznacznie rozumiane. Pogadanki radiowe językoznawców akceptują takie rozwiązanie.

[24] RISC jest to nowa technologia mikroprocesorowa, obecnie realizowana przez głównych producentów sprzętu komputerowego. Założeniem tej technologii jest redukcja listy wykonywanych przez procesor operacji do statystycznie najczęściej wykonywanych. Pozwala to na znaczne obniżenie kosztów produkcji procesorów i zwiększenie ich szybkości.



Rys. 3.1. Systemy otwarte według koncepcji Hewlett Packard

ale również urządzeń towarzyszących. Na rysunku 3.1 pokazano, że systemy otwarte dostarczają produktów różnorodnym zespołom zaangażowanym w wytwarzanie i użytkowanie systemów informatycznych.

Open View to rodzina zintegrowanych produktów i procedur usługowych dedykowanych zarządzającym systemem informacyjnym. Zawiera elementy środowiska wykorzystującego standardy sieci, które pozwalają na integracje zarządzania aplikacjami. Wpływa to na obniżenie kosztów i podniesienie jakości obsługi.

HP's Open Software Environment (OSE) jest ramową strategią używaną do rozwijania, pozyskiwania i aplikacji w przedsiębiorstwach. OSE jest określane jako teczka z narzędziami przeznaczonymi do prac projektowania, dokumentowania, pomocy w projektowaniu oprogramowania, implementacji i pielęgnacji zastosowań. Pozwala na tworzenie większej liczby aplikacji w krótszym czasie i wspomaga proces tworzenia aplikacji w całym cyklu życia systemu.

Trzecim kierunkiem jest wspomaganie użytkowników systemu informatycznego w swojej działalności zawodowej. Służy temu New Wave Office, czyli nowa technologia zarządzania i integracji dokumentów zawierających elementy danych tekstowych, graficznych i numerycznych. Pozwala ona na obniżenie kosztów obsługi użytkownika końcowego oraz podniesienie jego produktywności.

Przedstawione trzy kierunki działań są, według firmy HP, elementami systemów otwartych dostępnymi dla użytkownika dzisiaj. Przyszłość, do której firma będzie zmierzać, to „Cooperative Computing", czyli nowe środowisko, w którym obsługa komputerów klienta, będzie wykonywana wspólnie na równych prawach. Będzie to zintegrowane środowisko sieciowe łączące pojedyncze komputery i zastosowania w jeden wspólny system działania. Systemy otwarte są drogą do pełnej integracji zasobów.

Firma Digital Equipment Corporation proponuje własną strategię integracji zastosowań pracujących w środowiskach sprzętowo-programowych różnych firm komputerowych. Strategią wspomagającą koncepcję systemów otwartych jest NAS (Network Application Support)[25]. Zasady tej strategii przedstawia graficznie rysunek 3.2. NAS jest zbiorem produktów i narzędzi programowych wspomagających proces tworzenia systemu informatycznego. Każda aplikacja wyprowadza informacje dla użytkownika i przyjmuje jego polecenia, czyta i zapisuje dane, umożliwia dostęp do zasobów systemu cyfrowego i środowiska,

[25] Strategia NAS jest szczegółowo opisana w dokumentacjach, które przedstawiają ogólną koncepcję oraz niezbędne informacje szczegółowe, pozwalające na stosowanie NAS do tworzenia aplikacji. Na tę dokumentację składają się raporty (101, 95, 96, 98, 97, 99, 102).

w którym funkcjonuje, komunikuje się z innymi aplikacjami. Są to główne elementy przedstawione na rysunku 3.2.

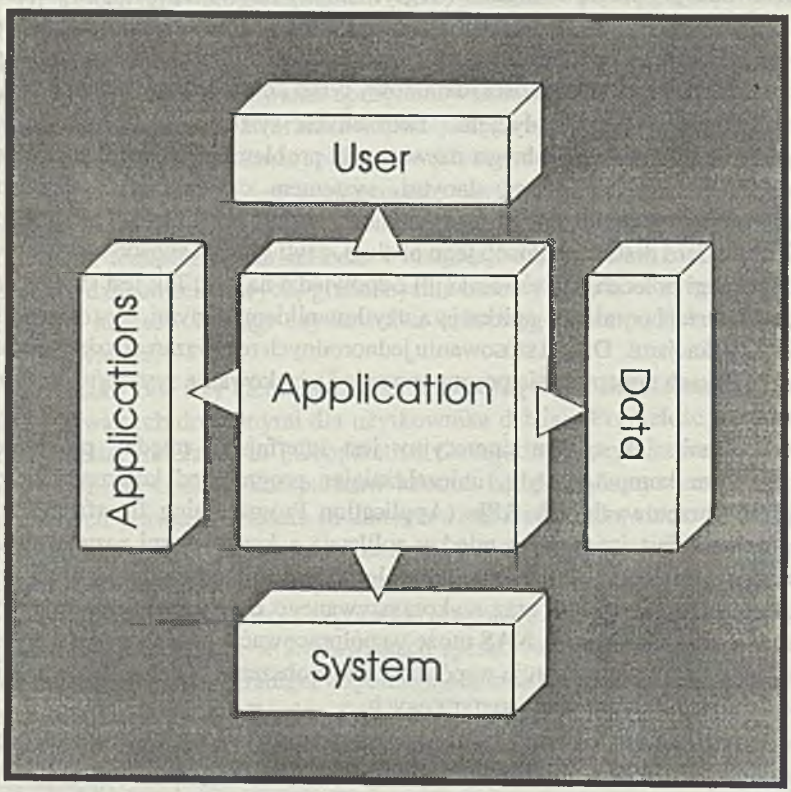
NAS dostarcza narzędzi, które umożliwiają dialog między tymi elementami. Nie jest siecią komputerową, ale umożliwia używanie sieci komputerowej dla bardziej efektywnego dzielenia i udostępniania informacji użytkownikowi. Jest to sposób na oszczędność czasu i pieniędzy. Założeniem koncepcji NAS jest integracja różnorodnych elementów. Nie jest to technologia zamknięta, statyczna, ale uwzględnia dynamikę rozwoju sprzętu i oprogramowania, akceptuje różne rodzaje sprzętu komputerowego i różne systemy operacyjne z komunikacją w niejednorodnym środowisku. Umożliwia to rozbudowę aplikacji o elementy sprzętowe i programowe firmy Digital i innych producentów, nie ogranicza bowiem możliwości rozbudowy tylko do produktów jednej firmy.

Oprogramowanie tradycyjnie tworzonych systemów informatycznych dawało możliwość dowolnego rozwiązania problemów komunikowania się aplikacji z użytkownikiem, danymi, systemem czy innymi aplikacjami. NAS proponuje przyjęcie jednego standardu komunikacyjnego. Definiuje pewien standard dialogu i sposób jego obsługi, czyli określa sposób konstruowania, obsługi poleceń, sposób reakcji i odpowiedzi na nie. Tak jest standaryzowany interfejs pomiędzy aplikacją a użytkownikiem, danymi, systemem i innymi aplikacjami. Dzięki stosowaniu jednorodnych rozwiązań w różnorodnych środowiskach upraszcza się programowanie i użytkowanie systemów informatycznych.

Podobnie jak system operacyjny jest interfejsem między programem a sprzętem komputerowym, uniezależniając program od konkretnych rozwiązań sprzętowych, tak APIs (Application Programming Interfaces), element NAS, jest interfejsem między aplikacją a konkretnymi rozwiązaniami sprzętowymi i programowymi, uniezależniając aplikację od rodzaju i typu sprzętu komputerowego oraz wykorzystywanego oprogramowania operacyjnego i narzędziowego[26]. NAS może współpracować z produktami SAA (opisanego dalej), co gwarantuje współdziałanie z obszerną rodziną IBM-owskich produktów sprzętowych i programowych.

Firmy Hewlett Packard i Digital Equipment Corporation propagują systemy otwarte jako rozwiązanie uniwersalne, akceptujące różnorodne środowiska sprzętowe i programowe, co nie ogranicza wyboru producenta. Pozwala to twórcom systemów informatycznych na swobodny wybór sprzętu i oprogramowania dla realizacji własnej koncepcji rozwiązania. Nieco inne założenie

[26] W pracy (100) wymienione są z nazwy konkretne produkty programowe wspomagające poszczególne kategorie i dziedziny komunikacji, proponowane przez NAS.



Rys. 3.2. Network Application Support firmy Digital (100)

legło u podstaw podobnych działań firmy IBM. Ten niekwestionowany lider światowy w produkcji sprzętu i oprogramowania działania integracyjne ogranicza do własnych produktów, propagując wiele możliwości w dziedzinie sprzętu i programowania. Otwartość systemu i przenośność oprogramowania jest — podobnie jak w przedstawionych wcześniej koncepcjach — celem tych działań, ale oferta narzędzi wspomagających nie wychodzi poza produkty IBM.

Biorąc pod uwagę wysokie koszty standaryzacji i przedsięwzięć zmierzających do powstawania systemów otwartych, firmy komputerowe zjednoczyły się w Open Software Foundation (OSF), której celem jest opracowywanie i wdrażanie standardów. Przykładem takich działań jest opracowanie i próby wdrożenia standardu dla systemu operacyjnego. Jako wersję podstawową przyjęto koncepcje systemu operacyjnego UNIX wraz z jego licznymi implementacjami. Portable Operating System Interface for Computer Environments (POSIX) jest koncepcją łączącą w sobie zbiór standardów projektowanych dla zapewnienia przenośności aplikacji przez grupy robocze IEEE 1003.0--1003.9. POSIX zawiera specyfikacje interfejsu systemu operacyjnego opisanego normą IEEE 1003.1 [27].

Inną organizacją promującą systemy otwarte jest X/Open Common Applications Environment. Zarówno OSF jak i X/Open nie są organizacjami standaryzacyjnymi, takimi jak ISO, IEEE lub ANSI, ale grupują producentów i użytkowników systemów otwartych. Zajmują się specyfikacją i promocją rozwiązań systemów otwartych. Wykorzystują do tych celów OSF's Application Environment Specification i X/Open Portability Guide, gdzie w niezwykle przystępny sposób publikują specyfikacje rozwiązań gwarantujących otwartość systemów. Celem tych organizacji jest próba przeciwstawienia się dominacji firmy IBM przez koncentracje środków dla opracowania własnych standardów, które mogłyby konkurować na rynku komputerowym.

Korporacja IBM, która dysponuje różnorodną ofertą w sensie mocy obliczeniowej, rozwiązań logicznych, sprzętu i oprogramowania operacyjnego, była zmuszona do uporządkowania produktów oferowanych klientom. Stałą dewizą IBM jest możliwość przenoszenia systemów użytkowych z modeli starszych na nowsze. Aby sprostać tym wymaganiom i podnieść konkurencyjność swojej oferty, IBM zaproponował standardową Architekturę Systemów Aplikacyjnych SAA (Systems Application Architecture) (122). Jest to koncepcja standardowej platformy konstrukcyjnej i zestaw standardowych narzędzi wytwarzania systemu informatycznego, aby uzyskać:

[27] Szczegóły związane z realizacja tej koncepcji w przypadku firmy Digital i jej produktu ULTRIX, który jest najbardziej zbliżony do koncepcji POSIX, można znaleźć w pracy (142).

— łatwość przenoszenia aplikacji w różne środowiska;
— spinanie różnych dziedzin działalności przez aplikacje;
— ułatwienie i ujednolicenie dostępu użytkownikowi do aplikacji.

Z uwagi na jednorodność środowiska sprzętowego i programowego warto lepiej poznać podstawowe założenia tej koncepcji, co ułatwi zrozumienie, na czym polega i jak jest w praktyce realizowana zasada tworzenia systemów otwartych.

3.3. Systems Application Architecture — SAA

3.3.1. Założenia i cechy SAA

W marcu 1988 roku firma IBM przedstawiła koncepcję SAA jako zestaw jednolitych zasad konstrukcji systemu informatycznego. Koncepcja została rozszerzona o nowe elementy kolejno w październiku tego samego roku, a następnie w czerwcu i październiku 1989 roku. Nie jest ona zamknięta, lecz stale uzupełniana i modyfikowana.

SAA jest zestawem wybranych programów pośrednictwa, wspomagających zasady i protokoły komunikowania się. Stanowi ramę, podstawowy szkielet dla tworzenia i rozwijania zintegrowanych zastosowań. Działania te mają na celu utrzymanie wysokiego stopnia zgodności i spójności w następujących obszarach:

— programowania — języki programowania, narzędzia obsługi tworzenia i pielęgnacji oprogramowania użytkowego;
— dostępu użytkownika — standardowe projekty ekranów użytkowych i techniką dialogu człowiek — komputer;
— wspomagania komunikacji — metody połączeń między systemami i programami;
— zastosowania — uniwersalne systemy oprogramowania dostarczane przez IBM i innych producentów aplikacji.

SAA ułatwia oraz zwiększa poziom zgodności i spójności produktów realizowanych według tej koncepcji, a w rezultacie:

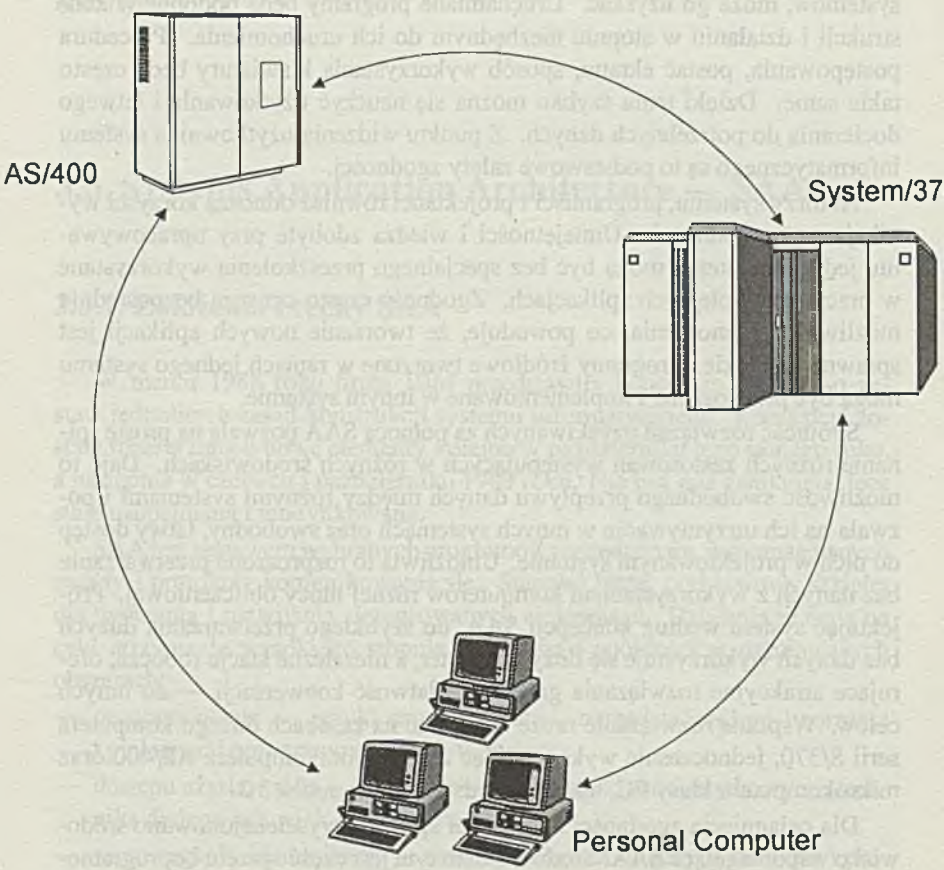
— tworzenie zastosowań jest tańsze i bardziej terminowe;
— działania i używane zasoby są optymalizowane;
— prostsze są przyszłe modyfikacje.

Zgodność rozwiązań uzyskiwana w wyniku stosowania SAA ma wiele zalet. Użytkownik, któremu dzisiaj potrzebny jest dostęp do danych z jednego systemu, a w przyszłości będzie potrzebował dostępu do danych z innych systemów, może go uzyskać. Uruchamiane programy będą podobne w konstrukcji i działaniu w stopniu niezbędnym do ich uruchomienia. Procedura postępowania, postać ekranu, sposób wykorzystania klawiatury będą często takie same. Dzięki temu szybko można się nauczyć użytkowania i łatwego docierania do potrzebnych danych. Z punktu widzenia użytkownika systemu informatycznego są to podstawowe zalety zgodności.

Autorzy systemu, programiści i projektanci również odnoszą korzyści wynikające ze zgodności. Umiejętności i wiedza zdobyte przy opracowywaniu jednego systemu mogą być bez specjalnego przeszkolenia wykorzystane w pracy przy kolejnych aplikacjach. Zgodność często oznacza bezpośrednią możliwość przenoszenia, co powoduje, że tworzenie nowych aplikacji jest sprawne i szybkie. Programy źródłowe tworzone w ramach jednego systemu mogą być przenoszone i implementowane w innym systemie.

Spójność rozwiązań uzyskiwanych za pomocą SAA pozwala na proste spinanie różnych zastosowań występujących w różnych środowiskach. Daje to możliwość swobodnego przepływu danych między różnymi systemami i pozwala na ich utrzymywanie w innych systemach oraz swobodny, łatwy dostęp do nich w projektowanym systemie. Umożliwia to rozproszone przetwarzanie baz danych z wykorzystaniem komputerów różnej mocy obliczeniowej. Projektując system według koncepcji SAA, do szybkiego przetwarzania dużych baz danych wykorzystuje się duży komputer, a niezależne stacje robocze, oferujące atrakcyjne rozwiązania graficzne i łatwość konwersacji — do innych celów. Wspólne rozwiązanie może operować na zasobach dużego komputera serii S/370, jednocześnie wykorzystując zasoby minikomputera AS/400 oraz mikrokomputera klasy PC, tak jak przedstawia to rysunek 3.3.

Dla osiągnięcia zgodności i spójności aplikacji wyselekcjonowano środowisko wspomagające SAA. Środowiskiem tym jest część sprzętu i oprogramowania z oferty IBM, dobranego pod kątem możliwości realizacji technicznej i programowej wszystkich usług. Środowisko to tworzą:
— TSO/E w Enterprise Systems Architecture/370;
— CMS w VM/System Product lub VM/Extended Architecture;
— Operating System/400 (OS/400);



Rys. 3.3. Platforma sprzętowa SAA (122)

— Operating System/2 (OS/2) Extended Edition;
— CICS/MVS w Enterprise Systems Architecture/370;
— IMS/VS Data Communications w Enterprise Systems Architecture/370.

Są to — jak widać — wybrane podstawowe elementy środowiska programowego komputerów serii S/370, AS/400 oraz PC. Nie jest to oczywiście lista zamknięta i należy się spodziewać, że nowe wersje elementów oprogramowania operacyjnego będą rozszerzały środowisko wspomagające SAA.

3.3.2. Standard dialogu

Standard dialogu (Common User Access) określa zasady kontaktu między człowiekiem a komputerem. Ustala, jakie informacje pojawiają się na ekranie użytkownika i jak powinien na nie reagować użytkownik. Zawiera on zasady pracy interakcyjnej. Proponuje standardowe rozmieszczenie danych na ekranie, czyli „formatki" ekranowe, procedurę przechodzenia z jednej formatki do drugiej (rozwijane menu), jednakowe zasady stosowania wyboru, użycia koloru, pomocy i terminologii stosowanej w dialogu. Wykorzystuje się tu najnowsze techniki i technologie oparte na koncepcji „z góry w dół".

W dialogu człowiek — komputer można wyróżnić:
— sposób komunikacji komputera z człowiekiem;
— sposób komunikowania się człowieka z komputerem;
— niezbędny zasób wiedzy użytkownika na temat dialogu.

Komunikacja komputera z człowiekiem polega na tym, w jaki sposób użytkownik postrzega komputer, czyli sprzęt i oprogramowanie, w jaki sposób są prezentowane dane i polecenia. Użytkownik powinien rozpoznać informację przeznaczoną dla niego, zrozumieć ją i wiedzieć, jak na nią odpowiedzieć — jest to sposób odpowiedzi użytkownika. Działanie użytkownika powinno polegać na wypisaniu odpowiedniego komunikatu na klawiaturze lub wciśnięciu klawisza myszy czy wykorzystaniu innego urządzenia. Istotna jest również wiedza o tym, jak użytkownik rozumie zasady dialogu, jego oczekiwania w stosunku do tych zasad, czyli pewien „model konceptualny" dialogu, który powinien być prosty w użyciu i opanowaniu. Dialog i połączenia potrzebne do jego prowadzenia powinny być zgodne w trzech płaszczyznach: fizycznej, syntaktycznej i semantycznej.

Zgodność fizyczna odnosi się do sprzętu, czyli układu klawiatury, położenia klawiszy, użycia myszy. Przykładowo, zgodność fizyczna dla klawiszy będzie

istniała wówczas, gdy klawisze funkcyjne będą wykorzystane do tych samych celów, niezależnie od tego w jakim systemie są używane, a ich miejsce na klawiaturze będzie identyczne.

Zgodność syntaktyczna — to powiązanie porządku i kolejności pojawiania się elementów na ekranie i kolejność naciskania klawiszy w celu uzyskania odpowiedzi na określone żądanie.

Zgodność semantyczna — to powiązanie znaczeń poszczególnych elementów używanych w dialogu. Przykładowo, zgodność oznacza identyczne znaczenie dla komendy CANCEL we wszystkich systemach.

Jest oczywiste, że uzyskanie cech zgodności zależy od konkretnej sytuacji. Projektowane połączenie dla mniej skomplikowanych końcówek czy mniejszej instalacji dużych komputerów nie będzie miało tych cech, co połączenie dla niezależnych stacji roboczych. Istotne jest znalezienie kompromisu między zgodnością a możliwościami technicznymi i technologicznymi.

Projekt SAA nie ustala wszystkich szczegółów dialogu, lecz jedynie podstawowe zasady organizacji ekranu i komunikowania się. Wszystkie szczegóły muszą być opracowane z uwzględnieniem specyfiki konkretnej aplikacji, lecz w ogólnie nakreślonych ramach. Podstawowe reguły zostały zdefiniowane przez IBM, a w odpowiednim opracowaniu można znaleźć wszystkie szczegóły proponowanego standardu (20, 21).

3.3.3. Standard programowania

Jedną z ważniejszych części SAA jest standard programowania (Common Programming Interface) (19), czyli języki programowania, komendy i inne udogodnienia stosowane przez programistów. Komponenty składające się na standard programowania podzielono na dwie główne kategorie:

— języki programowania — Generator Aplikacji, C, COBOL, FORTRAN, PL, RPG;

— obsługa — pośrednictwo (interface) komunikacyjne, obsługi baz danych, dialogu, prezentacji, zapytań.

Dla każdego elementu standardu firma IBM udostępnia pełną dokumentację specyfikacji, tak aby osiągnąć całkowitą zgodność nowych produktów rozszerzających środowisko SAA i promować standard.

Poszczególne elementy standardu mogą być używane samodzielnie, gwarantując zgodność ze standardem i spójność produktu finalnego, lub można je

wzajemnie łączyć również nie tracąc tych cech. Przykładowo, zintegrowana aplikacja może być zbudowana z kombinacji generatora aplikacji i programów w języku COBOL. W celu pobrania informacji z bazy danych można użyć komendy zapytań wmontowanej w programy pisane w języku FORTRAN. Program napisany w języku C może przetwarzać dane i prezentować je graficznie. Język procedur w połączeniu z pośrednictwem dialogu pozwala na powiązanie wielokrokowych aplikacji, zbudowanych z różnych części. Różnorodnych kombinacji jest bardzo wiele.

Standard programowania pozwala również na spinanie systemów w sieć, co oznacza, że można budować systemy przetwarzania oparte na rozproszonym przetwarzaniu. Obecnie SAA oferuje:

— dostęp do zdalnych relacyjnych baz danych;
— dostęp do zdalnych zintegrowanych rekordowo plików;
— komunikację między programami.

W zaawansowanych zintegrowanych zastosowaniach zachodzi potrzeba dostępu do danych przechowywanych na różnych komputerach. Prosty dostęp do takich danych jest możliwy dzięki stosowaniu SAA.

IBM oferuje narzędzia programowe, które są rodziną SAA z wbudowanymi mechanizmami języka SQL dostępu do relacyjnych baz danych. Rodzinę tę tworzą DB2, SQL/DS, OS/400 z systemem zarządzania bazą danych SQL/400 oraz OS/2 z podobnym rozwiązaniem. Współpraca tych produktów umożliwia dystrybucję i relacyjny dostęp do dużych, średnich i małych komputerów. Na wszystkich komputerach dostępne są mechanizmy SQL, które gwarantują dystrybucję i dostęp do wszystkich danych z zapewnieniem pełnego bezpieczeństwa. Komputery muszą być oczywiście połączone siecią, która również powinna spełniać wymogi SAA.

Rozwiązania SAA w ogóle nie ograniczają lokalnego wykorzystania danych, stwarzają natomiast nowe możliwości. Istnieje kilka poziomów dostępu do danych w sieci, który zawsze będzie podzbiorem możliwości lokalnych. Aplikacje umożliwiają dostęp do danych w różnych systemach. Przykładowo, z osobistego komputera połączonego z dużym komputerem można uruchomić konwersacyjny dialog, który wykorzysta dane zapisane w dużym komputerze, wspomagany szybkim dostępem i obliczeniami na dużym komputerze, a uruchamianym pod kontrolą OS/2.

3.3.4. Standard wspomagania komunikacji

Standard wspomagania komunikacji jest wykorzystywany do łączenia aplikacji, systemów, sieci i urządzeń. Procedury komunikacyjne występują jako bloki w różnych innych standardach, definiowanych w ramach SAA (14). Standard komunikacyjny jest oparty głównie na rozwiązaniach IBM-owskiego protokołu komunikacyjnego SNA (Systems Network Architecture) i innych międzynarodowych standardach, zwłaszcza Open Communication Architectures. Można wyróżnić kilka typów wspomagania standardów komunikacyjnych, takich jak: strumienie danych, obsługa aplikacji, obsługa sesji, sieci, kontrola połączeń, protokół otwartego systemu połączeń (OSI).

Standard strumienia danych to 3270, DCA i IPDS. Są one dokładnie wyspecyfikowane i udokumentowane. Sposób formatowania, wyświetlania i drukowania to standard 3270. Document Content Architecture (DCA) to definicja i zasady specyfikujące dokument tekstowy, wykorzystywany w przesyłaniu danych tekstowych. Intelligent Printer Data Stream (IPDS) jest najbardziej złożonym strumieniem danych, używanym do drukarek.

Na obsługę aplikacji składają się różnorodne protokoły komunikacji umożliwiające dostęp do zdalnych danych, takie jak Distributed Data Management (DDM), SNA Distribution Services (SNADS), Document Interchange Architecture (DIA). Protokoły te i inne szczegółowe zasady obsługi sesji, łączenia sieci oraz manipulowania danymi to elementy standardu SAA, które umożliwiają swobodny dostęp do danych umieszczonych w dowolnym miejscu sieci zintegrowanych systemów, zbudowanych zgodnie ze standardem SAA.

Obsługa połączenia danych jest realizowana przez powszechnie znane i akceptowane protokoły komunikacyjne Synchronous Data Link Control (SDLC), Token-Ring Network i X.25[28]. W standardzie przyjęto siedmiowarstwowy model Open System Interconnection (OSI), a dla każdej warstwy wyspecyfikowano odpowiedni element SAA.

3.3.5. Standard aplikacji

Standardową aplikację uzyska się wówczas, gdy konsekwentnie będą stosowane trzy opisane wcześniej standardy: dialogu, programowania, komunikacji.

[28] Informacje szczegółowe na temat przywołanych tu protokołów i innych elementów połączeń sieciowych można znaleźć w dokumentacjach technicznych, a ogólnie zainteresowani tą problematyką znajdą je w pracy (134).

Tworząc środowisko SAA, gwarantują one, że poruszając się w tym środowisku, uzyska się cechy zgodności i spójności opisane na wstępie.

Firma IBM dostarcza standardowe narzędzia i zasady postępowania, które umiejętnie wykorzystane przez IBM lub innych niezależnych producentów tworzą bogatą rodzinę systemów środowiska SAA. Rozwiązania szczegółowe wchodzące do tej rodziny mają zagwarantowaną możliwość komunikowania się i transferu danych między systemami i użytkownikami tych systemów. Nowe produkty z rodziny systemów SAA rozszerzają ją, stwarzając nowe możliwości, niezależnie od tego czy aplikacja jest realizowana na dużym komputerze, czy na komputerze osobistym, gdyż SAA wraz ze środowiskiem gwarantuje swobodę komunikowania się.

Niecelowe jest oczywiście łączenie wszystkich ze wszystkimi (chociaż technicznie możliwe), ponieważ obniżałoby to efektywność rozwiązań i niepotrzebnie angażowało zasoby. Wyróżnia się pewne klasy zastosowań, dla których celowe jest tworzenie zintegrowanych aplikacji. IBM proponuje w pierwszej kolejności stworzenie zintegrowanych aplikacji do obsługi biura, a w dalszej — aplikacji przemysłowych.

Wśród aplikacji obsługi biura wyróżnia się:

— przetwarzanie dokumentów;
— biblioteki dokumentów;
— obsługę pojedynczego użytkownika, pocztę elektroniczną;
— wspomaganie decyzji.

Tworzenie zintegrowanych aplikacji to proces długotrwały i wymagający współdziałania wielu zespołów. Każde specjalistyczne, oryginalne rozwiązanie, zbudowane zgodnie z zasadami SAA, rozszerza rodzinę standardów i tworzy nowe możliwości. Koncepcja SAA jest punktem wyjścia definiującym zasady budowy zintegrowanych systemów. Dalsza rozbudowa tej koncepcji jest procesem ewolucyjnym, w którym będą uczestniczyć zespoły różnych specjalistów. Kierunki rozwoju systemów aplikacji to:

— budowanie systemów użytkowych z wykorzystaniem obecnej specyfikacji SAA jako wystarczającej;
— wytwarzanie specyficznych systemów z wykorzystaniem elementów SAA na podstawie indywidualnej specyfikacji;
— dodatkowe oprogramowanie, które rozbudowuje środowisko SAA.

Szczególną uwagę zwraca się na tworzenie implementacji SAA w językach narodowych, co popularyzuje standard i zwiększa grono jego użytkowników.

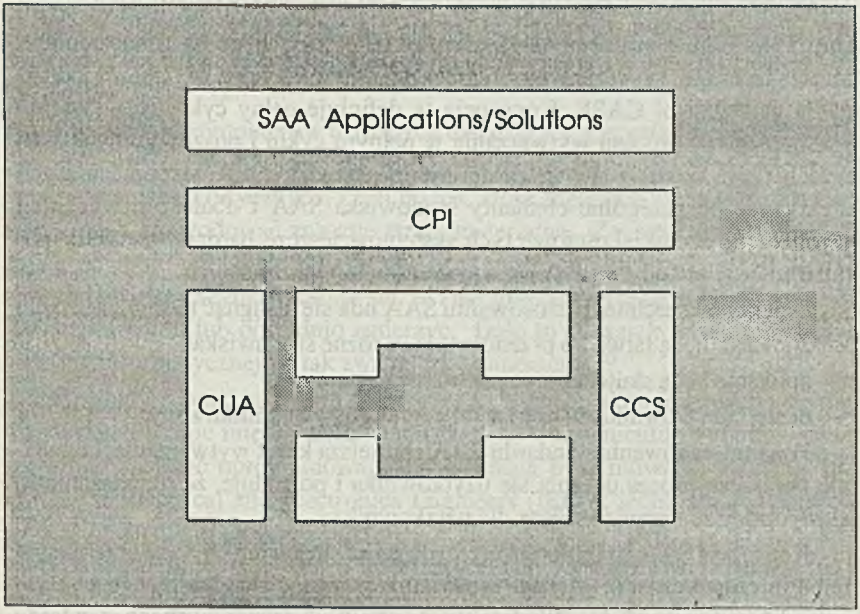
3.3.6. Zalety SAA

Zalety koncepcji SAA można przedstawić z różnych punktów widzenia. O spójności i podobieństwie systemów osiąganych dzięki stosowaniu SAA i wynikających z tego udogodnieniach dla programistów i użytkowników już wspomniano. SAA i środowisko sprzętowo-programowe wspomagające, tworzy pewną rodzinę systemów, która gwarantuje uzyskanie odpowiednich cech jakościowych. Należy również zwrócić uwagę na inne aspekty stosowania standardowych rozwiązań tej koncepcji.

SAA jest zbiorem narzędzi, które wspomagają tworzenie zintegrowanych projektów umożliwiających przetwarzanie danych za pomocą sieci. Gwarantuje ona połączenie fizyczne i logiczne różnorodnego sprzętu dla realizacji jednego projektu. W sieci tej mogą się pojawić różne rozwiązania sprzętowe i programowe z bogatego środowiska SAA. Ostatecznie użytkownik może to postrzegać jako wygodne „okienko" ekranowe wykorzystujące OS/2 na PC, który jest włączony w tę sieć. Pozwala to na swobodne i wygodne korzystanie ze zintegrowanych rozwiązań bezpośrednio z miejsca pracy za pomocą znanego sobie urządzenia PC, bez wnikania w szczegóły połączeń i fizyczne rozmieszczenie danych i algorytmów. Użytkownik pracuje tak, jakby wszystkie zasoby włączone w sieć były do jego wyłącznej dyspozycji i znajdowały się na jego stanowisku pracy. Dzięki takiemu rozwiązaniu użytkownik może być niezwykle efektywnie wspomagany w swej codziennej pracy.

Za tymi udogodnieniami kryje się ogrom prac związanych z synchronizacją procesów, umożliwieniem połączeń różnorodnego sprzętu, dostępem do rozproszonych baz danych, realizacją złożonych i pracochłonnych algorytmów. Atrakcyjność tego rozwiązania polega na tym, że środowisko SAA jest otwarte, a rodzina systemów jest stale rozbudowywana, co zwiększa możliwości konstrukcyjne.

SAA przynosi korzyści zarówno użytkownikowi, jak i zespołom wykonawczym. Użytkownik tworzący zastosowania na własne potrzeby może je szybko i sprawnie wykonać, używając prostych narzędzi wspomagających. Aplikacja tworzona z myślą o sprzedaży lub wielu wdrożeniach ma charakter otwarty; daje się łatwo modyfikować i implementować w różnorodnym środowisku. Z punktu widzenia użytkownika wydzielono pewne dziedziny, które standaryzowano, a tworząc narzędzia wspomagające — generowano środowisko umożliwiające realizację wygodnych zastosowań. Elementy składowe SAA przedstawia graficznie rys. 3.4.



Rys. 3.4. SAA — elementy składowe

3.3.7. Kierunki rozwoju

SAA jest opracowaną przez IBM koncepcją zasad wytwarzania systemów z myślą o użytkownikach sprzętu i oprogramowania tej firmy. Dewizą firmy IBM jest zapewnienie możliwości wzrostu systemów informatycznych wraz z otoczeniem, w którym pracują. Oprócz ogólnej koncepcji wytwarzania IBM dostarcza cały zestaw narzędzi ułatwiających realizację tego celu. Całość tworzy pewne środowisko SAA, które jest otwarte, modyfikowane i rozbudowywane o coraz nowsze elementy, zgodnie z ogólną zasadą standardu.

Naturalnym rozwojem tej koncepcji działania jest przedstawiona we wrześniu 1989 roku koncepcja AD/Cycle (1), jako odpowiedź na coraz popularniejszą technikę wytwarzania systemów informatycznych z użyciem wspomagania narzędziami CASE. Koncepcja ta definiuje pełny cykl życia systemu i wspomaganie procesu wytwarzania w pełnym cyklu i poszczególnych jego fazach (26). Wydaje się, że dostarczenie narzędzi CASE wspomagających efektywnie poszczególne elementy środowiska SAA i dbałość o wcześniej wymienione cechy jakościowe tych systemów jeszcze bardziej zwiększy popularność tej metody wytwarzania systemów informatycznych.

Dzięki powszechnemu stosowaniu SAA uda się osiągnąć następujące cele:
— aplikacje będą łatwe do przenoszenia w różne środowiska;
— aplikacje będą skutecznie spinać różne systemy;
— dostęp użytkownika do aplikacji będzie prosty i zunifikowany.

Ponadto stosowanie standardu SAA zmniejsza koszt wytwarzania, kolosalnie upraszcza proces uczenia się użytkownika i powoduje, że rozwiązania są elastyczniejsze i przyjaźniejsze dla użytkownika.

Koncepcja SAA jest zbiorem prostych zasad stosowanych do tworzenia systemu informatycznego; wymaga stosowania pewnego standardowego zestawu narzędzi wytwarzania (2). Wydaje się, że przyjęcie proponowanego standardu nie powinno sprawiać problemu z uwagi na popularność w Polsce sprzętu i oprogramowania operacyjnego zgodnego ze standardem SAA, zarówno na poziomie komputerów osobistych, jak i na poziomie mainframe. Poza tym obecnie nie ma zastosowań na sprzęcie klasy mini[29].

[29] Dosyć szczegółowe omówienie koncepcji IBM standaryzacji rozwiązań informatycznych podyktowane zostało pozycją tej firmy na rynku komputerowym. Warto wiedzieć, jak należy profesjonalnie podchodzić do problemów standaryzacji. Można tu odnotować zdanie jednego z pracowników IBM, „Jeśli ktoś uważa, że IBM to nie jest standard, to niech jeszcze raz pomyśli".

3.4. Walidacja systemów informatycznych

3.4.1. Problemy jakości oprogramowania

Standaryzacja rozwiązań informatycznych stwarza warunki do porównania i oceniania systemów informatycznych, co było niemożliwe w warunkach tradycyjnego wytwarzania z uwagi na dużą różnorodność rozwiązań i brak kryterium porównania produktów. Standaryzacja przynosi więc konkretne efekty — stworzyła warunki mierzalności procesów dotychczas niemierzalnych. Widoczne są również efekty w organizacji procesu wytwarzania, oceny i akceptacji systemów.

Stosowanie komputerów w różnych dziedzinach gospodarki spowodowało wzrost zainteresowania jakością oprogramowania, zwłaszcza w tak zwanych zastosowaniach krytycznych, czyli takich, które zagrażają życiu lub zdrowiu, albo mogą spowodować znaczne straty materialne. Zaczęto określać cechy, jakie powinno mieć oprogramowanie, aby można było mówić o jego dobrej lub złej jakości. Szybko okazało się, że znacznie łatwiej te cechy wyliczyć, niż je bezpośrednio lub pośrednio zmierzyć. Dało to początek nowej dziedzinie wiedzy informatycznej — tak zwanej programometrii[30].

Atrybuty jakości są różnie definiowane. W pracy skoncentrowano się nie na dyskusji różnic między definicjami, lecz na wymienieniu atrybutów, jakie powinno spełniać oprogramowanie, aby można było mówić o jakości. Institute of Electrical and Electronics Engineers (IEEE), jedna z największych profesjonalnych organizacji światowych zajmująca się teorią i praktyką standaryzacji i definiowania pojęć informatycznych wymienia w słowniku (61) około 30 cech jakości oprogramowania. Są to: dyspozycyjność, efektywność, elastyczność, integralność, możliwość konserwowania, krytyczność, modularność, nadmiarowość, niesprzeczność, niezawodność, odporność, ogólność, poprawność, prostota, przenośność, możliwość realizowania w różnych środowiskach, rozszerzalność, spoistość, śladowalność, możliwość testowania, odporność na błędy i uszkodzenia, użyteczność, możliwość równoczesnego

[30] W Polsce pierwsza wzmianka na ten temat pojawiła się w pracy (46). Bogaty spis literatury tematu można znaleźć w pracy (93).

użytkowania, współdziałanie, wydajność, zabezpieczenie, zespolenie, zgodność, złożoność[31].

Wymienione cechy są w większości niemierzalne, to znaczy nie można ich wyrazić w postaci liczbowej. Aby je określić ilościowo, trzeba skonstruować specjalne miary, tak zwane metryki. Według słownika (61) metryką nazywa się ilościową miarę stopnia, w jakim system, składowa lub proces ma określony atrybut. Przez atrybut rozumie się potocznie, pewną cechę, czyli ogólnie — charakterystykę obiektu. Bardziej szczegółowa definicja określa atrybut jakości jako cechę lub charakterystykę wpływającą na jakość obiektu.

W literaturze można znaleźć wiele bardzo różnych podejść do sposobu mierzenia jakości oprogramowania, lecz z praktycznego punktu widzenia istotniejsze jest, jak osiągnąć określone cechy jakościowe. Dobrą jakość produktu próbuje się zapewnić drogą narzucenia pewnych rygorów na proces wytwarzania oprogramowania. Można zaobserwować dwie drogi prowadzące do tego celu.

Pierwsze wysiłki skierowano na rozwój języków programowania i narzędzi wytwarzania oprogramowania. Zaowocowało to koncepcjami nowych języków programowania, takich jak Ada, Modula 2, oraz unowocześnieniem metod i narzędzi wytwarzania przez budowanie środowisk programistycznych. Inżynieria programowania i komputerowe narzędzia wspomagania, to kierunki mające na celu podniesienie jakości produktów programowych. Więcej na ten temat będzie w następnym rozdziale.

Zaczęto sobie zdawać sprawę, że procesu programowania nie należy ograniczać do wytwarzania kodu, lecz trzeba go utożsamiać z wytwarzaniem produktu programowego, tak jak ma to miejsce w technologii wytwarzania wyrobów w innych dziedzinach, czyli z podziałem na etapy, fazy i operacje. Podział procesu produkcji oprogramowania na wiele wyodrębnionych i dobrze odróżnianych faz, ze ściśle określonymi przejściami, ma dzięki przestrzeganiu zasad produkcyjnych ulepszyć końcowy produkt pod względem jakości.

Stosuje się wiele modeli cyklu życia systemu. Ponieważ procesy realizowane w cyklu produkcyjnym są bardzo zbliżone, więc normy i standardy opisujące poszczególne działania mają również powszechne zastosowanie. Wypracowano wiele dokumentów i działań mających na celu zapewnienie jakości i jej kontrolę w całym cyklu wytwarzania produktu.

Zgodnie z normą (62) w cyklu życia systemu przebiegają następujące procesy:

[31] Inna organizacja, mająca szczególny tytuł do zajmowania się jakością oprogramowania w zastosowaniach, Międzynarodowa Agencja Energii Atomowej, wylicza również około 30 cech, które są bardzo zbliżone do cech wymienionych przez IEEE (54).

— zarządzania przedsięwzięciem;
— przedprodukcyjne;
— produkcyjne;
— poprodukcyjne;
— integracyjne.

Procesy zarządzania i integracyjne trwają przez całe przedsięwzięcie, produkcyjne natomiast występują w czasie kolejno po sobie. Każdy z procesów polega na wykonaniu wielu ściśle określonych czynności.

Procesy integracyjne są niezbędne do skutecznego zakończenia przedsięwzięcia programistycznego. Dwa z nich odgrywają szczególną rolę — weryfikacja i atestacja oraz zarządzanie konfiguracją oprogramowania. Poświęcono im nawet po dwa oddzielne dokumenty normalizacyjne, określające zarówno same zalecenia (60, 63), jak i sposób korzystania z nich (57, 58).

Proces weryfikacji i atestacji obejmuje zaplanowanie i wykonanie wszystkich przeglądów, lustracji, testów i ocen w okresie istnienia oprogramowania. Przez weryfikację rozumie się stwierdzenie, czy wynik określonej fazy cyklu produkcyjnego oprogramowania spełnia wymagania ustalone w poprzedniej fazie. Atestacja natomiast oznacza ocenę oprogramowania na końcu cyklu produkcyjnego, czyli stwierdzenie czy produkt jest zgodny z wymaganiami zawartymi w specyfikacji. Celem zarządzania konfiguracją oprogramowania w cyklu produkcyjnym jest utrzymywanie informacji o aktualnym składzie wytwarzanego oprogramowania i kontrolowanie jego rozwoju.

3.4.2. Znaczenie walidacji

Decydując się na podjęcie opracowania (zlecenie opracowania systemu informatycznego), użytkownik powinien kierować się przeświadczeniem, że zastosowanie takiego systemu będzie wspomagać określoną sferę działalności jako wiarygodne i niezawodne narzędzie pracy. Użytkownik chciałby eksploatować system, który będzie realizował określone funkcje, a ponadto charakteryzował się określonymi parametrami funkcjonalnymi dotyczącymi czasu wykonywania poszczególnych zadań, rozmiarów danych przewidzianych do przetwarzania, liczby stanowisk dostępu do systemu, możliwości ochrony danych itp. Użytkownik ma określone wymagania i jeśli jest przekonany, że opracowany system spełni te wymagania, decyduje się na jego zastosowanie. Przed podjęciem decyzji użytkownik powinien wykonać wiele czynności, które

pozwolą mu na stwierdzenie, że opracowany system sprosta pierwotnie określonym wymaganiom. Ogół takich czynności nazywa się walidacją systemu informatycznego.

Czynności walidacji, mającej kluczowe znaczenie dla ostatecznej decyzji o zastosowaniu systemu, są uwarunkowane zachowaniem należytej staranności w całym toku definiowania i tworzenia systemu. Oznacza to w praktyce, że:

— należy dostatecznie dokładnie wyspecyfikować wymagania wobec systemu, tak aby mogły być jednoznaczną podstawą prac projektowych z jednej strony i kryterium oceny opracowanego systemu — z drugiej;
— należy tak udokumentować kolejne fazy opracowania systemu, by można się było odwoływać do poszczególnych, wyspecyfikowanych wymagań, a ponadto zidentyfikować czynności wykonane błędnie;
— należy wykonać czynności pozwalające na stwierdzenie, że opracowany system działa zgodnie ze sporządzoną specyfikacją wymagań.

Tylko spełnienie powyższych warunków jest dostateczną podstawą podjęcia decyzji o zastosowaniu systemu. Aby wykonać takie czynności, należy określić:

— osoby odpowiedzialne za ich zrealizowanie;
— wymagania normatywne co do sposobu prowadzenia określonych czynności i ich udokumentowania.

Dla użytkownika szczególne znaczenie w tym procesie mają dwa elementy:

— opracowanie specyfikacji wymagań dla systemu;
— dokonanie walidacji opracowanego systemu.

Specyfikacja wymagań dla systemu, sporządzona jako dokument, ma dla użytkownika istotne znaczenie, zarówno techniczne, jak i prawne.

Techniczne znaczenie specyfikacji wymagań dla systemu polega na tym, że jest materiałem niezbędnym dla czynności projektowych oraz kryterium walidacji. Specyfikacja wymagań sporządzona w taki sposób, który nie odzwierciedla potrzeb użytkownika lub odzwierciedla je nieprecyzyjnie, może spowodować opracowanie systemu częściowo lub całkowicie nieprzydatnego do późniejszych zastosowań. Brak specyfikacji wymagań uniemożliwia jednoznaczną ocenę czy system opracowano zgodnie ze zleceniem.

Należy zwrócić uwagę na prawne znaczenie sporządzania specyfikacji wymagań oraz samej specyfikacji jako dokumentu. Jeśli użytkownik zleca wykonanie systemu innej osobie (wykonawcy), to stosunek prawny między tymi dwoma podmiotami jest oceniany zgodnie z przepisami *Kodeksu Cywilnego*, dotyczącymi umowy o dzieło. Odnosi się to do ich wzajemnych obowiązków, uprawnień i odpowiedzialności. W tym kontekście specyfikacja wymagań dla

systemu jest „definicją dzieła”[32], która rozstrzyga o tym, co faktycznie zostało przez użytkownika zamówione.

Specyfikacja wymagań jest też kryterium decydującym o obowiązku „odebrania dzieła” i oceny zasadności ewentualnych roszczeń z tytułu rękojmi za jego wady. Brak specyfikacji wymagań dla systemu stwarza prawną niepewność wzajemnych praw, obowiązków i odpowiedzialności każdej ze stron. Walidacja i jej należyte udokumentowanie ma dla użytkownika istotne znaczenie techniczne i prawne.

Techniczne znaczenie czynności walidacji opracowanego systemu jest następujące:

— są przesłanką stwierdzenia, że system spełnia określone wymagania;

— powinny umożliwić stwierdzenie, czy system — spełniający określone wymagania — prawidłowo realizuje swoje funkcje.

Biorąc pod uwagę, że nieprawidłowo działający system może być przyczyną rozmaitych szkód, czynności walidacji, odpowiednio wykonane pod względem technicznym, uzasadniają decyzję o jego zastosowaniu.

Prawne znaczenie dokonania czynności walidacji i dokumentów sporządzonych w ich toku jest nawet szersze, niż ma to miejsce w przypadku specyfikacji wymagań. Jest to szczególnie istotne dla osoby podejmującej decyzję o skierowaniu systemu do stosowania. Jeśli bowiem w wyniku nieprawidłowego działania systemu powstanie szkoda (na przykład z powodu źle zaprojektowanego obiektu lub źle wytworzonego produktu), którą poniesie sam użytkownik lub inne osoby, odpowiedzialność będzie spoczywać na tym, kto podjął decyzję o jego stosowaniu, nie mając do tego odpowiednich podstaw.

Aby uzyskać właściwe efekty zastosowania systemu informatycznego, trzeba zachować należytą staranność w całym cyklu — od zdefiniowania wymagań dla systemu do odbioru systemu i skierowania go do stosowania, oraz w dokumentowaniu poszczególnych faz tego cyklu. Czynności walidacji wyróżniają się w tym cyklu tym, że są podstawą decyzji, w wyniku której efekty dadzą się poznać w praktyce.

[32] Użyte tu sformułowania są zaczerpnięte z *Kodeksu Cywilnego*.

3.4.3. Warunki prawidłowej walidacji

3.4.3.1. Specyfikacja wymagań dla systemu

Jak już wcześniej wspomniano, czynności walidacji nie są oderwane od reszty czynności wykonywanych w cyklu tworzenia systemu informatycznego, ale są jedną z faz tego cyklu. Jego prawidłowa organizacja, wynikająca ze stanu wiedzy technicznej w dziedzinie inżynierii oprogramowania, a zwłaszcza określenie faz, czynności, wymagań w odniesieniu do dokumentacji i organizacji zespołów ludzkich, jest przedmiotem wielu opracowań, na przykład prac (26, 149). Wynika z nich, że:

— treść i forma specyfikacji wymagań dla systemu, podstawowe kryterium walidacji, jest dziś przedmiotem normy, która — jeśli nawet nie obowiązuje formalnie — powinna być traktowana jako wytyczna do zachowania należytej staranności w sporządzaniu specyfikacji wymagań;

— czynności walidacji oraz treść i forma dokumentacji przeprowadzonych czynności są również objęte systemem norm o znaczeniu podobnym jak wyżej.

Specyfikacja wymagań to jeden z podstawowych dokumentów w toku wytwarzania oprogramowania. Jest tak dlatego, że z założenia zawiera ona informacje będące ostatecznym kryterium oceny czy wytworzone oprogramowanie spełnia ustalone na początku procesu oczekiwania, a tym samym może zostać zaakceptowane przez przyszłego użytkownika. Jest ona więc podstawowym kryterium atestacji. Ma również duże znaczenie prawne. W przypadku oprogramowania wytwarzanego na zamówienie jest podstawą zawarcia kontraktu i rozstrzygnięcia czy zostało ono wykonane zgodnie z jego postanowieniami.

Specyfikacja wymagań dla oprogramowania może być częścią szerszego dokumentu, czyli specyfikacji wymagań dla systemu odnoszącej się również do sprzętu oraz fizycznego kontekstu systemu informatycznego.

W stosunku do specyfikacji wymagań dla systemu wysuwa się na ogół kilka postulatów. Dokumentacja ta powinna być:

a) jednoznaczna, co oznacza, że każde z wyspecyfikowanych w niej wymagań powinno mieć tylko jedną interpretację semantyczną; jest to w istocie jednoznaczne z postulatem możliwie szerokiego stosowania sformalizowanych języków zapisu tych wymagań;

b) zupełna, co oznacza, że:

— żadne istotne wymaganie nie powinno zostać pominięte,

— musi być określona reakcja systemu (wynik przetwarzania, komunikat, sygnał) na każde przewidywane pobudzenie (dane wejściowe, komunikat, przerwanie zewnętrzne),
— na żadne z wymagań nie powinno pojawić się stwierdzenie typu: „zostanie określone później",
— dokumentacja musi być kompletna redakcyjnie, tzn. zawierać wszystkie elementy, do których są w niej odwołania, na przykład rysunki, tablice;

c) spójna, co oznacza, że nie wystąpią rozbieżności lub sprzeczności między jakimikolwiek wyspecyfikowanymi w niej wymaganiami;
d) możliwa do zweryfikowania, co oznacza, że dla każdego z wyspecyfikowanych wymagań istnieje określony sposób jednoznaczego rozstrzygnięcia (w skończonym czasie), czy wymaganie to zostało spełnione;
e) możliwa do modyfikowania — odnosi się to do struktury redakcyjnej dokumentacji, która powinna pozwolić na łatwe wprowadzanie zmian;
f) łatwa do prześledzenia, co oznacza, że dla każdego z wymagań łatwo ustalić zarówno materiały, na których podstawie zostało ono przyjęte, jak i elementy rozwiązań w dalszych fazach rozwoju, które są przez nie kształtowane.

Wyspecyfikowane wymagania powinny dotyczyć:

a) procesów, funkcji i operacji, dla których należy wyspecyfikowanie między innymi:
— algorytmów przetwarzania,
— danych wejściowych i wyjściowych,
— ograniczeń czasowych,
— powiązań z innymi procesami, funkcjami czy operacjami;
b) danych i komunikatów, a dla każdego zestawu danych czy komunikatu trzeba określić:
— sposób wprowadzania/wyprowadzania,
— źródło lub przeznaczenie,
— procesy, funkcje lub operacje korzystające z nich lub je generujące,
— strukturę,
— dokładność;
c) charakterystyk oprogramowania, takich jak:
— formy dialogu z użytkownikiem,
— współpraca z istniejącymi pakietami,
— łatwość adaptacji i zmian określonych wymagań,
— wymogi ochrony danych,

— szczególne wymagania związane z niezawodnością.

Dokumentacja specyfikacji wymagań powinna być uzupełniona opisem przebiegu procesu specyfikowania wymagań, przyjętych technik weryfikacji oraz raportem z wykonanych czynności. Dokumentacja powinna się odwoływać do przyjętych w procesie specyfikowania wymagań standardów i konwencji oraz wykorzystanych narzędzi. Zakres zawartości i forma specyfikacji wymagań dla oprogramowania jest przedmiotem standardu IEEE (116, 117). Jeśli stosuje się metodykę punktów węzłowych, specyfikacja może być rozwinięciem odpowiednich punktów węzłowych na niższych poziomach.

3.4.3.2. Walidacja systemu informatycznego

Zgodnie ze stanem współczesnej wiedzy technicznej w dziedzinie inżynierii programowania, walidacja polega na dokonaniu wielu czynności, właściwych dla kolejnych faz wytwarzania oprogramowania systemu informatycznego. Fazy te można różnie wyodrębniać, najczęściej jednak wyróżnia się:

— specyfikowanie wymagań;
— projektowanie;
— implementację;
— testowanie.

Walidacja jest więc wynikiem procesu równoległego z procesem tworzenia oprogramowania. Może ona być również dokonana ex post na podstawie udokumentowanych wyników kolejnych faz.

W procesie walidacji można wyróżnić czynności weryfikacji, polegające na stwierdzeniu, czy wynik kolejnej fazy (na przykład kod źródłowy) jest zgodny z wynikiem poprzedniej fazy (na przykład projekt). Walidacja powinna być szczegółowo zaplanowana i zorganizowana. Jednym z podstawowych warunków zapewnienia jakości oprogramowania jest sporządzenie, przed przystąpieniem do realizacji przedsięwzięcia, planu walidacji oprogramowania, w którym powinny znaleźć odzwierciedlenie zarówno przebieg procesu, jak i jego organizacja. Plan taki powinien uwzględniać konkretne warunki realizacji przedsięwzięcia, a zwłaszcza:

— charakter powstającego oprogramowania (oprogramowanie wykonywane na zamówienie czy powielane dla szerokiego kręgu użytkowników, stopień „krytyczności" itd.);
— właściwe dla tego przedsięwzięcia fazy w procesie rozwoju.

W swej części ogólnej plan powinien określać:

— strukturę organizacyjną jednostki prowadzącej proces walidacji, ogólny podział zadań, odpowiedzialności i kompetencji;
— wymagania dotyczące zawartości i formy dokumentowania zadań wykonanych w toku procesu oraz zasady zarządzania tą dokumentacją;
— zasady postępowania korekcyjnego w przypadku stwierdzenia nieprawidłowości badanych wyników;
— zasady dokonywania zmian w planie oraz inne wymagania ogólne w stosunku do organizacji procesu.

Dla każdej z faz rozwoju plan powinien sprecyzować:
— zadania do wykonania w ramach procesu walidacji;
— kryteria, metody i techniki weryfikacji wyników danej fazy;
— materiały, na których podstawie będą wykonywane poszczególne zadania (wejściowe), oraz materiały, które będą wynikiem wykonania tych zadań (wyjściowe);
— harmonogram realizacji zadań;
— narzędzia przewidziane do zastosowania przy realizacji poszczególnych zadań;
— znaczenie poszczególnych zadań w realizacji całego przedsięwzięcia;
— organizacyjny podział zadań, odpowiedzialności i kompetencji w toku procesu.

Należy zwrócić uwagę, że szczególnie złożone są zadania walidacji związane z procesem testowania. Obejmują one bowiem:
— zaplanowanie właściwej serii testów;
— dokonanie wyboru właściwych zestawów danych testowych;
— zaprojektowanie testów;
— przygotowanie oprzyrządowania dla testów;
— wygenerowanie procedur testowania;
— przeprowadzenie testowania i dokonanie analizy wyników testów.

Zadania związane z procesem testowania powinny zostać zaplanowane z maksymalnie dopuszczalnym przez procesy rozwoju wyprzedzeniem. Już w fazie specyfikowania wymagań powinno się zaplanować testowanie systemu i odbioru. Plany testowania modułów i integracji powinny być sporządzone w fazie projektowania, co tym samym wyznaczy strategię implementacji.

Przygotowanie danych i wygenerowanie procedur dla poszczególnych rodzajów testowania powinno być przeprowadzone w fazie implementacji. Wówczas zgodnie z przyjętą strategią rozpoczyna się proces testowania modułów i integracji. Najpierw należy zakończyć testowanie modułów

a następnie testuje się integrację. Na zakończenie przeprowadzone jest testowanie systemu oprogramowania i odbiór — zgodnie z wygenerowanymi procedurami.

Postać formalna i wymagania wobec zawartości planu walidacji oprogramowania są przedmiotem standardu IEEE (113). Standardami IEEE są również objęte wymagania wobec planowania procesu testowania i zawartości dokumentacji procesu testowania.

Rozważania te zwracają uwagę na inną dziedzinę standaryzacji, która jest związana z organizacją procesu wytwarzania i ma tak istotny wpływ na jakość produktu końcowego, jak sprawność procesu realizacji.

3.5. Zastosowanie norm

Zastosowanie norm i zaleceń w systemach informatycznych ma dwojakie znaczenie. Może służyć do opracowania bardziej szczegółowych zaleceń i wymogów tworzenia systemu informatycznego lub być podstawą licencjonowania, homologacji systemów informatycznych. Jest to szczególnie istotne w przypadku oprogramowania krytycznego, czyli takiego jak dla systemów:

— wojskowych (31),
— kosmicznych (37),
— jądrowych (4, 54, 56, 59),
— przemysłowych (112, 113).

Bezpośrednie korzyści ze stosowania norm są widoczne w praktycznym kwalifikowaniu programów komputerowych pod względem jakości.

W przypadku oprogramowania wytwarzanego dla zastosowań jądrowych kwalifikowania programów dokonuje się na poziomie producenta, kontraktu lub ogólnokrajowym przez nadawanie licencji. W pracy (36) przedstawiono przebieg atestacji oprogramowania wytwarzanego dla elektrowni jądrowej. Cykl produkcyjny był prowadzony według reguł inżynierii oprogramowania przez dwa oddzielne zespoły wykonawców[33]. Posługiwano się specyfikacją formalną, programowaniem strukturalnym, statystycznymi narzędziami testo-

[33] Dzięki temu dodatkowo uzyskano efekt zróżnicowania.

wania. Cały cykl nadzorował niezależny zespół weryfikacyjno-atestatycyjny, który przeprowadzał przeglądy specyfikacji, rewizję kodu i testy weryfikacyjne.

Ustawodawstwo wielu krajów wymaga licencjonowania systemów komputerowych stosowanych w elektrowniach jądrowych. Dotychczas nie rozwiązano jednoznacznie sprawy metodyki badań prowadzących do wydania licencji, choć wszystkie podejścia w dużej mierze akceptują przedstawiony model cyklu produkcyjnego. Przykładowo, włoski urząd ENEA wykorzystuje definicję jakości oprogramowania IEEE (27), co prowadzi do konieczności wyodrębnienia i oceny poszczególnych cech jakości. Uznano, że możliwe jest wydzielenie takich cech dla każdej z określonych klas systemów komputerowych według podziału zalecanego w normie IEC (55).

Niemiecki urząd TUV opracował własną procedurę licencjonowania systemów komputerowych czasu rzeczywistego (48), która oprócz przemysłu jądrowego objęła także naftowy, chemiczny i inne. Punktem wyjścia tych prac był podział zastosowań na klasy pod względem ich wagi dla bezpieczeństwa. Dla każdej z klas sformułowano inny zestaw kryteriów jakości w postaci cech, które powinien mieć licencjonowany system oprogramowania. Poszczególnym kryteriom przyporządkowano odpowiednie listy kontrolne w formie pytań, które służą do sprawdzania jakości podczas całego cyklu produkcyjnego. Kontroli podlegają: specyfikacja wymagań, projekt systemu i zrealizowany system. Metodyka ta jest dalej rozwijana, głównie pod względem doboru narzędzi wspomagających, których brak jest szczególnie odczuwalny. Na podstawie raportu TUV wnioskodawca może otrzymać licencje.

Podobną metodykę stosuje się w innych krajach, na przykład we Francji (18), w różnych dziedzinach zastosowań, na przykład takich jak transport (76), ze względu na szczególne wymagania wobec bezpieczeństwa. Prowadzi się działania mające na celu koordynację tych prac na szczeblu międzynarodowym (10, 84).

4. Komputerowe narzędzia wspomagania tworzenia systemu informatycznego

4.1. Przesłanki komputerowego wspomagania

Wzorowanie procesu wytwarzania systemu informatycznego na innych, podobnych w strukturze procesach wytwarzania konceptualnego szybko doprowadziło do przejęcia inżynierskiego sposobu organizowania tego procesu. Wytwarzanie bardziej złożonych systemów informatycznych, nie unormowane żadnymi zasadami, zależne tylko od inwencji i temperamentu twórców, nie przynosiło spodziewanych efektów w dłuższej perspektywie czasowej.

Faza programowania systemu, gdzie przede wszystkim koncentrowały się trudności, pierwsza została poddana modyfikacji, która miała na celu porządkowanie i typizację procedur postępowania. Ponieważ skorzystano z doświadczeń projektowania inżynierskiego, mówi się o inżynierii oprogramowania (software engineering)[34]. Celem tych działań było znalezienie i zastosowanie typowych procedur postępowania do wytwarzania elementów systemu informatycznego, zwłaszcza modułów programowych i programów.

Programowanie modularne i strukturalne to praktyczna realizacja tych działań. Podejście strukturalne w programowaniu znalazło zastosowanie

[34] Organizacja James Martin Association również lansowała pojęcia inżynieria informacji i metodologia inżynierii informacji (Information Engineering Methodology). Metodologia ta definiuje zestaw dyscyplin w konwencji czwartej generacji, gdzie celem jest budowa efektywnego systemu. Opiera się ona na podstawowym założeniu, że dane i ich struktura to centrum systemu przetwarzania (44).

w wielu metodach szczegółowych używanych do wytwarzania systemów informatycznych zarządzania (129). Pozytywne doświadczenia z zastosowaniem podejścia strukturalnego w fazie programowania zaowocowały różnorodnym wykorzystaniem go w innych fazach procesu wytwarzania, zwłaszcza w fazie projektowania systemu (106). Działania te spowodowały wyraźne ukształtowanie się procesu tworzenia systemu jako kaskadowo realizowanej procedury postępowania z dość dokładnie określonymi czynnościami, które należy wykonać w kolejnych krokach.

Dobrze zdefiniowane procedury postępowania dają się automatyzować, co umożliwia wspomaganie procesu. Zastosowanie komputera w procesie wspomagania, to naturalne użycie środków informatyki w środowisku odpowiednio zalgorytmizowanym. Naturalne jest również komputerowe wspomaganie procesu wytwarzania systemu informatycznego.

Praca programisty była wspomagana od momentu opracowania pierwszego assemblera. Stopień wspomagania był zależny od środowiska programowego, w jakim tworzono konkretną aplikację. Najprostsze sposoby wspomagania polegały na wykorzystywaniu kompilatorów języków programowania lub prostych programów narzędziowych. Prace nad rozwojem wspomagania programowania koncentrowały się głównie wokół fazy kodowania programów, a inne fazy cyklu życia systemu były pomijane. W związku z tym na przełomie lat sześćdziesiątych i siedemdziesiątych nastąpił dynamiczny rozwój języków programowania. Zaczęły również powstawać proste generatory kodu źródłowego na podstawie odpowiednio zdefiniowanych parametrów[35].

Tak rozumiane wspomaganie było tylko cząstkowe, dotyczyło bowiem wybranych problemów w całym procesie wytwarzania, głównie na etapie programowania. Dopiero inżynieria oprogramowania i jej wpływ na cały cykl życia systemu stworzyły warunki do wspomagania komputerowego. Przekazywanie wyników jednej fazy do następnej pozwoliło na pełniejsze wspomaganie. Aby osiągnąć ten cel, należało zintegrować prace i wyniki cząstkowe poszczególnych faz realizacji systemu. Integrację całego procesu umożliwiły narzędzia wspomagające. Do rozwiązania pozostały problemy metodologiczne. Narzędzia wspomagające są zanurzone w konkretnej metodyce wytwarzania systemu informatycznego.

Problemami technicznymi stały się obsługa bazy danych, gdzie przechowywane są dane o projektowanym systemie w takiej formie, aby mogły być

[35] Przykładem jest język programowania RPG (Report Program Generator), który zdobył dość liczną grupę zwolenników wśród programistów prostych programów aplikacyjnych w systemach przetwarzania danych.

wykorzystywane na kolejnych etapach prac, komunikowanie się zespołów oraz wykorzystywane narzędzia programowe wspomagające proces wytwarzania. Pomocne okazały się tu standardy stosowane w różnych fazach prac nad systemem.

Pierwsze doświadczenia ze stosowaniem narzędzi wspomagających pokazały, że nie tylko upraszcza się i przyśpiesza proces wytwarzania systemu informatycznego lecz przede wszystkim wzrasta jakość opracowywanego produktu (146). Stosowanie narzędzi wspomagających przynosi określone efekty (poprawa jakości, wzrost produktywności itp.), ale okupione jest to różnorodnymi uwarunkowaniami stosowania tych narzędzi. Komputerowo wspomagany może być standardowy proces odpowiednio ustrukturalizowany. Celem działań metodycznych jest właśnie takie zdefiniowanie cyklu życia systemu.

Inżynieria oprogramowania stworzyła podstawy budowy narzędzi komputerowego wspomagania. Akronim CASE (Computer Aided Software Engineering) zdobył sobie popularność i jest bardzo ciekawym chwytem reklamowym. Przez pojęcie CASE rozumie się narzędzia (tools) wspomagające proces wytwarzania systemu i metody postępowania. Często litera S w akronimie jest tłumaczona jako System, co ma oznaczać, że nie chodzi o wspomaganie tylko prac programistycznych, ale całego cyklu życia systemu. Narzędzia wspomagania są zanurzane w odpowiednie metodyki wytwarzania systemu, dlatego istota wspomagania tkwi nie tylko w samych narzędziach, ale w całym środowisku, na które składa się:

— metodyka postępowania;
— narzędzia komputerowego wspomagania;
— narzędzia integracji (por. (90)).

Takie spojrzenie na komputerowe wspomaganie procesu wytwarzania systemów informatycznych gwarantuje szybkie i sprawne opracowywanie systemów o zadowalającej, wysokiej jakości. Tworzy się nowa technologia wytwarzania systemów — technologia CASE.

Wspomaganie procesu wytwarzania systemu informatycznego stało się odrębną działalnością informatyczną, mającą własny rynek produktów (147). Organizuje się konferencje naukowe, na których są prezentowane doświadczenia z użytkowaniem narzędzi CASE oraz wymieniane poglądy zespołów prowadzących badania nad wspomaganiem procesu wytwarzania systemu informatycznego i produkujących narzędzia komputerowego wspomagania. Do największych należą organizowane corocznie od 1989 roku Skandynawskie Konferencje CAiSE. Również w kraju podejmuje się próby stworzenia rynku

narzędzi klasy CASE — odbywają się konferencje poświęcone tej tematyce (124, 90).

Technologia CASE jest wspomagana produktami programowymi, nazywanymi narzędziami CASE (case-tools). Wśród nich można wyróżnić następujące kategorie (por. (144)):

— pojedyncze narzędzia (Component Case Tools), pakiety programowe wspomagające jeden z etapów tworzenia systemu;
— pojedyncze narzędzia plus pomosty (Component Case Tools + Bridges), pakiety wcześniej wymienione wzbogacone o programy wspomagające przejście od jednego narzędzia do drugiego;
— narzędzia obsługi składnicy (Repository Case), programy i procedury obsługi transmisji do i ze składnicy systemu;
— narzędzia integracji (Integrated Case), produkty, których celem jest integracja w ramach kolejnych etapów całego cyklu życia systemu;
— środowisko wspomagania integracji projektu (Integrated Project Support Environment IPSE), produkty wspomagające cały proces wytwarzania.

Okazuje się, że narzędzia te są na tyle różnorodne, iż mogą wspomagać różne etapy wytwarzania i czynności wykonywane w procesie tworzenia, na przykład:

— integrację potrzeb informacyjnych,
— określenie celów systemu informacyjnego,
— zrozumienie wymagań organizacyjnych,
— porównanie sposobu rozumienia problemu przez projektanta i przyszłego użytkownika systemu,
— dostarczanie związków między poszczególnymi częściami,
— kontrolę procesu tworzenia,
— minimalizację skutków przyszłych zmian,
— dokumentację realizacji systemu,
— zarządzanie procesem.

Narzędzia wspomagające proces wytwarzania są pewnego rodzaju systemami ekspertowymi, których podstawowym elementem jest baza wiedzy i zaszyty w systemie motor wnioskowania (51). Ponieważ elementy te są nieodłączną cechą narzędzi wspomagania (126), można sobie pozwolić na takie porównanie. Narzędzia wspomagające oprócz realizacji konkretnych działań mają zaszytą w programie odpowiednio dużą wiedzę o realizowanej metodyce wytwarzania, dzięki czemu zapewniają spójność projektu.

Narzędzia wspomagania tworzenia mogą być różnie wykorzystywane w procesie wytwarzania. W literaturze wymienia się trzy filozofie ich użycia (143):

— restrykcyjną,
— przewodnikową,
— elastyczną.

Filozofia restrykcyjna polega na tym, że projektant systemu jest kontrolowany przez oprogramowanie, czy wykonał odpowiedni zestaw czynności przed przejściem do kolejnego kroku. Rozpoczęcie kolejnej fazy systemu jest blokowane w przypadku niewykonania zapisanego w programie zestawu czynności. Wymuszenie określonej sekwencji działań ma zapobiec niespójności i zapewnić kontrolę poprawności realizowanego procesu.

Filozofia przewodnikowa w odróżnieniu od restrykcyjnej dotyczy następstwa działań. Narzędzia wspomagania wskazują projektantowi możliwe do realizacji cele na aktualnym etapie zaawansowania prac. Przedstawiona lista działań wynika z analizy dotychczas wykonanych prac i jest przewodnikiem dla projektanta systemu.

Filozofia elastyczna pozwala na pełną swobodę realizacji systemu, pozostawiając projektantowi wszystkie decyzje dotyczące kolejności wykonania określonych czynności. Jej stosowanie wymaga dużego doświadczenia i głębokiej wiedzy o całym cyklu życia systemu. Nieprawidłowo stosowane narzędzia wspomagania, możliwość rozpoczęcia prac realizacyjnych w dowolnym punkcie cyklu życia, mogą prowadzić do niespójności systemu. Metoda elastyczna jest bardzo atrakcyjna, ale w praktyce wskazane jest jej wzbogacenie o pewne elementy kontroli realizacji procesu, czyli narzędzia klasy IPSE. Niezależnie od mocy narzędzi wspomagających wszystkimi pracami kieruje projektant i on odpowiada za projektowany system, a narzędzia CASE wspomagają tylko jego decyzje i czynności.

4.2. Klasyfikacja narzędzi wspomagania

Historia komputerowego wspomagania procesu tworzenia systemu informatycznego nie jest zbyt długa. Można jednak wydzielić okres, w którym

produkowano narzędzia wycinkowo wspomagające proces wytwarzania, oraz późniejsze próby wspomagania całego cyklu życia systemu. Mówi się więc o wycinkowych i zintegrowanych narzędziach CASE.

Innym kryterium klasyfikowania jest funkcjonalność narzędzi CASE. Mogą to być narzędzia zarówno wycinkowe, jak i zintegrowane. Dużą grupę systemów CASE tworzą narzędzia wspomagające funkcje graficzne realizowane w procesie projektowania. Należą do niej różnorodne edytory graficzne, które pozwalają na łatwe kreślenie, modyfikowanie i drukowanie schematów i wykresów oraz inne graficzne sposoby prezentacji działań w procesie tworzenia systemu. Edytory te mogą uwzględniać specyficzną dla danej metodyki notację i symbolikę bez wgłębiania się w badanie spójności kreślonych rysunków. Bardziej rozwinięte narzędzia CASE zajmują się ponadto badaniem spójności i poprawności metodycznej, której reguły są zaszyte w oprogramowaniu wspomagającym. Dokumentowanie prac jest najczęściej wykorzystywaną funkcją edytorów graficznych wspomagających proces tworzenia systemu. Łatwość utrzymania dokumentacji w aktualnym stanie, to główne zalety tej klasy narzędzi CASE.

Ponieważ kryteria klasyfikacyjne nie są zbyt precyzyjne i konkretne, narzędzia CASE można podzielić bardzo różnie. Firma doradcza CASE Research opracowała obszerną i stosunkowo precyzyjną klasyfikację (86). Wyróżniła ona następujące pakiety CASE:

— przechowywania danych (repository),
— modyfikacji, adaptacji systemów (re-engineering),
— wspomagania cyklu życia systemu,
— sterowania realizacją projektu,
— bieżącego doskonalenia jakości systemu.

4.2.1. Składnica informacji (repository)

Jednym z pierwszych sposobów wspomagania procesu tworzenia systemu informatycznego była idea słownika, w którym zapisywano wszystkie opisy obiektów definiowanych w poszczególnych fazach procesu realizacji. Założeniem koncepcji słownika było zapewnienie integracji wewnętrznej i uproszczenie problemów dokumentacji. Najbardziej wyrafinowanym przykładem tej idei jest pakiet PSL/PSA opisu i analizy danych, opracowany na Uniwersytecie Michigan.

Składnica informacji (repository) jest rozwinięciem idei słownika. Rysunek 4.1 przedstawia położenie składnicy względem innych elementów. Centralne położenie wskazuje na kluczową rolę odgrywaną w procesie wytwarzania systemu. Do słownika dodano mechanizmy gwarantujące bezpieczeństwo danych, zgodność wersji i zarządzanie obiektami definiowanymi w systemie. Składnica jest centralną bazą danych opracowywanego systemu z własnym systemem zarządzania. Podstawowe cechy składnicy informacji, to:

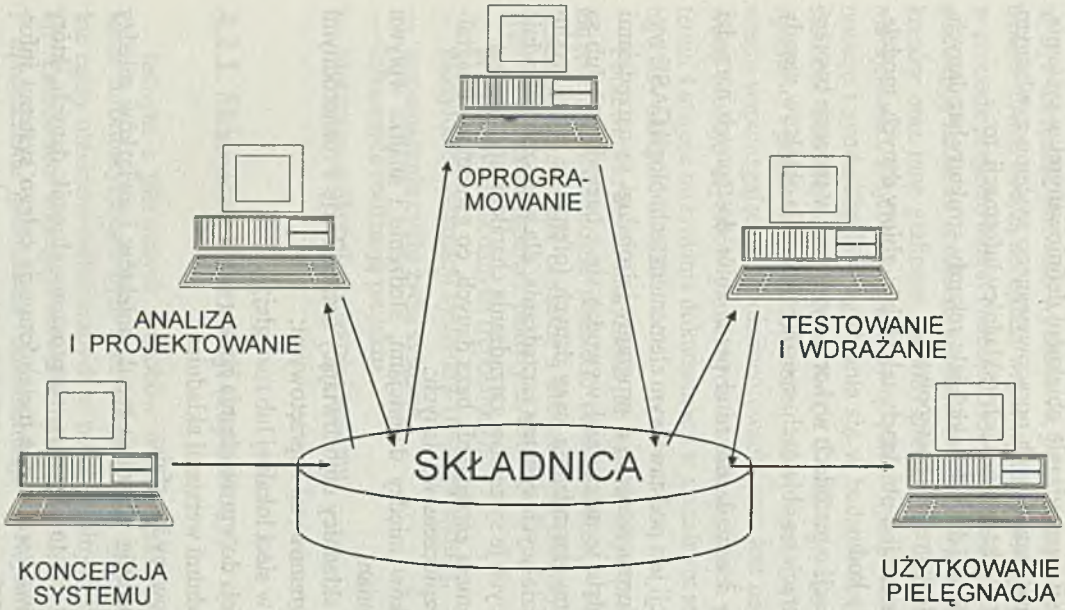
— łatwy dostęp do definicji danych, takich jak: rekordy, struktury baz danych, formularze, procedury i reguły postępowania;
— utrzymywanie informacji o obiektach, takich jak: zbiory danych, modele, dokumenty, moduły kodu;
— utrzymanie informacji o metodach wykorzystywanych w procesie tworzenia, takich jak: operacje na obiektach, sposoby tworzenia obiektów, zasady edycji;
— dzielenie obiektów i zarządzania nimi przez zbiór dostępnych narzędzi obsługi składnicy.

Składnica informacji jest podstawowym elementem technologii CASE wytwarzania systemu informatycznego, a programy jej obsługi są narzędziami wspomagania. Narzędzia te najczęściej wywodzą się z bazy danych lub są wmontowane w systemy zarządzania bazą danych (oferują je różni producenci). Mimo różnic dzielących systemy zarządzania, dla większości składnic informacji i obsługujących je systemów zarządzania charakterystyczne jest:

— powiązanie z systemem zarządzania bazą danych, co gwarantuje integralność, spójność i bezpieczeństwo danych;
— utrzymanie związków między definicjami, śledzenie i analiza wpływu wprowadzanych zmian;
— interfejs wywołań składnicy umożliwiający komunikację z różnorodnymi środowiskami programowymi i sprzętowymi;
— rozproszenie bazy w sieci lokalnej lub rozległej;
— interakcyjny interfejs do wprowadzania zmian;
— zgodność z poprzednimi wersjami składnic;
— obiektowo-zorientowany interfejs;
— możliwość definiowania własnych reguł, obiektów i związków między obiektami[36].

Składnica danych jest to aktywny, rozproszony słownik danych, który umożliwia kompleksowe zarządzanie metainformacją całego systemu infor-

[36] Ciekawą próbę zbudowania systemu, który ma takie cechy, podjął zespół prof. W. Olejniczaka na Uniwersytecie Szczecińskim (por. (104)).



Rys. 4.1. Koncepcja wytwarzania według AD/Cycle

matycznego. Wprowadza spójny i centralny opis danych wykorzystywanych w systemie przez różnorodne produkty, umożliwiając redukcję nadmiarowych i niekonsekwentnych definicji i zwiększając integralność danych.

Metainformacja przechowywana w składnicy zawiera takie atrybuty, jak położenie, typ, opis, format, rozmiar, i historie zmian. Każda zmiana metainformacji jest nadzorowana przez system słownika, co zapewnia wysoki stopień kontroli zarówno procesów zarządzania informacją, jak i tworzonych aplikacji, które korzystają z definicji zawartych w składnicy.

Tradycyjnie składnicą metainformacji był kod źródłowy każdej aplikacji. Potencjalnie każdy taki kod zawierał własne definicje danych. Nadmiarowość definicji i ich niespójność były więc regułą. Zmiana definicji w dowolnym programie mogła i najczęściej prowadziła do naruszenia integralności i spójności danych. Rozwiązaniem tego problemu jest centralne, wiarygodne i uniwersalne źródło definicji danych i innych obiektów występujących w systemie. Można wyróżnić trzy kategorie tego typu systemów:

— słownik pasywny,
— słownik dynamiczny,
— słownik aktywny.

Pasywny słownik danych służy jedynie do celów dokumentacji i tworzenia raportów. Informacja zawarta w słowniku nie może być wykorzystywana przez inne elementy systemu informatycznego, co bardzo ogranicza jego użyteczność.

W systemie wykorzystującym technologię dynamicznego słownika danych każda zmiana metainformacji jest natychmiast przekazana do wykorzystujących je aplikacji. Gwarantuje to zawsze aktualność wersji definicji. Programy sięgają do słownika po potrzebne definicje podczas uruchamiania i każdorazowo przy dostępie do danych. Główną wadą słownika dynamicznego jest jego ograniczona wydajność. Konieczność propagowania zmian definicji do wszystkich korzystających z niej aplikacji jest bardzo czasochłonna.

Metainformacja przechowywana w słowniku aktywnym jest wykorzystywana przez programy w procesie kompilacji. Zmiany definicji nie są odzwierciedlone w wykonywanym module aplikacji aż do czasu kolejnej kompilacji. Operacja ta może być natomiast wykonana bez ręcznego wprowadzania zmian definicji w kodzie źródłowym. Słownik tego typu nie zmniejsza wydajności systemu, metainformacje bowiem są wykorzystywane tylko w procesie kompilacji aplikacji.

Najczęściej składnica oferuje zalety wszystkich wymienionych typów słowników w zależności od sposobu jej wykorzystania. Ponadto jest

udostępniana architektura słownika rozproszonego, w której wiele fizycznie oddalonych słowników, istniejących na odrębnych węzłach sieci, pracuje jako jeden, spójny słownik metainformacji. Budowa składnicy wykorzystuje model obiekt — atrybut — relacja. Bezpośredni dostęp do obiektów przechowywanych w składnicy z poziomu języków programowania jest możliwy przez interfejs wywołań.

Z przedstawionej charakterystyki składnicy informacji wyraźnie widać, że jest to kluczowy element technologii CASE wytwarzania systemu informatycznego. Składnica gwarantuje integrację wewnętrzną systemu, a przez mechanizmy odpowiednich interfejsów zapewnia również integrację zewnętrzną tworzonego systemu i procesu wytwarzania w pełnym cyklu życia.

4.2.2. Pielęgnacja systemu (maintenance)

Liczne badania potwierdzają, że dużą część kosztów wytwarzania systemu informatycznego pochłania obsługa zmian i modyfikacji, które powstają w trakcie tworzenia systemu i jego eksploatacji użytkowej. Zmiany i modyfikacje wprowadzane do systemu są źródłem licznych komplikacji w procesie wytwarzania. Wspomaganie tych prac jest uzależnione od technologii wytwarzania.

Istota prawidłowego modyfikowania systemu tkwi w poprawnym przewidywaniu wpływu wprowadzonych zmian na inne elementy systemu. Sprowadza to się do niezbędnych zmian w innych fazach projektu. Wyróżnia się trzy podstawowe sytuacje:

— restrukturyzację systemu (restructuring);
— modyfikację wstępującą (rewerse engineering);
— modyfikację zstępującą (forward engineering).

Całkowita przebudowa projektu systemu może wynikać z potrzeby dostosowania systemu do istotnych zmian, jakie zaszły w obiekcie, gdzie stosowany jest system informatyczny. Ważne jest, aby restrukturyzację wykonać możliwie szybko, wykorzystując wcześniej opracowane elementy. Najczęściej wymaga to jednak przejścia przez cały cykl życia systemu.

Modyfikacje zstępujące to sytuacja, gdy zmiany we wstępnych fazach cyklu życia systemu implikują odpowiednie modyfikacje w dalszych fazach tego cyklu. Odwrotnie jest w przypadku modyfikacji wstępującej — zmiany są

inicjowane w fazie wdrożenia lub programowania, należy więc odpowiednio zmodyfikować specyfikację[37].

Stan faktyczny jest taki, że najczęściej brakuje aktualnej dokumentacji użytkowanych systemów. Aby temu zaradzić, rozwija się narzędzia, których podstawowym celem jest wytwarzanie specyfikacji na podstawie użytkowanego produktu. Działania takie (re-engineering) są niezbędne, gdy konieczna jest modyfikacja lub restrukturyzacja systemu nieodpowiednio udokumentowanego lub tworzonego w innej technologii. Dzięki możliwości powtórnego przekształcenia i modyfikacji można ochronić wcześniej poczynione inwestycje na prace projektowe. W pracy (82) opisano produkty firmy Bachman, które wchodzą w skład koncepcji IBM AD/Cycle, realizujące takie procedury.

4.2.3. Wspomaganie cyklu życia systemu

W literaturze można spotkać wiele różnych modeli cyklu życia systemu. Różnice między nimi nie są z reguły istotne i sprowadzają się do nazewnictwa lub podziału na fazy, etapy realizacji w takiej lub innej konfiguracji. Jednym z takich modeli jest przedstawiona metodyka punktów węzłowych.

Na potrzeby wspomagania metodyki punktów węzłowych stworzono prosty pakiet komputerowego wspomagania „EKSPERT" (130). Założenia i zasady metodyki zostały zaszyte w rozwiązaniach programowych pakietu programowego. Użytkownik pakietu ma możliwość twórczego rozwinięcia metodyki, poczynając od pewnego poziomu ogólności rozwiązań, zgodnie ze specyfiką tworzenia konkretnego systemu informatycznego. Pakiet ukierunkowany jest głównie na:

— kontrolę merytorycznej poprawności realizacji procesu tworzenia systemu;
— wspomaganie tworzenia i utrzymania w aktualnym stanie dokumentacji systemu[38].

Metodyka punktów węzłowych zaszyta w pakiecie programowym obejmuje pełny cykl życia systemu. W części objętej wspomaganiem komputerowym pakiet koncentruje uwagę na tym, jakie czynności należy wykonać i w jakim porządku w kolejnych fazach realizacji systemu informatycznego. Kontrola

[37] W działaniach tych bardzo pomocna może być sieć zależności między punktami węzłowymi przedstawiona na rys. 2.2 i opisana w ramach metodyki punktów węzłowych.

[38] Pakiet programowy „EKSPERT", autorstwa Z. Szyjewskiego i I. Szydłowskiego, jest produktem handlowym, oferowanym wraz z niezbędną dokumentacją przez Spółkę z o.o. „PROGRES".

poprawności realizacji sekwencji działań jest wymuszana na projektancie przez blokadę dostępu do kolejnych punktów węzłowych, jeżeli nie zostały wykonane wszystkie poprzedniki danego punktu węzłowego. Działania te to, zgodnie z propozycją opisaną w pracy (143), elementy filozofii restrykcyjnej.

W pakiecie jest zaszyta sieć zależności punktów węzłowych (opisana wcześniej), co umożliwia kontrolę poprawności logicznej realizacji poszczególnych czynności, ponadto wyświetla się informacja, jakie czynności można rozpocząć po zakończeniu prac nad danym punktem węzłowym. Lista możliwych do rozpoczęcia prac jest elementem filozofii przewodnikowej realizacji procesu. Elementem kontrolnym, który gwarantuje spójność realizacji, jest mechanizm akceptacji zakończenia prac nad danym punktem węzłowym. Istnieje możliwość zastrzeżenia mechanizmu odbioru zakończenia prac nad punktem węzłowym tylko dla kierownika projektu. Nie jest to kontrola programowa poprawności, ale jednoznacznie określa odpowiedzialność.

Pakiet jest przewodnikiem w realizacji systemu, który nie ogranicza inwencji twórców, gdyż oprócz zaszytych w nim powiązań między punktami węzłowymi, istnieje możliwość własnego kreowania czynności, poczynając od trzeciego poziomu ogólności punktów węzłowych. Użytkownik na podstawie własnej wiedzy i doświadczenia może określić skład czynności wchodzących w zakres punktu węzłowego lub skorzystać z wiedzy zaszytej w pakiecie. Własny opis zakresu tematycznego poszczególnych punktów węzłowych metodyki daje możliwość twórczej modyfikacji szkieletu metodyki zaszytego w pakiecie i adaptacji jej do konkretnych potrzeb tworzonego systemu informatycznego.

Pakiet „EKSPERT" nie tylko wspomaga kontrolę procesu realizacji, lecz również tworzenie dokumentacji systemu. Dostarcza bowiem wygodne mechanizmy edytorów tekstu w połączeniu ze standardowym spisem treści, który wywodzi się z listy punktów węzłowych (125). Wszelkie działania związane z realizacją systemu są zapisywane w jednym zbiorze danych, który można nazwać składnicą systemu. Mając na uwadze wszystkie usterki i stosunkowo prymitywne rozwiązania informatyczne, pakiet ten ma wszelkie cechy narzędzia wspomagania klasy CASE. Wykorzystanie dorobku teoretycznego w połączeniu z dostępnymi w kraju środkami informatyki daje próbę oryginalnego rozwiązania, które w prosty sposób może uczyć i propagować nowoczesną technologię wytwarzania.

Narzędzia wspomagające poszczególne fazy realizacji systemu informatycznego są klasyfikowane w zależności od tego, który etap prac wspomagają. Dzieli się je wówczas na:

— front-end CASE, narzędzia wspomagające fazę planowania, analizy i projektu,
— back-end CASE, narzędzia wspomagające fazę konstrukcji systemu, czyli projekt i oprogramowanie.

Inny podział to:
— upper-CASE,
— middle-CASE,
— lower-CASE,

dokonany w zależności od tego jaka faza cyklu życia jest wspomagana. Narzędzia klasy „lower" najczęściej wspomagają fazę kodowania, testowania i prototypowania, kompilacji programów oraz wdrożenia. Klasa „upper", to narzędzia wspomagające fazę planowania, analizy i specyfikacji[39].

4.3. Kierunki rozwoju narzędzi CASE

4.3.1. Rozwój oprogramowania narzędziowego

Żywiołowy rozwój narzędzi CASE spowodował konieczność ich uporządkowania, w efekcie powstały różne klasyfikacje. Niezależnie od tego zaczął rozwijać się rynek narzędzi CASE, a jego analiza może wskazywać na pewne trendy rozwojowe. W opracowaniu (34) dokonano wszechstronnego przeglądu bardzo różnych narzędzi komputerowego wspomagania. Ogólnie można wyróżnić dwie klasy produktów:
— pakiety narzędziowe (ang. toolkits);
— pakiety zintegrowane o charakterze zamkniętym (ang. workbenches).

Jednocześnie rozwijane jest oprogramowanie narzędziowe i operacyjne, które w dużej części przejmuje funkcje obsługiwane przez narzędzia CASE, głównie klasy „lower". Biorąc pod uwagę podstawowe znaczenie składnicy danych w metodologii CASE, która jest bazą danych tworzonego systemu, dla wspomagania procesu tworzenia systemu duże znaczenie mają możliwości

[39] Inne klasyfikacje i przykłady produktów, w tym krajowej produkcji, reprezentantów poszczególnych klas można znaleźć w pracy (43).

systemu zarządzania bazą danych, obsługującego składnicę. Jest to nowa jakość w tworzeniu systemu informatycznego, która każe inaczej spojrzeć na narzędzia CASE.

Dla wspomagania istotny jest rozwój systemów zarządzania bazą danych, które stają się pakietami wyposażonymi w bardzo mocne narzędzia wspomagania funkcji tworzenia systemu informatycznego. Wykorzystując mechanizmy systemu operacyjnego, użytkownikowi dostarcza się prosty mechanizm konstruowania, manipulowania i utrzymania bazy danych o systemie oraz realizacji algorytmów przetwarzania danych, będących przedmiotem działania tworzonego systemu. Aby ułatwić korzystanie z tych mechanizmów, wbudowano odpowiedni język komunikacji, pozwalający na skoncentrowanie się na zależnościach logicznych bez wikłania się w techniczne szczegóły. Języki te znane są w literaturze jako języki czwartej generacji (4GL), w odróżnieniu od dotychczas stosowanych języków proceduralnych, określanych jako języki trzeciej generacji (3GL).

Okazuje się, że coraz trudniej mówić o narzędziach CASE w oderwaniu od środowiska sprzętowo-programowego, w którym te narzędzia są zanurzone. Dużego znaczenia nabiera współdziałanie wielu elementów całego środowiska we wspomaganiu tworzenia systemu informatycznego. Wcześniej wskazywano na uzależnienie od metodyki postępowania, które wpływało na konceptualne uwarunkowania narzędzi CASE. Środowisko sprzętowo-programowe narzuca ograniczenia techniczno-technologiczne. Rozwój narzędzi CASE jest uzależniony od rozwoju środowiska i dostępnych mechanizmów. Wspomaganie jest realizowane nie jako jednorazowy przebieg jakiegoś pakietu programowego, ale odpowiednio przygotowane działanie kompleksowo wspomagające proces wytwarzania.

4.3.2. Systemy zarządzania bazą danych i języki 4GL

Patrząc na komputerowe narzędzia wspomagania tworzenia systemów informatycznych pod kątem praktyki postępowania, Z. Benedykt (w pracy (7)) wyróżnia trzy podstawowe elementy wspomagania:

— słownik danych,
— generatory programów,
— języki czwartej generacji.

Słownik danych to baza danych tworzonego systemu, gdzie zapisywane są wszystkie informacje potrzebne w procesie tworzenia, jest jednocześnie

kroniką całego procesu powstawania systemu. Z punktu widzenia procesu wytwarzania systemu istotne jest, jak są wspomagane procedury wykonywane na bazie danych przez system zarządzania bazą. Dobry system zarządzania bazą danych powinien zapewnić użytkownikowi:

1. **Wspomaganie wielodostępu.** Baza danych zawiera różne typy danych, logicznie powiązanych i pokrywających dość duży zakres tematyczny. Istotne jest umożliwienie wielu użytkownikom jednoczesnego korzystania z zasobów bazy danych, zarówno w trybie odczytu, jak i uaktualniania. Aby każdy z użytkowników mógł być pewny poprawnego wykonania polecenia skierowanego do bazy danych, system musi mieć mechanizmy synchronizujące ich pracę.

2. **Ochronę danych przed nielegalnym wykorzystaniem.** Dobry system musi zapewnić ochronę wszystkich zasobów, w tym danych zapisanych w bazie. Użytkownicy mają różne priorytety dostępu do danych, co najczęściej rozwiązuje odpowiedni system haseł.

3. **Ochronę danych przed awariami sprzętu lub systemu.** System powinien gwarantować poprawność danych niezależnie od mogących występować awarii sprzętu. Najczęściej osiąga się to przez okresowe kopiowanie zapisów z bazy lub bardziej wyrafinowane i droższe eksploatacyjnie duplikowanie danych. Wielokrotne aktualizowanie może również spowodować wewnętrzną niespójność. System zarządzania bazą danych powinien zawierać mechanizmy pozwalające na wykrywanie i lokalizowanie takich niezgodności.

4. **Obsługę wczytywania dużych ilości danych.** Umiejętna alokacja danych w bazach danych decyduje o jej parametrach wydolnościowych. System zarządzania powinien dostarczać mechanizmów programowych wspomagających twórcę systemu w odpowiednim umieszczaniu danych w bazie.

5. **Prowadzenie statystyki.** Mnogość różnorodnych operacji na bazie danych powoduje, że zmieniają się charakterystyki bazy w miarę jej użytkowania. Analiza tych zmian jest konieczna, aby zapewnić odpowiednie parametry funkcjonowania bazy. Do prowadzenia analiz i podejmowania działań zaradczych konieczne są odpowiednie statystyki informujące o zaistniałych faktach i przyczynach ich powstania.

6. **Możliwość restrukturyzacji.** Pielęgnacja systemu może zmusić do wprowadzenia zmian w strukturze logicznej bazy danych. Operacja taka jest niezwykle skomplikowana i należy ją przeprowadzać ze szczególną ostrożnością.

7. **Możliwość reorganizacji.** Reorganizacja w odróżnieniu od restrukturyzacji nie dotyczy logicznej struktury bazy danych, lecz jedynie jej struktury

fizycznej. Operacja ta jest prostsza, ale z uwagi na rozmiary baz danych wymaga ostrożności, by nie zgubić danych lub nie wprowadzić niespójności.

Na rynku oprogramowania jest wiele różnych systemów zarządzania bazą danych. Dystrybutorami są znane firmy komputerowe i firmy, których jedynym produktem rynkowym jest system zarządzania bazą danych mogący być użytkowany w różnych środowiskach sprzętowych. Przykładem pierwszej grupy produktów są systemy DB2 firmy IBM lub system Rdb firmy DIGITAL. Przykładem drugiej grupy produktów są systemy ORACLE, INGRES czy PROGRESS. Większość oferowanych systemów wykorzystuje relacyjny model bazy danych. Systemy te w różny sposób realizują obsługę oczekiwań użytkownika, co powoduje, że każdy z nich ma swoich zwolenników i przeciwników. Dokonuje się różnych zestawień i porównań zalet i wad poszczególnych systemów, co ma być pomocne w wyborze odpowiedniego systemu do obsługi odpowiedniej aplikacji. Wybór systemu dla konkretnego zastosowania jest zwykle bardzo trudny z uwagi na różnorodne kryteria oraz wiele czynników wpływających na użyteczność i parametry eksploatacyjne poszczególnych rozwiązań.

W raporcie (12) dokonano porównania sześciu najpopularniejszych produktów największych firm:

— Allbase/SQL firmy Hewlett Packard,
— DB2 firmy IBM,
— Ingres firmy Ingres,
— Oracle firmy Oracle,
— Rdb firmy Digital,
— Sybase firmy Sybase Software.

Kryteria porównania były bardzo zróżnicowane — od zastosowań w czterech różnych typach aplikacji, szybkości realizacji, do możliwości przenoszenia w różne środowiska systemów operacyjnych. Wyniki, przedstawione w tabelach i na wykresach, obrazują, w jakich zastosowaniach, jakie efekty daje stosowanie każdego z badanych systemów. Trudności z oceną tych systemów wynikają nie tylko z tego, że w prowadzonych zestawieniach uzyskano różne wyniki, ale również z faktu, że systemy te są ciągle doskonalone, na rynku pojawiają się coraz nowsze wersje, które eliminują poprzednie usterki i stwarzają nowe możliwości.

Próbę porównania popularnych na polskim rynku systemów zarządzania bazą danych podjęto w ramach Górskiej Szkoły PTI (121). Rozważania teoretyczne i próby praktyczne również nie dały jednoznacznej odpowiedzi, który z badanych systemów jest najlepszy.

Systemy zarządzania bazą danych są ciągle rozwijane, dostarczają użytkownikom coraz wygodniejszych narzędzi tworzenia systemów informatycznych. Systemy najczęściej wykorzystują mechanizmy systemu operacyjnego i innych programów narzędziowych, realizując typowe funkcje operowania na danych. Z drugiej strony doskonalone są metody komunikowania się z użytkownikiem, tak aby definiowanie zadań dla systemu było zbliżone do języka naturalnego. W ten sposób system zarządzania bazą danych staje się łatwym w użyciu pośrednikiem w wykorzystaniu skomplikowanych mechanizmów oprogramowania operacyjnego.

Użytkownicy muszą mieć możliwość natychmiastowej reakcji na zmiany zachodzące w otoczeniu. Znaczne uzależnienie się od systemów informatycznych powoduje, że reakcja ta związana jest zazwyczaj z nagłą zmianą w oprogramowaniu.

Projektant aplikacji dąży do podnoszenia własnej produktywności. Nadzieją na to są narzędzia 4GL, które gwarantują, że oprogramowanie zostanie wytworzone szybciej i będzie lepszej jakości. Języki czwartej generacji są naturalnym rozwojem generatorów programów; postrzega się je jako pierwszy krok w świat narzędzi CASE.

Podobnie jak wiele innych pojęć związanych z informatyką, języki czwartej generacji nie mają jednoznacznej i powszechnie przyjętej definicji. Nazwa zawiera w sobie elementy reklamy, a w środowisku informatycznym jest rozumiana intuicyjnie. Różne produkty określane tym mianem mają następujące cechy wspólne:

— udostępnianie przez język podstawowych własności obecnych w językach poprzednich generacji, takich jak: możliwość budowania iteracji, instrukcja wyboru, funkcje i procedury;
— istnienie zdań umożliwiających dostęp na bardzo wysokim poziomie abstrakcji do danych oraz realizacja funkcji kontroli poprawności, spójności i bezpieczeństwa danych;
— język ma możliwość formatowania i obsługi ekranu;
— możliwość generacji raportów na podstawie danych zawartych w bazach danych;
— możliwość pracy wielu użytkowników i mechanizmy synchronizacji dostępu do danych;
— dostęp do funkcji systemu operacyjnego z poziomu języka;
— istnienie zintegrowanego środowiska rozwoju aplikacji (42).

Język czwartej generacji, który jest użyteczny w prostych aplikacjach, może nie być dość silny w bardziej skomplikowanych. Dla przykładu, niektóre języki

czwartej generacji pozwalają na interaktywne tworzenie prostych projektów, ale wymagają „ręcznej" korekty kodu i rekompilacji przy wykorzystaniu bardziej skomplikowanych funkcji. Język czwartej generacji powinien sprostać następującym wymaganiom:

— technologii **klient-serwer;**
— szybkiego tworzenia aplikacji;
— łatwego dostępu do bazy danych.

Główną cechą charakterystyczną języków czwartej generacji jest ich deklaratywność. Programista deklaruje, co chce uzyskać, a problemem systemu jest, jak to wykonać. Zazwyczaj niecały program jest jednostką deklaratywną[40]. Są w tych językach mechanizmy pozwalające określić kolejność wykonania poszczególnych poleceń. Polecenia te dotyczą dużych jednostek semantycznych, jak aktualizacja pól czy aktualizacja zmiennych programu, wyszukiwanie danych w bazie, organizacja ekranu itp. Dla programisty są to niepodzielne jednostki semantyczne. W językach czwartej generacji można używać zmiennych, dzielić program na podprogramy, używać procedur podobnie jak w językach proceduralnych.

Obecna presja rynku powoduje, że programiści są zmuszeni do skrócenia cyklu tworzenia aplikacji, tak aby jak najszybciej osiągnąć końcowy efekt. Strategia szybkiego tworzenia aplikacji (RAD — Rapid Application Development), umożliwia osiągnięcie tego celu, dzięki temu, że:

— twórcy systemu w fazie analizy i projektu mają ciągły kontakt z użytkownikiem;
— szybkie jest prototypowanie i wielokrotne poprawki przed fazą produkcji kodu;
— jest stosowana predefinicja, czyli krótki czas do oddania produktu do eksploatacji.

Cały ten proces tworzenia oprogramowania (cykl życia) realizuje jednocześnie wielu programistów. W tradycyjnym modelu każda faza była wykonywana oddzielnie. W modelu RAD projekt, kodowanie i inne funkcje nakładają się na obszary, które były poprzednio pozostawione na koniec projektu. To równoległe tworzenie znacznie redukuje czas potrzebny na wykonanie aplikacji[41].

---

[40] Typowym przykładem jednostki deklaratywnej może być polecenie SELECT w SQL.

[41] Przykładowo, pakiet DEC RALLY używając danych z bazy Rdb/VMS, może wygenerować całą aplikację w sposób domyślny. Dzięki obiektowemu charakterowi, RALLY pozwala na zmianę nawet skomplikowanego programu w ciągu kilku minut.

Natychmiastowa reakcja na żądanie użytkownika, możliwa dzięki stosowaniu narzędzi 4GL, nie tylko skraca czas tworzenia, ale również pomaga w ustaleniu kierunku rozwoju aplikacji. Właściwość szybkiego prototypowania minimalizuje koszty pomyłek i pomaga w zrozumieniu tego, co wyobraża sobie programista, a co użytkownik o tworzonej aplikacji. Język 4GL umożliwia wspólne tworzenie aplikacji. W przypadku zmiany wymagań projektant może szybko stworzyć prototyp i nową wersję aplikacji.

4.3.3. Środowisko COHESION

4.3.3.1. Założenia koncepcji środowiska COHESION

Informatyzacja przedsiębiorstw, urzędów państwowych, banków i innych podmiotów gospodarczych powinna być realizowana szybko i sprawnie w zmieniającej się rzeczywistości gospodarczej. Przyszli użytkownicy systemów informatycznych mają do wyboru dwie drogi szybkiej realizacji koncepcji informatyzacji: adaptacja gotowego rozwiązania lub tworzenie systemu „na miarę" za pomocą nowoczesnej technologii wytwarzania. Pierwsze rozwiązanie najczęściej nie w pełni spełnia oczekiwania użytkownika, a drugie jest związane z pełną pułapek, ryzykowną drogą tworzenia systemu za pomocą narzędzi używanych po raz pierwszy lub tylko częściowo opanowanych przez twórców.

Nowoczesne technologie wytwarzania to języki czwartej generacji, systemy zarządzania bazą danych oraz strukturalna analiza i projektowanie. Pomimo niezaprzeczalnych zalet tych technik i stosowania narzędzi CASE często nie osiąga się pełnego sukcesu. Oferowane na rynku narzędzia CASE mogą być stosowane w różnych fazach realizacji. Twórca ma możliwość wyboru z gamy różnorodnych dobrych narzędzi. Trudności zaczynają się wówczas, gdy trzeba dobrać narzędzia działające sprawnie w pełnym cyklu życia projektu, przekazujące wyniki z jednego etapu do drugiego. Często okazuje się, że nie można tego zrealizować zgodnie z potrzebami twórców.

W odpowiedzi na to wyzwanie stworzono w pełni zintegrowane środowisko CASE (Integrated Computer Aidded Software Engineering — ICASE) (108). Podjęła się tego firma DIGITAL przedstawiając strategię COHESION.

COHESION jest zbiorem produktów firmy DIGITAL tworzących środowisko CASE do produkowania oprogramowania, jego rozwijania i zarządzania nim. Centralnym elementem tego środowiska jest wspólny słownik

— CDD/Repository (rozwinięta wersja wcześniejszego CDD/Plus) (17). CDD/Repository to aktywny, rozproszony słownik, który umożliwia organizację, zarządzanie, kontrolę i integralność wszystkich narzędzi i aplikacji.

COHESION obejmuje cały proces tworzenia oprogramowania, od analizy potrzeb i projektu aż po zarządzanie działającym systemem. Podstawowym celem budowy tego środowiska jest stworzenie warunków do szybkiego tworzenia oprogramowania o wysokiej jakości. Charakterystyczne cechy środowiska, to:

— możliwość tworzenia wszystkich typów aplikacji;
— heterogeniczna platforma sieciowa, która umożliwia połączenie różnorodnego pod względem mocy obliczeniowej i źródeł pochodzenia sprzętu komputerowego;
— różne platformy docelowe tworzonego oprogramowania, co oznacza, że tworzenie oprogramowania na jakiejś platformie sprzętowo-programowej nie ogranicza możliwości stosowania produktów tylko do tej platformy; praktyczna realizacja możliwości przenoszenia oprogramowania — cecha jakościowa postulowana przez inżynierię oprogramowania;
— zintegrowana składnica danych;
— konsultacje, szkolenie i współpraca z przyszłym użytkownikiem, co gwarantuje spełnienie oczekiwań użytkownika.

Na środowisko COHESION składają się cztery podstawowe elementy:

— NAS (Network Application Support);
— struktura integracji narzędzi CASE;
— narzędzia CASE;
— wsparcie i usługi.

Architektura NAS, przedstawiona na rysunku 3.2, umożliwia tworzenie przenośnych aplikacji w rozproszonym, heterogenicznym środowisku sieciowym. Network Application Support jest ideą i zbiorem programów usługowych, które tworzą model projektowania i rozwijania aplikacji działających w środowisku sieciowym. Założenie rozproszenia i połączeń sieciowych jest podstawowe dla tej koncepcji. Dodatkowym założeniem jest możliwość włączenia w sieć dowolnego rodzaju sprzętu komputerowego. Użytkownik włączający się z mikrokomputerem produkcji firmy Macintosh ma mieć takie same możliwości komunikowania się ze wszystkimi zasobami, jak za pomocą dużego komputera VAX czy IBM. Architektura NAS, oparta na koncepcji klient — serwer[42], umożliwia swobodne roz-

[42] Klient — serwer to specyficzny sposób połączeń logicznych w sieci komputerowej, polegający na tym, że istnieje jeden serwer lub wiele obsługujących wspólne zasoby i wielu klientów, zapewniających przede wszystkim bezpośredni interfejs z użytkownikiem.

budowywanie sieci o nowych użytkowników korzystających ze wspólnych zasobów. Komunikacja jest gwarantowana dzięki pełnej zgodności NAS z większością międzynarodowych standardów dotyczących systemów informatycznych. W jednej sieci komputerowej mogą pracować komputery VAX, DEC, IBM czy Macintosh, obsługiwane przez różne systemy operacyjne, takie jak VMS, UNIX, OS czy MS-DOS. Widać z tego, że architektura NAS nie ogranicza się tylko do produktów firmy DIGITAL, choć je oczywiście faworyzuje, ale umożliwia włączenie popularnego sprzętu innych producentów. Twórca systemu informatycznego, który ma takie możliwości techniczne, może się skoncentrować na rozwiązaniach logicznych.

Struktura integracji narzędzi CASE jest realizowana na trzech płaszczyznach:

— prezentacji,
— kontroli,
— danych.

Integracja na poziomie prezentacji odnosi się do wspólnego interfejsu użytkowego wszystkich narzędzi i aplikacji. Obsługa tej integracji jest realizowana przez DECwindows Motif[43].

Integracja na poziomie kontroli i współdziałania aplikacji, to odpowiednie procedury programowe, które umożliwiają współdziałanie, kontrolę i komunikację między aplikacjami.

Integrację na poziomie danych gwarantują CDD/Repository, co pozwala na wspólne wykorzystywanie danych przez aplikacje w pełnym cyklu życia tworzonego systemu.

Narzędzia CASE produkowane przez DIGITAL i inne firmy pokrywają pełny cykl życia systemu. W pracy (115) przedstawiono wykaz głównych reprezentantów narzędzi CASE wspomagających poszczególne fazy realizacji systemu[44]. Znalazły się tam narzędzia bardzo wyrafinowane, produkcji firmy DIGITAL i jej partnerów, oraz dość powszechnie znane i stosowane kompilatory i elementy oprogramowania narzędziowego.

Środowisko COHESION (firmy Digital) oferuje użytkownikowi wsparcie i usługi w dziedzinie konsultacji i szkolenia. Pozwala to na wspólne definiowanie zadań przez zespół wykonawców oraz wzajemne wspomaganie się w całym procesie tworzenia i wdrażania systemu.

---

[43] Rozwiązanie analogiczne do CUA w koncepcji SAA firmy IBM.

[44] Pełny wykaz narzędzi wspomagających jest podany w formie kilkustronicowej tabeli 1.1 w opracowaniu (17), zawarto tam również krótkie charakterystyki poszczególnych narzędzi.

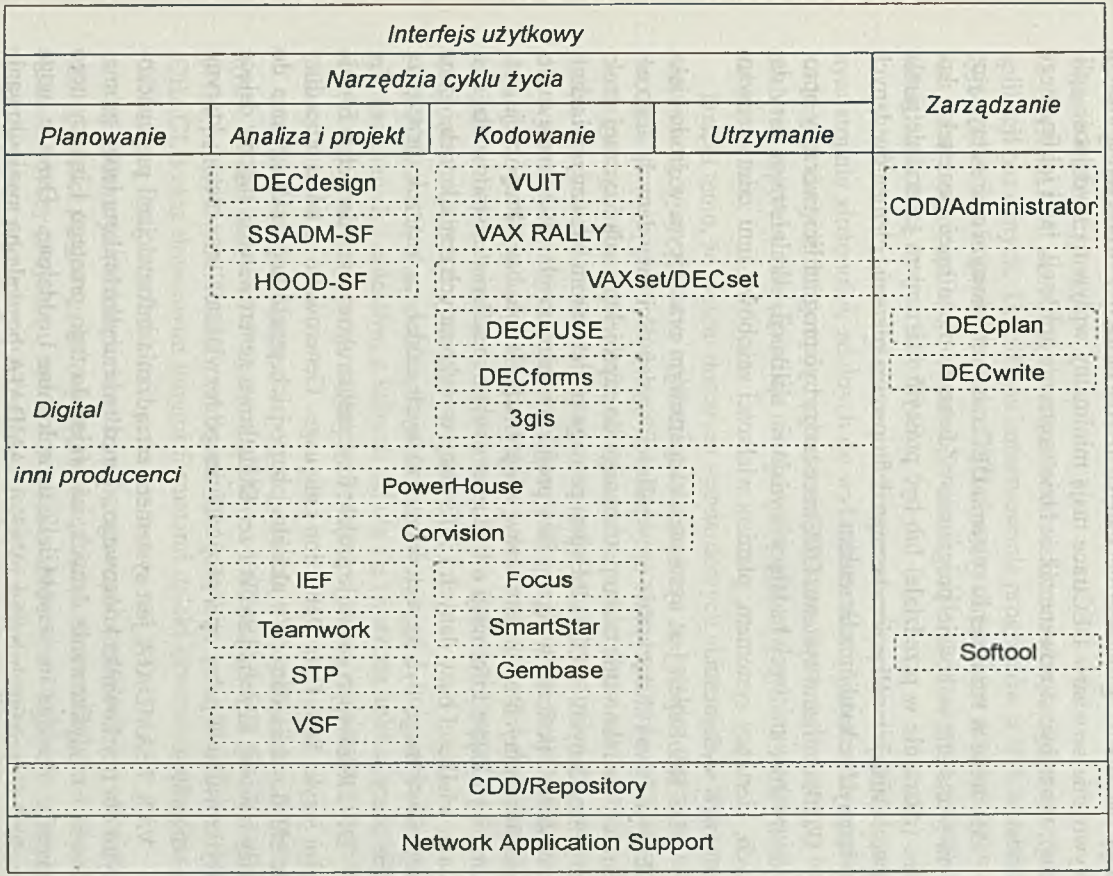
4.3.3.2. Narzędzia wspomagania firmy DIGITAL

Środowisko COHESION jest zbiorem narzędzi o różnym stopniu złożoności. Podstawowym podzbiorem tego zbioru są narzędzia produkcji firmy DIGITAL, opracowane w środowisku systemu operacyjnego VMS, kierowane do rodziny komputerów VAX. Z uwagi na otwartość rozwiązań nie jest to ograniczenie. Stosowana architektura gwarantuje współpracę sieci lokalnych i rozległych różnych firm. W ramach sieci mogą współpracować ze sobą maszyny z różnymi systemami operacyjnymi: VMS, ULTRIX, UNIX wielu producentów, DOS, OS/2, Macintosh. Możliwy jest też dostęp do dużych maszyn IBM pracujących w ramach sieci SNA. Dzięki temu fakt, że narzędzia te pracują pod systemem VMS na maszynach VAX, nie jest żadnym ograniczeniem. Stosując architekturę klient — serwer, możliwy jest dostęp z dowolnego węzła sieci, na przykład DOS, OS/2. Serwer pracuje na maszynie VAX w pełni wykorzystując bogate możliwości tych komputerów, takie jak wieloprocesorowość, VAXcluster, zwiększone bezpieczeństwo danych itp. Z drugiej strony aplikacje klienta pracują w środowisku znanym dla użytkownika (MSWindows, Macintosh).

Rysunek 4.2 przedstawia fazy cyklu życia systemu, które są wspomagane przez wybrane narzędzia CASE produkcji firmy DIGITAL i innych producentów. Podstawowe produkty to:

— DECtrace,
— DEC RdbExpert,
— VAX TEAMDATA,
— VAX RALLY,
— DECforms,
— VAXset,
— VAX Language-Sensitive Editor/Source Code Analyzer,
— VAX Performance and Coverage Analyzer,
— VAX DEC/Test Manager,
— VAX ACMS,
— DECtp Desktop for ACMS,
— VAX Software Project Manager,
— DECdesign.

DECtrace jest produktem umożliwiającym gromadzenie danych generowanych przez określone zdarzenia, które występują podczas wykonania aplikacji. Dane te mogą być pomocne w analizie wydajności, planowaniu wymagań, optymalizowaniu bazy danych i wielu innych przypadkach, gdzie analiza danych



Rys. 4.2. Narzędzia środowiska COHESION (115)

o zdarzeniach może być użyteczna. DECtrace gromadzi te dane z każdej aplikacji, która ma wywołania odpowiednich serwisów DECtrace. Programiści mogą umieszczać takie wywołania we własnych aplikacjach, dzięki czemu mają możliwość gromadzenia danych o zdarzeniach z własnych aplikacji. Wywołania serwisów DECtrace mają minimalny wpływ na wydajność aplikacji, mogą być stosowane tak w fazie tworzenia aplikacji, jak i jej finalnego działania.

Zdarzenie w rozumieniu systemu DECtrace jest pewnym charakterystycznym miejscem w kodzie programu. Zdarzenie może mieć początek i koniec (zdarzenie w przedziale) lub być prostym zdarzeniem (zdarzenie punktowe). Istnieje możliwość własnego definiowania zdarzeń i elementów danych związanych z każdym zdarzeniem.

Użytkownikami systemu DECtrace mogą być programiści tworzący oprogramowanie, analitycy badający wydajność aplikacji, administrator bazy danych, kierujący systemem, planista wielkości zasobów i inni zainteresowani danymi o wydajności.

DEC RdbExpert jest systemem ekspertowym pracującym w środowisku DECwindows i służącym do optymalizacji wydajności bazy danych. Jego celem jest zredukowanie czasu potrzebnego do zaprojektowania fizycznej struktury bazy danych. DEC RdbExpert pomaga projektantom baz danych i administratorowi podczas wstępnej fazy projektowania danych. Wspomaga mniej zaawansowanych administratorów w optymalizowaniu baz. Propozycje systemu wynikają z informacji o logicznej strukturze schematu, środowisku systemu, wielkości bazy danych i informacji o wykonanych transakcjach. Dane wejściowe mogą być importowane z różnych źródeł, co znacznie oszczędza czas pracy.

DEC RdbExpert generuje projekt fizycznej struktury bazy danych w formie zdań języka SQL Data Definition Language. Generowane są także procedury do składowania danych z aktualnej bazy i ich powtórnego załadowania do bazy o nowej fizycznej strukturze. Dodatkowo generowane są raporty dające użytkownikowi pełny opis proponowanego rozwiązania oraz metod, którymi je osiągnięto.

VAX TEAMDATA jest systemem zarządzania informacjami przeznaczonymi dla użytkownika końcowego. Umożliwia użytkownikom łatwe gromadzenie i modyfikowanie danych, używając do tego prostego i intuicyjnego aparatu. Pomaga im współdzielić dane lokalne i oddalone. Dane te mogą pochodzić z różnych źródeł. VAX TEAMDATA pozwala na manipulowanie własnymi, współdzielonymi i odległymi danymi w postaci tablic, arkuszy kal-

kulacyjnych, raportów, grafiki. Użytkownicy mogą dokonywać tych operacji korzystając z systemu menu lub używając komend do ich tworzenia. Dane mogą być gromadzone, sortowane, zmieniane. Istnieje możliwość zadawania pytań i generowania raportów. TEAMDATA ma również podstawowe cechy arkusza kalkulacyjnego.

VAX RALLY jest zintegrowanym środowiskiem służącym do generowania aplikacji baz danych. Umożliwia konstruowanie programów w trybie interakcyjnym bez konieczności stosowania tradycyjnych języków programowania. Jest generatorem aplikacji czwartej generacji, ściśle związanym z systemem zarządzania bazą danych Rdb/VMS.

VAX RALLY stanowi platformę do konstruowania aplikacji o zróżnicowanym stopniu złożoności, zdolnych do wykonania skomplikowanych operacji na bazach danych, które mają rozbudowany system menu, pełny system formularzy, raportów i objaśnień dla użytkownika oraz wielopoziomową strukturę zabezpieczeń.

Dzięki temu, że system dostarcza standardowych elementów, a tryb pracy jest interakcyjny, prototyp aplikacji może powstać w krótkim czasie w bardzo efektywny sposób. W następnym etapie procesu tworzenia, prototyp ten może być rozbudowany i udoskonalony zgodnie z potrzebami użytkownika. Ta forma pracy nad tworzeniem oprogramowania może być atrakcyjna zarówno dla wykwalifikowanych programistów, którzy w krótkim czasie otrzymują działający prototyp aplikacji, podlegający następnie testom i doskonaleniu, jak i dla mniej doświadczonego personelu, który otrzymuje narzędzia nie wymagające umiejętności programowania.

VAX RALLY dostarcza narzędzi do interakcyjnego definiowania i tworzenia baz danych i obiektów, z których składa się tworzona aplikacja: definicji źródeł danych, formularzy, raportów, systemów menu, objaśnień i systemu zabezpieczeń.

Możliwa jest jednoczesna współpraca aplikacji VAX RALLY z różnorodnymi bazami danych i formatami plików: VAX Rdb/VMS, RMS, DBMS przez pakiet DATATRIEVE, tablicami TEAMDATA, bazą DB2 firmy IBM, ORACLE oraz dowolnymi innymi formatami dzięki otwartemu interfejsowi ODI (Open Data Interface).

DECforms jest zbiorem narzędzi programowych umożliwiających tworzenie tej części programu, która jest odpowiedzialna za interakcje z użytkownikiem. Dotyczy to systemu menu, formularzy i raportów. Aplikacje mogą być wyświetlane na dowolnych terminalach w pełni wykorzystując ich szczególne właściwości, jak kolor, liczba wierszy i kolumn.

DECforms jest implementacją standardu proponowanego przez ANSI/ISO o nazwie Form Interface Management System (FIMS). Podstawowymi cechami modelu FIMS, zaimplementowanymi w DECforms, są:
— odseparowanie formularzy od funkcjonalnej części programu;
— rozproszenie przetwarzania formularzy;
— łatwość użycia;
— niezależność od języka programowania.

Pakiet VAXset (Software Engineering Tools) składa się z pięciu produktów:
— VAX Language-Sensitive Editor/Source Code Analyzer,
— VAX Performance and Coverage Analyzer,
— VAX DEC/CMS,
— VAX DEC/MMS,
— VAX DEC/Test Manager.

Produkty te zawierają zbiory narzędzi wspomagające pracę inżyniera programisty w kolejnych etapach procesu tworzenia oprogramowania: projekt, tworzenie, testowanie i utrzymanie gotowej aplikacji.

VAX LSE/SCA jest zintegrowanym środowiskiem programowym, które daje programistom możliwość edytowania i statycznej analizy kodu programu. Jest wielojęzykowym, wielomodułowym i wielooknowym narzędziem, niezbędnym do projektowania, tworzenia i utrzymania programów. Pakiet VAX LSE/SCA składa się z dwóch komponentów:
— wrażliwego na język edytora (LSE);
— analizatora kodu źródłowego (SCA).

LSE ma szablony składniowe kodu źródłowego i elementy projektowania na niskim poziomie, wykorzystując wbudowany pseudokod. Umożliwia również prezentowanie kodu na różnych poziomach szczegółowości, dostęp do pomocnika z informacjami specyficznymi dla danego języka i używanych pakietów bibliotecznych.

SCA wspomaga użytkownika w zrozumieniu skomplikowanego programu. Umożliwia analizowanie całego systemu lub poszczególnych modułów w fazie tworzenia i utrzymania programu. Pozwala na statyczną analizę kodu programu lub pseudokodu. Informacje dla SCA powstają w trakcie kompilacji kodu źródłowego programu przez kompilatory języków systemu VAX.

VAX LSE/SCA pracuje w środowisku DECwindows lub na standardowych terminalach. Użytkownik może redefiniować pewne elementy modułu LSE. Zmiany mogą być zapamiętane i wykorzystywane w przyszłości.

VAX Performance and Coverage Analyzer (VAX PCA) jest narzędziem umożliwiającym analizę aplikacji podczas wykonania. VAX PCA odgrywa

dwie role. Może wskazać na wąskie gardła i inne problemy efektywnościowe podczas wykonania aplikacji oraz pozwala na testowanie wybranych części programu za pomocą różnych testów.

VAX PCA pracuje w trybie tekstowym na standardowych terminalach alfanumerycznych oraz w środowisku DECwindows. VAX PCA występuje w ramach pakietu VAXset wraz z innymi narzędziami wspomagającymi pracę programistów. Szczególnie przydatny jest w fazie testowania programów.

VAX PCA składa się z dwóch części. Pierwsza to Collector, który gromadzi dane cząstkowe działającej aplikacji. Druga to Analyzer, który umożliwia analizowanie i wyświetlanie zgromadzonych w poprzedniej fazie danych. Collector gromadzi dane i zapisuje je w zbiorze. Analyzer pobiera ten zbiór jako dane wejściowe i wyświetla je w formie wykresów i tabel.

VAX DEC/CMS (Code Management System) jest systemem bibliotecznym ułatwiającym tworzenie i utrzymywanie oprogramowania. Aplikacja jest dzielona na wiele różnych funkcjonalnie klas, w których skład wchodzą zbiory dyskowe. DEC/CMS pomaga zarządzać tymi zbiorami przez ich gromadzenie w bibliotece projektu. Gromadzi również informacje o zmianach i monitoruje dostęp do biblioteki. Umożliwia także manipulowanie zbiorami wewnątrz bibliotek i między bibliotekami. Czynności te można sformalizować i wykonywać automatycznie.

VAX DEC/MMS (Module Management System) jest narzędziem zwiększającym produktywność procesu tworzenia oprogramowania. Ma możliwość powiązania zmienionych komponentów projektu i optymalnie regeneruje aplikacje. Kiedy pewne moduły aplikacji zostały zmienione, zależne moduły powinny zostać przekompilowane. VAX DEC/MMS zapamiętuje, które moduły są zrekompilowane, aby aplikacja powstała z konsolidacji i kompilacji ostatnich wersji modułów.

VAX DEC/Test Manager (VAX DTM) automatyzuje proces testowania programów przez wykonywanie zdefiniowanych przez użytkownika testów i porównywanie ich rezultatów z oczekiwaniami. VAX DTM daje użytkownikowi swobodę w organizacji testów, wyborze testów do wykonania oraz w ich weryfikacji i analizie rezultatów. Aplikacje wsadowe i interakcyjne, tak tekstowe, jak i pod DECwindows, mogą być testowane za pomocą VAX DTM.

VAX DTM daje użytkownikom następujące możliwości:

— tworzenie opisów testów i gromadzenie zbiorów w katalogach VMS lub bibliotekach CMS; katalogi lub biblioteki mogą znajdować się na oddalonych maszynach w sieci;

— łączenie testów w grupy, które można następnie wykorzystywać wspólnie;
— wyświetlanie i modyfikowanie opisów testów lub grup testów;
— specyfikowanie zbiorów komend jako szablonów testów do sprawdzania programów w trybie interakcyjnym;
— wykonywanie specyficznych testów, grup testów lub kombinacji tych grup;
— zapamiętanie wyników wykonania aplikacji, powstałego podczas nagrywania testu lub konkretnego wykonania jako wzoru, z którym będą porównywane rezultaty następnych testów;
— porównywanie rezultatów testu z testem wzorcowym; w wyniku wyświetlane są różnice obu testów;
— zmiana testów wzorcowych.

VAX ACMS jest monitorem systemów transakcyjnych tworzonych w środowisku systemu operacyjnego VMS, dając użytkownikom kompletne i elastyczne środowisko uruchomieniowe i narzędziowe do tworzenia systemów transakcyjnych, takich jak: śledzenie zleceń, rachunkowość, obsługa sklepu czy system ubezpieczeń. VAX ACMS wspomaga cały proces tworzenia systemu transakcyjnego — od instalacji, planowania, projektowania przez tworzenie, testowanie, zarządzanie i nadzorowanie skomplikowanych aplikacji transakcyjnych.

System VAX ACMS umożliwia tworzenie i projektowanie aplikacji w sposób modularny. Użytkownik ma możliwość tworzenia aplikacji charakteryzujących się rozproszonym przetwarzaniem. Część aplikacji związana z operacjami wejścia-wyjścia, czyli przetwarzanie formularzy (front-end processing), może być oddzielona od części odpowiedzialnej za obliczenia i dostęp do bazy danych (back-end processing).

VAX ACMS używa języka wysokiego poziomu o składni zbliżonej do języka naturalnego. Zawiera kompletny system uruchomieniowy do wykonywania definicji aplikacji. Ma narzędzia dla definiowania, monitorowania i modyfikowania charakterystyk aplikacji. W systemie użytkowym terminal, menu i inne operacje wejścia-wyjścia są odseparowane od bazy danych, obsługi zbiorów i funkcji obliczeniowych.

VAX ACMS ma możliwości uruchamiania i kończenia transakcji oraz samonaprawy bazy danych, wykorzystując do tego właściwości systemu zarządzania bazą danych. System współpracuje ze wspólnym słownikiem danych. Mechanizm kolejek systemu umożliwia użytkownikom pełną kontrolę nad zadaniami w kolejkach. Zawiera bogaty system dokumentacji, która dotyczy nie tylko opisu samego systemu, ale także jest pełnym studium tworzenia systemów transakcyjnych.

DECtp Desktop for VMS (nazywane także Desktop ACMS) jest produktem umożliwiającym aplikacjom, które pracują w środowisku systemów DOS lub Macintosh, dostęp do systemu transakcyjnego, opartego na ACMS i pracującego na maszynach VAX/VMS.

Desktop ACMS wykorzystuje architekturę klient — serwer. Część oprogramowania rezyduje na maszynie VMS (serwer), podczas gdy druga udostępnia tworzenie aplikacji na maszynie klienta (DOS, Macintosh). Maszyny klienta powinny działać w sieci na podstawie produktów serii PATHWORKS[45].

Desktop ACMS ma zbiór narzędzi i bibliotek niezbędny do tworzenia oprogramowania współpracującego bezpośrednio z aplikacjami ACMS. Narzędzia te pozwalają na pisanie programów "klienckich" bez specjalnej znajomości komunikacji w sieci czy sposobu, w jaki jest tworzona aplikacja transakcyjna.

VAX Software Project Manager (SPM) jest zintegrowanym, graficznym, wielodostępnym systemem zarządzania projektami, ułatwiającym planowanie, szacowanie kosztów i kontrolę nad średniej i dużej wielkości projektami programowymi. SMP automatyzuje wiele czasochłonnych czynności związanych z planowaniem i zarządzaniem projektami oraz ułatwia i systematyzuje analizę kosztów i zasobów związanych z projektem.

Dostępne narzędzia analizują i przetwarzają dane dostarczane przez projektanta oraz ustalają przybliżony koszt, czas trwania, niezbędny personel i inne nakłady konieczne do realizacji projektu. Wyniki mogą być prezentowane w postaci graficznej lub raportów.

DECdesign jest programem wspomagającym proces tworzenia oprogramowania w fazie analizy i projektu, pracującym w środowisku DECwindows. Dzięki graficznej prezentacji, modele użyte do analizy i projektu opisują tworzenie aplikacji w taki sposób, jak rysunek techniczny opisuje budowę budynku. Inżynier programista używając DECdesign ma możliwość tworzenia i modyfikowania modeli opisujących aplikację na podstawie reguł danej techniki modelowania. DECdesign ma możliwość współpracy z trzema technikami modelowania:

— Metodą Yourdona (106),
— Metodą Gane'a & Sarsona (119),
— Metodą MERISE (85).

DECdesign zawiera system uczący, umożliwiający użytkownikom samodzielne uczenie.

[45] PATHWORKS jest zbiorem produktów dla architektury klient — serwer i różnych systemów operacyjnych oraz platform sprzętowych, które umożliwiają integrację komputerów personalnych w sieciach lokalnych i rozległych (107).

Zestawienie porównawcze (tabela 4.1) przedstawia wspomaganie różnymi narzędziami CASE poszczególnych etapów metodyki punktów węzłowych. W górnej części tabeli, opracowanej na podstawie tabeli z pracy (147), zestawiono narzędzia różnych producentów oraz z przeznaczeniem na różne płaszczyzny sprzętowe i programowe. W dolnej części tabeli zestawiono narzędzia wchodzące w skład opisanego środowiska COHESION. Analizując zestawienie, można zauważyć, że wszystkie narzędzia wspomagają dokumentowanie prac, co należy odczytywać jako wspomaganie dokumentacji odpowiedniego etapu prac. Większość narzędzi z górnej części tabeli wspomaga początkowe etapy procesu tworzenia. Narzędzia środowiska COHESION wspomagają pełny cykl życia systemu, dostarczając narzędzi wspomagania głównie dla prac implementacji i pielęgnacji projektu.

Ten stosunkowo obszerny opis elementów środowiska COHESION nie wyczerpuje wszystkich aspektów zmian, które zachodzą w zupełnie nowej technologii wytwarzania oprogramowania. Możliwości stwarzane przez nowe technologie ujawniają różnice między dotychczas stosowanym sposobem pisania oprogramowania, gdzie inwencja i umiejętność pisania kodu miały decydujący wpływ na ostateczny sukces, a zupełnie nowymi sposobami produkcji programu, które gwarantują wysoką jego jakość i odpowiednie udokumentowanie całego procesu powstawania.

Środowisko COHESION wydaje się być najszerzej nakreśloną koncepcją wspomagania procesu tworzenia systemu informatycznego. Ścisłe powiązanie z pracami organizacji międzynarodowych i wdrażanie zatwierdzonych standardów oraz zakres oferty sprzętowej i programowej firmy DIGITAL stawiają to rozwiązanie w czołówce światowych dokonań w tej dziedzinie. Dodatkowym atutem tego rozwiązania jest pozostawienie interfejsów, które umożliwiają łatwe włączenie produktów innych firm do środowiska COHESION[46]. Osiąga się to dzięki stosowaniu standardów międzynarodowych w produktach składających się na to środowisko oraz wysokiemu stopniowi automatyzacji procesu tworzenia z wykorzystaniem narzędzi CASE.

---

[46] Szósty rozdział pracy (17) dokładnie przedstawia, jak współpracuje środowisko COHESION z produktami firmy IBM w różnorodnych konfiguracjach — od prostego włączenia, przez SNA/GETWAY, do re-engineering funkcjonujących aplikacji.

Tabela 4.1

Zestawienie porównawcze narzędzi CASE wspomagania poszczególnych etapów metodyki punktów węzłowych (147)

Nazwa produktu CASE	Etapy metodyki punktów węzłowych						
	1	2	3	4	5	6	7
Zerotime	x	x	x	x		x	
Promod	x	x	x	x		x	
IEW/GAMMA	x	x	x	x		x	x
PAGEFIT	x	x	x	x		x	x
Foundation	x	x	x	x	x	x	
IEF	x	x	x	x		x	
CASE 2000	x	x	x	x	x	x	
System Developer	x	x	x	x		x	
Auto-Mate	x	x	x			x	
TEAMWORK	x	x			x	x	
Excelerator	x	x			x	x	
Prokit*Workbench	x	x	x			x	
Systems Analyst	x	x	x		x	x	
Maestro	x	x			x	x	
Analyst Toolkit	x	x				x	
APS			x	x		x	
CorVision				x	x	x	
Recorder						x	x
Multi-Cam					x	x	
DECtrace					x	x	x
DECRdbExpert			x	x		x	x
VAX TEAMDATA		x	x			x	
VAX RALLY			x	x	x	x	
DECforms				x		x	
VAXset	x	x	x	x	x	x	x
VAX ACMS		x	x	x	x	x	x
VAX SPM	x	x	x			x	
DECdesign			x	x		x	

4.4. Nowa jakość systemów informatycznych — systemy otwarte

Jednym z podstawowych postulatów inżynierii programowania jest jakość systemu oprogramowania. Wcześniej przedstawiono cechy, które składają sie na jakość systemu oprogramowania, i sposoby ich uzyskania. System informatyczny to nie tylko system oprogramowania, ale wiele innych elementów składowych. Wysoka jakość systemu informatycznego w dużej mierze zależy od jakości systemu oprogramowania, ale również od innych rozwiązań, które dotyczą głównie współdziałania systemu z otoczeniem. Od pewnego czasu pojawiło się pojęcie systemy otwarte (open systems), które jest przypisywane produktom o nowej, wyższej jakości[47].

U podstaw koncepcji systemów otwartych była chęć podniesienia produktywności zespołów wykonawczych i szybkości oraz ułatwienia adaptowania aplikacji do zmieniającego się otoczenia. Drogą prowadzącą do tego celu jest standaryzacja rozwiązań i komputerowe wspomaganie procesu wytwarzania.

Rozwiązania stosowane w systemach otwartych są przeniesieniem rozwiązań stosowanych w inżynierii oprogramowania na poziomie całego systemu informatycznego. Idea procedur czy modułów, która była podstawą programowania modularnego, została zmodyfikowana i przeniesiona na poziom całego systemu informatycznego, budowanego ze standardowych elementów wmontowanych w standardowe procedury wykonawcze. Działania takie są możliwe dzięki szerokiemu zasięgowi międzynarodowych standardów, stosowanych nie tylko w informatyce, ale również w obiektach, gdzie ma funkcjonować system. Aby idea systemów otwartych mogła być wdrożona, konieczne jest więc odpowiednie środowisko, które nie może się zawężać tylko do rozwiązań informatycznych. Miara otwartości zmienia się wraz z postępem technologii i wdrażania nowych rozwiązań sprzętowych i programowych.

Jednym z założeń koncepcji systemów otwartych było złamanie monopolu firm komputerowych. Użytkownik, który kupił sprzęt komputerowy lub oprogramowanie w jakiejś firmie, nie może uzależniać rozwoju systemu od pierwszego dostawcy pod groźbą straty poczynionych inwestycji. Po-

[47] W. Iszkowski w artykule (69) uzasadnia, że należy mówić nie o systemach otwartych, lecz o systemach otwartych możliwości. Stanowisko takie zajmuje firma Digital. Wydaje się, że takie sformułowanie pozwala uniknąć wielu nieporozumień i różnic interpretacyjnych, o jakich autor pisze w artykule. Obecnie można spotkać się z wieloma definicjami systemów otwartych. Często mają one charakter komercyjny i są lansowane przez duże firmy komputerowe.

winien mieć możliwość włączenia w funkcjonującą aplikację nowych elementów, zgodnie z potrzebami rozwojowymi systemu, niezależnie od tego kto jest producentem nowego elementu. Standardowe rozwiązania interfejsów, wsparte międzynarodowymi standardami, powinny gwarantować swobodny wybór i konfigurowanie systemu z elementów pochodzących z różnych źródeł. Podobna sytuacja panuje już w wielu innych, pokrewnych dziedzinach, jak telefonia czy urządzenia elektryczne. Możliwość swobodnego łączenia urządzeń i produktów różnych dostawców zwalnia producenta z powielania funkcjonujących produktów w nowym środowisku sprzętowym i programowym. Umożliwia to również międzynarodowy podział pracy i specjalizację w stosunkowo wąskiej grupie produktów.

Międzynarodowa organizacja X/Open przyjęła definicję pojęć możliwość przenoszenia, możliwość łączenia i współdziałanie, które gwarantują otwartość systemów. Pojęcia te są podstawą technologii przetwarzania rozproszonego w architekturze klient — serwer.

Możliwość przenoszenia (ang. portability) jest cechą, która gwarantuje przeniesienie aplikacji z jednego środowiska sprzętowo-programowego w inne tylko po rekompilacji kodu źródłowego. Osiąga się to stosując interfejs oprogramowania aplikacji API (Application Programming Interface).

Możliwość łączenia (ang. connectivity) pozwala na połączenie ze sobą dwóch lub więcej systemów za pomocą standardowych interfejsów komunikacyjnych, wykorzystywanych przez łaczone aplikacje.

Współdziałanie (ang. interoperability) oznacza możliwość dostępu z jednej aplikacji do innych aplikacji przez standardowy interfejs. Może nim być standard OSF[48] rozproszonego środowiska komunikacyjnego ze zdalnym wywołaniem procedur (OSF DCE — Distributed Communnication Environment z RPC — Remote Procedure Call).

Ujednolicenie platform stosowania aplikacji jest związane z rozwojem sieci komputerowych, w których zachodzi potrzeba wymiany danych i komunikowania się różnorodnych aplikacji, oraz coraz większym zasięgiem stosowania systemów informatycznych. Organizacje administracji rządowych krajów wysoko rozwiniętych jako pierwsze zażądały od producentów aplikacji ujednolicenia środowisk, w których mają funkcjonować użytkowane produkty. Skala problemu miała wpływ na dynamikę prac nad koncepcjami systemów otwartych.

Wymagania te zbiegły się z dążeniami firm komputerowych do obniżenia kosztów wytwarzania aplikacji. W opracowaniu (69) zestawiono w tabeli ele-

[48] Open Software Foundation, organizacja założona przez główne firmy komputerowe.

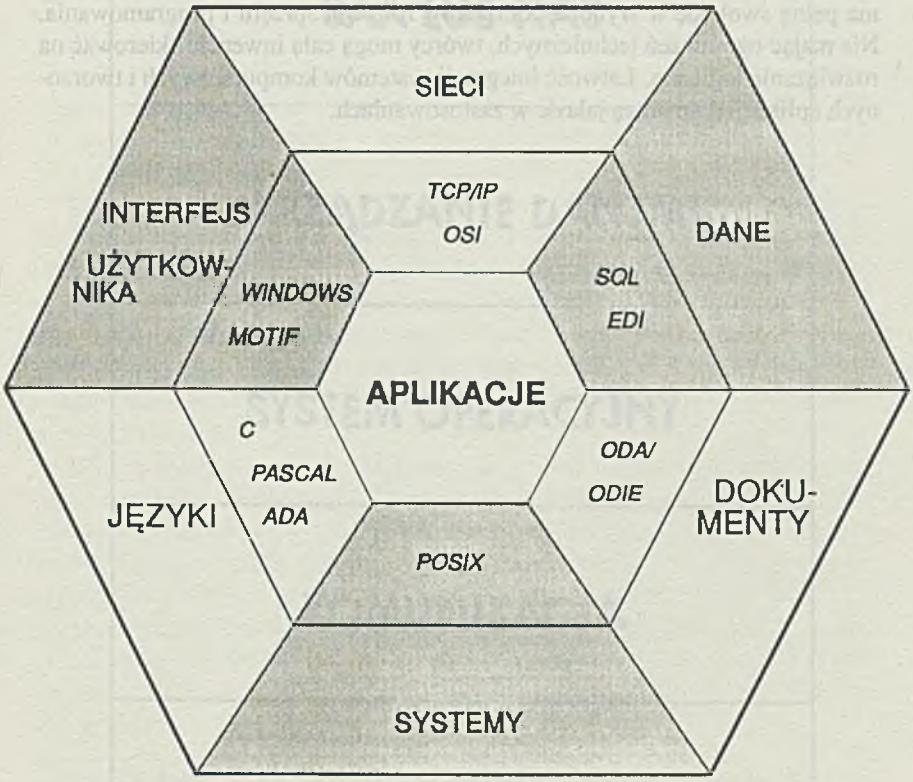
menty systemu informatycznego, których wytworzenie w przypadku systemu otwartego może być znacznie tańsze. W przypadku stosowania technologii systemów otwartych oszczędność kosztów szacuje się od 5% do 60% w poszczególnych elementach składających się na system.

W technologii systemów otwartych środowisko, w którym powstaje system informatyczny, nie ogranicza się do sprzętu komputerowego i systemu operacyjnego, ale rozszerza się na otoczenie. Otoczenie to tworzą terminale, łącza komunikacyjne, różnorodne struktury danych i inne aplikacje. Aplikacja oprócz naturalnej wewnętrznej poprawności musi gwarantować poprawność współdziałania z otoczeniem. Rzeczywiste środowisko jest więc znacznie bardziej rozbudowane, niż to się rozumie tradycyjnie. Chodzi nie tylko o środowisko wytwarzania aplikacji ale również o środowisko funkcjonowania.

Należy rozważyć różne interfejsy, tak jak to przedstawia rys. 4.3. Wyróżnia się więc interfejs:

— użytkownika, określający sposób współpracy oprogramowania z urządzeniami wejścia/wyjścia oraz sposób dostępu użytkownika do aplikacji; standardy OSF/Motif i X/Windows, wykorzystujące grafikę w komunikacji z użytkownikiem zdobyły sobie największą popularność; komunikacja z wykorzystaniem kolorowych ikon jest czytelna i prosta w użytkowaniu, nie wymaga od użytkownika specjalistycznej wiedzy;
— komunikacji, określający sposób komunikowania się oprogramowania z innymi aplikacjami w sieci komputerowej; wykorzystywany jest siedmiowarstwowy model OSI z protokółami X.25, Ethernet, TCP/IP lub protokół poczty elektronicznej X.400; — dostępu do danych, określający sposób dostępu do danych; standard języka SQL i standard bezpośredniej wymiany informacji EDI (Electronic Data Interface);
— dokumentów, określający formaty i formy wymiany informacji według standardów ODA/ODIF (Open Document Architecture/ Open Document Interface Format);
— systemowy, określający formaty zleceń do systemu operacyjnego i sposób komunikowania się z użytkownikiem przez polecenia i komendy; wykorzystuje się tu standard POSIX;
— języka programowania aplikacji, określający sposób zapisu kodu źródłowego oprogramowania w kilku wybranych językach programowania, jak C, Pascal, ADA, itp.

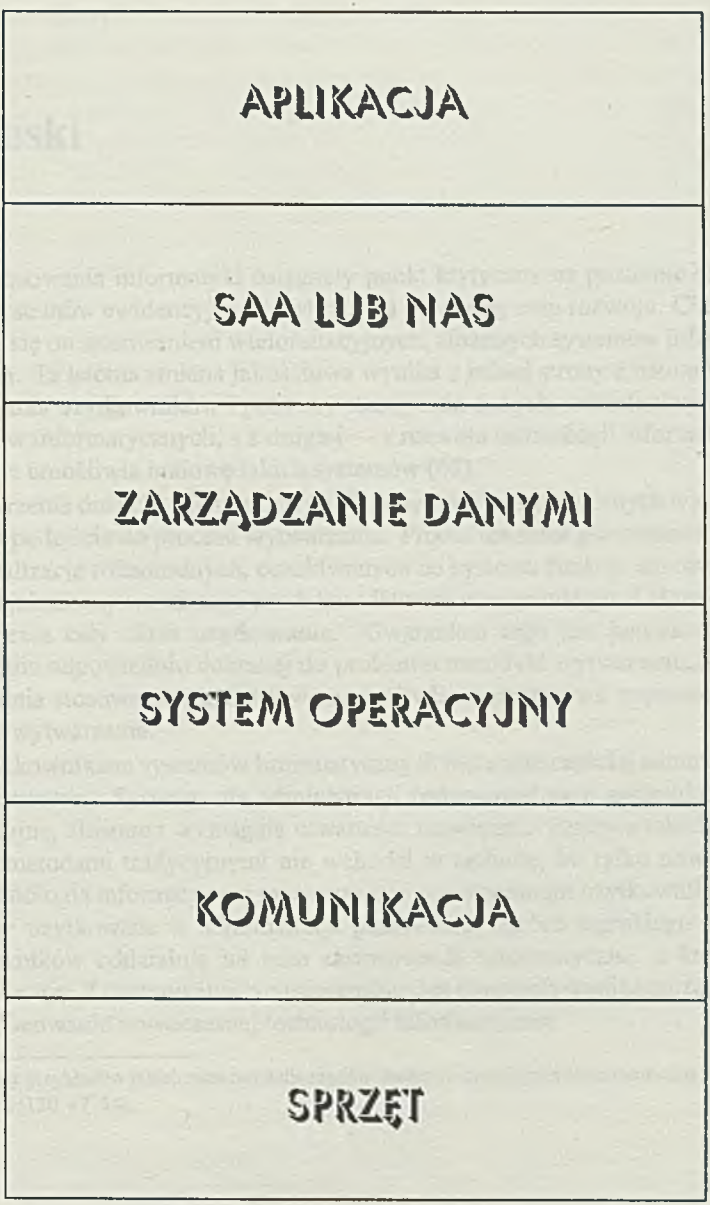
Firmy komputerowe dostarczają zestawy pakietów oprogramowania, które realizują poszczególne elementy środowiska. Przykładem takich zestawów są wcześniej opisane SAA firmy IBM czy NAS firmy Digital. Pomiędzy sprzęt



Rys. 4.3. Model heksagonalny systemu otwartych możliwości (68)

komputerowy a użytkownika współpracującego z aplikacją jest wkładana kolejna warstwa, tak jak to przedstawia rysunek 4.4. Wprowadzenie tej warstwy wynika z potrzeby uproszczenia użytkowania coraz bardziej złożonych systemów informatycznych, które współpracują w różnorodnych konfiguracjach sprzętowych, często rozproszonych terytorialnie (132).

Dzięki warstwowej konstrukcji struktury systemu można łatwo dopasować system do różnorodnego środowiska sprzętowo-programowego. Użytkownik nie jest związany z jednym dostawcą sprzętu czy oprogramowania. Oprócz naturalnego sposobu komunikowania się aplikacji różnych autorów, użytkownik ma pełną swobodę w wyborze kierunków rozwoju sprzętu i programowania. Nie mając ograniczeń technicznych, twórcy mogą całą inwencję skierować na rozwiązania logiczne. Łatwość integracji systemów komputerowych i tworzonych aplikacji daje nową jakość w zastosowaniach.



Rys. 4.4. Warstwowa struktura systemu

APLIKACJA
SAA-BUS-MAS
ZARZĄDZANIE DANYMI
SYSTEM OPERACYJNY
KOMUNIKACJA
SPRZĘT

Wnioski

Zastosowania informatyki osiągnęły punkt krytyczny na poziomie stosowania systemów ewidencyjnych i wkraczają w kolejny etap rozwoju. Charakteryzuje się on stosowaniem wielofunkcyjnych, złożonych systemów informatycznych. Ta istotna zmiana jakościowa wynika z jednej strony z naturalnego dojrzewania użytkowników i potrzeby stosowania dużych, wielofunkcyjnych systemów informatycznych, a z drugiej — z rozwoju technologii informatycznej, która umożliwia budowę takich systemów (68).

Tworzenie dużych, wielofunkcyjnych systemów informatycznych wymaga nowego podejścia do procesu wytwarzania. Proces ten musi gwarantować nie tylko realizację różnorodnych, oczekiwanych od systemu funkcji, ale również wysoką jakość wytworzonego produktu i łatwość utrzymania go w aktualnym stanie przez cały okres użytkowania. Gwarantem tego jest konsekwentne stosowanie odpowiednio dobranej do problemu metodyki wytwarzania, która uwzględnia stosowanie standardów i umożliwia komputerowe wspomaganie procesu wytwarzania.

Użytkownikiem systemów informatycznych jest coraz częściej administracja państwowa. Systemy dla administracji państwowej są z założenia wielofunkcyjne, złożone i wymagają otwartości rozwiązań. Budowa takich systemów metodami tradycyjnymi nie wchodzi w rachubę, bo tylko nowoczesna technologia informatyczna może sprostać oczekiwaniom użytkowników[49] Systemy użytkowane w administracji państwowej oprócz szerokiego kręgu użytkowników oddziałują na inne zastosowania informatyczne, z którymi współpracują. Z usytuowania tych systemów i ich znaczenia wynika szczególna rola w kreowaniu nowoczesnej technologii informatycznej.

[49] Opisy przykładów takich rozwiązań dla rządów Holandii, Szwajcarii i Niemiec można znaleźć w pracach (120, 47, 24).

Terytorialne rozproszenie użytkowników systemów wymusiło stosowanie przetwarzania rozproszonego z wykorzystaniem sieci komputerowych. Ze zrozumiałych względów są to sieci heterogeniczne, co w naturalny sposób skłania do standaryzacji rozwiązań. Fakt rozproszenia użytkowników systemu jest obecnie traktowany jako podstawowe założenie funkcjonalne projektowanych systemów. Wyjście zastosowań na stanowisko pracy użytkownika wymusza nie tylko konieczność stosowania teletransmisji i odpowiedniej technologii przetwarzania, ale również inny, łatwiejszy sposób komunikowania się systemu z użytkownikiem. Dużą popularność zdobył graficzny sposób komunikowania się. Obraz stał się standardem prezentacji i komunikowania, częściowo wyeliminował problem językowy, a ponadto jest bardziej zwięzły i czytelniejszy w użytkowaniu[50]. Nie bez znaczenia są również efekty ekonomiczne — ograniczone zużycie papieru w archiwowaniu i przesyłaniu danych (80).

Użytkownikiem systemu często jest klient bez przygotowania informatycznego, dla którego kontakt z systemem jest incydentalny. Inny użytkownik ma głęboką wiedzę informatyczną i wykorzystując dostępne, przyjazne oprogramowanie, chce na danych z systemu wykonać własne, specjalne procedury obliczeniowe. Serwis obsługi użytkownika jest różny w zależności od jego wiedzy. System informatyczny musi gwarantować obsługę zarówno jednego, jak i drugiego typu użytkownika. Realizacja praktyczna tego postulatu jest możliwa dzięki stosowaniu standardów światowych i przyjaznych w użytkowaniu narzędzi komputerowego wspomagania.

Nowe, większe wymagania stawiane systemom informatycznym to wyzwanie rzucone projektantom systemów. Systemy muszą być elastyczne, przyjazne dla różnych użytkowników i mieć wysoką jakość. Technologia pracy powinna uwzględniać najnowsze osiągnięcia techniki informatycznej, oprócz tradycyjnego tekstu wykorzystywać obraz i głos.

Rozwój sprzętu, oprogramowania i technologii informatycznej pozwala na spełnienie tych większych wymagań i oczekiwań użytkownika. Pozostaje tylko uświadomienie tego twórcom i stworzenie warunków organizacyjnych do realizacji nowej generacji systemów informatycznych. Jednym z tych warunków jest przyjęcie odpowiednich standardów wraz z uregulowaniami prawnymi za powszechnie obowiązujące.

Nowe środowisko pracy informatyka powinno sprzyjać wytwarzaniu systemów nowej generacji. Na środowisko to składa się:

[50] Potwierdza to sukces rynkowy i ciągle rosnącą popularność pakietu Windows produktu firmy Microsoft. Windows stał się w krótkim czasie niekwestionowanym standardem komunikacji nie tylko z systemem operacyjnym. Liczne aplikacje ogólnego stosowania wykorzystują środowisko Windows.

— metodyka,
— standardy,
— narzędzia komputerowego wspomagania.

Wytworzenie takiego środowiska jest przedsięwzięciem niezwykle kosztownym i długotrwałym. Mogą więc sobie na to pozwolić duże, najbogatsze firmy komputerowe, które od długiego czasu prowadzą w tej dziedzinie intensywne prace. Rozwiązaniem jest przyjęcie standardów akceptowanych przez organizacje międzynarodowe. Akceptuje je również większość tworzonego oprogramowania i sprzęt różnych producentów, co gwarantuje łatwy rozwój i pielęgnację projektowanych systemów.

Przedstawiona w pracy metodyka punktów węzłowych wraz z wspomagającym pakietem programowym jest zalążkiem oryginalnego środowiska tworzenia systemów informatycznych. Dalszy rozwój powinien iść w kierunku opisanych przedsięwzięć standaryzacyjnych największych firm komputerowych oraz metod i dostępnych narzędzi komputerowego wspomagania. Otwartość rozwiązań, lansowana przez większość producentów, pozwoli na swobodny dobór niezbędnych elementów pod warunkiem przestrzegania standardów. Wytwarzanie systemów informatycznych, zgodnych z oczekiwaniami użytkownika, musi być wspomagane metodyką, standardowymi rozwiązaniami i technologią CASE.

Klasyczny model procesu wytwarzania systemu informatycznego należy zmodyfikować do postaci:
metodyka ⟶ standardy ⟶ komputerowe wspomaganie.

Wytwarzanie systemu według tego modelu zmusza twórców do wyboru metodyki postępowania adekwatnej do rozwiązywanego problemu. Metodyka ta powinna zawierać w sobie standardy postępowania, notacji, komunikowania, dokumentowania i inne, lub uwzględniać odpowiednią fazę realizacji dla opracowania własnych standardów na potrzeby tworzonego systemu. Przyjęta metodyka i standardy ograniczają możliwości wyboru, lecz jednocześnie powinny inspirować do stosowania w procesie wytwarzania komputerowych narzędzi wspomagania.

Taki model realizacji w niczym nie ogranicza inwencji autorów i gwarantuje wysoką jakość wytwarzanego systemu. Jednym z elementów jakości jest łatwość modyfikowania i utrzymania systemu w aktualnym stanie przez cały czas użytkowania. Wytworzony system może rosnąć razem ze zwiększającymi się potrzebami użytkownika bez ograniczeń wynikających z przyjętych w czasie realizacji rozwiązań sprzętowych czy programowych. Niezależność systemu

od środowiska sprzętowego i programowego jest szczególnie istotna w warunkach integracji kilku systemów w jeden produkt o nowych możliwościach.

Przykładem takiego modelu realizacji systemu jest przedstawiona w pracy metodyka punktów węzłowych. Dedykowana głównie systemom informatycznym zarządzania, uwzględnia (jako jeden z kroków realizacji) wybór standardów światowych lub opracowanie własnych na potrzeby realizacji systemu. Metodyka dostarcza autorom przewodnik realizacji w postaci sieci działań, co wspomaga proces wykonawczy nie ograniczając inwencji twórców. Opracowany na potrzeby tej metodyki pakiet programowy wspomaga komputerowo realizację i inspiruje do stosowania innych narzędzi, adekwatnych do rozwiązywanego problemu.

Nakreślony tu kierunek prac zmierza do tworzenia zintegrowanego środowiska wytwarzania systemów informatycznych. Jest to długotrwały i kosztowny proces. Korzystając z doświadczeń światowych i dostępnych produktów rynkowych, każdy zespół powinien dążyć do stworzenia własnego środowiska, warsztatu wytwarzania, co będzie gwarantować wysoką jakość tworzonych systemów i możliwość szybkiego reagowania na zmiany zachodzące w otoczeniu.

Literatura

1. *AD/Cycle Concept,* IBM raport No GC26-4531.
2. *AD/Cycle Digest,* IBM raport G520-6654-00,1990.
3. *An American National Standard, IEEE Guide to Software Requirements Specification,* ANSI/IEEE nr 830-1984.
4. *ANS; Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Generating Stations,* ANS Std 7-4.3.2--1982, American Nuclear Society, La Grange Park (IL), July 1982.
5. Baker F.: *Chief Programer Team management of production programing,* „IBM Systems Journal" 1972, vol. 11, nr 1.
6. Benedykt Z.: *Komputerowe wspomaganie budowania systemów metodą Jacksona, doświadczenia w wykorzystaniu pakietów PDF i SPEEDBUILDER,* [w:] *Komputerowe wspomaganie tworzenia systemów informatycznych,* Materiały konferencyjne Trzeciej Wiosennej Szkoły PTI. Świnoujście 1990.
7. Benedykt Z.: *Struktury danych,* [w:] *Systemy relacyjnych baz danych: INFORMIX, INGRES, ORACLE, PROGRESS,*Materiały konferencyjne Górskiej Szkoły PTI. Szczyrk 1991-1992.
8. Birrel N.D., Ould M.A.: *A Practical Handbook for Software Development,* Cambridge University Press, Cambridge 1985.
9. „Biuletyn PTI" 1987, nr 6.
10. Bloomfield R.E., Ehrenberger W.D.: *Licensing issues associated with the use of computers in the nuclear industry,* Report EUR-11147. Commission of the European Communities, Luxemburg 1987.
11. Bogdanowicz Z., Kolbusz E., Kram E.: *Projektowanie systemów APD,* OBRI, Warszawa 1974.

12. Butler M., Bloor R.: *Comparative Database Performance on VAX,* A Computer Research Report 1990.
13. Cargill C.F.: *Information technology standarization. Theory, process, and organizations,* Digital Press 1989.
14. *CCS Summary,* IBM raport No GC31-6810.
15. Chenn P.P.: *The Entity Relationship Model — A Basis for the Enterprise View of Data,* Massachusetts Institute of Technology, Sloan School of Management, Center for Information Systems, Research CISR 30, WP 913-77, Cambridge 1977.
16. Chenn P.P.: *The Entity Relationship Model — Toward a Unified View of Data,* „ACM Transactions on Database Systems" 1976, vol 1, nr 3.
17. *COHESION Environment for CASE: A Single Environment for All Your Development Needs,* Digital Equipment Corporation raport No EC-H0986--48/91 05 43 70.0 MCG/BUO.
18. Colling B. et al: *Digital Integrated Protection System SPIN Qualitation Process,* Report IAEA SM-265/58. International Atomic Energy Agency, Vienna 1984.
19. *CPI Summary,* IBM raport No GC26-4675.
20. *CUA Advanced Interface Design Guide,* IBM raport No SC26-4582.
21. *CUA Basic Interface Design Guide,* IBM raport No SC26-4583.
22. Dahl O., Dijkstra E., Hoare C.: *Structured programming,* Academic Press, London and New York 1972.
23. *DEC Zurich keeping local government informed,* „Password Magazine" 1992, nr 17, September.
24. De Marco T.: *Structured Analysis and Design,* Yourdon Inc., New York 1978.
25. Dennie van Tassel: *Praktyka programowania,* WNT, Warszawa 1978.
26. *Dependability of Critical Computer Systems,* Pr. zbior. pod red. F.J. Redmilla. Opracowanie Komitetu Technicznego nr 7 w ramach EWICS (European Workshop on Industrial Computer Systems) w 1988 r.
27. Di Marco G., Pasquini A.: *Licensing of Real Time Systems in Nuclear Power Plants,* [w:] *Hardware and Software for Real Time Process Control,*Proc. IFIP/IFAC/EWICS Conf. Warsaw 1989.
28. *Digital Program Metodology — Life Cycle Volume,* Digital Equipment Corporation raport No A-MN-ELCP701-00-0, 1989.
29. Dijkstra E.: *GO TO statements Considered Harmful,* „Communication ACM" 1968, nr 3.
30. *DoD-STD-2167A,* Washington 1989.

31. *DOD; Defense System Software Development, MIL-Std–2167,* US Department of Defense, Washington (DC) 1984.
32. *DRIVE '91, The DRIVE Programme in 1991,* Commision of the European Communities, DRI202, Brussels 1991.
33. *Economic and Social Council,* UN/EDIFACT, raport TRADE/WP.4/R.690, August 1990.
34. Elek A., Rawiński T., Wrycza S.: *Charakterystyka wybranych narzędzi komputerowego wspomagania tworzenia systemów informatycznych,* Prace Badawcze Politechniki Gdańskiej nr 162, Gdańsk 1989.
35. Elzer P.F.: *Experiences from Prototyping w Advanced Information Systems Engineering,* Proc. Second Nordic Conference CAiSE '90. Springer-Verlag, Stockholm 1990.
36. *EPRI; Validation of Real-Time Software for Nuclear Plant Safety Applications,* Report NP-2646. Electric Power Research Institute, Palo Alto (CA) 1982, November.
37. *ESA; Software Engineering Standards,* European Space Agency, Darmstadt 1983.
38. Flakiewicz W.: *Typologia systemów informatycznych,* [w:] *Systemy informatyczne,* Materiały seminarium SPIS '82. Warszawa 1982.
39. Folkmanis J.: *ESPRIT Today — An Overwiew,* Springer-Verlag in Advanced Information Systems Enginering, Second Nordic Conference CAiSE '90, Stockholm 1990, May.
40. Fry J.P., Teorey T.J., Oberlander L.B.: *Database Design Tools and Metods: A Survey and Tutorial,* Technical Report 78DB14. Database Systems Research Group, Graduate School of Business Administration, The University of Michigan, Michigan 1978.
41. Fryźlewicz Z., Huzar Z.: *Rozległe sieci komputerowe,* „Informatyka" 1991, nr 12.
42. Fuglewicz P.: *Języki czwartej generacji a systemy CASE,* Materiały konferencyjne INFOGRYF '92. TNOiK, Szczecin 1992.
43. Fuglewicz P.: *Systemy CASE — problemy, techniki, rozwiązania,* [w:] *Komputerowe wspomaganie tworzenia systemów informatycznych,* Materiały konferencyjne Trzeciej Wiosennej Szkoły PTI. Świnoujście 1990.
44. Gajewski K.: *Inżynieria informacji Jamesa Martina,* [w:] *Komputerowe wspomaganie tworzenia systemów informatycznych,* Materiały konferencyjne Trzeciej Wiosennej Szkoły PTI. Świnoujście 1990.
45. Gane C., Sarson T.: *Structured Systems Analysis, Tools and Techniques,* Prentice Hall, New York 1979.

46. Gasik S.: *Wprowadzenie do programometrii,* „Informatyka" 1985, nr 11-12.
47. *German local authorities follow open systems trend,* „Password Magazine" 1992, No 17, September.
48. Gloe G., Westhausser R.: *Assessment and Licensing of Control Computers in the FRG,* [w:] *Software Quality Assurance,* Vol 1. Practice and Implementation Prentice-Hall, Englewood Cliffs (NJ) 1988.
49. Gościński J.: *Projektowanie systemów zarządzania,* Warszawa 1971.
50. *Guide to Information Systems,* A Framework for Success. Digital Equipment Corporation, raport No EC-G0768-54/9007 14 50.0 BUO.
51. Hiemstra Y.: *Expert Systems in Portfolio Management: An Application to Tactical Asset Allocation,* [w:] Materiały konferencyjne INFOGRYF '92. TNOiK, Szczecin 1992.
52. *HIPO — A design aid and documentation technique. Instalation management series, GC 20-1851.* IBM, Mechaniesburg 1974.
53. Hoffman L.J.: *Poufność w systemach informatycznych,* WNT, Warszawa 1982.
54. *IAEA: Manual on Quality Assurance for Computer Software. Final Version,* International Atomic Energy Agency, Vienna 1987.
55. *IEC; Application of Digital Computers to Nuclear Reactor Instrumentation and Control, IEC 643,* International Electronical Commission, Geneva 1979.
56. *IEC; Software for Computers in the Safety Systems of Nuclear Power Stations, IEC 880,* International Electrotechnical Commision, Geneva 1979.
57. *IEEE Guide to Software Configuration Management Planning,* IEEE Std 1042-1987.
58. *IEEE Guide for Software Verification and Validation Plans,* IEEE Project 1059.
59. *IEEE Recommended Practice for Application of IEEE Std 828 to Nuclear Power Generating Stations,* IEEE Std 1033-1985.
60. *IEEE Standard for Software Configuration Management Plans,* IEEE Std 826-1983.
61. *IEEE Standard Glossary of Software Engineering Terminology,* IEEE Std 729-1989.
62. *IEEE Standard for Software Life Cycle Processes,* IEEE Project 1074.
63. *IEEE Standard for Software Verification and Validation Plans,* IEEE Std 1012-1986.

64. *Informatyka. Poradnik dla ekonomistów,* Pr. zbior. pod red. E. Niedzielskiej. PWE, Warszawa 1982.
65. *Introducing IBM Office Vision,* IBM raport No GH21-0448.
66. Irlik J.: *O definiowaniu systemu informatycznego,* [w:] *Metody, techniki i narzędzia tworzenia systemów informatycznych,* Materiały konferencyjne Drugiej Wiosennej Szkoły PTI. Świnoujście 1989.
67. *IS 7498 — Information processing systems — Open Systems Interconection — Basic Reference Model,* Wyd. I 1984.
68. Iszkowski W.: *Informatyzacja sukcesem biznesu,* „DEC Forum" 1992, nr 3.
69. Iszkowski W.: *Systemy otwartych możliwości,* „DEC Forum" 1992, nr 2.
70. Jackson M.: *Principles of program design,* Academic Press, New York 1975.
71. Klapper M.: *Metodyka ustalania faktów,* [w:] *Metodyki projektowania systemów informatycznych zarządzania,* Materiały konferencyjne Pierwszej Wiosennej Szkoły PTI. Świnoujście 1988.
72. Klir G.J.: *Ogólna teoria systemów,* WNT, Warszawa 1976.
73. *Komputery w gospodarce socjalistycznej,* Pr. zbior. pod red. T. Wierzbickiego. PWE, Warszawa 1974.
74. Kopetz H.: *Niezawodność oprogramowania,* WNT, Warszawa 1980.
75. Kotarbiński T.: *Traktat o dobrej robocie,* Wrocław-Warszawa-Kraków 1969.
76. Krebs H.: *Verification of Safety Related Programs for a Maglev System,* Working Paper 520. European Workshop on Industrial Computer Systems, TC 1986, July, 7, 21.
77. Lange O.: *Ekonomia polityczna,* T. 1. Warszawa 1961.
78. Longworth G.: *Standards in programing Methods and Procedures,* The National Computing Centre Limited Oxford Road, Oxford 1981.
79. Lucas H.C. jr.: *The Analysis, Design and Implementation of Information Systems,* McGraw Hill, New York 1985.
80. Luray M.: *Imaging works its magic,* „Enterprise" 1992, October.
81. Łukasik-Makowska B.: *Standaryzacja systemów powielarnych,* „Informatyka" 1991, nr 9.
82. Madrjas K.: *AD/Cycle IBM-owska filozofia inżynierii projektowania oprogramowania,* [w:] *Metodologie tworzenia systemów informatycznych a CASE,* Materiały konferencyjne Piątej Wiosennej Szkoły PTI. Świnoujście 1992.

83. *Making Open Systems Work for You,* Heweltt Packard. Raport No 5091--1235-LE.
84. Malagardis N.: *Certification and Validation in Europe,* [w:] *Nuclear Power Plants,* [w:] *Hardware and Software for Real Time Process Control,* Proc. IFIP/IFAC/EWICS Conf. Warsaw 1989.
85. Marcinkiewicz J.: *MERISE i REMORA — metody rozwoju systemów informacyjnych,* [w:] *Metody, techniki i narzędzia tworzenia systemów informatycznych,* Materiały konferencyjne Drugiej Wiosennej Szkoły PTI. Świnoujście 1989.
86. Marlyn V., Boone G.: *Case Product Classification Model,* „CASE Bulletin" 1989, March.
87. Martin J.: *Fourth Generation Languages,* Printice Hall, New York 1985.
88. Mayhew P.: *Software Prototyping: Implications for the People Involved in Systems Development,* [w:] *Advanced Information Systems Engineering,* Proc. Second Nordic Conference CAiSE '90. Springer-Verlag, Stockholm 1990.
89. Maynard J.: *Modular Programming,* London 1972.
90. *Metodologie tworzenia systemów informatycznych a CASE,* Materiały konferencyjne Piątej Wiosennej Szkoły PTI. Świnoujście 1992.
91. *Metody, techniki i narzędzia tworzenia systemów informatycznych,* Materiały konferencyjne Drugiej Wiosennej Szkoły PTI. Świnoujście 1989.
92. Mills H.: *Top Down Programming in large systems, Debugging techniques in large systems,* Rustin R. Prentice 1971.
93. *MUSE Project, Alist of references on metrics,* Internal Report, ESPRIT Project No 1257, Brussels 1989.
94. *Myers G.J.:* Projektowanie niezawodnego oprogramowania,WNT, Warszawa 1980.
95. *NAS Application Style Guide,* Digital Equipment Corporation, raport No AA.PE7ZA.TK.
96. *NAS Guide to Developing Portable Software,* Digital Equipment Corporation, raport No AA.PE81A.TK.
97. *NAS Guide to Designing a Portable Programming Interface,* Digital Equipment Corporation, raport No AA.PE82A.TK.
98. *NAS Guide to Documenting Multiplatform Products,* Digital Equipment Corporation, raport No AA.PE80A.TK.
99. *NAS Guide to Using a portable Programming Interface,* Digital Equipment Corporation, raport No AA.PE83A.TK.

100. *NAS Handbook: Developing Applications in a Multivendor Environment,* Digital Equipment Corporation, raport No EC.H0477.48/900543200.0BUO.
101. *NAS Overview,* Digital Equipment Corporation, raport No AA.PE7YA.TK.
102. *Network Application Support (NAS) Application Catalog,* Digital Equipment Corporation, raport No EC.J1036.58.
103. Niedzielska E. (red.): *Projektowanie systemów informatycznych,* PWE, Warszawa 1977.
104. Olejniczak W., Szydłowski I.: *FACTOR: System wspomagania projektowania systemów informatycznych,* [w:] *Komputerowe wspomaganie tworzenia systemów informatycznych,* Materiały konferencyjne Trzeciej Wiosennej Szkoły PTI. Świnoujście 1990.
105. Olejniczak W. Szydłowski I.: *Globalna metoda projektowania systemów informatycznych,* [w:] *Metody, techniki i narzędzia tworzenia systemów informatycznych,* Materiały konferencyjne Drugiej Wiosennej Szkoły PTI. Świnoujście 1989.
106. Page-Jones Meilir: *The practical guide to structured systems design,* Yordon, New York 1980.
107. *PATHWORKS Just how far are you planning to take your PC network?* Digital Equipment Corporation raport No EC-N0821-69.
108. Piotrowski Z.A.: *ICASE — zintegrowane środowisko CASE.* [w:] *Metodologie tworzenia systemów informatycznych a CASE,* Materiały konferencyjne Piątej Wiosennej Szkoły PTI. Świnoujście 1992.
109. Pollack S., Hicks H., Harrison W.: *Tablice decyzyjne.* PWN, Warszawa 1975.
110. *Propozycja strategii rozwoju informatyki i jej zastosowań w Rzeczypospolitej Polskiej,* Pr. zbior. Polskie Towarzystwo Informatyczne, Warszawa 1991.
111. *Propozycje strategii Rządu Rzeczypospolitej Polskiej w dziedzinie informatyki,* Biuro Informatyki Urzędu Rady Ministrów, Warszawa 1991.
112. Redmill F. (ed.): *Dependability of Critical Computer Systems,* Vol. 1. Elsevier Applied Science, London 1988.
113. Redmill F. (ed.): *Dependability of Critical Computer Systems,* Vol. 2. Elsevier Applied Science, London 1989.
114. Ross D.T., Schoman K.E.: *Structured Analysis for Requirements Definition,* IEEE Transactions on Software Engineering January 1977, vol SE-3, nr 1, January.

115. Sobolewski P.: *Środowisko COHESION*, [w:] *Metodologie tworzenia systemów informatycznych a CASE*, Materiały konferencyjne Piątej Wiosennej Szkoły PTI. Świnoujście 1992.
116. *Software Requirements Specification*, IEEE Std 830-1984.
117. *Software Verification & Validation Plans*, IEEE Std 1012-1986.
118. *Standaryzacja w systemach informacji społeczno-gospodarczej*, Materiały konferencyjne SPIS '91, GUS 1991.
119. Stąpor K.: *Strukturalna analiza i projektowanie systemów informatycznych — metodyka Gane'a-Sarsona*, „Informatyka" 1992, nr 2 i 3.
120. *Streamlining bureaucracy, HP and Uniface take part in extensive 4GL project.* „Password Magazine" 1992, No 17, September.
121. *Systems Application Architecture. An overview*, IBM raport GC26-4341--04. Wyd. piąte. Grudzień 1990.
122. *Systemy informatyczne zarządzania*, Pr. zbior. pod red. T. Wierzbickiego. PWE, Warszawa 1985.
123. *Systemy relacyjnych baz danych: INFORMIX, INGRES, ORACLE, PROGRESS*, Materiały konferencyjne Górskiej Szkoły PTI. Szczyrk 1991-1992.
124. Szyjewski Z.: *CASE w Polsce.* „Biuletyn PTI" 1990, nr 11.
125. Szyjewski Z.: *Komputerowe wspomaganie dokumentowania tworzenia systemów informatycznych*, [w:] *Komputerowe wspomaganie tworzenia systemów informatycznych*, Materiały konferencyjne Trzeciej Wiosennej Szkoły PTI. Świnoujście 1990.
126. Szyjewski Z.: *Koncepcja tworzenia systemu IBM AD/Cycle a programowanie z bazą wiedzy*, [w:] *Problemy wspomagania decyzji*, Zeszyty Naukowe nr 78. Uniwersytet Szczeciński, Szczecin 1991.
127. Szyjewski Z.: *Metoda Jacksona projektowania oprogramowania*, [w:] *Metody, techniki i narzędzia tworzenia systemów informatycznych*, Materiały konferencyjne Drugiej Wiosennej Szkoły PTI. Świnoujście 1989.
128. Szyjewski Z.: *Metodyka tworzenia systemów informatycznych zarządzania*, Zeszyty Naukowe ITS. Warszawa 1990.
129. Szyjewski Z.: *Programowanie strukturalne w przetwarzaniu danych*, PWE, Warszawa 1984.
130. Szyjewski Z.: *Projektowanie systemów informatycznych zarządzania*, [w:] *Metodyki projektowania systemów informatycznych zarządzania*, Materiały konferencyjne Pierwszej Wiosennej Szkoły PTI. Świnoujście 1988.
131. Szyjewski Z.: *Standaryzacja w systemach informatycznych*, [w:] *Alternatywne źródła pozyskiwania danych statystycznych*, Materiały konferencyjne SPIS '90. PAN-GUS, Smardzewice 1990.

132. Szyjewski Z.: *Systemy otwarte — moda czy konieczność,* „Informatyka" 1992, nr 7-8.
133. Szyjewski Z.: *Wpływ rozwoju oprogramowania na proces projektowania systemów informatycznych zarządzania,* Praca doktorska. Politechnika Szczecińska, Szczecin 1979.
134. *Sieci komputerowe — zastosowania,* Materiały konferencyjne Czwartej Wiosennej Szkoły PTI. Świnoujście 1991.
135. Tavolato P., Vincena K.: *A Prototyping Methodology and its Tool w Budde et al./eds/, Approaches to Prototyping,* Springer, Berlin 1984.
136. Teichrose F., Bastarache M.: *PSL user's manual,* University of Michigan Dept. of Industrial Engineering, ISDOS Working Paper 98, Michigan 1975.
137. *The Standardization of Interfaces for Computer Intergrated Manufacturing (CIM),* DIN Technical Report No 15. German Standards Institute, Berlin--Koln 1988.
138. Tsichritzis D.C., Lochovsky F.H.: *Modele danych,* WNT, Warszawa 1990.
139. *Turski W.M.:* Metodologia programowania,WNT, Warszawa 1978.
140. Turski W.M.: *Struktury danych,* WNT, Warszawa 1976.
141. *Uchwała III Walnego Zjazdu PTI,* „Biuletyn PTI" 1990, nr 9-10 i 11.
142. *ULTRIX Handbook,* Digital Equipment Corporation, raport No EC-H0592--43/91A044320.0, 1991.
143. Vessey I., Jarvenpaa S.I., Tranctinsky N.: *CASE tools and Methodology Companions,* „Communications of the ACM" 1992, April.
144. Wieringen J.J.: *Case-tools; tools or new technology?* Materiały konferencyjne INFOGRYF '92. TNOiK, Szczecin 1992.
145. Wierzbicki T., Szyjewski Z.: *The milestones of information-systems design,* [w:] *Current trends in information systems development methodologies,* Polish-Skandynawian Seminar. Politechnika Gdańska, Paraszyno 1988.
146. Wijers G.H., van Dort H.E.: *Experiences with the use of CASE-tools in the Netherlands,* [w:] *Advanced Information Systems Engineering,* Proc. Second Nordic Conference CAiSE '90. Springer-Verlag, Stockholm 1990.
147. Wrycza S.: *Aktualne trendy komputerowo wspomaganego tworzenia systemów informatycznych,* [w:] *Komputerowe wspomaganie tworzenia systemów informatycznych,* Materiały konferencyjne Trzeciej Wiosennej Szkoły PTI. Świnoujście 1990.
148. Wrycza S.: *Współczesne metodyki tworzenia systemów informatycznych,* PTC, Zeszyty Problemowe nr 5. Gdańsk 1989.
149. Zalewski J.: *Zalecenia normalizacyjne stosowane w okresie istnienia oprogramowania,* „Materiały Szkoleniowe PTI" 1989, nr 1.

SPIS RYSUNKÓW I TABEL

Rysunki

1.1. Środowisko pracy twórcy systemu informatycznego (148) 43
1.2. Cykl życia systemu według metodyki firmy Digital (28) 46
1.3. Realizacja systemu według metodyki firmy Digital (28) 47
1.4. Koszty i czas realizacji prowadzonych prac (148) 49
2.1. Sieć głównych powiązań punktów węzłowych — poziom 1 89
2.2. Sieć głównych powiązań punktów węzłowych — poziom 2 91
3.1. Systemy otwarte według koncepcji Hewlett Packard 111
3.2. Network Application Support firmy Digital (100) 114
3.3. Platforma sprzętowa SAA (122) 118
3.4. SAA — elementy składowe 125
4.1. Koncepcja wytwarzania według AD/Cycle 146
4.2. Narzędzia środowiska COHESION (115) 161
4.3. Model heksagonalny systemu otwartych możliwości (68) 173
4.4. Warstwowa struktura systemu 175

Tabele

1.1. Porównanie technik tworzenia systemów informatycznych (147) 55
2.1. Zależności między punktami węzłowymi 83
4.1. Zestawienie porównawcze narzędzi CASE wspomagania poszczególnych etapów metodyki punktów węzłowych (147) 169