9 1993

informatyka

WYDAWNICTWO SIGMA NOT

Cena 25 000 zł

informatyka

nr 9 1993
Wrzesień
Rok wydania XXVIII

Miesięcznik
ISSN 0542-9951
INDEKS 36124

WYDAWCA:
Wydawnictwo Czasopism i Książek
Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych
redakcja nie zwraca

W numerze:

W najbliższych numerach:

- **Mirosław Wieczorek charakteryzuje podstawowe zagadnienia programowania zorientowanego obiektowo – polimorfizm i dziedziczenie klas w języku C++.**
- **Zbigniew Benedykt odpowiada na pytanie: Co daje obiektowe podejście do analizy i projektowania systemów informatycznych?**
- **Zbigniew Nowacki prezentuje pakiet narzędziowy POLY do języka C oraz przedstawia jego rozszerzenia – szablony i listy elastyczne.**

Warunki prenumeraty na 1993 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573

Wpłaty na prenumeratę od marca br. przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 25 000 zł, ulgowa 18 750 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 75 000, półroczna 150 000, roczna 300 000
Ulgowa: kwartalna 56 250, półroczna 112 500, roczna 225 000.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 320/93

Piąta Jubileuszowa Wyższa Międzynarodowa Górska Szkoła PTI,
Szczyrk 21–25.06.1993 r.

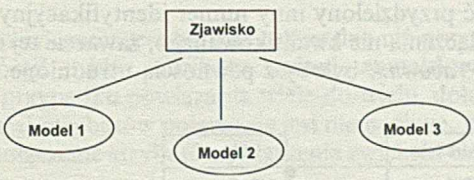
ARTUR KASPRZYK
Uniwersytet Wrocławski

Obiektowo-zorientowane analiza i projektowanie (1)

Cykl życia systemu informatycznego rozpoczyna się od momentu sformułowania problemu. Następnymi etapami są: analiza, projektowanie, implementacja, testowanie. W tym momencie gotowy produkt zostaje oddany do rąk użytkownika. Nie kończy się jednak cykl jego życia. Rozpoczyna się bowiem etap jego pielęgnacji i rozszerzania (może to być, i z reguły jest, proces cykliczny).

Celem niniejszego artykułu jest omówienie etapów obiektowo-zorientowanej analizy i projektowania, które w dalszej części będą nazywane modelowaniem. Jest to ta część cyklu życia systemu, który rzutuje na całą przyszłą jego strukturę oraz zachowanie. Jej efektem jest gotowy model systemu. Modelowanie problemu nie jest rzeczą nową. Od stuleci inżynierowie i naukowcy najpierw budowali modele, by po ich akceptacji rozpocząć konstrukcję konkretnego przedmiotu. Testowanie modelu jest o wiele prostsze i mniej kosztowne niż testowanie samego produktu końcowego. Nie ma już zakładów motoryzacyjnych, które najpierw budowałyby samochód, by dopiero potem stestować jego działanie. Do tego służą komputerowe symulacje różnorodnych procesów. Co więcej, dzięki modelom możliwe jest zmniejszenie złożoności całego przedsiębiorstwa przez jego podział na mniejsze części. Ta właściwość jest jednak odczuwalna dopiero przy projektowaniu większych systemów i być może stąd wynika powszechne lekceważenie tych etapów.

Każdy model jest pewną abstrakcją opisywanego wycinka rzeczywistości.



Rys. 1. Każdy model dotyczy jednego aspektu rozpatrywanego zjawiska

Abstrakcje w swojej naturze są niekompletne bowiem opisują jedynie pewien sposób widzenia zjawiska, a dzięki temu w znacznym stopniu je upraszczają. Nie oznacza to, iż przez to nastąpi utrata części wiedzy o zjawisku, gdyż można skonstruować kilka jego różnych modeli (rys. 1). Przykładem może być tu instrukcja obsługi radioodbiornika, w której jest on opisany za pomocą schematu, rysunku poglądowego oraz słownego opisu (zauważmy, że słowa też są jedynie abstrakcjami, a jednak spełniają swoje zadanie znakomicie).

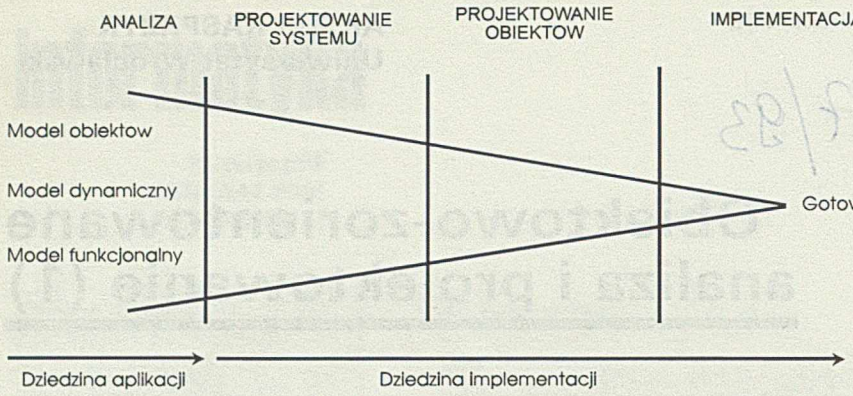
Object Modelling Technique (OMT)

Jedną z obiektowo-zorientowanych metodologii tworzenia systemów informatycznych jest *Object Modelling Technique*. Jak można się domyślić, podstawowym elementem tej techniki jest obiekt, na który składają się struktury danych oraz operacje, a jej celem jest identyfikacja i organizacja obiektów występujących w dziedzinie aplikacji (nie implementacji!). Cały proces modelowania jest przeprowadzany niezależnie od jakiegokolwiek języka programowania. Co ciekawsze, nie zakłada on, iż system będzie zaimplementowany w języku obiektowym. Można z powodzeniem wykorzystać do tego celu języki proceduralne oraz systemy baz danych.

Metodologia OMT składa się z czterech podstawowych etapów: analizy, projektowania systemu, projektowania obiektów oraz implementacji[1)] (rys. 2). W każdym z nich stworzone na etapie analizy modele obiektów, dynamiczny oraz funkcjonalny, przechodzą swoistą ewolucję. Modele te odzwierciedlają konstruowany system, widziany na trzy różne – choć powiązane ze sobą – sposoby. Model obiektów odpowiada statycznej strukturze aplikacji. Dostarcza on informacji na temat „wydobytych" z rzeczywistości obiektów, ich atrybutów i operacji, a także wzajemnych powiązań. Model ten powinien zawierać tylko te obiekty, które są ważne z punktu widzenia aplikacji. Model obiektów w metodologii OMT jest reprezentowany graficznie za pomocą diagramów, składających się z klas (ich atrybutów oraz operacji) tworzących hierarchie oraz połączonych ze sobą wiązaniami. Model obiektów jest strukturą, nad którą są konstruowane dwa pozostałe modele.

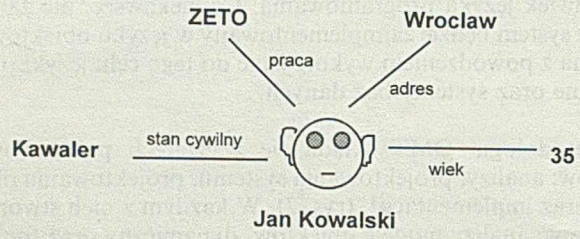
Model dynamiczny odpowiada zmianom stanu aplikacji w czasie. Podstawowymi pojęciami z nim związanymi są: zdarzenie (ang. *event*) oraz stan (ang. *state*) – rozumiany jako kolekcja powiązań obiektu z innymi obiektami. Rysunek 3 opisuje obiekt *Jan Kowalski* w postaci jego powiązań z innymi obiektami (inaczej mówiąc – wartościami jego atrybutów). Stanem obiektu jest w tym przypadku kolekcja (35, *Wrocław, ZETO, Kawaler*). Sposób reakcji na zdarzenie zależy od stanu, w jakim obiekt się znajduje. Efektem zdarzenia może być (choć nie musi) zmiana stanu obiektu. Graficzną reprezentację modelu dynamicznego jest zbiór diagramów stanów (ang. *state diagram*) – skończonych, działających niezależnie od siebie, automatów deterministycznych. Każdy diagram odpowiada klasie, której zachowanie w czasie jest istotne z punktu widzenia analizy systemu. Ostatnim modelem jest model funkcjonalny. Przedstawia on obliczenia dokonywane w systemie (kolejne przekształcenia wartości wejściowych) bez względu na to, kiedy są one wykonywane. Graficznym przedstawieniem modelu funkcjonalnego są diagramy przepływu danych (ang. *data flow diagram*).

[1)] Metodologia OMT została zaproponowana przez J. Rumbaugh'a, Michaela Blaha, Williama Premerlani, Fredericka Eddy'ego oraz Williama Lorensena (General Electric Research and Development Center Schenectady, New York). Składa się z czterech etapów. Ostatni z nich – implementacja – został w artykule pominięty z racji tego, iż przejście od projektowania do implementacji wydaje się być bardzo naturalne.



Rys. 2. Kolejne składowe metodologii OMT

Jak widać, z każdym modelem jest związana jego graficzna reprezentacja. Poniżej zostaną przybliżone ich podstawowe składowe.



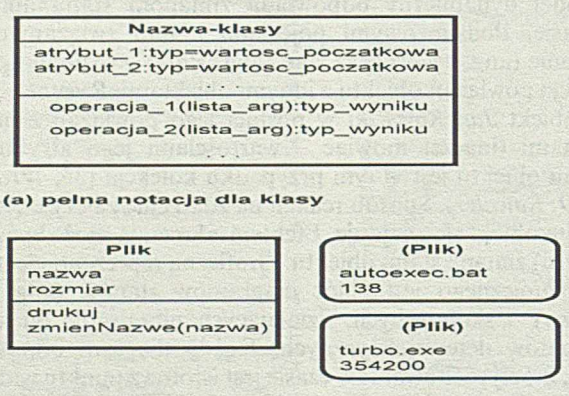
Rys. 3. Powiązania obiektu *Jan Kowalski* z innymi obiektami

Model obiektów

Jak wspomniano wcześniej, na model obiektów składają się obiekty wraz z atrybutami i operacjami oraz ich wzajemne powiązania.

Obiekty i klasy

Graficzną reprezentację pojęć pokazuje rys. 4.

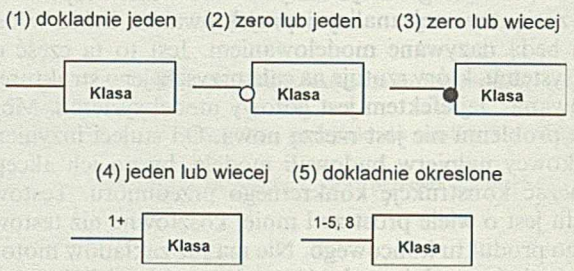


Rys. 4. Notacja dla klas i obiektów

Połączenia, powiązania i kompozycja

Między połączeniem (ang. *link*) i powiązaniem (ang. *association*) istnieje taka sama różnica, jak między obiektem i klasą. Połączenie wiąże ze sobą dwa lub więcej obiektów. Na przykład, obiekty **Jan Kowalski** i **Wrocław** wiąże połączenie *Mieszka*. Jednak rozpatrując klasy **Osoba** i **Miasto**, związek *Mieszka* nazwiemy powiązaniem, a nie połączeniem.

Powiązania są z natury dwustronne. Jeżeli osoba mieszka w mieście, to można także stwierdzić, iż miasto jest zamieszkane przez osoby. Warto w tym miejscu zwrócić uwagę na jeszcze jedną sprawę. Otóż, o ile jedna osoba w określonym czasie może mieszkać tylko w jednym mieście, o tyle miasto może być zamieszkane przez wiele osób. Wielowartościowe powiązania przedstawia rysunek 5.

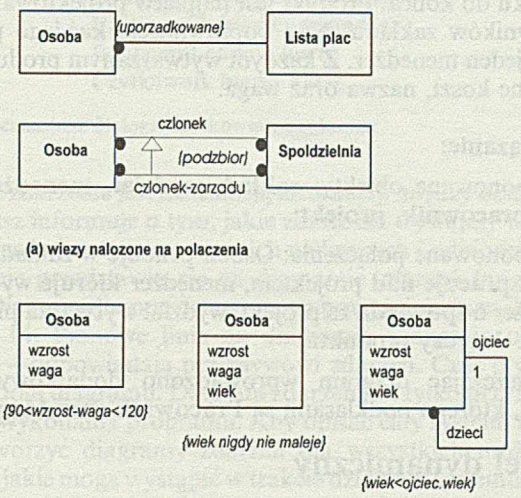


Rys. 5. Różnowartościowość powiązań

Wartościowość powiązania można zmniejszyć z „wiele" do „jeden" przez zastosowanie powiązania kwalifikowanego (ang. *qualified association*); dotyczy to powiązań „wiele-do-wielu" oraz „jeden-do-wielu". Sposób tworzenia powiązań ilustruje rysunek 6. Zauważmy, iż w przypadku (b) zastosowanie kwalifikatora daje dodatkową korzyść. Ponieważ pracownik może pracować w więcej niż jednym zakładzie pracy, w każdym z nich może mieć przydzielony inny numer identyfikacyjny. W przypadku połączenia nie kwalifikowanego, zawarcie tej informacji w klasie *Pracownik* byłoby z pewnością utrudnione.

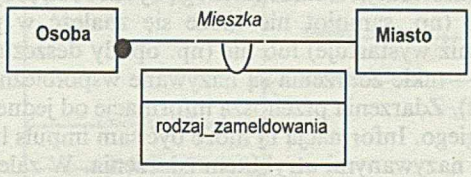
Rys. 6. Połączenia kwalifikowane i niekwalifikowane

Czasem jest pożądane oznaczenie obiektów, które na końcu wiązania „wiele" są w jakiś sposób uporządkowane. Przykładem może być lista płac, na której umieszczono osoby w porządku alfabetycznym. Uporządkowanie jest specjalnym przypadkiem więzów, jakie można nakładać na wartości atrybutów oraz połączenia. Pojęcie to jest znane z problematyki baz danych. Przykłady więzów przedstawia rysunek 7.



Rys. 7. Więzy można nakładać na połączenia i atrybuty

Wiązania, podobnie jak klasy, mogą mieć również atrybuty. W przypadku powiązania *Osoba mieszka w Mieście* (rys. 8), atrybutem powiązania *mieszka* może być **rodzaj_zameldowania** (stały, czasowy).

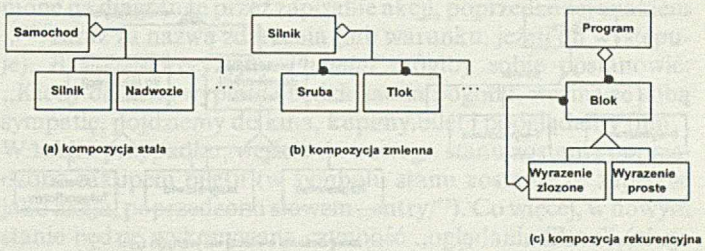


Rys. 8. Atrybut połączenia dla powiązania *wiele-do-jednego*

Takie rozwiązanie jest bardziej naturalne niż przyporządkowanie którejś z klas, atrybutu **rodzaj_zameldowania**. Co więcej, w przypadku powiązania wiele-do-wielu, dołączenie do którejś z klas atrybutów połączenia jest niemożliwe. Widać więc jasno, iż dołączenie atrybutów połączenia zwiększa elastyczność systemu, gdyż przyszłe rozszerzenia systemu mogą zmienić wiązanie do niewygodnego w tym przypadku wiele-do-wielu.

Uogólnieniem tego przypadku jest modelowanie powiązania jako obiektu. Wówczas powiązanie ma dodatkowo nazwę oraz operacje.

Szczególnym przypadkiem powiązania jest kompozycja (ang. *composition, aggregation*). Można intuicyjnie przyjąć, że kompozycja jest połączeniem w rodzaju *składa-się, jest-zbudowane* itp. Obiekt, który jest *skomponowany* z innych składowych, będzie dalej nazywany agregatem (ang. *aggregate*). Różnica między kompozycją a zwykłym połączeniem jest bardzo subtelna. Polega ona na tym, że agregat jest traktowany semantycznie jako jeden obiekt. Przenosząc jednostkę centralną komputera ze stołu na biurko, nie zastanawiamy się, czy osobno przenieść sterowniki, a osobno kartę graficzną. Przykład agregatu (klasa **Jednostka centralna**) przedstawia rysunek 9. Widać z niego, iż kompozycja jest relacją przechodnią. Z tego, że jednostka centralna składa się z płyty głównej, a w skład płyty głównej wchodzą układy pamięci wynika, że układy pamięci wchodzą również w skład jednostki centralnej. Kompozycje można podzielić na stałe, zmienne oraz rekurencyjne. Różnice przedstawia rysunek 10.

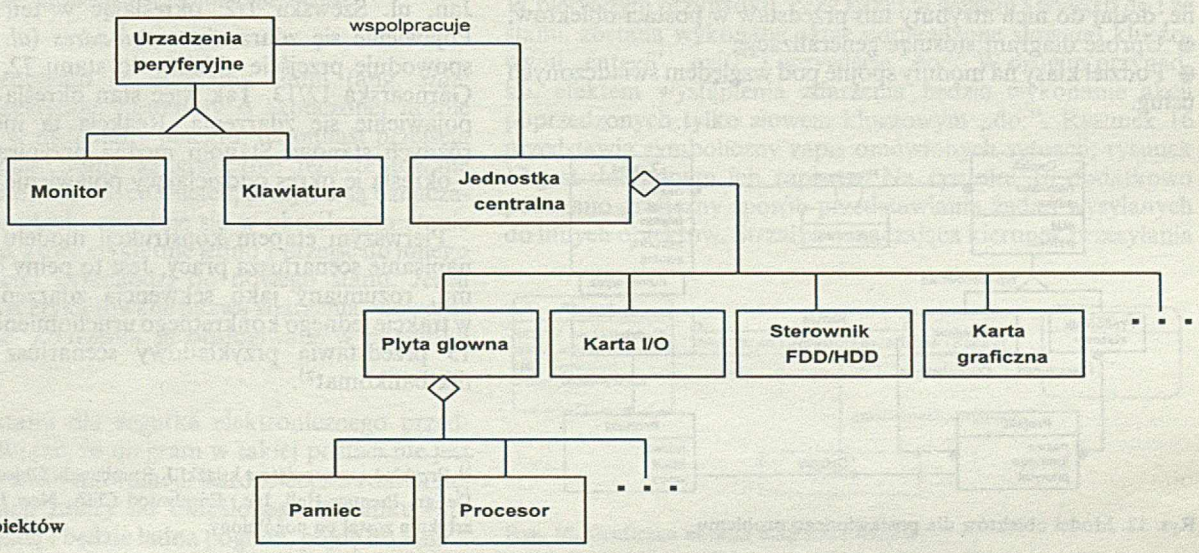


Rys. 10. Różne rodzaje kompozycji

Dziedziczenie i generalizacja

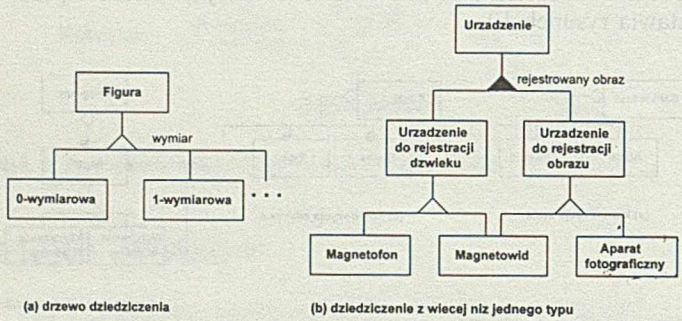
Generalizacja odzwierciedla związki między klasami, często nazywanymi superklasami (ang. *superclass*) oraz ich podklasami (ang. *subclass*). Dziedziczenie jest implementacją genelizacji. Ponieważ generalizacja jest związkiem przechodnim, wprowadza się pojęcie *następcy* (ang. *ancestor*). Znaczenia tych pojęć są następujące:

- klasa A jest **poprzednikiem** klasy X, jeżeli istnieją klasy $A1,...,An$ takie, że $A1$ jest podklasą A, $A2$ jest podklasą $A1$, X jest podklasą An,
- klasa A jest **następcą klasy** X (analogicznie).



Rys. 9. Kompozycja obiektów

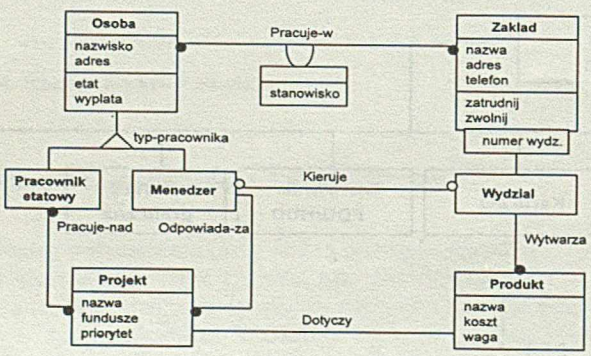
Wprowadzoną dla generalizacji notacją jest trójkąt łączący superklasę z jej podklasami. Przykłady hierarchii dziedziczenia przedstawia rysunek 11. Na rysunku (a) widać proste drzewo dziedziczenia dla wielowymiarowych figur. Klasą bazową jest **Figura**, a jej podklasami figury **0-wymiarowe, 1-wymiarowe, 2-wymiarowe** itd. Trzykropek oznacza, iż przedstawiona hierarchia nie jest kompletna, gdyż z jakichś względów nie przedstawiono jej dalszej części (być może nie jest to ważne z punktu widzenia aplikacji). Słowo „wymiar", położone przy symbolu generalizacji, jest nazywane dyskryminatorem (ang. *dyscriminator*). Określa ono właściwość, która została wzięta pod uwagę podczas dokonywania podziału. Na rysunku (b) widać hierarchię dziedziczenia dla urządzeń audiowizualnych. Klasą bazową jest klasa **Urządzenie**. Z niej są wyprowadzone pozostałe typy. Zaczerniony trójkąt oznacza, iż dokonany podział nie jest rozłączny (ang. *disjoint*), tzn., że istnieją klasy obiektów, będące zarówno urządzeniami do rejestracji dźwięku jak i obrazu (np. magnetowid). Każda klasa znajdująca się poniżej klasy bazowej jest jej następnikiem, ale tylko **Urządzenie do rejestracji dźwięku** oraz **Urządzenie do rejestracji obrazu** są jej podklasami. Klasa **Magnetowid** jest przykładem klasy łączonej – mającej więcej niż jedną superklasę (ang. *join class*).



Rys. 11. Wielopoziomowe hierarchie dziedziczenia

Przedstawione wyżej koncepcje stanowią podstawowe narzędzia potrzebne do stworzenia prostego modelu obiektów (rys. 12). Proces jego tworzenia może przebiegać następująco:

- Ze specyfikacji zadania wybierz kandydatów na obiekty. Najprostszą metodą jest wybranie rzeczowników ze słownego opisu zadania i wstępne przyjęcie ich jako kandydatów na obiekty. Następnie należy wyeliminować te z nich, które dotyczą implementacji oraz nie są istotne z punktu widzenia tworzonego systemu;
- Dodaj do obiektów atrybuty i metody;
- Utwórz połączenia między obiektami. Jeżeli jest to konieczne, dodaj do nich atrybuty lub przedstaw w postaci obiektów;
- Uprość diagram stosując generalizację;
- Podziel klasy na moduły spójne pod względem świadczonych usług.



Rys. 12. Model obiektów dla postawionego problemu

Przykład. Mając dany opis słowny problemu, utwórz dla niego model obiektów.

Opis zadania. Zakład składa się z wielu wydziałów, które są ponumerowane w sposób jednoznaczny. Większość z nich ma własnego menedżera. Jednak nie każdy z zatrudnionych w zakładzie menedżerów zarządza własnym wydziałem. Każdy wydział wytwarza przynajmniej jeden produkt, zawsze od początku do końca. Produkt jest najpierw projektowany przez pracowników zakładu. Nad poprawnością każdego projektu czuwa jeden menedżer. Z każdym wytwarzanym produktem są związane koszt, nazwa oraz waga.

Rozwiązanie:

- proponowane obiekty: **zakład, wydział, menedżer, produkt, pracownik, projekt:**
- Proponowane połączenia: Osoba **pracuje** w zakładzie, pracownik **pracuje nad** projektem, menedżer **kieruje** wydziałem, menedżer **odpowiada** za projekt, wydział **wytwarza** produkty, projekt **dotyczy** produktu;
- upraszczając diagram wprowadzono dodatkowy obiekt **Osoba,** którego podklasami są **Pracownik** oraz **Menedżer**.

Model dynamiczny

Zadaniem modelu dynamicznego jest przedstawienie zmian, jakim podlegają obiekty w trakcie działania systemu. Podstawowymi pojęciami, na których opiera się model są zdarzenia (ang. *events*), stany (ang. *states*) oraz przejścia od jednego stanu do innego (ang. *transition*).

Zdarzeniem jest to, co wydarzyło się w określonym momencie. Zdarzenie nie ma czasu trwania – jest jakby impulsem niosącym informację. Zdarzenia mogą być ze sobą przyczynowo powiązane (np. samolot nie może się znaleźć w powietrzu wcześniej niż wystartuje) lub nie (np. opady deszczu w Polsce i Tunezji) – takie zdarzenia są nazywane współbieżnymi (ang. *concurrent*). Zdarzenia przenoszą informacje od jednego obiektu do drugiego. Informacją tą może być sam impuls lub impuls z danymi, nazywanymi atrybutem zdarzenia. W zależności od potrzeb zdarzenie będzie rozumiane bądź jako klasa zdarzeń (np. opady deszczu), bądź jako pojedyncze zdarzenie (np. opady deszczu na południu Tunezji).

Stan obiektu można rozumieć jako zbiór wartości jego atrybutów. Niech **Jan Kowalski** będzie obiektem klasy **Osoba**, mającym atrybuty, **wiek, nazwisko, imię** i **adres**. Niech aktualnymi wartościami atrybutów będą kolejno 72, Kowalski, Jan, ul. Szewska 1/2, określając w ten sposób jego stan. Pojawienie się zdarzenia *zmień-adres* (*ul. Garncarska* 12/13) spowoduje przejście obiektu do stanu 72, Kowalski, Jan, ul. Garncarska 12/13. Tak więc stan określa reakcję obiektu na pojawienie się zdarzenia. Reakcja ta może być różna dla różnych stanów. Stanom można przypisać czas ich trwania – określa je okres oddzielający pojawienie się dwóch zdarzeń.

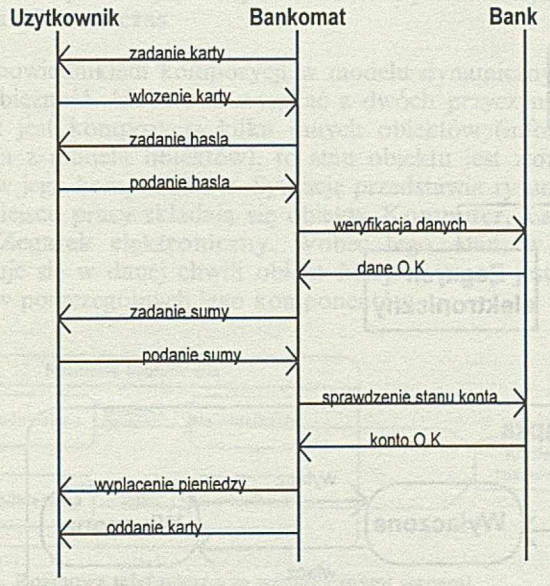
Pierwszym etapem konstrukcji modelu dynamicznego jest napisanie scenariusza pracy. Jest to pełny cykl działania systemu, rozumiany jako sekwencja zdarzeń pojawiających się w trakcie jednego konkretnego uruchomienia systemu. Rysunek 13 przedstawia przykładowy scenariusz dialogu użytkownik-bankomat[2].

[2] Przykład zaczerpnięto z książki J. Rumbaugh: Object-Oriented Modeling and Design. Prentice-Hall, Inc., Engelwood Cliffs, New Jersey, 1991. Dla potrzeb artykułu został on uogólniony.

Bankomat żąda włożenia karty.
Użytkownik wkłada kartę.
Bankomat żąda podania hasła.
Użytkownik podaje hasło.
Bankomat weryfikuje hasło.
Bankomat żąda podania sumy.
Użytkownik podaje sumę.
Bankomat sprawdza stan konta.
Bankomat wydaje pieniądze.
Użytkownik bierze pieniądze.
Bankomat oddaje kartę.
Użytkownik bierze kartę.

Rys. 13. Scenariusz dialogu użytkownik-bankomat

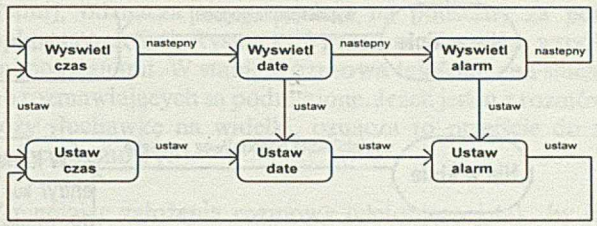
Każde zdarzenie jest transmisją informacji między obiektami. Scenariusz informuje o tym, jakie zdarzenia wystąpiły w czasie sesji, nie mówi natomiast nic o ich nadawcach i odbiorcach. Tę informację przedstawia się na diagramie zdarzeń (ang. *event trace*). Diagram dla przedstawionego powyżej dialogu ilustruje rysunek 14. Pionowe linie na diagramie oznaczają obiekty, poziome – odpowiadają przepływowi zdarzeń. Czas płynie od góry do dołu diagramu. Diagram zdarzeń jest tylko przykładem jednego wykonania programu. Aby opisać cały system, należałoby utworzyć diagramy zdarzeń dla wszystkich możliwych sytuacji, jakie mogą wystąpić w trakcie działania systemu. Takie podejście wydaje się niezbyt rozsądne.



Rys. 14. Diagram zdarzeń dla dialogu użytkownik-bankomat

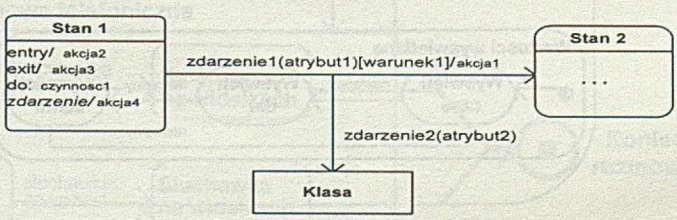
Rozwiązaniem problemu są diagramy stanu (ang. *state diagram*) – deterministyczne automaty o skończonej liczbie stanów. Stanami są w nich stany obiektu, natomiast przejścia reprezentują zdarzenia. Stany są graficznie reprezentowane przez prostokąty o zaokrąglonych rogach; przejścia są oznaczane strzałkami. Jeżeli obiekt znajduje się w określonym stanie i pojawi się zdarzenie, które etykietuje jedno z przejść do innego stanu, wówczas obiekt przechodzi do nowego stanu. Jeżeli żadne z przejść nie jest etykietowane nazwą zdarzenia, wówczas jest ono ignorowane. Zdarzenia są obsługiwane w porządku w jakim nadchodzą.

Prosty diagram stanu dla zegarka elektronicznego przedstawia rysunek 15. Widać, że diagram w takiej postaci nie jest w stanie przedstawić wielu informacji. Bardzo często zdarza się, że wykonanie operacji zależy od określonego warunku, np. „Kiedy dostanę wypłatę i będzie ładna pogoda, pójdę do kina".



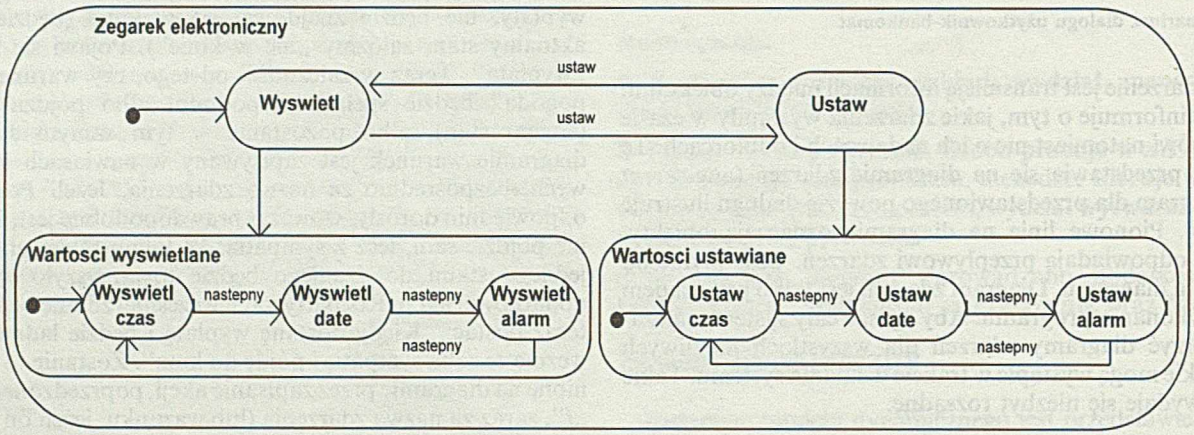
Rys. 15. Diagram stanów dla zegarka elektronicznego

Przeanalizujmy to zdanie. *Podmiot*, w momencie otrzymywania wypłaty, nie będzie znajdował się w kinie (będzie to jego aktualny stan, załóżmy „nie w kinie"). Pojawi się zdarzenie „wypłata". Teraz w zależności od tego, czy warunek „ładna pogoda" będzie spełniony, podmiot albo pójdzie do kina (zmieni stan), albo pozostanie w tym samym stanie. Na diagramie warunek jest zapisywany w nawiasach kwadratowych, bezpośrednio za nazwą zdarzenia. Jeżeli *Podmiot* jest odpowiednio dorosły, wówczas prawdopodobne jest, że do kina nie pójdzie sam, lecz z sympatią. W takim razie, przejściu od jednego stanu do drugiego będzie towarzyszyło wykonanie dodatkowej akcji. Rozpatrywane wcześniej zdanie będzie miało teraz postać: „Kiedy dostanę wypłatę i będzie ładna pogoda, wezmę ze sobą sympatię i pójdę do kina". Zostanie to uwidocznione na diagramie przez zapisanie akcji, poprzedzonej znakiem „/", zaraz za nazwą zdarzenia (lub warunku, jeżeli on występuje). Bardziej precyzyjny *Podmiot* mógłby sobie postanowić: „Kiedy dostanę wypłatę i będzie ładna pogoda, wezmę ze sobą sympatię, pójdziemy do kina, kupimy bilet i pooglądamy film". W takim przypadku, wejście do nowego stanu zostanie poprzedzone zakupem biletu (w symbolu stanu zostanie to zapisane jako akcja, poprzedzona słowem „entry/"). Co więcej, w nowym stanie będzie wykonywana czynność „oglądanie filmu" (akcja zostanie zapisana w symbolu stanu i poprzedzona słowem „do:"). Jeżeli *Podmiot* planuje przed wyjściem z kina kupić karnet dwuosobowy na Konfrontacje, dodatkową czynnością wykonywaną przed opuszczeniem stanu będzie: „kupno karnetu" („/exit"+akcja). Może się też zdarzyć, iż film, który obejrzeli, tak bardzo im się spodobał, że postanowili obejrzeć go jeszcze raz. Wobec tego, bez opuszczania kina dokupili jeszcze bilety i obejrzeli film ponownie. Jest to przykład zdarzenia wewnętrznego (ang. *internal action*), to znaczy takiego, które zostaje „obsłużone" nie powodując przy tym zmiany stanu – co więcej, nawet z niego nie wychodząc. Na diagramie akcja, która ma być wykonana w przypadku wystąpienia zdarzenia jest poprzedzona nazwą tego zdarzenia. Widać więc różnicę między zdarzeniem, które wychodzi ze stanu, wchodząc do niego ponownie, a tym, które ze stanu w ogóle nie wychodzi. W pierwszym przypadku, przy każdym wejściu i wyjściu do i ze stanu, zostaną wykonane akcje poprzedzone słowami kluczowymi „entry/", „exit/" i oczywiście „do:". W drugim przypadku, efektem wystąpienia zdarzenia będzie wykonanie akcji poprzedzonych tylko słowem kluczowym „do:". Rysunek 16 przedstawia symboliczny zapis omówionych sytuacji; rysunek 17 jest dokładnym ich zapisem. Na rysunku 16 dodatkowo pokazano graficzny sposób przedstawiania żądań wysyłanych do innych obiektów. Strzałka oznaczająca kierunek przesyłania

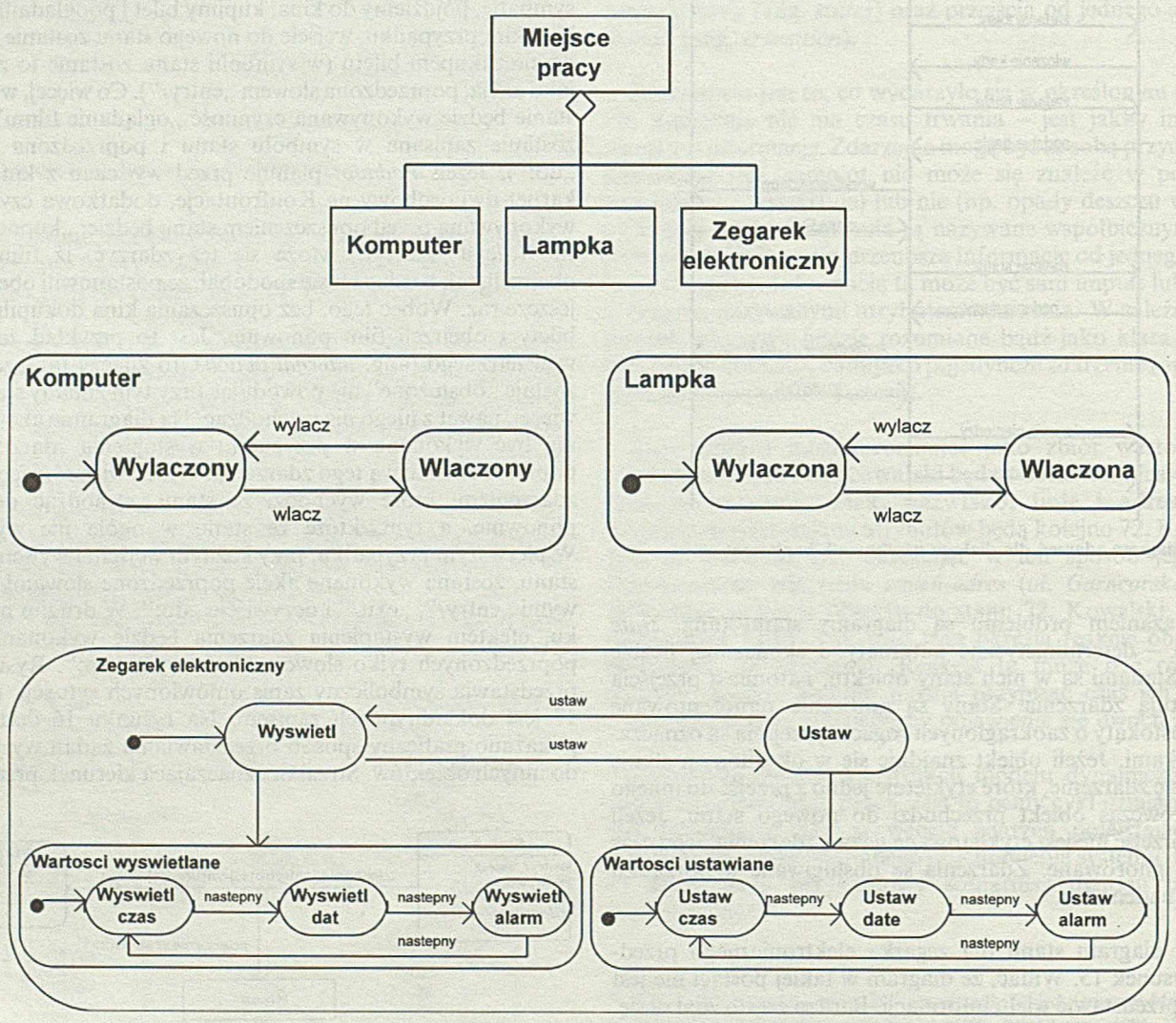


Rys. 16. Graficzna notacja diagramu stanów

Rys. 17. Przykłady diagramów stanów



Rys. 18. Prosty diagram zagnieżdżony



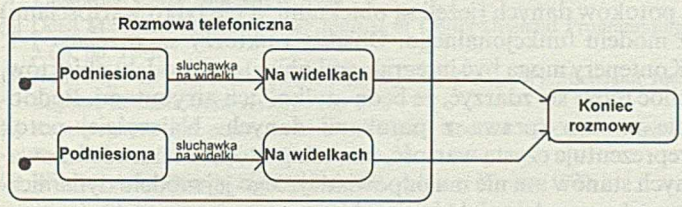
Rys. 19. Kompozycja i jej współbieżny diagram stanów

informacji powinna być, z jednej strony, połączona ze zdarzeniem, które rozpoczyna transmisję (w ten sposób informując, że wysłanie zdarzenia 2 zależy od pojawienia się zdarzenia 1), a z drugiej, ze zdarzeniem w diagramie stanów obiektu docelowego (by pokazać związek między nim a przesłanym zdarzeniem).

Podobnie jak obiekty, również stany można grupować w hierarchie lub je komponować z innych stanów. Odpowiednikiem generalizacji z modelu obiektów jest zagnieżdżenie diagramów (ang. *nested diagrams*), natomiast kompozycji – stany współbieżne (ang. *concurent state*). Zagnieżdżanie diagramów umożliwia płynne przejście od rozważań ogólnych do bardziej szczegółowych. Stany diagramów zagnieżdżonych (podstanów) dziedziczą przejścia swoich przodków. Oznacza to, iż każde przejście i każda akcja, mające zastosowanie do określonego stanu, stosują się do wszystkich jego podstanów (oczywiście pod warunkiem, że nie zostaną przedefiniowane). Przykład zagnieżdżonych diagramów przedstawia rysunek 18. Zagnieżdżonymi diagramami są **Wartości wyświetlane** oraz **Wartości ustawiane**. Zaczerniony okrąg na początku przejścia oznacza stan początkowy diagramu.

Stany: **Wyświetl czas, Wyświetl datę** oraz **Wyświetl alarm** dziedziczą po stanie **Wyświetl** przejście „ustaw". Oznacza to, że będąc w którymkolwiek z podstanów stanu **Wyświetl**, zdarzenie „ustaw" spowoduje przejście do stanu **Ustaw**. Nieetykietowane przejście (ε – przejście) od **Ustaw** do diagramu **Wartości ustawiane** powoduje, że system automatycznie znajdzie się w stanie **Ustaw czas**.

Odpowiednikiem kompozycji w modelu dynamicznym jest współbieżność. Może ona wynikać z dwóch przyczyn. Jeżeli obiekt jest kompozycją kilku innych obiektów (informacja wynika z modelu obiektów), to stan obiektu jest złożeniem stanów jego komponentów. Sytuację przedstawia rysunek 19. Na miejscu pracy składają się obiekty **Komputer, Lampka** oraz **Zegarek elektroniczny**. Wobec tego, stan, w jakim znajduje się w danej chwili obiekt **Miejsce pracy** jest sumą stanów poszczególnych jego komponentów.



Rys. 20. Rozmowa telefoniczna ze współbieżnymi stanami

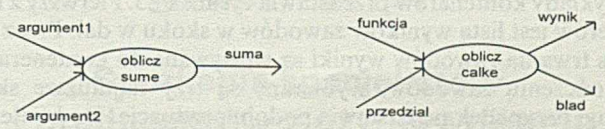
O współbieżności można mówić również w obrębie jednego obiektu. Pojawia się ona wtedy, gdy obiekt można podzielić na podzbiory atrybutów i powiązań, z których każdy ma swój własny poddiagram. Wówczas, na stan obiektu składają się stany wszystkich jego poddiagramów. Podział wewnątrz pojedynczego stanu złożonego (oczywiście ze współbieżnymi podstanami), odznacza się podziałem na podstany za pomocą kropkowanych linii. Rysunek 20 przedstawia stan ze współbieżnymi podstanami. W stanie **Rozmowa telefoniczna** słuchawki osób rozmawiających są podniesione. Jeżeli jeden z rozmówców położy słuchawkę na widełki, oznacza to przejście do stanu **Koniec rozmowy**.

Zmieniając założenia rozmowy telefonicznej tak, by jej zakończenie następowało w momencie, gdy dwóch rozmówców położy słuchawki, dochodzimy do problemu synchronizacji zdarzeń. Obiekt wyjdzie więc ze stanu **Rozmowa telefoniczna** w momencie, gdy stan słuchawek obydwu rozmówców będzie **Na widełkach**. Graficzny zapis tej sytuacji przedstawia rysunek 21. Wprowadzony dodatkowo element etykietowany **Koniec rozmowy** oznacza stan końcowy diagramu.

Model funkcjonalny

Model funkcjonalny pokazuje rezultaty obliczeń, bez określania, kiedy zostają one wykonane; określa znaczenie operacji w modelu obiektów oraz akcji w modelu dynamicznym. Na model funkcjonalny składa się zbiór diagramów przepływu danych (ang. *data flow diagram*). Z kolei, w ich skład wchodzą procesy (ang. *process*), potoki danych (ang. *data flow*), aktorzy (ang. *actor*) oraz kontenery (ang. *data store*).

Zadaniem procesów jest przekształcanie danych. Proces otrzymuje dane wejściowe, przetwarza je, produkując w ten sposób dane wyjściowe. Przykładem procesu są wszelkie funkcje – są to procesy najniższego poziomu. Procesem na wyższym poziomie, można nazwać cały diagram przepływu danych. Przykłady procesów przedstawia rysunek 22. Proces obliczający sumę dwóch argumentów jest bardzo prosty, podczas gdy dokładne numeryczne obliczenie całki może sprawić trochę kłopotów. Symbolem procesu w diagramie przepływu danych jest elipsa zawierająca opis przekształcenia. Z każdym procesem związana jest określona liczba potoków danych wejściowych i wyjściowych.



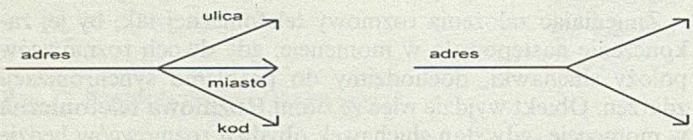
Rys. 22. Przykłady procesów

Potoki danych łączą ze sobą procesy i obiekty. Początek potoku jest wynikiem działania procesu, koniec potoku – argumentem innego procesu (lub obiektu). Potok danych nie zmienia wartości przenoszonych danych, chociaż może je rozkładać na składowe. Przykłady potoków danych przedstawia rysunek 23.

Potoki danych stanowią wyniki działania określonych procesów. Jeżeli są to końcowe obliczenia, to mogą one stanowić wartości znaczące z punktu widzenia aplikacji. Potoki wew-

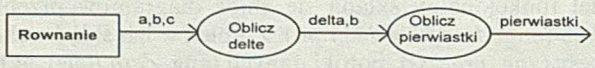
Rys. 21. Rozmowa telefoniczna z synchronizacją współbieżnych stanów

nętrzne odpowiadają wartościom tymczasowym i, z reguły, nie stanowią znaczącej informacji. Potoki zewnętrzne (tzn. takie, które znajdują się na końcach diagramu przepływu), mogą być nie połączone z niczym (wówczas diagram stanowi poddiagram większej struktury) lub mogą być połączone z obiektem. Takie obiekty nazywane są aktorami.



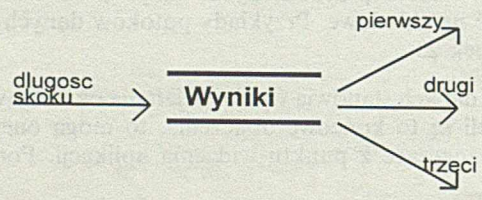
Rys. 23. Rozdzielenie złożonej wartości na składowe oraz kopia danych w potoku

Aktor nie tylko konsumuje, ale może również dostarczać dane do diagramu. Zawsze jednak leży na „skraju" diagramu. Graficzną reprezentacją aktora jest prostokąt z wpisaną wewnątrz nazwą. Na rysunku 24 aktorem jest obiekt **Równanie**.



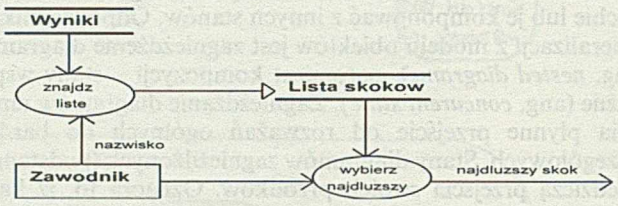
Rys. 24. Obiekt *Równanie* (aktor) dostarcza dane dla diagramu

Ostatnią składową diagramu przepływu danych są kontenery. Są one wykorzystywane do przechowywania danych. Jedynym zadaniem kontenerów jest reagowanie na żądania umieszczania w nich danych oraz ich pobierania. Kontenery pozwalają na pobieranie danych w innym porządku, niż były do nich dostarczone. Kontenery są reprezentowane w diagramie jako para równoległych odcinków, oddzielonych nazwą kontenera. Do kontenera mogą dochodzić potoki danych, i wówczas są to operacje wstawiania, usuwania bądź modyfikowania przechowywanego elementu, oraz mogą z niego wychodzić, oznaczając odczytanie danych. Ponieważ kontenery są obiektami, podobnie jak aktorzy, muszą być zdefiniowane w modelu obiektów. Przykłady kontenerów przedstawia rysunek 25. Pierwszy z kontenerów jest listą wyników zawodów w skoku w dal. Przez cały czas trwania zawodów wyniki są wpisywane do kontenera. Po zakończeniu zawodów, wybierane są trzy najdłuższe skoki. Drugi przypadek przedstawia podobną sytuację. Do kontenera są wpisywane długości skoków oraz nazwiska sportowców. Jeżeli jest to drugi skok, do kontenera jest wpisywany dłuższy z nich. Proces **znajdź odległość** na podstawie podanego nazwiska wyszukuje na liście wyników odległość, na jaką skoczył zawodnik.



Rys. 25. Kontenery

Wynikiem odczytu z kontenera może być nie tylko pojedyncza wartość, lecz również inny obiekt. Po zmianie sposobu wpisywania wyników zawodów na listę tak, aby zapamiętywane były wszystkie skoki, żądanie wyszukania najdłuższego skoku będzie wymagało „dynamicznego" utworzenia obiektu. Obiekt taki będzie wyróżniony dodatkowo trójkątem na końcu potoku danych. Przykład ilustruje rysunek 26.



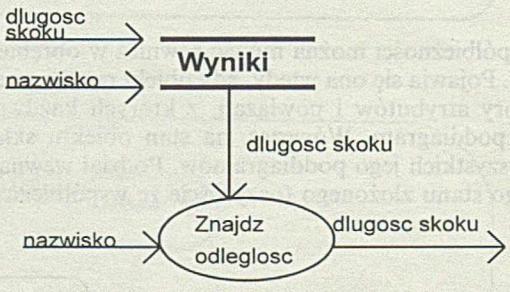
Rys. 26. Odczyt dający w wyniku obiekt

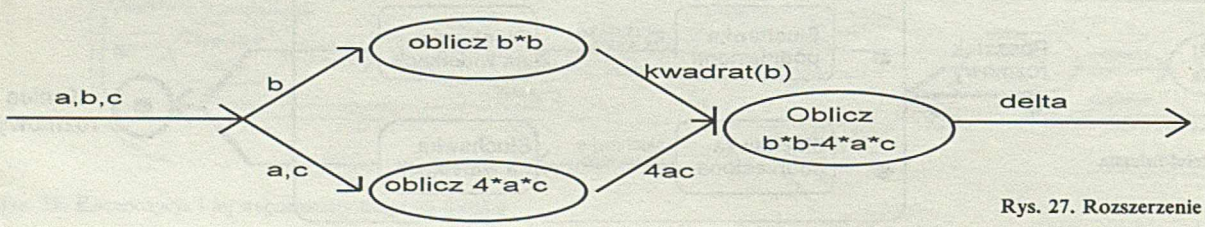
Diagramy przepływu danych mogą zagnieżdżać się w sobie. Oznacza to, że proces na wyższym poziomie może być reprezentowany przez osobny diagram na poziomie niższym. Nowy diagram może zawierać obiekty, których na poziomie wyższym nie było. Danymi dla diagramu są potoki danych wchodzące do procesu, a jego rezultatem – potoki wychodzące ze stanu. Rysunek 27 pokazuje diagram przepływu danych, odpowiadający procesowi **oblicz deltę** z rysunku 24.

Wzajemne związki modeli

Każdy z opisanych modeli pokazuje inny aspekt tego samego systemu. Bazę dla pozostałych modeli stanowi model obiektów. Zawiera on kluczową informację: *z czego* składa się system. Informacja w nim zawarta jest statyczna. Wiadomo wprawdzie, które obiekty ze sobą współpracują, ale nie wiadomo kiedy i jak. Pierwszą informację dostarcza model dynamiczny. Dodaje on „ducha" statycznym obiektom, pokazując *kiedy* mają rozpocząć działanie i w jakim porządku akcje mają występować Model funkcjonalny dopełnia informację pokazując, jak wykonywać operacje.

Model obiektów definiuje strukturę aktorów, kontenerów i potoków danych (jeżeli są obiektami, a nie tylko wartościami) z modelu funkcjonalnego. Obiekty i aktorzy są w relacji 1-1. Kontenery mogą być integralnymi obiektami modelu obiektów, choć może się zdarzyć, że będą „tylko" ich atrybutami. Podobnie ma się sprawa z potokami danych. Najczęściej potok reprezentuje czystą wartość, tzn, taką, która nie ma zdefiniowanych stanów ani nie ma odpowiadającego jej modelu dynamicznego (wprawdzie niektóre języki programowania traktują wszy-





Rys. 27. Rozszerzenie procesu *Oblicz deltę*

stko jako obiekty, ale nie zmienia to sytuacji). Są od tego wyjątki, co pokazuje rysunek 25. Procesy, w zależności od poziomu abstrakcji diagramu przepływu danych, odpowiadają operacjom lub zbiorom operacji na obiektach. Procesy reprezentowane przez zwykłe funkcje można utożsamić z pojedynczymi metodami. Im bardziej abstrakcyjny staje się diagram, tym większemu zbiorowi operacji odpowiada proces.

Model dynamiczny pokazuje dokładnie kolejność wykonywania procesów w modelu funkcjonalnym. Kolejność ta zależy od napływu zdarzeń pojawiających się w systemie.

Model funkcjonalny definiuje akcje i czynniki wykonywane w poszczególnych stanach modelu dynamicznego.

Nie wszystkie modele są jednakowo ważne w każdym problemie. Prawie w każdym z nich model obiektów gra pierwszoplanową rolę. Jest on odzwierciedleniem statycznej struktury modelowanego wycinka rzeczywistości. Jeżeli obraz problemu zmienia się znacząco wraz z upływającym czasem, wówczas ważny jest także model dynamiczny. Systemy mające do czynienia z dużą liczbą obliczeń będą, z kolei, miały złożony model funkcjonalny.

Analiza systemu nie jest procesem liniowym. Wymaga ona stałej współpracy z klientem oraz ekspertami z dziedziny problemu. Najczęściej proces tworzenia modeli rozpoczyna się od konstrukcji na wysokim poziomie abstrakcji, zawierających bardzo ogólne wiadomości. Nieustanne konsultacje oraz dokładniejsza analiza problemu pozwalają na przechodzenie do rozwiązań bardziej szczegółowych. Ostateczna postać modeli stanowi bazę dla etapów projektowania.

LITERATURA

[1] Davis A.M.: A comparison of techniques for the specification of external system behavior. Communications of the ACM, 31 (9), pp. 1098–1115
[2] Harel D.: On visual formalism. Communications of the ACM, 31 (5), pp. 514–529
[3] Martin J., Odell J.: Object-Oriented Analysis and Design. Prentice-Hall, Engelwood Cliffs, New Jersey, 1992
[4] Odell M.: What is object state? Journal of Object-Oriented Programming, 5 (2), pp. 19–22
[5] Rumbaugh J.: Object-Oriented Modelling and Design. Prentice-Hall Inc., Engelwood Cliffs, New Jersey, 1991
[6] Rumbaugh J.: An object or not an object? Journal of Object-Oriented Programming, 5 (3), pp. 20–25
[7] Windlad A.L.: Object-Oriented Software. Addison-Wesley, 1990.

W związku z wprowadzeniem powszechnego podatku dochodowego i koniecznością informowania właściwych Urzędów Skarbowych o wszystkich wypłatach zwracamy się do Autorów wysyłających teksty do opublikowania o podawanie następujących danych:

- **nazwisko, imiona (pierwsze i drugie),**
- **imiona: ojca i matki,**
- **miejsce i data urodzenia,**
- **numer identyfikacyjny PESEL (wpisany przez Biuro Meldunkowe do dowodu osobistego – nie mylić z numerem dowodu osobistego!),**
- **dokładny adres (miejsce zameldowania),**
- **adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora (bardzo ważne – bez tej informacji nie możemy przekazać honorarium do wypłaty!).**

JAN GOLIŃSKI
Katedra Systemów Informacyjnych
Szkoła Główna Handlowa
Warszawa

Informatyka gospodarcza

Informatyka Gospodarcza (*Business Informatics, Wirtschaftsinformatik*) stanowi obszar informatyki znajdujący coraz szersze zrozumienie i upowszechnienie. Wiele uczelni europejskich wyodrębniło „Informatykę Gospodarczą" jako osobny kierunek studiów. Szkoła Główna Handlowa SGH w Warszawie rozpoczęła kształtowanie tego kierunku w Polsce.

Historyczne i aktualne zależności między informatyką, informatyką gospodarczą a nauką o zarządzaniu

Informatykę można, najogólniej rzecz biorąc, podzielić na cztery podstawowe obszary:
- informatykę teoretyczną,
- informatykę techniczną,
- informatykę praktyczną – zajmującą się systemami informacyjnymi i programowymi,
- informatykę stosowaną, w której mieszczą się wszystkie dziedziny zastosowań.

W latach siedemdziesiątych przyjął się termin **informatyka przedsiębiorstwa** (niem. *Betriebsinformatik*), określający naukę o budowie, funkcjonowaniu i tworzeniu wewnątrzzakładowych systemów informacyjnych. Rozróżniano wówczas pojęcie **informatyka gospodarcza** oznaczające wiedzę o zewnętrznych systemach informacyjnych, tzn. łączących większą liczbę przedsiębiorstw (koncerny), a w przypadku szczególnym wszystkie przedsiębiorstwa gospodarki narodowej. Problematyką „informatyki przedsiębiorstwa" zajmowali się i pojęcie to propagowali: Seibt (1974), Wedekind (1980), Stahlknecht (1980), Scheer (1980), Schmitz (1980), Müller-Mehrbach (1981). W latach osiemdziesiątych coraz szerzej i powszechniej stosowano termin „informatyka gospodarcza". Można w tym miejscu postawić pytanie, czy nie należy wyłączyć informatyki gospodarczej z nauk o gospodarce i połączyć z „informatyką"? Nie wydaje się to wskazane z kilku powodów:

- Zastosowanie technologii informacyjnej nie jest nigdy celem samym w sobie. Technologia informacyjna powinna zawsze podporządkowywać się celom przedsiębiorstw, czy też celom innych organizacji (np. państwowych). Prymat celów nietechnicznych grozi zagubieniem, jeżeli zadania konstrukcyjne oraz technologiczne systemów za bardzo „usamodzielniają" się i są wyodrębniane z konkretnej sytuacji zastosowaniowej. Powiązanie z zastosowaniem jest zagrożone wtedy, gdy tak, jak świadomie to czyni informatyka, próbuje się wypracowywać wspólne cechy i ogólne zasady dla wszystkich rodzajów systemów aplikacyjnych. Informatyka gospodarcza w znacznym stopniu „żyje" ze szczególnych przypadków, charakteryzujących przetwarzanie informacji w przedsiębiorstwach i instytucjach.

- Informatyka gospodarcza nie tylko ma cechy nauki inżynierskiej, ale oprócz tego wiele cech pragmatyczno-konstruktywnie zorientowanej nauki o organizacji (Grochla 1971). Obiekt, który rozpoznaje i tworzy informatyka gospodarcza jest wielowarstwowy. Wspomagane komputerem systemy człowiek-maszyna dla przedsiębiorstw, władz administracyjnych i organizacji obejmują cztery warstwy, a mianowicie: sprzęt komputerowy, oprogramowanie, organizację oraz ludzi jako użytkowników. Jeżeli takie systemy mają pozytywnie i użytecznie funkcjonować, musi nastąpić proces tworzenia, powodujący powstanie uzgodnionych działań oraz uzyskanie zintegrowanych wyników we wszystkich czterech wymienionych warstwach. Kierowanie takim procesem tworzenia systemu nie może znajdować się w gestii informatyków, lecz powinno być przekazane tym, którzy odpowiadają za realizację celów organizacyjnych.

- Przy tworzeniu systemów informacyjnych jako systemów człowiek-maszyna szczególną wagę przywiązuje się do:
  - celów ekonomicznych (ewentualnych ograniczeń),
  - specyfiki wynikającej z zadań,
  - jednostkowych właściwości budowy i struktury przebiegów,
  - istniejącej organizacji pracy,
  - specyfiki politycznej, a zwłaszcza personalno-politycznej.

  Elementy te mają taką samą wagę, jeżeli nie większą, niż kryteria dotyczące sprzętu komputerowego i oprogramowania. Wiedza o specyfice organizacyjnej, np. przedsiębiorstw, urzędów państwowych i instytucji jest zawarta w dyscyplinach zarządzania. Wiedza ta, konieczna przy tworzeniu systemów zastosowaniowych, może być tym lepiej wykorzystana przez informatykę gospodarczą, im lepiej i ściślej będzie ona współpracowała z dyscyplinami zarządzania.

- Postulat ścisłego powiązania informatyki gospodarczej z naukami gospodarczymi powinien mieć charakter dwukierunkowy. Przez wprowadzenie technologii informacyjnej, dyscypliny te uzyskują wiele nowych istrumentów, służących zwiększeniu efektywności ich badań i dydaktyki.

- Tendencją rozwojową rokującą wiele nadziei są systemy oparte na bazach wiedzy. Stwierdzenie to dotyczy zarówno informatyki gospodarczej, jak i samej informatyki. Nie może jednak budzić wątpliwości, że ta nowa technologia ma tylko wtedy szansę sukcesu, kiedy eksperci od problemów gospodarczych czy państwowych będą nie tylko dostawcami wiedzy, ale jednocześnie jej użytkownikami oraz tymi, którzy systemy te rozwijają. W tym kontekście informatyce gospodarczej przypada rola katalizatora teorii i praktyki.

Informatyka gospodarcza zajmuje się problemami zastosowania komputerów w podmiotach gospodarczych. Jest zatem dyscypliną bardziej specjalizowaną niż informatyka, jako że komputery stosuje się nie tylko w przedsiębiorstwach. Jednocześnie jest jednak szerzej pojmowana niż informatyka, ponieważ jej przedmiotem są wspomagane komputerowo informacyjne systemy zarządzania działające jako systemy człowiek-ma-

szyna-oprogramowanie, każdorazowo w specyficznym organizacyjnym i ekonomicznym kontekście przedsiębiorstwa zawierającego znacznie więcej komponentów systemowych niż tylko sprzęt i oprogramowanie komputera.

Informatycznie wspierane systemy zarządzania przedmiotem informatyki gospodarczej

Przedmiotem informatyki gospodarczej są komputerowo wspierane systemy informacyjne jako systemy człowiek-maszyna przy uwzględnianiu odpowiedniego organizacyjno-ekonomicznego kontekstu przedsiębiorstwa. Pojęcie „system informacyjny" jest przez większość środowiska pojmowany jako złożone pojęcie dla wszystkich rodzajów systemów planowania, decyzji, dyspozycji, komunikacji itd., gdyż wszystkie te rodzaje systemów mają jako cel główny wspieranie procesów przetwarzania informacji. W podobnie szerokim rozumieniu używa się pojęcia „system informacyjny" w literaturze anglosaskiej.

Pojęcie „wsparte komputerowo" uważa się dzisiaj za zbyt wąskie, nie odpowiadające szerokości i głębi obecnego stanu rozwoju techniki informacyjnej. Technika ta obejmuje nie tylko technikę komputerową, ale cały szereg sąsiadujących technik, jak na przykład technikę telekomunikacyjną, automatycznego przetwarzania mowy i obrazów, sztucznej inteligencji, robotykę, nowoczesną technikę audiowizualną. Pewna specjalna właściwość, która niejako jest wspólną cechą tych różnorodnych technik informacyjnych, polega na tym, że wszystkie te techniki z wielką prędkością zbliżają się do siebie, wzajemnie zazębiają i przenikają, a jednocześnie każda z nich szeroko i niezależnie rozprzestrzenia się.

Wewnątrzzakładowe systemy informacyjne stanowią podstawowy przedmiot badań informatyki gospodarczej. Nie budzi najmniejszej wątpliwości fakt, że w przyszłości coraz więcej uwagi zwracać się będzie na systemy łączące przedsiębiorstwa ze sobą oraz na tworzone przez te przedsiębiorstwa związki.

Komponenty systemu

W praktyce często równoważnie używa się pojęcia „system informacyjny" oraz „oprogramowanie zastosowaniowe" (aplikacyjne). Występuje to szczególnie w przypadku, gdy mamy do czynienia z systemowo „dużym" oprogramowaniem zastosowaniowym. Negatywne konsekwencje mieszania tych dwóch pojęć, to niewystarczające uwzględnienie organizacyjnych i ludzkich komponentów systemu informacyjnego, co prawie zawsze powoduje nie akceptowanie systemu przez organizację oraz ludzi, którzy mieli mieć z tego systemu pożytek.

Wsparty komputerowo informacyjny system zarządzania powinien zawsze odpowiadać na wymagania i specyfikę tego przedsiębiorstwa, dla którego jest realizowany (projektowany „na miarę") i tylko w takim przypadku może być on wdrożony z powodzeniem. Stąd wynika, że przy tworzeniu systemu należy uwzględnić następujące problemy:

- ograniczenia, które w fazie tworzenia systemu nie mogą być zmienione lub zmieniane jedynie w ograniczonym zakresie; Są to warunki brzegowe danego przedsiębiorstwa natury technicznej, gospodarczej, organizacyjnej socjalnej i personalnej;
- różnorodne, często sprzeczne wyobrażenia celów kierownictwa przyszłego użytkownika systemu oraz osób „uszczęśliwionych" systemem;
- obszar rozwiązań, dopuszczalnych ze względu na specyfikę przedsiębiorstwa, jego ograniczenia ekonomiczne, organizacyjne, socjalne i personalne; W tym obszarze rozwiązań dopuszczalnych należy znaleźć rozwiązanie najlepsze.

Gotowe komponenty systemowe dla wspartych komputerowo systemów informacyjnych zarządzania, można znaleźć jedynie na płaszczyźnie sprzętu i oprogramowania. Jednak i te składowe standardowe, są w większości przypadków nie do wykorzystania jako składowe standardowe, a więc większym i mniejszym nakładem sił i środków muszą zostać dopasowane do specyfiki organizacyjnej przedsiębiorstwa. Należy więc odrzucić wciąż jeszcze pokutujący pogląd, że wystarczy kupić komputer i oprogramowanie, a wszystkie problemy przetwarzania informacji zostaną w ten sposób rozwiązane. Jedno z najważniejszych zadań informatyki gospodarczej polega na uzmysłowieniu przedsiębiorstwom, zwłaszcza średniej wielkości, a także całej gospodarce, że pogląd taki ma bardzo często negatywne skutki.

Informatyka gospodarcza jest uzależniona od informatyki jako dyscypliny inżynierskiej i nie jest wystarczająco kompletna w zakresie tworzenia komponentów sprzętowych i programowych systemów informacyjnych. Konieczną do tego wiedzą dysponuje z zasady informatyka. Odwracając jednak problem należy stwierdzić, że informatyka jest uzależniona od informatyki gospodarczej, jako że ta pierwsza nie zajmuje się strukturami organizacyjnymi i ekonomicznymi, w których muszą się mieścić wsparte komputerowo systemy informacyjne zarządzania, aby przynosiły one oczekiwane korzyści. Wiedzę o tego rodzaju implementacji posiada informatyka gospodarcza.

Zmiany w organizacji przetwarzania systemów zarządzania

Dynamicznie rosnąca liczba wprowadzonych do przedsiębiorstw komputerów osobistych z jednej strony, a potencjał obliczeniowy komputerów zakładowych i międzyzakładowych wraz z sieciami komunikacyjnymi z drugiej, prowadzą do daleko idących zmian w organizacji przetwarzania danych. Zmiany te rozciągają się na trzy różne płaszczyzny:

- Różne klasy systemów informacyjnych przedsiębiorstw stwarzają różne problemy przy ich projektowaniu. Systemy informacyjne zarządzania można bowiem podzielić na różne klasy, aby według tego podziału stosować odpowiednio różne metody ich tworzenia. Często wprowadza się rozróżnienie między systemami administracyjnymi, dyspozycyjnymi i planowania (Mertens i Griese 1988). Inna klasyfikacja odwołuje się do wielkości organizacyjnego obszaru obejmowanego przez system. Obok dominujących wciąż jeszcze systemów wydziałowych (np. dla księgowości, wydziałów personalnych lub zbytu) występują lokalne systemy zorientowane na miejsce pracy oraz infrastrukturalne systemy globalne (Seibt 1985). W związku z tym poza problemami projektowania technologicznego należy zwrócić baczną uwagę na problemy zapewniające użytkownikowi swobodę w kształtowaniu i użytkowaniu systemu. Zwiększa to różnorodność systemów informacyjnych, wykazując przy tym różnorodność problemów, z którymi ma do czynienia informatyka gospodarcza.

- W procesy tworzenia systemu wciąga się coraz więcej ludzi wykonujących określone zadania w przedsiębiorstwie. W procesie tworzenia systemu angażuje się coraz więcej przedstawicieli różnych branż. Za proces ten odpowiadało kiedyś centrum obliczeniowe, tzn. profesjonalna komórka realizacji systemu. Obecnie coraz więcej systemów tworzą działy fachowe przedsiębiorstwa, a więc użytkownik końcowy, korzystający samodzielnie z prostych języków i narzędzi programowania. Proces tworzenia systemu zostaje w ten sposób istotnie uproszczony wskutek zmiejszenia się problemów komunikacji.

dokończenie na s. 15

Piąta Jubileuszowa Wyższa Międzynarodowa Górska Szkoła PTI,
Szczyrk 21–25.06.1993 r.

MAREK WIERZBICKI
Wydział Elektryczny – Automatyka
Politechnika Łódzka

Tworzenie obiektów przez dziedziczenie mono- i polimorficzne

Programowanie obiektowe jest najnowszym sposobem programowania. Idea ta powstała jako dalsze udoskonalenie i rozszerzenie programowania strukturalnego. Umożliwia ukrycie wszystkich szczegółów (które dotąd zaciemniały obraz sytuacji) tylko po to, aby całość rozwiązania była jak najprostsza i najbardziej przejrzysta. Zbliża również do rzeczywistości modele świata opisywane w komputerze, przez co programowanie staje się bardziej naturalne.

Dla osób, które nigdy dotąd nie miały styczności z programowaniem obiektowym, najważniejsze będą trzy innowacje:

- Nowa klasa zmiennych: programowanie obiektowe realizuje się z wykorzystaniem zmiennych nowego typu. Typy te są projektowane przez użytkownika podobnie jak rekord. Oprócz danych mogą jednak zawierać także procedury i funkcje. Cała taka struktura, składająca się z danych i procedur, jest nazywana właśnie obiektem.
- Dziedziczność: jeśli do definicji nowego obiektu używamy opisu innego, wcześniej skonstruowanego, wtedy ten nowy obiekt będzie dziedziczył wszystkie cechy (zmienne i procedury) po obiekcie, którego użyliśmy do jego opisu (będzie je miał w swojej strukturze.
- Poliformizm: najatrakcyjniejsza cecha programowania obiektowego, umożliwiająca zmniejszenie pracy do minimum przy pisaniu programu korzystającego z obiektów. Idea poliformizmu polega na tym, że każdy obiekt wykorzystuje tylko swoje własne sposoby działania, nawet jeśli korzystamy z tych sposobów pośrednio (to znaczy używając procedur nie opisanych konkretnie w tym obiekcie, a tylko dziedziczonych).

Te trzy innowacje dają już pełne możliwości programowania obiektowego. Dokładnie takie same możliwości dostępne są w językach C++, Turbo Pascal, Simula, SmallTalk czy Eiffel. Warto jednak zwrócić uwagę na różne nazewnictwo i trochę inne sposoby konstrukcji obiektów w różnych językach – wynikają one najczęściej ze struktury samego języka, który posłużył za podstawę realizacji idei obiektowej. W poniższym artykule będę korzystał ze składni i nazewnictwa stosowanego w Turbo Pascalu. Na zakończenie przedstawię główne różnice między Turbo Pascalem a C++ – drugim najpopularniejszym językiem programowania, udostępniającym możliwości programowania obiektowego.

Zajmiemy się teraz podstawami programowania obiektowego i przyjrzyjmy się dokładnie wszystkim innowacjom wprowadzonym do języka strukturalnego. Dotychczas w Turbo Pascalu dostępna była tylko jedna klasa zmiennych, która mogła być projektowana przez użytkownika. Był to rekord – struktura, która umożliwia połączenie (zgromadzenie) kilku zmiennych prostych opisujących np. jeden przedmiot, pod jedną nazwą. Takie połączenie ułatwia operacje na wszystkich danych zawartych w rekordzie. Jeśli chcielibyśmy opisać za pomocą zmiennych kilka cech człowieka, należałoby skonstruować rekord, który zawierałby takie pola (poszczególne zmienne), jak: wzrost, waga, wiek, płeć, kolor skóry itd. Wykorzystanie takich rekordów np. w bazach danych bardzo ułatwia pracę. Zmienne *Człowiek_1* i *Człowiek_2*, które są tego samego typu, mają dokładnie taką samą strukturę. Dość łatwo można więc zaprojektować procedury, które zewnętrznie oddziaływują na te zmienne. Jednak do potrzeb symulacji komputerowych takie rozwiązanie było niewystarczające. Wiadomo, że każdy człowiek, mimo ogólnych podobieństw, zachowuje się jednak trochę inaczej. Najbardziej rzucającym się w oczy rozwiązaniem było więc przypisanie do każdej zmiennej złożonej całego zestawu procedur, opisujących działania tylko i wyłącznie tej zmiennej. Od takiego rozwiązania był już tylko mały krok do stworzenia obiektu – połączenia w jednej strukturze danych i procedur opisujących działania na tych danych. Cała taka struktura oglądana z zewnątrz dawała już dużo bliższą symulację życia codziennego. Wystarczyło do każdej struktury danych dodać procedury opisujące działania na niej (podobne, lecz nie jednakowe) aby otrzymać modele niezbędne w symulacji.

Z języków symulacyjnych obiekty szybko przeniesiono do zwykłych języków programowania. Co więcej, każdy z nas nieświadomie używa kilku konstrukcji, które są typowo obiektowymi rozwiązaniami. Potrzebujecie przykładu? Proszę bardzo. Wyobraźmy sobie, że mamy dwie pary zmiennych: *aw* i *bw* – liczby typu Word oraz *ar* i *br* – liczby typu Real. Jeśli w programie wykonujemy mnożenie liczb *aw* i *bw* oraz *ar* i *br*, używamy do tego (w obu przypadkach) operatora*. Jednak te dwa mnożenia wykonane będą na dwa różne sposoby (inny zestaw instrukcji zastąpi mnożenie liczb typu Word, a inny mnożenie liczb typu Real). Nie jest to nic innego, jak skojarzenie pewnych procedur (metod działania) z niektórymi typami zmiennych. Oczywiście w Turbo Pascalu nie jest to rozwiązane w sposób obiektowy (razem z każdą zmienną nie trzymamy w pamięci kopii wszystkich procedur i funkcji działających na tej zmiennej). Teoretycznie można by wykonać to właśnie tak (nie zważając na ograniczenia pamięci i niemożność przeciążania operatorów w Turbo Pascalu). Oczywiście tworzenie tak prostych obiektów służy tylko zaprezentowaniu możliwości i wprawkom programistycznym. Naprawdę atrakcyjne efekty osiąga się dopiero przy tworzeniu dużych struktur i programów.

Wyobraźmy sobie, że przygotowujemy się do stworzenia dużego programu komputerowego. W poniższym artykule zajmiemy się sprawą komunikacji tego programu z użytkownikiem (nie będziemy prowadzili rozważań o jego merytorycznym działaniu). Podstawowym obiektem naszej biblioteki będzie obiekt *Prostokąt*, którego definicja pokazana jest poniżej:

```
type
  Prostokat=object
    AX, AY: Integer;
    BX, BY: Integer;
  end;
```

Poszczególne pola oznaczają położenie lewego górnego rogu (*AX, AY*) i prawego dolnego rogu (*BX, BY*) we współrzędnych ekranowych. Cały obiekt określa tylko współrzędne prostokąta i jako taki nie jest jeszcze zbyt użyteczny. Będzie jednak mógł być używany do dalszego dziedziczenia. Przykładem stworzenia obiektu, który nadal nie ma żadnych metod jest definicja pokazana poniżej:

```
type
  Okno_Dane=object(Prostokat)
    Ramka: String[8];
    Wnetrze: Pointer;
    Tlo: Pointer;
    CienX, CienY: Byte;
    CienAttr: Byte;
  end;
```

Obiekt *Okno_Dane* zawiera najważniejsze informacje o oknie. Oprócz mechanizmu dziedziczenia, w definicji tego obiektu nie ujawniają się żadne inne cechy programowania obiektowego. Pokazana powyżej definicja określa obiekt, który logicznie ma postać:

```
type
  Okno_Dane=object
    AX, AY: Integer;
    BX, BY: Integer;
    Ramka: String[8];
    Wnetrze: Pointer;
    Tlo: Pointer;
    CienX, CienY: Byte;
    CienAttr: Byte;
  end;
```

Jak widać, mimo że pola określające położenie prostokąta nie zostały jawnie zadeklarowane, istnieją one w tym obiekcie. Jak dotąd obiekt niewiele różni się od standardowego rekordu. Nową możliwością jest tylko dziedziczenie. Dodajmy teraz do obiektu procedury, które będą działaniami operującymi na obiekcie (czyli na własnych danych). Poniżej przedstawiam definicję obiektu *OknoBazoweM*, które oprócz pól (dziedziczonych z obiektu *Okno_Dane*) zawiera procedury na nim działające (rysujące to okno, przesuwające je itd.):

```
type
  OknoBazoweM=object(Okno_Dane)
    procedure ZapamietajTlo;
    procedure OdtworzTlo;
    procedure Rysuj;
    procedure RysujRamke;
    procedure RysujCien;
    procedure RysujWnetrze;
    procedure Przesun;(DX, DY: Integer);
  end;
```

Interesujące nas w tych rozważaniach i wymienione w definicji metody mają następującą implementację:

```
procedure OknoBazoweM.Przesun
                  (DX, DY: Integer);
begin
  OdtworzTlo;
  AX:=AX+DX;
  BX:=BX+DX;
  AY:=AY+DY;
  BY:=BY+DY;
  Rysuj;
end;
```

```
procedure OknoBazoweM.Rysuj;
begin
  ZapamietajTlo;
  RysujCien;
  RysujRamke;
  RysujWnetrze;
end;

procedure OknoBazoweM.RysujWnetrze;
begin
end;
```

Jak widać, implementacja metod jest bardzo zbliżona do implementacji zwykłych procedur. Mimo tego, że metoda *RysujWnętrze* nie wykonuje żadnej akcji, jest ona wywoływana z metody *Rysuj* w celu zachowania konwencji. Obiekt *OknoBazoweM* może być teraz rozszerzany na wiele różnych sposobów. Możemy z jego użyciem stworzyć okno dialogowe, okno edycyjne itp. Dla przykładu pokażę fragment definicji i deklaracji obiektu *OknoDialogoweM:*

```
type
  OknoDialogoweM=
              object(OknoBazoweM)
    procedure Przesun(DX, DY:
                      Integer);
    procedure Rysuj;
    procedure RysujWnetrze;
  end;

procedure
          OknoDialogoweM.Przesun
              (DX, DY: Integer);
begin
  OdtworzTlo;
  AX:=AX+DX;
  BX:=BX+DX;
  AY:=AY+DY;
  BY:=BY+DY;
  Rysuj;
end;

procedure OknoDialogoweM.Rysuj;
begin
  ZapamietajTlo;
  RysujCien;
  RysujRamke;
  RysujWnetrze;
end;

procedure
      OknoDialogoweM.RysujWnetrze;
begin
  {rysowanie wnętrza w sposób}
  {charakterystyczny dla okna}
  {dialogowego na podstawie danych}
  {opisanych strukturą}
  {znajdującą się pod adresem}
  {określonym przez Wnetrze}
end;
```

W implementacji metod obiektu *OknoDialogoweM* należy zwrócić uwagę na metody *Rysuj* i *Przesuń*. Jak widać, są one identyczne z metodami o tych samych nazwach obiektu, po którym *OknoDialogoweM* dziedziczy własności. Dlaczego więc ponowna implementacja (a właściwie skopiowanie) tych metod jest konieczna? Dla procedury *Przesuń* problemem jest działanie metody *Rysuj*, a dla *Rysuj* – działanie metody *RysujWnętrze*. *RysujWnętrze*, które znajduje się na końcu kolejki wywołań, zostało zmienione w stosunku do metody *RysujWnętrze* obiektu *OknoBazoweM*. Naturalne wydaje się więc, że wszystkie te metody, które pośrednio lub bezpośrednio odwołują się do *Rysujwnętrze*, muszą zostać zmienione.

Takie podejście wydaje się naturalne dla osób, które nie miały dotąd styczności z programowaniem obiektowym i z obiektami polimorficznymi. Poniżej przedstawię fragmenty definicji dwóch obiektów polimorficznych, które funkcjonalnie będą takie same, jak przedstawione wcześniej. *OknoBazoweP* jest obiektem polimorficznym, który może służyć jako podstawa do tworzenia następnych obiektów (okna dialogowego, edycyjnego itp.). Definicja i fragment implementacji tego obiektu są następujące:

```
type
  OknoBazoweP=object(Okno_Dane)
    constructor Init(NAX, NAY, NBX,
                     NBY: Integer);
    destructor Done;
    procedure ZapamietajTlo;
    procedure OdtworzTlo;
    procedure Rysuj;
    procedure RysujRamke;
    procedure RysujCien;
    procedure RysujWnetrze;
                          virtual;
    procedure Przesun(DX, DY:
                      Integer);
  end;

procedure OknoBazoweP.Przesun
                (DX, DY: Integer);
begin
  OdtworzTlo;
  AX:=AX+DX;
  BX:=BX+DX;
  AY:=AY+DY;
  BY:=BY+DY;
  Rysuj;
end;
```

```
procedure OknoBazoweP.Rysuj;
begin
  ZapamietajTlo;
  RysujCien;
  RysujRamke;
  RysujWnetrze;
end;

procedure OknoBazoweP.RysujWnetrze;
begin
end;
```

W stosunku do obiektu *OknoBazoweM* obiekt ten ma dwie modyfikacje:

- metoda *RysujWnętrze* jest metodą wirtualną (polimorficzną);
- dołożono dwie specjalizowane metody: konstruktor *Init* oraz destruktor *Done*.

Możliwości i zalety metody *RysujWnętrze*, która ma teraz nową kwalifikację, omówione zostaną później. Jak widać, metoda ta, podobnie jak w obiekcie *OknoBazoweM* jest pusta (nie wykonuje żadnej akcji). Specjalizowane metody – konstruktor i destruktor służą do prawidłowej inicjacji i dezaktywacji obiektu. Implementacja konstruktora i destruktora prawie nie różni się od metod standardowych. Najważniejsze są operacje, które dopisywane są automatycznie przez kompilator, zarówno do konstruktora, jak i destruktora. Operacje te są istotne ze względu na techniczny sposób przechowywania informacji o obiektach i metodach polimorficznych (inny, niż w przypadku obiektów monoformicznych). Zalety polimorfizmu ujawniają się dopiero w przypadku dziedziczenia i tworzenia obiektu o pochodnych własnościach. Definicja oraz implementacja obiektu *OknoDialogoweP* mają następującą postać:

```
type
  OknoDialogoweP=
                object(OknoBazoweP)
    procedure RysujWnetrze;
                          virtual;
  end;
```

```
procedure
      OknoDialogoweP.RysujWnetrze;
begin
  {rysowanie wnętrza w sposób}
  {charakterystyczny dla okna}
  {dialogowego na podstawie danych}
  {opisanych strukturą}
  {znajdującą się pod adresem}
  {określonym przez Wnetrze}
end;
```

Jak widać, jedyną metodą, która odróżnia obiekt *OknoDialogoweP* od obiektu *OknoBazoweP*, jest *RysujWnętrze* (jest to metoda polimorficzna – wirtualna). Mimo tego obiekty *OknoDialogoweP* i *OknoDialogoweM* działają tak samo. Jak to możliwe? Otóż jest to właśnie zaleta polimorfizmu. Wszystkie metody wirtualne są wywoływane w inny sposób niż metody standardowe (statyczne). Można to wyjaśnić przedstawiając logiczną implementację metody *Rysuj*:

```
procedure OknoBazoweP.Rysuj
                      (var Self);
begin
  OknoBazoweP.ZapamietajTlo;
  OknoBazoweP.RysujCien;
  OknoBazoweP.RysujRamke;
  Self.RysujWnetrze;
end;
```

Jak widać, wszystkie odwoływania do metod statycznych dotyczą tych metod, które są zaimplementowane w tym obiekcie (lub są dziedziczone przez ten obiekt). Inaczej jest z wirtualną metodą *RysujWnętrze*. Odwołanie do niej jest realizowane przez deskryptor *Self*. Określa on wskazanie do aktualnie używanego obiektu (niekoniecznie do *OknoBazoweP*). Jeśli więc metoda *Rysuj* zostanie wywołana jako metoda obiektu *OknoDialogoweP*, wtedy jako *Self* zostanie podstawiony adres tego obiektu i metoda ta będzie miała postać:

```
procedure OknoDialogoweP.Rysuj;
begin
  OknoBazoweP.ZapamietajTlo;
  OknoBazoweP.RysujCien;
  OknoBazoweP.RysujRamke;
  OknoDialogoweP.RysujWnetrze;
end;
```

Jak widać, mimo tego, że w chwili projektowania obiektu *OknoBazoweP* nic nie było jeszcze wiadomo o obiekcie *OknoDialogoweP* (jeszcze on nie istniał), metoda *Rysuj* wywoła metodę *RysujWnętrze* obiektu *OknoDialogoweP*.

Czy to wszystko? Tak! Naprawdę idea polimorfizmu jest bardzo prosta i polega na dynamicznym sposobie wywoływania własnych metod danego obiektu (odwołania do tych metod są konstruowane w czasie pracy programu, a nie jego konsolidacji). Metody te będą prawidłowo wywoływane nawet wtedy, jeśli w chwili tworzenia procedur wywołujących nie były znane obiekty końcowe. Jakie są zalety stosowania obiektów polimorficznych w porównaniu do obiektów standardowych (monomorficznych)? Przede wszystkim szybkość i łatwość tworzenia nowych obiektów. Porównajmy definicję i implementację obiektów *OknoDialogoweP* i *OknoDialogoweM*. Obiekt polimorficzny jest dużo prostszy w definicji i dużo szybszy w implementacji. Mimo tego, że ta sama metoda, zrealizowana jako polimorficzna, zajmuje więcej miejsca (dochodzą tu niewidoczne dla programisty nagłówki), program wykorzystujący obiekty polimorficzne jest mniejszy (nie trzeba przechowywać w pamięci zestawów jednakowych procedur). Dodatkowo, co częściowo było założeniem twórców programowania zorientowanego obiektowo, łatwiej jest ukrywać niektóre szczegóły implementacyjne przez tworzenie tzw. obiektów abstrakcyjnych (obiekt abstrakcyjny to jakby szkielet, wewnątrz którego w procesie dziedziczenia są implementowane tylko metody polimorficzne, a szkielet pozostaje stały). Dobrze zaprojektowany bazowy obiekt abstrakcyjny ponad dwukrotnie przyspiesza pracę przy tworzeniu programu. Różnicę między Turbo Pascalem a C++ w programowaniu obiektowym nie są zbyt wielkie. Ponieważ w swoim artykule używałem składni i określeń charakterystycznych dla Turbo Pascala, czuję się w obowiązku zaznaczyć różnice związane z omawianymi powyżej problemami, które będą łatwe do wychwycenia dla znawców języka C:

- W C++ do definicji obiektów (klas) używa się tych samych słów kluczowych, jakie były używane do definiowania rekordów (*class, struct, union*);
- Konstruktory i destruktory mają w C++ nazwy obligatoryjne, pochodzące od nazw klas. W Turbo Pascalu nazwy są dowolne, choć preferowane są odpowiednio *Init* oraz *Done*;
- Konstruktory i destruktory klas są w C++ uruchamiane automatycznie w chwili pojawienia się deklaracji użycia zmiennej obiektowej (przydzielanie pamięci) oraz przy zakończeniu jej życia (zwalnianie pamięci). W Turbo Pascalu konstruktory i destruktory trzeba wywoływać jawnie;
- W Turbo Pascalu może istnieć więcej niż jeden destruktor (w C++ jest to niemożliwe). Jeśli nie definiujemy żadnego destruktora, to go po prostu nie ma (w C++ tworzony jest domyślny – pusty destruktor).

Informatyka gospodarcza

dokończenie ze s. 10

Z drugiej strony, w istotny sposób staje się on bardziej skomplikowany wskutek potrzeby globalnych ustaleń i centralizacji nakładów. Kierowanie tymi procesami wiąże się z nieznanymi dotychczas problemami.

- Silnie zmieniają się powstające w realnie istniejącej rzeczywistości procesy tworzenia i konserwacji systemu, jak również schematy działania. Nowoczesne założenia inżynierii oprogramowania mają istotny wpływ na tworzone systemy zarządzania. Obok tego powstają nowe formy organizacyjne (zespoły, samodzielne stanowiska pracy) ułatwiające wypracowanie najlepszej organizacji.

Modele jako punkty wyjścia dla tworzenia systemów

Wychodząc z określonych modeli informatyka gospodarcza zajmuje się tworzeniem, wdrażaniem i konserwacją systemów zarządzania, jak również ich dostosowaniem do właściwej organizacji. Obiektami badań i tworzenia są wspierane komputerowo systemy (DSS – *Decision Support Systems*):

- sterowania i planowania produkcji,
- sterowania służbami zewnętrznymi,
- gospodarki towarowej,
- płatności bankowych,
- finansowo-księgowe,
- rachunkowości,
- informacyjno-kadrowe,
- rewidenckie.

Wybrane dziedziny badań informatyki gospodarczej

Kurbel i Strunz (1990) wyodrębnili następujące dziedziny badawcze informatyki gospodarczej:

- architektura zastosowań w gospodarce i administracji,
- konstruowanie i realizacja zastosowań systemów zarządzania,
- systemy oparte na bazie wiedzy,
- komputerowo wsparte stanowisko pracy,
- kierowanie informacją,
- rynki informatyczne,
- podstawowe technologie.

Zarysowałem jedynie niektóre problemy informatyki gospodarczej. Jest ich znacznie więcej. Sądzę, że będziemy zmuszeni częściej do nich wracać. Wiąże się to z naturalnym naciskiem, jaki na tak zdefiniowany podzbiór informatyki wywiera otaczający nas świat oraz wynikające problemy gospodarki.

LITERATURA

[1] Kurbel K., Strunz H.: Wirtschaftsinformatik – eine Einführung. In: Handbuch der Wirtschaftsinformatik. Hrsg. von K. Kurbel und H. Struntz, Stuttgart 1990, S. 1–25.

[2] Müller-Merbach H.: Betriebswirtschaftslehre nach dem Jahr 2000. In: Zukunftsaspekte der anwendungsorientierten Betriebswirtschaftslehre. Hrsg. von E. Gaugler, H.G. Meissner und N. Thom, Stuttgart 1986, S. 497–511

[3] Seibt D.: ADP – Application Systems. In: Handbook of German Business Management, Edited by E. Grochla et al. Stuttgart, Heidelberg, Berlin, 1990, pp. 112–126.

TADEUSZ WITKOWSKI
Kombinat Przemysłu Narzędziowego VIS
Warszawa

Wspomaganie decyzji o wyborze zintegrowanego systemu CAD/CAM

Zintegrowane systemy CAD/CAM komputerowego wspomagania projektowania i technologicznego przygotowania produkcji są podstawą do wdrażania zintegrowanych systemów zarządzania przedsiębiorstwem. Takie systemy zapewniają żywność przedsiębiorstwu i stwarzają możliwość jego rozwoju przy obecnej silnej konkurencji rynkowej. Prawidłowe zaprojektowane i właściwie wybrane przez firmę użytkownika systemy CAD/CAM mogą stworzyć środek strategiczny decydujący o jej przetrwaniu i rozwoju. Wdrożenie takich systemów zwiększa konkurencyjność firmy również ze względu na taktyczne ich znaczenie dla działalności produkcyjno-finansowej.

W artykule zwrócono uwagę na dwa zagadnienia. Po pierwsze – przedstawiono kompleks modeli dla komputerowego wspomagania decyzji w warunkach ryzyka i niepewności, po drugie – poruszono niektóre problemy związane z oceną wybieranego w przetargu najlepszego wariantu zintegrowanego systemu CAD/CAM. Szczególną uwagę skierowano na wspomaganie procesu podejmowania decyzji przez dostawcę systemu, gdyż problem wyboru przez kupującego w warunkach przetargu przedstawiono szerzej w pracy [14].

Celowość zakupu zintegrowanego systemu informatycznego CAD/CAM

Przy planowaniu przez firmę specyfiki i asortymentu wyrobów powinny być rozpatrywane jednocześnie wymagania dotyczące procesów produkcyjnych, co nie zawsze jest szczegółowo analizowane ze względu na mały stopień integracji między strategiami rozwoju wyrobów i procesów produkcyjnych. Na przykład, jeżeli sytuacja rynkowa wymaga znacznego zmniejszenia kosztów produkcji, to, aby firma uzyskała konkurencyjną cenę wyrobu, jest ona zmuszona do modyfikacji swojej polityki produkcyjnej, głównie przez jej uelastycznienie. Chociaż zakup i wdrożenie systemu CAD/CAM może w takim przypadku okazać się pomocnym, to jednak taka decyzja powinna uwzględniać opłacalność wdrożenia tego rodzaju systemu. Należy zauważyć, że systemy CAD/CAM skracają czas produkcji o ok. 20%, natomiast jakość wyrobów ulega poprawie o kilkanaście procent. Chociaż systemy te są bardzo kapitałochłonne, to jednak okres ich amortyzacji jest stosunkowo krótki bo wynosi 28–60 miesięcy.

Projekty systemów CAD/CAM mają głównie na celu uelastycznienie procesów związanych z projektowaniem wyrobów i technologicznym przygotowaniem produkcji. Powinny one uwzględniać m.in. następujące wymagania [2, 4, 8, 11]:
- mieć łączność sieciową dla ułatwienia procesów zarządzania i przesyłania informacji między gniazdami i wydziałami produkcyjnymi oraz działami funkcjonalnymi firmy;
- zapewniać rozproszoną konfigurację systemu celem uniknięcia nadmiernej koncentracji w jednym miejscu stanowisk analizy i przetwarzania informacji;
- uwzględniać możliwość adaptacji oprogramowania pozwalającą na łatwe i szybkie jego wykorzystanie dla różnych zastosowań;
- przewidywać możliwość wykorzystania technicznych środków automatyzacji projektowania wyrobów i procesów technologicznych pochodzących od różnych dostawców.

Podstawowymi taktycznymi korzyściami z wdrożenia w przedsiębiorstwie systemu CAD/CAM są [2, 4, 6, 8, 9, 10, 11, 12]:
- wzrost jakości wyrobów i skrócenie czasu ich dostawy do klienta;
- wzrost wartości użytkowej wyrobów;
- skrócenie czasu opracowania nowych procesów technologicznych;
- zmniejszenie liczby błędów i łatwość wprowadzania zmian konstrukcyjnych;
- łatwość i precyzja odtwarzania dokumentacji konstrukcyjnej;
- możliwość wspomagania komputerowego sterowania procesów produkcyjnych (programowanie obrabiarek sterowanych numerycznie i robotów);
- zwiększenie elastyczności działania wydziałów produkcyjnych dzięki układom sterowania numerycznego;
- wprowadzenie do obliczeń konstrukcyjnych metod wymagających zastosowania techniki komputerowej, np. metody elementów skończonych i wielokryterialnej optymalizacji konstrukcji;
- pełna zgodność konstrukcji z obowiązującymi normami i listami preferencyjnymi, przechowywanymi i bieżąco aktualizowanymi w bazie danych systemu;
- zastosowanie symulacyjnych metod sprawdzania poprawności działania programów sterujących numerycznymi urządzeniami technologicznymi;
- możliwość sprawdzania bezkolizyjności działania urządzeń technologicznych;
- powiązanie wszystkich działów przedsiębiorstwa wspólną bazą danych o łatwym i szybkim dostępie;
- zmniejszenie kosztów wyszukiwania informacji;
- zmniejszenie kosztów projektowania nowych wyrobów i technologicznego przygotowania produkcji.

Proces wyboru dostawcy zintegrowanego systemu CAD/CAM

Etapy wyboru najlepszego dostawcy zintegrowanego systemu CAD/CAM są następujące [4, 13]:
- formułowanie zestawu wymaganych właściwości systemu (przygotowanie zapytania ofertowego dla dostawcy);

- wstępna ocena wariantów rozwiązań alternatywnych systemu;
- rozmowy handlowe z dostawcami systemów;
- przeprowadzenie testów poziomu technicznego wyrobów wybranych potencjalnych dostawców;
- opracowanie macierzy wspomagającej proces podejmowania decyzji (końcowa ocena wariantów rozwiązań alternatywnych);
- wybór końcowy najlepszego wariantu systemu.

Jest oczywiste, że proces podejmowania decyzji przez kupującego w formie przetargu jest bardzo ściśle powiązany z procesem podejmowania decyzji przez sprzedawcę. Dlatego też niektóre modele wykorzystywane do wspomagania decyzji są podobne, przy czym kupujący w większości przypadków dokonuje wyboru na podstawie informacji stosunkowo pewnej, chociaż niepełnej, natomiast sprzedawca ocenia własne możliwości wygrania przetargu w warunkach ryzyka i niepewności. W takich przypadkach celowe jest zastosowanie aparatu matematycznego wykorzystującego teorię zbiorów rozmytych. Podstawowymi określeniami tej teorii są: funkcja przynależności oraz zmienna lingwistyczna. Tak np. dla oceny niezawodności systemu ocena eksperta może być wyrażona za pomocą zmiennej lingwistycznej „niezawodność", której zbiór terminów można przedstawić np. następująco: niezawodny, bardzo niezawodny,.., zawodny, bardzo zawodny. Przeprowadzenie oceny wymaga przy tym wprowadzenia pojęcia bazowej zmiennej lingwistycznej. Jako zmienną bazową dla wspomnianej zmiennej „niezawodność" można przyjąć spadek (lub wzrost) efektywności funkcjonowania systemu w procentach.

Zarówno dla sprzedającego jak i kupującego bardzo istotny jest problem wyboru jednego lub większej liczby dostawców. W przypadku systemu CAD/CAM motywacją przemawiającą za wyborem jednego dostawcy są m.in. takie okoliczności, jak [6, 9, 10, 12]:

- małe doświadczenie użytkowników w zakresie wdrożenia systemów CAD/CAM oraz nieznaczna ilość sprzętu informatycznego;
- możliwość stopniowego zastosowania systemu do rozwiązywania niektórych problemów zarządzania produkcją;
- stosunkowo mała liczba stanowisk roboczych (np. kilka) do rozwiązania danego problemu.

Z kolei motywacją przemawiającą za dostawą systemu CAD/CAM przez kilku dostawców są m.in. takie okoliczności, jak:

- konieczność jednoczesnego uwzględnienia wielu dziedzin zastosowań (czego raczej nie podejmuje się żaden z dostawców systemu CAD/CAM);
- brak wymagania w zakresie szczególnych warunków przetwarzania i przesyłania informacji w różnych dziedzinach zastosowań;
- istnienie w przedsiębiorstwie zastosowań zrealizowanych już przez kilku dostawców;
- dysponowanie przez użytkownika doświadczeniem w zakresie wdrożenia systemu CAD/CAM;
- istnienie i wykorzystywanie złożonego oprogramowania do obliczeń optymalizacyjnych.

Analizując powyższe okoliczności można zauważyć, że większość współczesnych przedsiębiorstw ma już pewne doświadczenie w dziedzinie komputerowego przetwarzania informacji, a często dysponuje także znaczną ilością sprzętu informatycznego. Z drugiej strony, niewiele jest firm, które mogłyby spełnić wszystkie wymagania związane z wdrożeniem tak skomplikowanego systemu informatycznego. Ponadto, ze względu na szybszy zwrot nakładów inwestycyjnych, celowe jest dążenie do wykorzystania systemu w możliwie najszerszym zakresie, ponieważ tylko wtedy możemy znacznie podwyższyć efektywność jego działania. Uwzględniając powyższe przyjmiemy, że kupujący zdecydował się na zakup systemu od kilku dostawców również ze względów finansowych, gdyż obecność konkurentów przyczynia się do obniżenia cen. Wysłane do dostawców zapytania ofertowe dotyczą zarówno dostawy kompleksowej, jak i dostawy poszczególnych grup oprogramowania i sprzętu informatycznego.

Niech w naszym przypadku rynkiem będzie przedsiębiorstwo, które jest w stanie zakupić sprzęt i oprogramowanie dla systemu CAD/CAM. Zestawiając grupy sprzętu i oprogramowania x_i (rys. 1) w zależności od wypełnianych przez nie zadań funkcjonalnych oraz integrujących, możemy dla naszego problemu wydzielić 12 segmentów rynku (rys. 2). Zakładamy ponadto, że kupujący (organizujący przetarg) zamierza zakupić system od kilku dostawców, a mianowicie od dostawcy $A = \{x_1, x_4, x_7, X_{10}\}$, od dostawcy $B = \{x_2, x_5, x_8, x_{11}\}$ oraz od dostaw-

CAD/ CAM	Oprogramowanie dla integracji systemów CAD/CAM	Oprogramowanie dla integracji CAD i komputerowego systemu sterowania produkcją	Oprogramowanie dla współpracy systemów CAD z oprogramowaniem do obliczeń optymalizacyjnych oraz dwuwymiarowego i trójwymiarowego odwzorowania konstrukcji	Systemy informatyczne wspomagające funkcje zarządzania przedsiębiorstwem oraz środki biurotyki
CAM	Systemy zarządzania bazami danych oraz metody klasyfikacji i kodowania dokumentacji konstrukcyjnej i technologicznej	Oprogramowanie oraz pakiety automatyzacji programowania dla obrabiarek sterowanych numerycznie (procesory geometryczne oraz postprocesory)	Oprogramowanie dla sterowania robotów oraz systemy komputerowego modelowania robotów	Oprogramowanie dla realizacji zadań w zakresie organizacji i technologicznego przygotowania produkcji
CAD	Środki komputerowego dwuwymiarowego odwzorowania konstrukcji	Środki komputerowego trójwymiarowego odwzorowania konstrukcji (modelowania geometrycznego)	Oprogramowanie dla analizy oraz obliczeń optymalizacyjnych (np. pakietyoprogramowania metody elementów skończonych - MES)	Oprogramowanie rysunkowe oraz formy przedstawienia informacji wejściowej i wyjściowej

Rys. 1. Przykładowe grupy oprogramowania i sprzętu informatycznego systemu CAD/CAM

cy $C - \{x_3, x_6, x_9, x_{12}\}$. Dostawcy mogą jedynie snuć domysły o decyzji kupującego, tym bardziej, że może ona być zmieniona w rezultacie różnych okoliczności.

Rozpatrzmy model wspomagania decyzji dotyczący określenia szans sprzedawcy w określonym segmencie rynku. Jest to o tyle ważne, że w sensie praktycznym firma powinna opracować taki kompleks przedsięwzięć, który pozwoliłby stworzyć możliwość sprzedaży swoich produktów i aby jej produkt znalazł się w zestawie wyboru [7] ustalonym przez kupującego. Oprócz tego istotne jest określenie produktów, wchodzących wg dostawcy w zestaw wyboru, celem poznania konkurentów oraz określenia ich „użyteczności informacyjnej", co w rozmowach handlowych pozwoli przygotować odpowiednią argumentację o wyższości własnego produktu.

Wybór segmentów rynku przez uczestników przetargu

Model podziału rynku na segmenty w zastosowaniu do naszego problemu jest następujący:

- istnieje rynek X (przedsiębiorstwo organizujące przetarg na zakup określonych grup sprzętu i oprogramowania) i segmenty x_i, $X = \{x_1, x_2, ..., x_n\}$;
- w przetargu uczestniczy zbiór Z firm konkurencyjnych z_i, $Z = \{z_1, z_2, ..., z_m\}$;
- firmy są oceniane na podstawie zbioru Y kryteriów y_i, $Y = \{y_1, y_2, ..., y_p\}$;
- firma z_i ma przewagę nad firmą z_j na podstawie kryterium y_i, jeżeli jej ocena jest bardziej zbliżona do wymagań kupującego.

Niech $A_1, A_2, ..., A_m$ – ograniczone wypukłe podzbiory rozmyte, określające stopnie preferencji dla firm $z_1, z_2, ..., z_m$, scharakteryzowane funkcjami przynależności $\mu_{A_1}, \mu_{A_2}, ..., \mu_{A_m}$.

Ponadto niech $\alpha_R : X \times Y \to [0, 1]$ jest funkcją przynależności rozmytej relacji binarnej R.

Dla wszystkich $x \in X$ oraz dla wszystkich $y \in Y$ funkcja $\alpha_R(x, y)$ oznacza stopień ważności kryterium, określony przez kupującego X, przy ocenie przez niego priorytetu firmy z_i na dostawę poszczególnych grup sprzętu i oprogramowania.

Relację R można przedstawić w postaci macierzy:

$$R = \begin{vmatrix} & y_1 & y_2 & \dots & y_p \\ x_1 & \alpha_R(x_1, y_1) & \alpha_R(x_1, y_2) & \dots & \alpha_R(x_1, y_p) \\ x_2 & \alpha_R(x_2, y_1) & \alpha_R(x_2, y_2) & \dots & \alpha_R(x_2, y_p) \\ . & & & & \\ . & & & \dots & \\ . & & & & \\ x_n & \alpha_R(x_n, y_1) & \alpha_R(x_n, y_2) & \dots & \alpha_R(x_n, y_p) \end{vmatrix}$$

Niech $\beta_S : Y \times Z \to [0, 1]$ jest funkcją przynależności rozmytej relacji binarnej S. Dla wszystkich $y \in Y$ oraz wszystkich $z \in Z$, $\beta_S(y, z)$ – stopień przynależności firmy z_i do wymagań klienta na podstawie kryterium y_i. Postać macierzy relacji S jest następująca:

$$S = \begin{vmatrix} & z_1 & z_2 & \dots & z_m \\ y_1 & \beta_S(y_1, z_1) & \beta_S(y_1, z_2) & \dots & \beta_S(y_1, z_m) \\ y_2 & \beta_S(y_2, z_1) & \beta_S(y_2, z_2) & \dots & \beta_S(y_2, z_m) \\ . & & & & \\ . & & & \dots & \\ . & & & & \\ y_p & \beta_S(y_p, z_1) & \beta_S(y_p, z_2) & \dots & \beta_S(y_p, z_m) \end{vmatrix}$$

Następnie jest konstruowana macierz T, której elementami są funkcje przynależności $\mu_{A_i}(x, z_i)$ określane jako:

$$\mu_{A_i}(x, z_i) = \frac{\sum_y \alpha_R(x, y)\, \beta_S(y, z_i)}{\sum_y \alpha_R(x, y)} \qquad \text{dla } x \in X,\ y \in Y \text{ i } z \in Z.$$

Suma $\sum_y \alpha_R(x, y)$ jest równa stopniowi rozmytego podzbioru mającego postać

$$l < \max_x \min\, [\mu_{A_1}(x), \mu_{A_2}(x)] = \sup \mu_{A_1 \cap A_2}(x),$$

który przedstawia liczbę ważniejszych kryteriów y, którą dostawca wykorzystuje dla oceny grupy sprzętu i oprogramowania x_i firmy z_i, a $\mu_{A_i}(x, z_i)$ jest interpretowana jak ważona ocena priorytetu firmy z_i odnośnie segmentu x_i (grupy sprzętu i oprogramowania).

Następnie jest konstruowana macierz W o następującej postaci:

$$W = \begin{vmatrix} \mu_{A_1}(x_1,z_1) \wedge \mu_{A_2}(x_1,z_2) & \dots & \mu_{A_{m-1}}(x_1,z_{m-1}) \wedge \mu_{A_m}(x_1,z_m) \\ \mu_{A_1}(x_2,z_1) \wedge \mu_{A_2}(x_2,z_2) & \dots & \mu_{A_{m-1}}(x_2,z_{m-1}) \wedge \mu_{A_m}(x_2,z_m) \\ & \dots & \\ \mu_{A_1}(x_n,z_1) \wedge \mu_{A_2}(x_n,z_2) & \dots & \mu_{A_{m-1}}(x_n,z_{m-1}) \wedge \mu_{A_m}(x_n,z_m) \end{vmatrix}$$

W przedstawionym modelu [5] próg segmentacji rynku jest ograniczony warunkiem $l < \min_{ij} \max_x \min\, [\mu_{A_i}(x, z_i),\ \mu_{A_j}(x, z_j)]$ i wtedy „wypadkowe" segmenty rynkowe M_i, $i = 1, 2, ..., m$ są przedstawione za pomocą zbioru:

$$M_i = \{x | \mu_{A_i}(x) \geqslant \min_{ij} \max_x \min\, [\mu_{A_i}(x, z_i),\ \mu_{A_j}(x, z_j)]\}$$

dla wszystkich $x \in M_i$.

Przykład 1. Niech $X = \{x_1, x_2, ..., x_{12}\}$ – segmenty rynkowe (rys. 2), na które są dostarczane odpowiednie grupy sprzętu i oprogramowania systemu CAD/CAM; $Z = \{z_1, z_2, z_3, z_4\}$ – zbiór najbardziej konkurencyjnych firm uczestniczących w przetargu (w tym także oferta firmy przeprowadzającej analizę); $Y = \{y_1, y_2, y_3, y_4\}$ – zbiór kryteriów wykorzystywanych do oceny poszczególnych firm; y_1 – stopień kompatybilności sprzętu i oprogramowania systemu; y_2 – dobra jakość; y_3 – wysoka wydajność; y_4 – niska cena.

x_3	x_6	x_9	x_{12}
x_2	x_5	x_8	x_{11}
x_1	x_4	x_7	x_{10}

Rys. 2. Rynek (Przedsiębiorstwo organizujące przetarg ofertowy) jako zbiór poszczególnych użytkowników grup oprogramowania i sprzętu systemu CAD/CAM

Niech macierz R rozmytej relacji binarnej ma postać przedstawioną w tabeli 1. W takiej macierzy elementy każdego wiersza określają względne stopnie ważności kryteriów przy podejmowaniu decyzji przez kupującego. Ponadto elementy każdej kolumny macierzy S określają ocenę poszczególnych firm konkurencyjnych na podstawie kryteriów y_i (tabela 2). Np. firma z_1 jest charakteryzowana jako dostawca oferujący sprzęt

i oprogramowanie o wysokim stopniu kompatybilności, średnim poziomem jakości i wydajności oraz niskiej cenie, natomiast firma z_2 oferuje sprzęt o wysokiej jakości, wysokiej cenie oraz niskim stopniu kompatybilności i wydajności.

Tabela 1. Macierz wartości określających stopnie ważności kryteriów y_i na podstawie oceny kupującego x_i

x_i \ y_i	y_1	y_2	y_3	y_4
x_1	0,8	0,2	0,0	0,0
x_2	0,0	1,0	0,0	0,0
x_3	0,0	0,0	1,0	0,0
x_4	0,0	0,0	0,0	1,0
x_5	1,0	0,7	1,0	0,7
x_6	0,7	0,3	0,4	0,8
x_7	0,8	0,4	0,5	0,9
x_8	0,6	0,9	0,7	0,1
x_9	0,5	0,9	0,5	0,7
x_{10}	0,6	0,7	0,7	0,5
x_{11}	0,4	0,5	0,3	0,1
x_{12}	0,0	0,0	0,8	0,9

Tabela 2. Oceny poszczególnych firm z_i na podstawie kryteriów y_i

y_i \ z_i	z_1	z_2	z_3	z_4
y_1	0,8	0,4	0,5	0,6
y_2	0,5	0,9	0,4	0,5
y_3	0,4	0,2	0,9	0,4
y_4	0,9	0,3	0,5	0,6

Tabela 3. Macierz wartości określających stopnie priorytetu firmy z_i na podstawie oceny kupującego x_i

x_i \ z_i	z_1	z_2	z_3	z_4
x_1	0,740	0,500	0,480	0,580
x_2	0,500	0,900	0,400	0,500
x_3	0,400	0,200	0,900	0,400
x_4	0,900	0,300	0,500	0,600
x_5	0,641	0,424	0,597	0,521
x_6	0,723	0,395	0,559	0,550
x_7	0,712	0,404	0,562	0,546
x_8	0,565	0,530	0,583	0,500
x_9	0,646	**0,508**	0,542	0,527
x_{10}	0,624	0,464	0,584	0,516
x_{11}	0,600	0,538	0,554	0,515
x_{12}	0,665	0,253	0,688	0,506

Wykorzystując przedstawiony wyżej wzór dla funkcji przynależności $\mu_{A_i}(x, z_i)$ konstruujemy macierz T (tabela 3), a na podstawie przedstawionej w niej informacji otrzymujemy macierz W, której wartości elementów przedstawiono w tabeli 4. Następnie na podstawie macierzy W otrzymujemy:

$$\max_x \min [\mu_{A_1}(x, z_1), \quad \mu_{A_2}(x, z_2)] = 0{,}538$$
$$\max_x \min [\mu_{A_1}(x, z_1), \quad \mu_{A_3}(x, z_3)] = 0{,}665$$
$$\max_x \min [\mu_{A_1}(x, z_1), \quad \mu_{A_4}(x, z_4)] = 0{,}600$$
$$\max_x \min [\mu_{A_2}(x, z_2), \quad \mu_{A_3}(x, z_3)] = 0{,}538$$
$$\max_x \min [\mu_{A_2}(x, z_2), \quad \mu_{A_4}(x, z_4)] = 0{,}515$$
$$\max_x \min [\mu_{A_3}(x, z_3), \quad \mu_{A_4}(x, z_4)] = 0{,}550$$

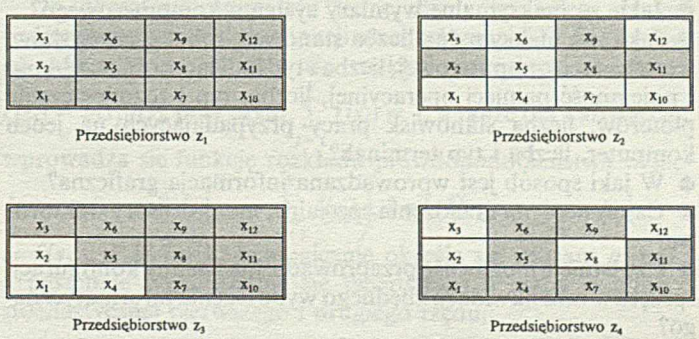
Jak widać z wyżej przedstawionych obliczeń, minimalną wartością $\min_{ij} \max_x = 0{,}515$. Następnie z tablicy T (tab. 3) wybieramy dla l największą możliwą wartość, która jednocześnie jest mniejsza niż 0,515. Otrzymujemy więc $l = 0{,}508$. Przyjmując tę wartość jako próg rozdziału (segmentacji rynku), określamy następujące segmenty rynku dla poszczególnych dostawców:

$$M_1 = \{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}\}$$
$$M_2 = \{x_2, x_8, x_9, x_{11}\}$$
$$M_3 = \{x_3, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}\}$$
$$M_4 = \{x_1, x_4, x_5, x_6, x_7, x_9, x_{10}, x_{11}\}$$

Tabela 4. Macierz wartości określonych na podstawie „operacji Λ" dla poszczególnych par funkcji $\mu_i(x, z_i)$

0,500	0,480	0,580	0,480	0,500	0,480
0,500	0,400	0,500	0,400	0,500	0,400
0,200	0,400	0,400	0,200	0,200	0,400
0,300	0,500	**0,600**	0,300	0,300	0,500
0,424	0,597	0,521	0,424	0,424	0,521
0,395	0,559	0,550	0,395	0,395	**0,550**
0,404	0,562	0,546	0,404	0,404	0,546
0,530	0,565	0,500	0,530	0,500	0,500
0,508	0,542	0,527	0,508	0,508	0,527
0,464	0,584	0,516	0,464	0,464	0,516
0,538	0,554	0,515	**0,538**	**0,515**	0,515
0,253	**0,665**	0,506	0,253	0,253	0,506

Jak widać z rys. 3, w przypadku dostawy systemu przez kilka firm największe szanse, zgodnie z przedstawioną wyżej polityką kupującego (tj. różni dostawcy sprzętu i oprogramowania dla CAD, CAM oraz CAD/CAM) mają: firma z_1 i z_4 na dostawę systemu CAD, firma z_1, z_2 i z_3 na dostawę systemu CAM, oraz firma z_1 i z_3 na dostawę systemu CAD/CAM. Natomiast w przypadku kompleksowej dostawy zintegrowanego systemu CAD/CAM kupujący dałby zapewne priorytet firmie z_1 i z_3.



Rys. 3. Najbardziej preferowane segmenty rynkowe dla poszczególnych dostawców (uczestników przetargu) systemu CAD/CAM rozpatrywanego przykładu

Przeprowadzenie takiej analizy przez firmę-dostawcę pozwala wybrać najbardziej racjonalne segmenty rynkowe dla sprzedaży swojej produkcji oraz zwrócić szczególną uwagę na te właśnie grupy sprzętu i oprogramowania podczas negocjacji handlowych i testowania oferowanych wyrobów.

Przeprowadzanie rozmów handlowych oraz testów technicznych

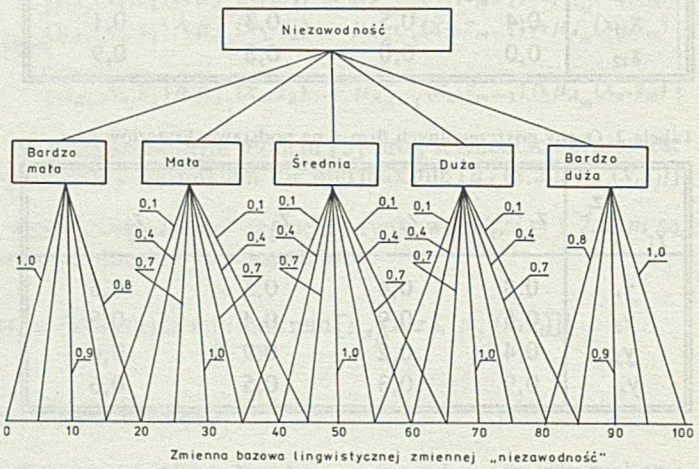
Problemy związane z przeprowadzeniem rozmów handlowych między kupującym i sprzedawcą przedstawiono szerzej w artykule [14], gdzie omówiono wpływ osobowości uczestników rozmów na prawdopodobieństwo zawarcia kontraktu. Tutaj rozpatrzymy dokładniej niektóre problemy związane z dostawą systemu CAD/CAM, a mianowicie te, które zwykle są omawiane między menedżerami dostawców oraz kupującego, korzystającego z usług doradców technicznych.

Przedstawmy teraz niektóre z aspektów, jakie mogą być włączone do rozmów handlowych na temat dostawy systemu CAD/CAM. Problemy te można sformułować w postaci „banku pytań" wspomagających proces analizy [4, 6, 9]:

- W jaki sposób będą przetwarzane i wprowadzane do systemu istniejące zbiory i ewentualnie informacja graficzna?
- Czy system CAD/CAM wykorzystuje standardowy system operacyjny, z którego korzystali dotąd informatycy użytkownika?
- Czy stanowiska pracy spełniają wymagania ergonomiczne?
- Jakie są perspektywiczne plany produkcyjne dostawcy?
- Jakie są możliwości wykorzystania w systemie CAD/CAM oprogramowania pochodzącego od innych dostawców?
- Jakie są możliwości dostawców w zakresie rozwiązywania złożonych problemów oprogramowania?
- Czy dostawca jest w stanie dostarczyć niezbędne pakiety oprogramowania specjalnego?
- Czy istnieje możliwość niezawodnej wymiany danych w systemie pochodzącym od wielu dostawców?
- Jakie są techniczne wymagania instalacyjne systemu komputerowego oraz stanowisk pracy?
- Jakie są doświadczenia ze współpracy z dostawcą?
- Czy istnieje kompatybilność między różnymi rodzajami sprzętu komputerowego i systemami operacyjnymi?
- Czy i jakie unikalne właściwości wykazuje system?
- Czy dostawca zobowiązuje się oddelegować doświadczonych specjalistów do wdrożenia systemu?
- Jaka jest gotowość dostarczenia do magazynu użytkownika (kupującego) najczęściej używanych części zapasowych, niezbędnych do szybkiego usunięcia awarii systemu?
- Czy dostawca przewiduje możliwość wykorzystania szybkiej i bardzo pojemnej pamięci?
- Jaka jest jakość dostarczanej dokumentacji technicznej?
- Jaka jest struktura zbiorów bazy danych?
- Jakie są maksymalne wymiary systemu komputerowego?
- Jaka jest maksymalna liczba stanowisk roboczych, dysków, węzłów sieci komputerowej, liczba i typ linii łączności, szybkość i pojemność pamięci operacyjnej, liczba, typ i rozmieszczenie ploterów, liczba stanowisk pracy przypadających na jeden komputer, liczba i typ terminali?
- W jaki sposób jest wprowadzana informacja graficzna?
- Czy system ma urządzenia specjalne, np. procesory wektorowe?
- Czy istnieje możliwość przeprowadzenia zmiany konfiguracji sprzętu w celu likwidacji zbędnego wyposażenia informatycznego?
- Jaka jest możliwość przeprowadzenia szkolenia personelu użytkownika u dostawcy i jakie są jego koszty?
- Jaki jest poziom usatysfakcjonowania klientów przez dostawcę?
- Jak skomplikowana będzie organizacja interfejsu z informatycznym systemem zarządzania przedsiębiorstwem, jak będzie on realizowany i jakie będą koszty tego przedsięwzięcia?

Bardzo ważnym etapem, związanym z podejmowaniem decyzji dotyczących zakupu systemu jest przeprowadzenie testów technicznych u dostawcy. Testy te, mające na celu określenie poziomu technicznego systemu CAD/CAM, dotyczą sprawdzenia prawidłowości jego funkcjonowania, a także jego wydajności.

Celem testu jest również określenie stopnia wykonalności zdefiniowanych przez użytkownika zadań (funkcji). W wyniku przeprowadzenia przeglądu systemu, modelowania strumieni informacyjnych oraz analizy dokumentacji technicznej kupujący ma możliwość sformułowania własnego poglądu na temat przydatności i jakości systemu. Ocena wyników testów poziomu technicznego jest przeprowadzana na podstawie oceny punktowej, głównie przy użyciu zmiennych lingwistycznych. Stopnie oceny, np. niezawodności systemu, można określić na podstawie struktury hierarchicznej tej zmiennej w sposób pokazany na rys. 4.



Rys. 4. Hierarchiczna struktura zmiennej lingwistycznej przedstawienie wartości funkcji przynależności zmiennej lingwistycznej „niezawodność" dla wartości bardzo mała, ..., bardzo duża, na podstawie zmiennej bazowej w zakresie [0, 100]

Test dotyczący sprawdzania wydajności systemu służy do określenia czasu potrzebnego na wykonanie poszczególnych zadań. W przypadku testowania wydajności istotne jest prawidłowe określenie możliwości systemu komputerowego do wspomagania takiej liczby stanowisk roboczych, jaką planuje się zainstalować dla wykonania poszczególnych zadań funkcjonalnych. Na przykład, określony system CAD może zapewnić satysfakcjonujący użytkownika czas pracy kilku stanowisk roboczych wykonujących proste prace graficzne, natomiast w przypadku stanowisk roboczych wykonywujących złożony proces modelowania geometrycznego może wspomagać tylko 3–4 razy mniejszą ich liczbę.

Dlatego też przy przeprowadzaniu testów dotyczących wydajności systemu należy dokładnie badać i analizować wydajność systemu w zależności od rodzaju procesu przetwarzania informacji i liczby stanowisk roboczych. Jest oczywiste, że duża liczba błędów podczas testowania systemu znacznie obniża ocenę tego systemu na tym etapie procesu podejmowania decyzji.

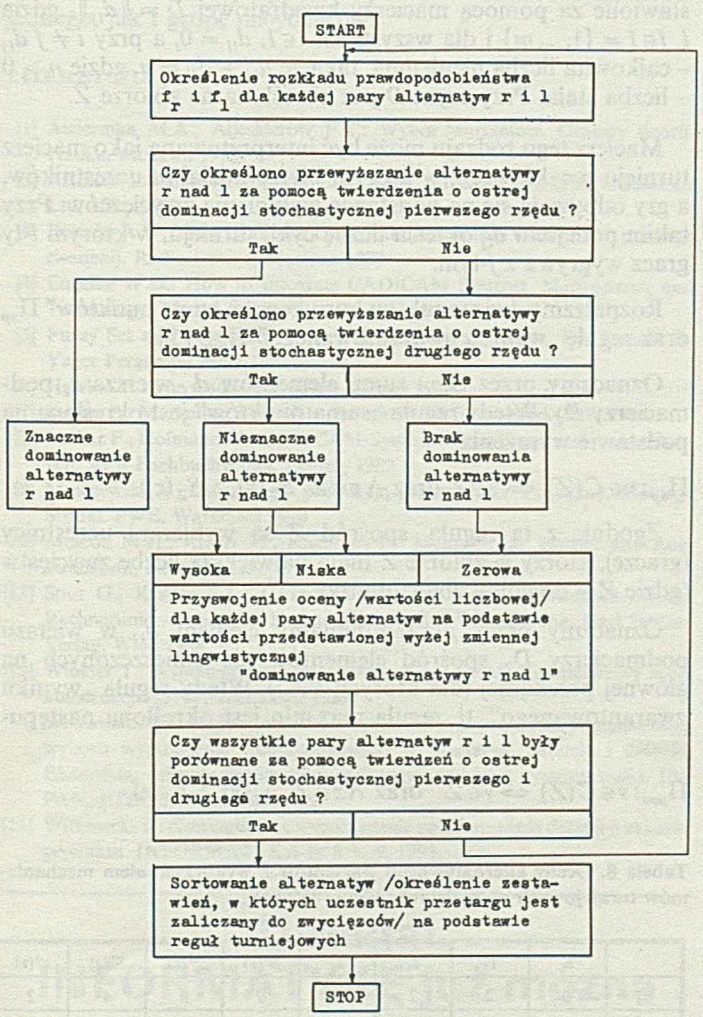
Podczas rozmów handlowych oraz testowania systemu dostawca systemu może uzyskać więcej informacji na temat priorytetów, jakie ma kupujący w stosunku do sprzętu i oprog-

ramowania. Np. użytkownik może przyjąć na siebie wykonanie części zadań ze względu na fakt zatrudnienia wykwalifikowanego i stosunkowo licznego zespołu informatyków. Dlatego w rozmowach szczególną uwagę zwraca na kompatybilność sprzętu i oprogramowania, a także postać informacji wejściowej oraz wyjściowej. Natomiast dostawca, znając w przybliżeniu (a nieraz dokładnie) swoich głównych konkurentów, może wspomagać proces decyzyjny, a na jego podstawie podejmować decyzje dotyczące np. ceny systemu lub jego poszczególnych elementów.

Ocena alternatyw oraz określenie „prawdopodobnego zestawu wyboru" przez dostawcę

Realizując operacje elementarne w procesie podejmowania decyzji, ludzie posługują się zazwyczaj różnymi strategiami. Jeżeli wybór jest dokonywany spośród 6–10 alternatyw decydenci porównują je parami, pozostawiają lepszą i przechodzą do następnej. Natomiast przy większej liczbie alternatyw i kryteriów (około 12) stosują często strategie mieszane.

Rozpatrzmy ocenę alternatyw w warunkach ryzyka i niepewności oraz określenia prawdopodobnego zestawu wyboru przez dostawcę, zgodnie ze schematem przedstawionym na rys. 5.



Rys. 5. Schemat blokowy procedury oceny alternatyw z wykorzystaniem twierdzeń o rozmytej dominacji stochastycznej i opracowania „prawdopodobnego zestawu wyboru" na podstawie reguły turniejowej

Niech na podstawie przeprowadzonych rozmów handlowych oraz testów technicznych systemów CAD/CAM wynika, że efektywność większości zadań realizowanych na tych systemach jest prawie jednakowa. W takim przypadku naturalnym dążeniem kupującego jest wybór takiego wariantu rozwiązania alternatywnego, który wymaga najmniejszych kosztów na przeprowadzenie niezbędnych zmian zgodnie z wymaganiami systemu CAD/CAM. Ponadto, przyjmiemy, że analiza dokumentacji technicznej oraz doświadczenia w zakresie eksploatacji wariantów systemu a_r oraz a_l prowadzi do wniosków, że przy wyborze każdego z nich są możliwe następujące przypadki:

s_1 – jest wymagana istotna zmiana oprogramowania,
s_2 – jest wymagana modyfikacja oprogramowania,
s_3 – jest wymagana zmiana formy wejściowej i wyjściowej informacji,
s_4 – wymagane zmiany są nieznaczne,
s_5 – zmiany nie są wymagane.

Przy założeniu, że ceny ofertowe konkurentów są jednakowe, uczestnik przetargu modelując proces podejmowania decyzji może określić najlepsze oferty poszczególnych firm.

Uporządkowanie preferencyjne dla kupującego przy wyborze poszczególnych firm jest następujące:

$$s_1 < s_2 < s_3 < s_4 < s_5$$

przy czym

$$u(s_1) < u(s_2) < u(s_3) < u(s_4) < u(s_5),$$

gdzie $u(s_i)$ – użyteczność alternatywnych rozwiązań systemu s_i, $i \in 1,5$.

Na początku analizy uczestnik przetargu przeprowadza porównanie swojej oferty z ofertami najgroźniejszych konkurentów.

Udział w przetargu jest celowy w przypadku, gdy jego uczestnik ma szanse wejść do zestawu wyboru formułowanego przez kupującego. W przeciwnym przypadku nie ma on żadnych szans na wygranie przetargu i tylko ponosi koszty przygotowania oferty i rozmów handlowych, a także koszty, np. opracowania i testowania oprogramowania.

O ile etap analizy związany z określeniem segmentów rynkowych miał zadanie jedynie stwierdzenia celowości udziału w przetargu, to etap szczegółowej analizy z wykorzystaniem dokładniej określonych priorytetów kupującego system ma na celu przygotowanie się dostawcy do prowadzenia negocjacji na temat ewentualnego kontraktu.

Do określenia priorytetu jednej firmy względem drugiej wykorzystano model dominacji stochastycznej [3]. Dla każdego wariantu a_f określa się rozkład prawdopodobieństwa $f(s_j) = p(s_j|a_f)$, przy czym $\sum_{j=1}^{k} f(s_j) = 1$, a także dla każdego a_f, wprowadza się funkcję rozkładu prawdopodobieństwa względem preferencji $F_p\left(s_k = \sum_{j=1}^{k} f(s_j)\right)$, $k \in \overline{1,N}$; przy czym $F_p(s_1) = f(s_1)$, $F_p(s_k) = 1$. Analogicznie określa się wariant a_g i G_p, a następnie wykorzystuje się twierdzenia o ostrej dominacji stochastycznej pierwszego i drugiego rzędu,

$$F_p^2\left(s_k = \sum_{j=1}^{k} F_p^1(s_j)\right); \quad k \in 1, N-1 \text{ gdzie } F_p^2(s_i) = \max\{F_p^1(s_j)\}$$

Przykład 2. Rozkłady prawdopodobieństwa dla alternatywnych rozwiązań a_r i a_l są przedstawione w tabeli 5. Obliczamy $F_{p_r}(s_j)$ oraz $F_{p_l}(s_j)$ zgodnie ze wzorem $F_{p_k}(s_k) = \sum_{j=1}^{k} f_k(s_k)$, $k \in 1, N$ i wyniki przedstawiamy w tabeli 6.

Tabela 5. Rozkłady prawdopodobieństw $f_r(s_j)$ i $f_l(s_j)$ dla alternatywnych wariantów a_r i a_l

f_k \ s_j	s_1	s_2	s_3	s_4	s_5
$f_r(s_j)$	0,1	0,3	0,1	0,3	0,2
$f_l(s_j)$	0,2	0,1	0,1	0,5	0,1

Tabela 6. Wartości $F_{p_r}(s_j)$ i $F_{p_l}(s_j)$ dla alternatywnych wariantów a_r i a_l

F_{p_k} \ s_j	s_1	s_2	s_3	s_4	s_5
$F(s_j)$	0,1	0,4	0,5	0,8	1,0
$F(s_j)$	0,2	0,3	0,4	0,9	1,0

Ponieważ $F_{p_r}(s_2) > F_{p_l}(s_2)$ i $F_{p_r}(s_3) > F_{p_l}(s_3)$ oraz $F_{p_r}(s_1) < F_{p_l}(s_1)$ i $F_{p_r}(s_4) < F_{p_l}(s_4)$, to nie jest możliwe ani $a_r >_1 a_l$, ani $a_l >_1 a_r$, co oznacza, że na podstawie tego twierdzenia nie można uporządkować preferencyjnie tych dwóch alternatywnych rozwiązań (gdzie $>_k$ znak dominacji stochastycznej k-go rzędu, alternatywy a_r nad a_l).

Dlatego też decydent w trybie konwersacyjnym dostarcza dodatkową informację o użyteczności poszczególnych alternatyw, a więc powinien uporządkować preferencyjnie następujące wartości przedstawionych poniżej czterech różnic dla każdej różnicy $u(s_{j+1}) - u(s_j)$ tj.: $u(s_2) - u(s_1)$, $u(s_3) - u(s_2)$, $u(s_4) - u(s_3)$ oraz $u(s_5) - u(s_4)$.

W tym celu system konwersacyjny przedstawia decydentowi pytania następującego rodzaju: „określ czy alternatywa pewność (konieczność) modyfikacji oprogramowania (s_2) jest bardziej priorytetowa od wystąpienia alternatywy s_1 oraz s_3 z prawdopodobieństwem 0,5, gdzie jako wynik występuje alternatywa s_1 i s_3 ?".

W przypadku twierdzącej odpowiedzi oznacza to, że

$$u(s_2) - u(s_1) > u(s_3) - u(s_2)$$

Wtedy wartości $F^2_{p_r}(s_j)$ i $F^2_{p_l}(s_j)$ mają następujące wartości (tabela 7). Ponieważ dla wszystkich $j \in 1, 4$, $F^2_{p_l}(s_j) \geqslant F^2_{p_r}(s_j)$, więc $a_r >_2 a_l$ tj. alternatywa a_r nieznacznie dominuje nad a_l.

Tabela 7. Wartości $F^2_{p_r}(s_j)$ i $F^2_{p_l}(s_j)$ dla alternatywnych wariantów a_r i a_l

$F^2_{p_k}$ \ s_j	s_1	s_2	s_3	s_4
$F^2_{p_r}(s_j)$	0,8	0,9	1,3	1,8
$F^2_{p_l}(s_j)$	0,9	1,1	1,4	1,8

Załóżmy, że dostawca określa sytuację, gdy kupujący ogranicza swój zestaw wyboru do kilku (np. trzech) firm. W tym przypadku do określenia szans wygrania w przetargu wykorzystamy regułę turniejową [1], której oprogramowanie [13] pozwala na szybkie określenie zestawień, gdzie firma wchodzi jako zwycięzca. W odróżnieniu od kupującego, który swój wybór przeprowadza na podstawie określonej liczby ofert i ściśle sprecyzowanych kryteriów, dostawca swoją ocenę przeprowadza na podstawie niepewnej liczby ofert, wspomagając się przybliżonymi kryteriami ocen oraz wartościami użyteczności alternatyw.

Otrzymane za pomocą twierdzeń o dominacji stochastycznej (rys. 5) uporządkowanie preferencyjne wykorzystuje się następnie do określenia zestawień przy wykorzystaniu funkcji wyboru (reguły turniejowej), która ma następującą postać logiczną:

$$C^i(\tilde{Z}) = z_i \bigwedge_{j=1}^{n} (\bar{z}_j \vee \lambda_{ij}(\tilde{Z}^{ij})),$$

gdzie: z_i – zmienna boolowska, odpowiadająca uczestnikowi i; $\tilde{Z}$ – boolowski zestaw $(z_i, \ldots z_n)$ opisujący zbiór uczestników przetargu (zbiór wierzchołków grafu zorientowanego, odpowiadających uczestnikom, w których łuk (i, j) przeprowadza się w przypadku preferencji i nad j); $\lambda_{ij}(\tilde{Z})$ – brzegowa funkcja boolowska dla $i, j \in \tilde{Z}$, równa $\lambda_{ij}(\tilde{Z}) = 1 \Leftrightarrow l_i(\tilde{Z}) - l_j(\tilde{Z}) \geqslant 0$, o postaci liniowej $l_{ij}(\tilde{Z}) = l_i(\tilde{Z}) - l_j(\tilde{Z})$. Najlepszym uczestnikiem przetargu jest taki uczestnik i, dla którego wielkość $|R(i)| - R^{-1}(i)|$ (gdzie R – relacja binarna na zbiorze $\tilde{Z}$) osiąga wartość maksymalną.

Pokażmy na prostym przykładzie niektóre problemy związane z określaniem zwycięzcy w przetargu w zależności od liczby uczestników oraz charakteru kryterium oceny.

W mechanizmach turniejowych oceny alternatyw są przedstawione za pomocą macierzy kwadratowej $D = \| d_{ij} \|$, gdzie $i, j \in I = \{1, \ldots, m\}$ i dla wszystkich $i \in I$, $d_{ii} = 0$, a przy $i \neq j$ d_{ij} – całkowita liczba nieujemna, taka że $d_{ij} + d_{ji} = n$, gdzie $n > 0$ – liczba stała. Przy czym D jest określona na zbiorze $\tilde{Z}$.

Macierz tego rodzaju może być interpretowana jako macierz turnieju n cyklowego, w którym bierze udział m uczestników, a gry odbywają się na podstawie wyłonienia zwycięzców. Przy takim podejściu d_{ij} określa liczbę cykli turnieju, w którym i-ty gracz wygrywa z j-tym.

Rozpatrzmy dwie reguły wyboru: regułę „suma punktów" Π_{sp} oraz regułę „wyniku gwarantowanego" Π_{mm} [1].

Oznaczmy przez $S(z_i)$ sumę elementów d_{ij} wiersza z_i podmacierzy D_Z. Wtedy reguła „suma punktów" jest określana na podstawie wyrażenia

$$\Pi_{sp}: v \in C(Z) \Leftrightarrow v \in Z \text{ oraz } \Lambda z \in Z,\ S_Z(v) \geqslant S_Z(z)).$$

Zgodnie z tą regułą, spośród Z są wybierani uczestnicy (gracze), którzy w zbiorze Z mają największą liczbę zwycięstw (gdzie Z – dowolny zbiór graczy).

Oznaczmy przez $\rho(z_i)$ minimalny element d_{ij} w wierszu podmacierzy D_Z spośród elementów nie umieszczonych na głównej przekątnej (dla których $i \neq j$). Wtedy reguła „wyniku gwarantowanego", tj. reguła max min jest określona następująco:

$$\Pi_{mm}: v \in C(Z) \Leftrightarrow v \in Z \quad \text{oraz } \Lambda z \in Z,\ \rho_Z(v) \geqslant \rho_Z(z).$$

Tabela 8. Oceny alternatywnych wariantów z wykorzystaniem mechanizmów turniejowych na podstawie reguły Π_{sp} i Π_{mm}

	z_1	z_2	z_3	z_4	$S^4(z)$	$\varsigma^4(z)$	$S^3(z)$	$\varsigma^3(z)$
z_1	0	2	2	5	9	2	4	2
z_2	4	0	3	1	8	1	7	3
z_3	4	3	0	2	9	2	7	3
z_4	1	5	4	0	10	1	-	-

W tabeli 8 przedstawiono przykład macierzy (tablicy) turniejowej, w którym bierze udział czterech oraz trzech uczestników z_1, z_2, z_3 i z_4. Na podstawie reguły Π_{sp} zwycięzcą jest uczestnik z_4, a na podstawie reguły Π_{mm} – uczestnicy z_1 i z_3. W przypadku tablicy turniejowej zmniejszonej do trzech uczestników z_1, z_2 i z_3, zwycięzcą zarówno na podstawie reguły Π_{sp}, jak i na podstawie Π_{mm} są uczestnicy z_2 i z_3.

Wszystkie wyżej przedstawione modele wspomagania decyzji zostały zaprogramowane w języku FORTRAN oraz C [13] i uruchomione na komputerze IBM PC XT/AT.

Przedstawiony wyżej oraz w [14] system wspomagania jest przeznaczony do podejmowania decyzji w sytuacjach, gdy należy wybrać jedną lub kilka możliwych wariantów (alternatyw), charakteryzujących się szeregiem ilościowych i jakościowych atrybutów.

System składa się z zestawu metod niezbędnych do analizy wielokryterialnej, zarówno na etapie oceny wstępnej (mniej formalnej), jak i oceny końcowej (z większym stopniem formalizacji procesu wyboru) w warunkach pewności oraz przy niepełnej i niepewnej informacji.

System jest łatwym i wygodnym środkiem, przy wykorzystaniu którego można sformułować zadanie, określić zbiór alternatyw oraz charakteryzujących je atrybutów wraz z podaniem ich wartości, a także ocenić i preferencyjnie uporządkować alternatywy, zarówno w przypadku podejmowania decyzji przez jednego, jak i grupę decydentów.

LITERATURA

[1] Aizierman M.A., Aljeskjerow F.T.: Wybor wariantow. Osnowy tieorii. Nauka, Moskwa 1990

[2] Artificial Intelligence. Implication for CIM. Ed. Kusiak A. IFS Publications Ltd, UK 1988

[3] Borisow A.W. (red.): Obrabotka niecziotkoj informacji w sistiemach priniatja rieszienij. Radio i swiaz, Moskwa 1989

[4] Engelke W.D.: How to Integrate CAD/CAM Systems. Management and Technology. Marcel Dekker Inc., New York 1987

[5] Fuzzy Set and Possibility Theory. Recent Developments. Ed. Ronald R. Yager Pergamon Press 1982

[6] Hawkes B.: The Cadcam Process. Pitman 1988

[7] Kotler P.: Marketing Essentials. Prentice Hall Englewood Cliffs 1984

[8] Lothar F., Hofmann M.: CAD/CAM-Systeme. Grundlagen und Anwendungen. VEB Fachbuchverlag, Leipzig 1989

[9] Matczewski A.: Zarządzanie produkcją przemysłową. Problemy. Metody. Środki. PWE, Warszawa 1990

[10] Mescon M.H., Albert M., Khedouri F.: Management. Harper and Row Publishers, New York 1988

[11] Spur G., Krause F.L.: CAD-Technik, Lehr – und Arbeitsbuch für die Rechnerunterstützung in Konstruktion und Arbeitsplanung. Carl Hanser Verlag, Wien 1984

[12] Winkler T.: Wspomaganie komputerowe CAD/CAM. Komputerowy zapis konstrukcji. WNT, Warszawa 1989

[13] Witkowski T.: Wielokryterialna, wieloetapowa procedura optymalnego wyboru wyposażenia technologicznego w przetargu. Modele i decyzje. Ekonomia, ochrona środowiska, medycyna, technika, systemy walki. IBS PAN, PTBOiS, Warszawa 1991

[14] Witkowski T.: Komputerowe wspomaganie podejmowania decyzji o zakupie produktu. INFORMATYKA nr 8, s. 9, 1993.

Wskazówki dla Autorów

Nadsyłane artykuły nie mogą być publikowane lub przeznaczone do opublikowania w innych czasopismach.

Materiał oprócz tekstu zasadniczego powinien zawierać:

- ★ *krótki życiorys zawodowy Autora i jego zdjęcie,*
- ★ *wykaz literatury,*
- ★ *tabele,*
- ★ *materiał ilustracyjny (rysunki, zdjęcia czarno-białe, wydruki) dołączony do artykułu (nie wklejać materiału w tekst),*
- ★ *podpisy pod ilustracje.*

Na osobnej stronie prosimy podać tytuł naukowy, imię i nazwisko, nazwę zakładu pracy, adres domowy (koniecznie!), numer telefonu oraz informację, jaką drogą przekazać honorarium – kasa Wydawnictwa, poczta, bank.

Tekst artykułu prosimy dostarczyć w formie maszynopisu lub wydruku komputerowego (pisany jednostronnie, z podwójnym odstępem – bardzo ważne! – czyli 30 wierszy na stronie i 60 znaków w wierszu).

Wykaz literatury w porządku alfabetycznym.

Tabele – każda na osobnej stronie – powinny być ponumerowane, opatrzone tytułem oraz ściśle związane z tekstem (zaznaczone miejsce tabeli w tekście).

Rysunki – winny być czytelne (zaznaczyć ich miejsce w tekście), mogą być wykonane ołówkiem.

Zdjęcia – czarno-białe, kontrastowe, na błyszczącym papierze.

Wydruki – czytelne, kontrastowe, wykonane na białym papierze. Format – maksymalnie 18 cm w podstawie.

Każde pierwsze słowo opisu rysunków powinno być pisane dużą literą.

Podpisy pod ilustracje powinny być napisane na oddzielnej stronie, oprócz kolejnego numeru powinny zawierać tytuł (rysunku, zdjęcia, wydruku) i ew. legendę dotyczącą poszczególnych elementów.

Autor opublikowanego w INFORMATYCE artykułu otrzymuje honorarium i bezpłatny egzemplarz okazowy.

Materiałów nie zamówionych redakcja nie zwraca.

Przedruk z czasopisma „Machine GRAPHICS & VISION."
Vol.1, No. 1–2, 1992.
Wydawca: Instytut Podstaw Informatyki PAN

JAROSŁAW PROKOP
Instytut Informatyki
Politechnika Warszawska

System CHARLIE II modelowanie i animacja sylwetek ludzkich

Modele sylwetek ludzkich są używane w czasie projektowania obiektów i urządzeń używanych przez człowieka i umożliwiają tanią weryfikację takich projektów. Są też używane do produkcji filmów oraz w zastosowaniach związanych z medycyną [2, 3, 4, 5]. Modelowanie sylwetek ludzkich składa się z:

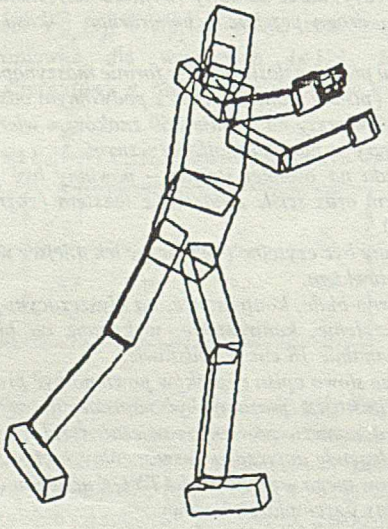
- modelowania ciała (modele: szkieletowe, powierzchniowe, objętościowe),
- specyfikacji typów ruchu (ruch w otoczeniu, ruch elementów modelu, ruchy twarzy: mimika i synchronizacja z wypowiadanymi fonemami),
- modelowania ruchu (modele kinematyczne lub dynamiczne),
- specyfikacji parametrów ruchu (kadry wiodące, algorytmy, poziom logiczny).

Podstawowe cechy systemu Charlie II

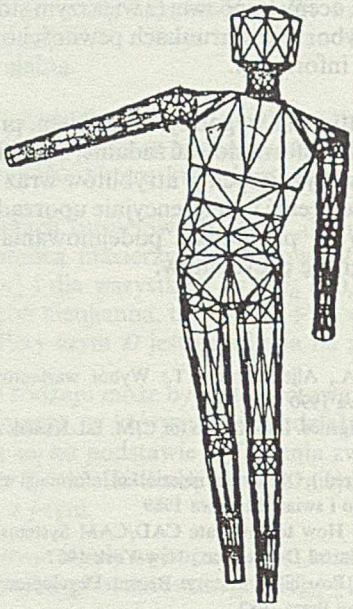
System Charlie II pozwala modelować i wyświetlać sylwetki ludzkie. Jest to modyfikacja programu Recovman, napisanego przez C. Attwooda w Intelligent Systems Group na Wydziale Informatyki Uniwersytetu w Reading.

Program Charlie II został napisany przez autora na Wydziale Informatyki Uniwersytetu w Reading (Wielka Brytania). Pobyt w Reading był sponsorowany przez program TEMPUS.

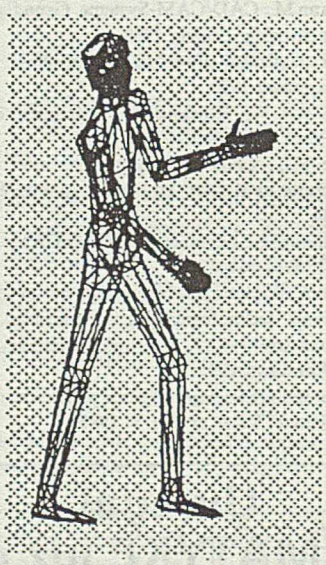
Oryginalny program jest przeznaczony do rozpoznawania pozycji człowieka w trzech wymiarach na podstawie dwuwymiarowych danych (np. jednego rzutu – zdjęcia) [1]. Program ten zawierał moduł generowania sylwetki w trzech wymiarach i wyświetlania jej na ekranie. Model składał się z 16 walców, które przyporządkowane głównym segmentom ciała uwidoczniały przyjętą pozę. Podstawowym zadaniem programu było rozpoznanie tej pozy i jej klasyfikacja (np. klęczy, stoi, siedzi). Rysunek 1 przedstawia model używany w programie Recovman (ściany walców podzielone na cztery prostokąty) z dodanymi segmentami palców prawej ręki.



Rys. 1. Model szkieletowy używany w programie Recovman



Rys. 2. Płynne łączenie segmentów



Rys. 3. Kompletny model używany w programie Charlie II

W systemie Charlie II wprowadzono następujące nowe elementy:

- rozbudowę modelu sylwetki o nowe segmenty (dłonie, palce, twarz),
- płynną interpolację między segmentami,
- uogólnienie metody wyświetlania w celu umożliwienia opisu skomplikowanych powierzchni sylwetki,
- stworzenie modułu animacji modelu i generacji sekwencji obrazów.

Rysunki 1, 2 i 3 ilustrują fazy rozwoju systemu Charlie II.

Interpolacja między segmentami

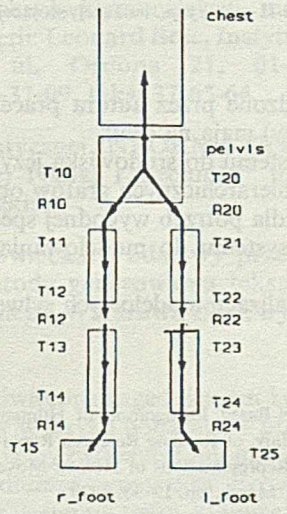
Standardowe programy modelujące nie pozwalają na łatwą modyfikację powierzchni sylwetek (np. przy zmianie długości kończyn modelu). Aby uniknąć tego problemu zdecydowano się podzielić segmenty ciała na dwa typy:

- walcowe (np. uda, ramiona, palce), modelowane uogólnionymi walcami (podstawami uogólnionych walców są elipsy),
- inne (np. korpus, miednica, dłoń); kształt takich segmentów jest zróżnicowany; w systemie Charlie II można opisać powierzchnie takich segmentów jednym z dwóch sposobów: albo zdefiniować kilka elips z wyróżnionymi punktami i podać sposób rozpięcia trójkątów na tych elipsach albo załadować listę trójkątów zdefiniowaną poza systemem (ten sposób zastosowano do uzyskania modelu powierzchni twarzy).

Wszystkie segmenty na brzegach stawów kończą się elipsami. Pozwala to wypełnić przestrzeń między segmentami dwoma uogólnionymi walcami sklejonymi ze sobą podstawami. Wspólna podstawa jest dwusieczną między płaszczyznami wyznaczonymi przez elipsy sąsiadujących segmentów. Druga podstawa każdego ze stawowych walców jest „doklejona" do odpowiedniego segmentu. Prawidłowe położenie walców stawowych musi zapewnić funkcja pozycjonująca wspólną podstawę w przestrzeni między segmentami. Ostateczną reprezentacją powierzchni segmentów i stawów jest lista trójkątów. Pozwala to na wygodną wizualizację powierzchni.

Specyfikacja ruchu w otoczeniu

Rysunek 4 przedstawia organizację grafu opisującego połączenia między stawami i segmentami dla pozycji spoczynkowej.



Rys. 4. Translacje w pozycji spoczynkowej

Segmentem głównym jest miednica. Składanie przekształceń lokalnych układów segmentów do układu globalnego odbywa się wzdłuż strzałek. Przekształcenie punktów z układu lokalnego stopy do układu globalnego polega na złożeniu odpowiednich macierzy lokalnych. Macierze T są macierzami translacji łączących środki lokalnych układów współrzędnych przylegających do siebie segmentów i stawów. Macierze R są złożeniem lokalnych obrotów w stawach. W pozycji spoczynkowej kąty obrotów są równe zero, można więc pominąć macierze R:

l_foot: $T25\ T24\ T23\ T22\ T21\ T20$
r_foot: $T15\ T14\ T13\ T12\ T11\ T10$

Miednica jest związana z otoczeniem macierzą *ENV* i dopóki *ENV* jest stałą, to ciało nie poruszy się w otoczeniu (tzn. względem układu związanego z otoczeniem): zmiana lokalnych kątów obrotu stawów nóg spowoduje, że manekin będzie machał nogami „w powietrzu". Jeżeli zapewni się możliwość przeorganizowania grafu opisującego połączenia segmentów i stawów, to możliwe będzie definiowanie złożonych ruchów w środowisku bez operacji na macierzy ENV. Zmiana organizacji grafu polega na zmianie kierunku składania przekształceń i wybieraniu nowego segmentu głównego bez utraty informacji o lokalnych macierzach przekształcenia globalnego.

Sposób specyfikacji ruchu zostanie zilustrowany następującym przykładem składania przekształceń i organizacji grafu dla jednego kroku modelu – wykrok lewą nogą i dostawienie prawej (operacja *inv*( ) odwraca macierz):

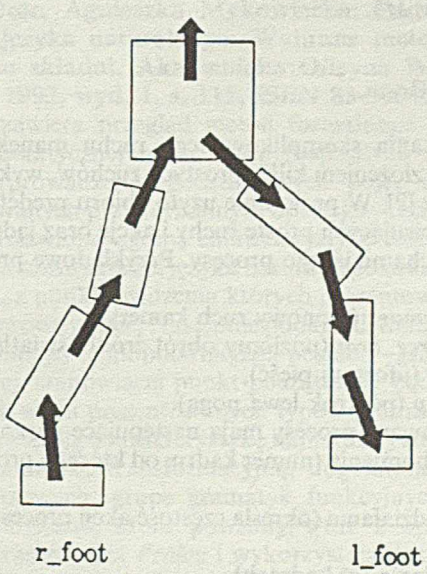
1. Segmentem głównym jest miednica; chwilowe macierze globalne:

l_foot: $M1 \leftarrow T25\ R24\ T24\ T23\ R22\ T22\ T21\ R20\ T20$
r_foot: $M2 \leftarrow T15\ R14\ T14\ T13\ R12\ T12\ T11\ R10\ T10$

2. Zapamiętanie $S2 \leftarrow M2$ w segmencie prawej stopy:
Pierwsza faza kroku – prawa stopa stoi na ziemi, lewa w rosnącym wykroku:

3. Przekształcenie grafu do postaci jak na rysunku 5; chwilowe macierze globalne:

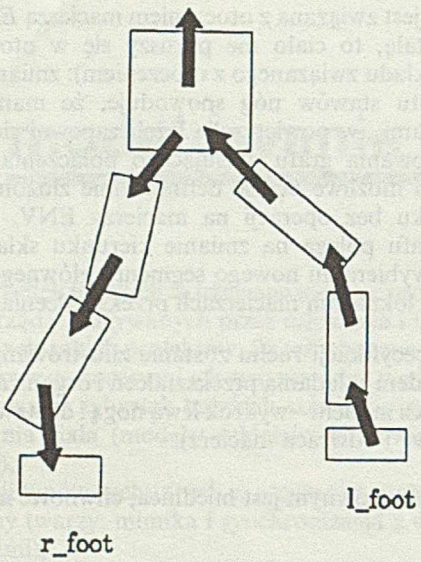
r_foot: $M4 \leftarrow S2 = const$
l_foot: $M3 \leftarrow T25\ R24\ T24\ T23\ R22\ T22\ T21\ R20\ T20$
$inv(T10)\ inv(R10)\ inv(T11)\ inv(T12)\ inv(R12)\ inv(T13)$
$inv(T14)\ inv(R14)\ inv(T15)\ M4$



Rys. 5. Pierwsza faza kroku

4. Zmiana lokalnych kątów w ustalonych kwantach od min do max.

Druga faza kroku – lewa stopa stoi na ziemi, prawa zmienia położenie:
5. Zapamiętanie ostatniej chwilowej macierzy lewej stopy z fazy pierwszej: $S3 \leftarrow M3$.
6. Zmiana organizacji grafu do postaci jak na rysunku 6.



Rys. 6. Druga faza kroku

7. Nadawanie lokalnym kątom obrotu wartości od max do min (ujemne kwanty); chwilowe macierze globalne:

l–foot: $M5 \leftarrow S3 = const$
r_foot: $M6 \leftarrow T15\ R14\ T14\ T13\ R12\ T12\ T11\ R10\ T10$ $inv(T20)\ inv(R20)\ inv(T21)\ inv(T22)\ inv(R22)\ inv(T23)$ $inv(T24)\ inv(R24)\ inv(T25)\ M5$

Każda chwilowa macierz globalna dla lewej stopy zawiera informację o:
- kierunkach w grafie w danej chwili (macierze *T*),
- lokalnych kątach obrotu,
- historii wszystkich poprzednich ruchów w środowisku, zakumulowaną w *M5*.

Animacja

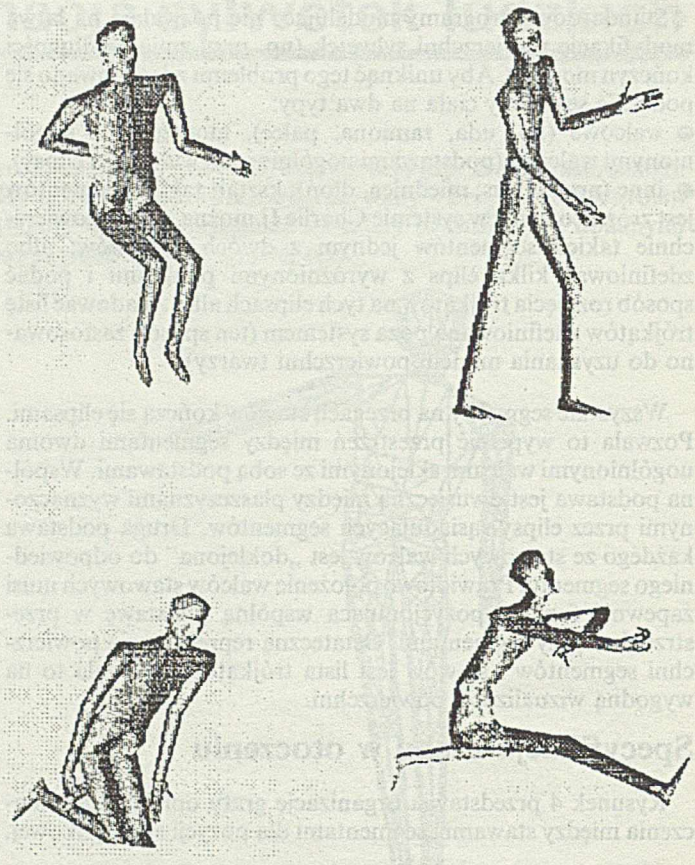
Do uzyskania skomplikowanego ruchu manekina można posłużyć się złożeniem kilku prostych ruchów, wykonywanych jednocześnie [2]. W programie użyto zbioru predefiniowanych procesów opisujących proste ruchy i akcje oraz jądra animacji, kolejno uruchamiającego procesy. Przykładowe predefiniowane procesy:
- *camera–vmove* (pionowy ruch kamery),
- *light–source_hrot* (poziomy obrót źródła światła),
- *form–fist* (uformuj pięść),
- *l–half-step* (półkrok lewą nogą).

Predefiniowane procesy mają następujące parametry:
- czas uruchomienia (numer kadru, od którego proces zaczyna działać),
- szybkość działania (określa częstość akcji procesu w akcjach /kadry),
- czas działania (w kadrach),
- dodatkowe parametry (np. długość kroku, zakres zmienności, kąta itp.).

Ponieważ najwięcej czasu zajmuje odwzorowanie na ekranie, to istnienie takich procesów jak *camera–on* i *camera–off* pozwala dzielić pracę nad zdefiniowaną sekwencją między kilka komputerów.

* * *

System modelowania sylwetki ludzkiej i jej ruchu, Charlie II, został napisany w języku POP-11, dostępnym na komputerach Sun. Zrealizowano 25 s (ok. 300 kadrów) filmu animowanego, pokazującego proste ruchy manekina, i zapisano ten film na taśmie wideo. Kilka przykładowych sylwetek narysowanych z eliminacją powierzchni niewidocznych i z zastosowaniem cieniowania Gourauda pokazano na rysunku 7.



Rys. 7. Model Charlie II w pozycji siedzącej, stojącej, klęczącej i przypadkowej

Obecnie prowadzone przez autora prace (stanowiące jego projekt dyplomowy) mają na celu:
- przeniesienie systemu do środowiska języka C++,
- implementację hierarchicznych grafów opisu struktury segmentów i stawów dla potrzeb wygodnej specyfikacji ruchu,
- wykorzystanie systemu do modelowania innych form niż ludzkie,
- ulepszenie wizualizacji modelowych sylwetek.

LITERATURA

[1] Attwood C.: Model-Based Recognition of Human Posture Using Single Synthetic Images. Univ. of Reading Research Report, 1987
[2] Badler N.: Digital Representations of Human Movement. Computing Surveys, Vol. 11, No 1, March, pp. 19–38, 1979
[3] Badler N.: A Spherical Representation of a Human Body for Visualizing Movement. Proceedings of the IEEE, Vol. 67, No. 10, October, pp. 1396–1403, 1979
[4] Badler N.: Multi-Dimensional Input Techniques and Articulated Figure Positioning by Multiple Constraints. Univ. of N.. Carolina Workshop on Interactive 3D Graphics, 1986
[5] Dooley M.: Antropometric Modeling Programs – A Survey. IEEE Computer Graphics and Applications, November, pp. 17–25, 1982
[6] O'Rourke J.: Model Based Image Analysis of Human Motion Using Constraint Propagation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 6, November, pp. 522–536, 1980

AKADEMICKA OFICYNA WYDAWNICZA RM

Akademicka Oficyna Wydawnicza (ul. Mokotowska 58, 00-534 Warszawa, tel. faks 29-80-21) rozpoczęła swoją działalność w połowie ubiegłego roku.

W pierwszej serii tej Oficyny „Problemy Współczesnej Nauki i Techniki" będą ukazywały się systematycznie głównie małe monografie prezentujące aktualny stan wiedzy w wybranych dziedzinach nauki, w tym również tłumaczenia wybitnych autorów zagranicznych.

Podstawowym celem Oficyny będzie jednak szybkie i tanie wydawanie prac autorów polskich, przyczyniając się do rozpowszechniania ich osiągnięć badawczych w kraju.

Szczegółowy program wydawniczy serii, podzielony na grupy tematyczne, będzie opracowywany we współpracy ze znanymi naukowcami z kraju i zagranicy.

Dotychczas we wspomnianej serii były wydawane książki z grupy tematycznej INFORMATYKA, które również w przyszłości będą stanowić większość wydawanych tytułów serii. Szczególną uwagę zwróci się tu na tytuły z zakresu sztucznej inteligencji.

Publikacje serii adresowane są nie tylko do polskich, lecz również zagranicznych środowisk naukowych, a także do wszystkich tych, którzy są zainteresowani pogłębianiem swojej wiedzy oraz rozwinięciem własnych zainteresowań.

Oficyna zaprasza wszystkich zainteresowanych tą nową inicjatywą wydawniczą do współpracy. Manuskrypty – przygotowane zgodnie ze wskazówkami wydawcy – można przysyłać pod adresem edytora serii: **Leonard Bolc, Instytut Podstaw Informatyki PAN, ul. Ordona 21, 01-237 Warszawa, tel.: 36-28-41, 36-37-09, faks: 37-65-64.**

W grupie tematycznej INFORMATYKA serii „Problemy Współczesnej Nauki i Techniki" ukazały się dotąd następujące książki:

Agnieszka Mykowiecka: Podstawy przetwarzania języka naturalnego. Metody generowania tekstów. Akademicka Oficyna Wydawnicza, Warszawa 1992, wyd. 1, s. 134, ISBN 83-900451-2-5

Książka jest poświęcona zagadnieniom komputerowego generowania tekstów w języku polskim. W pracy przedstawiono najważniejsze problemy związane z automatyczną budową zdań i łączeniem ich w dłuższe wypowiedzi. Rozważane są zarówno kwestie związane z wyborem treści, które należy przekazać, jak i sposobu doboru słów i konstrukcji składniowych odpowiednich dla ich wyrażenia. W pracy scharakteryzowano wybrane systemy, w których zaimplementowano moduły generujące pojedyncze zdania bądź teksty. Następnie opisano rodzaj danych niezbędnych do realizacji tego zadania oraz metody ich reprezentowania. Proces generowania wypowiedzi został przedstawiony w podziale na trzy główne nurty zagadnień związanych odpowiednio z planowaniem treści tekstu, ustalaniem jego struktury oraz nadaniem mu ostatecznej formy.

Książka jest przeznaczona dla wszystkich zainteresowanych komputerowym przetwarzaniem języka naturalnego zarówno z punktu widzenia sztucznej inteligencji, jak i lingwistyki komputerowej. Może też stanowić wprowadzenie do zagadnienia automatycznego generowania wypowiedzi dla studentów kierunków informatycznych.

Elżbieta Dobryjanowicz: Podstawy przetwarzania języka naturalnego. Wybrane metody analizy składniowej. Akademicka Oficyna Wydawnicza, Warszawa 1992, wyd. 1, s. 163, ISBN 83-900451-8-4

Książka dotyczy zagadnienia komputerowej analizy składniowej zdań i fraz języka naturalnego. Stanowi przegląd istniejących metod budowania rozbiorów syntaktycznych wypowiedzi w języku naturalnym, którego struktury są opisane za pomocą reguł pewnej gramatyki formalnej. Pracę rozpoczyna wprowadzenie do metod formalnego opisu języków naturalnych oraz opis sposobów tworzenia analizatorów składniowych wykorzystujących gramatyki bezkontekstowe. Następnie przedstawiono najczęściej stosowane metody budowy rozbiorów syntaktycznych zdań języka naturalnego. W pracy opisano też analizator opracowany w Instytucie Podstaw Informatyki PAN dla wybranego podzbioru języka polskiego. Ostatni rozdział zawiera podsumowanie omówionych metod analizy syntaktycznej i porównanie ich pod pewnymi względami istotnymi z obliczeniowego punktu widzenia.

Książka jest przeznaczona dla wszystkich zainteresowanych komputerowym przetwarzaniem języka naturalnego zarówno z punktu widzenia sztucznej inteligencji, jak i lingwistyki komputerowej.

Leonard Bolc, Agnieszka Mykowiecka: Podstawy przetwarzania języka naturalnego. Wybrane metody formalnego zapisu składni. Akademicka Oficyna Wydawnicza, Warszawa 1992, wyd. 1, s. 115, ISBN 83-900451-9-2

Książka zawiera przegląd metod formalnego opisu zasad budowy syntaktycznej zdań. Pracę rozpoczyna rozdział poświęcony wprowadzeniu podstawowych pojęć matematycznych, stosowanych przy formalnym opisie języków. W kolejnej części przedstawiono ogólną charakterystykę różnych podejść do problemu opisu składni języka naturalnego oraz najważniejsze kryteria, z punktu widzenia których prezentowane metody mogą być porównywane. Następnie podano definicję gramatyki formalnej oraz jej najpopularniejszej postaci – gramatyki bezkontekstowej, stanowiącej punkt początkowy bądź układ odniesienia dla wielu prezentowanych typów gramatyk. Kolejne rozdziały zawierają charakterystykę najbardziej znanych teorii. Opisano gramatykę transformacyjną, uogólnioną gramatykę struktur frazowych, grupę gramatyk funkcyjnych oraz tzw. gramatyki logiczne, które powstały wraz z opracowaniem języka programowania *Prolog* i wykorzystują dostępne w nim możliwości.

Książka jest przeznaczona dla wszystkich zainteresowanych komputerowym przetwarzaniem języka naturalnego, może też stanowić wprowadzenie do problemu formalnego opisu składni języka naturalnego dla lingwistów.

Dokończenie w następnym numerze

W PRZEDPŁACIE DUŻO TANIEJ!

Miło nam poinformować, że już ukazało się na rynku drugie wydanie

Słownika skrótów angielskich stosowanych w elektronice, informatyce i telekomunikacji

najobszerniejsze, zawierające około 100 000 haseł, nowoczesne i bardzo poszukiwane, o czym świadczy fakt, że pierwsze wydanie rozeszło się błyskawicznie!
Słownik można otrzymać pocztą pod wskazany adres, wpłacając preferencyjną, taką samą jak przy pierwszym wydaniu (pomimo inflacji!), dużo niższą niż w księgarniach kwotę 50 000 zł na konto:

PBK S.A. III/O Warszawa nr 370015-1573-139-11
Wydawnictwo SIGMA-NOT Sp. z o.o., Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004

Zamówienie drugiej edycji słownika w przedpłacie jest gwarancją otrzymania go po cenie ulgowej, bez wysokiej marży sieci handlowej i kosztów wysyłki.

Kasprzyk A.: Obiektowo-zorientowane analiza i projektowanie (1) INFORMATYKA 1993, nr 9, s. 1 Pierwsza część charakterystyki amerykańskiej obiektowo-zorientowanej metodologii tworzenia systemów informatycznych OMT (*Object Modelling Technique*), obejmująca omówienie jej pierwszego etapu – analizy oraz tworzenia modeli obiektów.	**Kasprzyk A.: Object-oriented analysis and design (1)** INFORMATYKA 1993, No. 9, p. 1 First part of characteristics of the OMT (*Object Modelling Technique*), an american object-oriented methodology for data processing system building, which includes discussion of its first phase – analysis and of object models building.	**Kasprzyk A.: Objektorientierte Analyse und Projektierung (1)** INFORMATYKA 1993, Nr. 9, S. 1 Erster Teil einer Charakteristik von der amerikanischen OMT (Object Modelling Technique)--Methodologie, einer der objektorientierten Methodologien für EDV-Systemebau, der eine Besprechung ihrer ersten Etappe – der Analyse, sowie des Baues von Objektenmodellen, umfasst.
Goliński J.: Informatyka gospodarcza INFORMATYKA 1993, nr 9, s. 10 Charakterystyka informatyki gospodarczej jako nowego kierunku studiów wyższych w Polsce.	**Goliński J.: Business informatics** INFORMATYKA 1993, No. 9, p. 10 Characteristics of problems connected with business informatics as a new discipline of university studies in Poland.	**Goliński J.: Wirtschaftsinformatik** INFORMATYKA 1993, Nr. 9, S. 10 Eine Charakteristik von Problemen, die mit Wirtschaftsinformatik als neuer Hochschulstudienrichtung in Polen verbunden sind.
Wierzbicki M.: Tworzenie obiektów przez dziedziczenie mono- i polimorficzne INFORMATYKA 1993, nr 9, s. 12 Omówienie podstaw i możliwości programowania obiektowego przy użyciu składni oraz nazewnictwa języka Turbo Pascal, z podaniem różnic w przypadku użycia języka C++.	**Wierzbicki M.: Object building through mono- and polymorfic succession** INFORMATYKA 1993, No. 9, p. 12 Discussion of object programming foundations and possibilities using Turbo Pascal syntax and terminology, as well as showing differences in case of C++ use.	**Wierzbicki M.: Bau von Objekten durch mono- und polymorfisches Ererben** INFORMATYKA 1993, Nr. 9, S. 12 Eine Besprechung von Grundlagen und Möglichkeiten der Objektprogrammierung mit Verwendung der Turbo Pascal-Syntax und Terminologie, sowie mit Angabe von Differenzen im Falle der Verwendung von C++ -Sprache.
Witkowski T.: Wspomaganie decyzji o wyborze zintegrowanego systemu CAD/CAM INFORMATYKA 1993, nr 9, s. 16 Prezentacja modeli komputerowego wspomagania decyzji w warunkach ryzyka i niepewności oraz charakterystyka problemów związanych z oceną najlepszego wariantu zintegrowanego systemu CAD/CAM.	**Witkowski T.: Computer assisted decision making for integrated CAD/CAM system** INFORMATYKA 1993, No. 9, p. 16 Presentation of computer assisted decision making under conditions of risk and uncertainty, as well as characteristics of problems connected with estimation of the CAD/CAM integrated system best variant.	**Witkowski T.: Entschlussunterstützung bei der Wahl eines integrierten CAD/CAM-Systems** INFORMATYKA 1993, Nr. 9, S. 16 Eine Präsentation von computerunterstützten Entschlussfassung unter Bedingungen der Risiko und Unsicherheit, sowie eine Charakteristik von Problemen einer Bewertung des besten Variantes von einem integrierten CAD/CAM-System.
Prokop J.: Modelowanie i animacja sylwetek ludzkich – system CHARLIE II INFORMATYKA 1993, nr 9, s. 24 Charakterystyka rozwiązań pakietu graficznego CHARLIE II do modelowania i animacji sylwetek ludzkich.	**Prokop J.: Modelling and estimation of human shapes – the CHARLIE II system** INFORMATYKA 1993, No. 9, p. 24 Characteristics of the CHARLIE II graphic package for human shape modelling and animation.	**Prokop J.: Modellierung und Animation von menschlichen Silhouetten – das CHARLIE II-System** INFORMATYKA 1993, Nr. 9, S. 24 Eine Charakteristik des CHARLIE II-Graphikprogrammes für Modellierung und Animation von menschlichen Silhouetten.

POLSKIE LITERY + NOWE MOŻLIWOŚCI

AKREM (90-950 Łódź 1, box 308, tel. 36-48-74) oferuje pakiety programowe:

POLY do CLIPPERA 87/5

- indeksowanie wg polskiego alfabetu w dowolnym standardzie,
- wprowadzanie, kontrola i manipulacje na danych o polskojęzycznym obrazie (GET-y, funkcje UPPER, ISALPHA itd.),
- praktycznie wszystkie znane metody programowego definiowania i symulacji polskich liter:
  - symulacja na Herkulesie w trybie znakowym i graficznym,
  - nie znikające polskie litery na EGA, VGA, SVGA we wszystkich trybach (w tym tryby tekstowe 25/28/43/50 wierszy, tryby graficzne, 512 fontów w trybie tekstowym),
  - polskie klawiatury (Alt i/lub dowolnie wybrany "klawisz polskiej litery", możliwość zmiany sposobu wprowadzania "w locie", liczne dodatkowe udogodnienia, jak np. automatyczna zmiana wielkości już wprowadzonych liter, "inteligentny "CapsLock, itd.),
  - na drukarce trzy metody , w tym nowość: nie znikający "download" (automatycznie samodefiniujący się po każdym włączeniu drukarki, działa bez żadnych przeróbek na każdej drukarce emulującej standard Epson lub IBM) oraz druk w trybie graficznym i najprostsza metoda "nakładania",
- dynamiczne dostosowanie programu do dowolnego (nawet własnego) standardu polskich liter,
- komunikacja programów w CLIPPERZE z rezydentami pakietu,
- bezbłędna współpraca z dowolnymi wersjami DOS-u,
- a ponadto system tablica/dysk, tablice wielowymiarowe o zwiększonej pojemności, szybkie operacje na tablicach, system pakowania danych (40-50% oszczędności pamięci dyskowej bez straty szybkości i miejsca w pamięci operacyjnej), opcja szyfrowania danych (przydatna dla użytkowników pracujących w sieci), menu wielopoziomowe, dwuwymiarowe i logiczne (wybór wielu opcji jednocześnie), manipulowanie atrybutami i kolorami ekranu, wyświetlanie dużych tekstów w okienkach, uniwersalne podejście okienkowe, polski kalendarz i liczby słownie, możliwość indeksowania i wyszukiwania danych wg polskiej wymowy, opcja definiowania własnych szablonów formatu, "trójwymiarowe" rozszerzenie funkcji DBEDIT, możliwość pracy z ekranami o zwiększonej liczbie wierszy i kolumn, klawisze F11 i F12 itd.

W skład pakietu wchodzą: biblioteka funkcji, cztery programy rezydentne (nakładki na DOS), edytor własnych fontów dla Herkulesa, VGA i drukarki (Draft i NLQ), programy pomocnicze, pliki rozmaitych fontów, programy przykładowe oraz doskonała dokumentacja (podręcznik oczywiście po polsku), a w przypadku CLIPPERA 5.x dodatkowo plik QUICKPOL umożliwiający polonizację bez zmian w programie źródłowym oraz plik NEWORDER definiujący strukturalno-obiektowy język NewOrder (włączający do CLIPPERA m.in. obiekty klasy MENU).

POLY-C

Zawiera wymienione wyżej programy rezydentne i narzędziowe, podręcznik napisany specjalnie dla C oraz bibliotekę blisko 300 funkcji dostarczanych w postaci źródłowej, przystosowanych do użycia wraz z dowolnym kompilatorem języka C na IBM PC (standard ANSI lub C++, szablony dla Borlanda 3.1), a także możliwych do wykorzystania przez znających C programistów w CLIPPERZE, FOX PRO czy też w PARADOX-ie.

Biblioteka funkcji POLY-C zawiera większość funkcji z biblioteki POLY dla CLIPPERA, a ponadto kilka funkcji napisanych specjalnie dla C. W bibliotece została uwzględniona kompleksowa (ekran, klawiatura, drukarka, manipulacje znakowe, konwersja) wzorowo zorganizowana polonizacja programu, a ponadto takie zagadnienia, jak: dowolne operacje na datach i łańcuchach znaków, pakowanie i szyfrowanie danych, pełnoekranowe wprowadzanie danych (obiekty GET), wyświetlanie dużych tekstów w okienkach, uniwersalne podejście okienkowe, funkcja INTPOLY symulująca operacje BIOS-u w pamięci ekranu, manipulacje ekranowe, drukowanie danych, organizacja menu wielopoziomowych, dwuwymiarowych i logicznych (obiekty MENU i PROMPT, menu strumieniowe), przetwarzanie list, help kontekstowy, język NewOrder, komunikacja programu w C z rezydentami pakietu, itd.

NEW INDEX

Już od października br. zostanie wprowadzony do sprzedaży nowy pakiet obsługi baz danych w C (debiut naSOFTARG'u 93). Zawiera obsługę zbiorów DBF i DBT z możliwością pakowania pól oraz zbiorów NDX, NTX i NZX (pliki indeksowe najnowszej generacji, dowolnie wiele indeksów różnych rodzajów w jednym zbiorze, możliwość programowego ustawienia wielkości strony). Nabywcy POLY-C będą mogli zakupić NEW INDEX jako upgrade (za 50% ceny).

POLY-DOS

Zawiera programy rezydentne i narzędziowe oraz podręcznik (50 stron). Oprócz wszechstronnej polonizacji (tryb tekstowy i graficzny, dowolny standard, opcja zmiany standardu "w locie") umożliwia m.in. przedefiniowanie klawiatury (makrodefinicje dowolnej długości), definiowanie własnych fontów na ekranie i drukarce, a także komunikację z rezydentami pakietu z programu w dowolnym języku (przy pomocy tzw. funkcji POLY-DOS-u).

MECZ

Piłkarska baza danych (z podręcznikiem) na IBM PC zawierająca dowolne (definiowane przez użytkownika) tabele i prognozy.

* * *

A oto ceny pakietów:

POLY 87 (do Clippera 87)	- 1.200 tys. zł
POLY 5 (do Clippera 5.0, 5.01 i 5.2)	- 1.400 tys. zł
POLY 5 + 87	- 1.800 tys. zł
POLY-C (źródła)	- 2.200 tys. zł
NEW INDEX (źródła)	- 2.000 tys. zł
POLY-DOS	- 700 tys. zł
MECZ	- 550 tys. zł

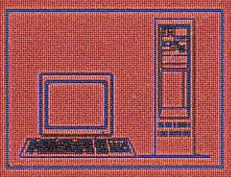
plus opłata za transport. Wysyłka za pobraniem pocztowym. W przypadku łącznego zakupu POLY do C i CLIPPERA dodatkowe 500 tys. zł zniżki.

Oferujemy również oprogramowanie firm COMPUTER ASSOCIATES, BORLAND i MICROSOFT. W tym przypadku obowiązuje zamówienie pisemne wraz z kserokopią dowodu wpłaty na konto **AKREM, PBK SA V/O ŁÓDŹ, nr 374606-11729-136.** W przypadku jednoczesnego zakupu programów zagranicznych i AKREM-u udzielamy 3% rabatu od całości zamówienia.

0/1/93

JUNISOFTEX®

Przedsiębiorstwo Produkcyjno-Usługowe Innowacji Technicznych Biuro Techniczno-Handlowe 44-100 Gliwice ul. Konstytucj 11.
tel. 31-57-00, 31-51-31, 31-75-10, fax 31-75-10, centr. 31-90-81 do 88, telex 036233
Zespół wdrożeń i analiz systemów komputerowych wewn. 15 ,25, 282, Zespół inżynierii oprogramowania wewn. 14, 20, 245,
Zespół dla serwisu systemów Novell i sprzętu komputerowego wewn. 16, 245, Dział Handlowy wewn. 18, 28, 201,
Szkoła Komputerowa wewn. 282, Sekretariat wewn. 250, Prezes wewn. 13, 272, Dyrektor wewn. 12, 272, Główny Księgowy wewn. 256,
** Konto: PKO I/O Gliwice 27661-225298-136 **



OFERUJEMY

Kompleksową komputeryzację przedsiębiorstw obejmującą dostawę i wdrożenie zintegrowanych systemów komputerowych takich jak system:

finansowo-kosztowy (automatyczne rozliczanie kosztów)
+
gospodarka materiałowa
+
środki trwałe
+
wyposażenie
+
obrót towarowy fakturowanie sprzedaży
+
kadry-płace
+
techniczne przygotowanie produkcji

Sprzęt komputerowy w tym:

Komputery
Monitory
Dyski twarde
Drukarki (mozaikowe i laserowe)
osprzęt sieciowy (karty sieciowe, kable zwykłe i światłowodowe)

Systemy i oprogramowanie licencjonowane

DOS
Novell
Btrieve
SQL Base
Windows

Szkolenia komputerowe

podstawy obsługi komputerów
arkusze kalkulacyjne (Qpro, Lotus, EXCEL)
edytory tekstów (WordPerfect, TAG, ChiWriter)
bazy danych (dbase IV, Clipper, Paradox)
systemy operacyjne (DOS, Novell)

SOFTWARE SUPPORT FOR DOS, NOVELL 2.2, 3.11, 4.0, SQL BASE firmy GUPTA

*

OPROGRAMOWANIE DLA PRZEDSIĘBIORSTW

oferujemy własnej produkcji zakładowy system informatyczny PHU-Perfect(N) obejmujący swoim zakresem:
system finansowo-kosztowy umożliwiający automatyczne rozliczanie kosztów, atomatyczną dekretację, analizę rozrachunków
system obrotu materiałami, wyrobami, towarami (kody paskowe) w tym: stany magazynowe, ewidencje zamówień, fakturowanie sprzedaży (VAT) z automatyczną dekretacją rozdzielników kosztów i sprzedaży do systemu finansowo-kosztowego,
system ewidencji majątku trwałego i wyposażenia
system kadrowo-płacowy (SQL) z automatyczną dekretacją rozdzielników płacowych do systemu finansowo-kosztowego
system technicznego przygotowania produkcji (uk. 1993 r)
(oprogramowanie w systemie SQL BASE w przygotowaniu)

*

OPROGRAMOWANIE DLA URZĘDÓW MIAST I GMIN

oferujemy własnej produkcji system finansowo-podatkowy UMG-Perfect(N) obejmujący swoim zakresem:
system finansowo-księgowy,
systemy podatkowe obejmujące: ustalanie wymiarów, naliczanie i kontrolę płatności należności podatkowych dla dowolnych podatników i/lub płatników podatków na ich kontach i kontach dochodów ze swobodną korektą dowolnych informacji w dowolnym okresie a w tym:
ewidencja zapłat z wykorzystaniem kodów paskowych, emisja wezwań do zapłaty, naliczanie odsetek.
system ewidencji majątku

*

Zapraszamy do współpracy w zakresie wdrożeń naszych systemów partnerów, - firmy informatyczne z całego kraju !!!

0/7/92