DRUGA WIOSENNA SZKOŁA PTI

METODY, TECHNIKI I NARZĘDZIA TWORZENIA SYSTEMÓW INFORMATYCZNYCH

Organizowana przez POLSKIE TOWARZYSTWO INFORMATYCZNE

Do użytku wewnętrznego

Świnoujście, 15-19 maj 1989 r.

DRUGA WIOSENNA SZKOŁA PTI

METODY, TECHNIKI I NARZĘDZIA TWORZENIA SYSTEMÓW INFORMATYCZNYCH

Organizowana przez POLSKIE TOWARZYSTWO INFORMATYCZNE

Do użytku wewnętrznego

Świnoujście, 15-19 maj 1989 r.

SŁOWO WSTĘPNE

Polskie Towarzystwo Informatyczne w ramach prowadzonej działalności statutowej spopularyzowało mało dotychczas wykorzystywaną formułę spotkań - Szkołę. Obecna druga edycja Szkoły Wiosennej PTI jest organizowana przez Koło PTI w Szczecinie oraz Sekcje Informatyki Stosowanej w Zarządzaniu. Tradycyjnie Szkoła Wiosenna jest monotematyczna. Druga jej edycja poświęcona jest metodom, technikom i narzędziom tworzenia systemów informatycznych. Wpisując się w formułę szkoły, pragniemy nauczyć lub przypomnieć znane już metody i techniki tworzenia systemów informatycznych oraz zaprezentować nowości w tej dziedzinie.

Przeglądając spis treści referatów można zauważyć metody, które były popularne przed laty i pragniemy je odkurzyć, lub przybliżyć nowym adeptom informatyki. Prezentowane będą również metody mniej znane, których poznanie może wpłynąć na ich spopularyzowanie. Metody i techniki tworzenia systemów informatycznych, przedstawione w referatach, mają różne pola zastosowań, sądzimy więc, że każdy z uczestników Szkoły znajdzie coś interesującego dla siebie. Narzędzia występujące w tytule Szkoły, są najsłabiej reprezentowane, ale taki chyba jest stan polskiej informatyki stosowanej obecnie.

W programie Szkoły, jak również w materiałach wyróżnione zostały:

- wykłady, wygłaszane w sesji przedpołudniowej,
- referaty, uzupełniające tematykę w sesji popołudniowej.

Pięć pierwszych pozycji materiałów zawiera treść wykładów wygłaszanych w postaci trzech 45-cio minutowych spotkań na sesji porannej. Pozostałe pozycje to referaty, wygłaszane w ciągu 45-ciu minut w ramach spotkań po południu. Zarówno referaty jak i wykłady zostały przygotowane specjalnie na zamówienie organizatorów. Zdajemy sobie sprawę, że nie udało nam się osiągnąć pełnej reprezentacji poglądów w tematyce Szkoły, ale mamy nadzieję, że wymiana poglądów w trakcie trwania Szkoły, wypełni przynajmniej w części tę lukę.

Liczymy również na życzliwe uwagi i sugestie uczestników odnośnie organizacji i programu następnej edycji Szkoły Wiosennej.

Zdzisław Szyjewski

Szczecin, marzec, 1989r.

SPIS TREŚCI

mgr Mariusz Klapper (Sekcja Metodyki i Dokumentowania Projektów i Programów PTI)
Tablice decyzyjne

dr inż. Barbara Łukasik-Makowska (Akademia Ekonomiczna Wrocław)
Standaryzacja systemów powielarnych

dr Jerzy Marcinkiewicz (Uniwersytet Szczeciński)
MERISE i REMORA - metody rozwoju systemów informacyjnych

doc. dr hab. Wojciech Olejniczak (Uniwersytet Szczeciński)
Globalna metoda projektowania systemów informatycznych

dr inż. Marek Valenta (AGH Kraków)
Metoda tworzenia systemów ekspertowych

doc. dr hab. inż. Zdzisław Gomółka (Uniwersytet Szczeciński)
Metoda wzorca stanów w projektowaniu systemów informatycznych

dr Jacek Irlik (Instytut Systemów Sterowania Katowice)
O definiowaniu systemu informatycznego

dr Leszek Kotulski (Uniwersytet Jagielloński)
Koncepcja dwupoziomowego systemu generacji oprogramowania dla lokalnych sieci komputerowych

dr Mieczysław Lech-Owoc (Akademia Ekonomiczna Wrocław)
Struktury danych w systemach powielarnych

dr Marek Skwarnik (Akademia Ekonomiczna Wrocław)
Standaryzacja warunków eksploatacji systemów powielarnych

mgr Ireneusz Szydłowski (Uniwersytet Szczeciński)
Metoda wspomaganego projektowania systemów informatycznych

dr Zdzisław Szyjewski (Uniwersytet Szczeciński)
Metoda Jacksona projektowania oprogramowania

doc. dr hab. Stanisław Wrycza (Uniwersytet Gdański)
Narzędzia komputerowo wspomaganego tworzenia systemów informatycznych w świetle CASExpo EUROPE

mgr inz. specj. I st.
Mariusz KLAPPER
przew. Sekcji

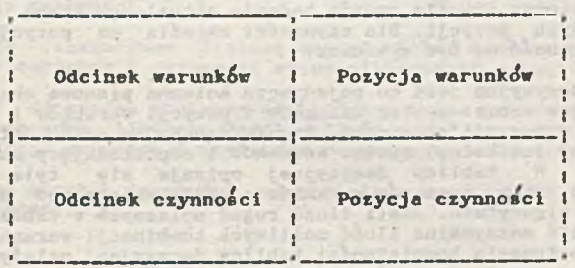
Kraków, luty 1989r.

TABLICE DECYZYJNE

1. POJĘCIA PODSTAWOWE.

Tablica decyzyjna jest specyficznym sposobem zapisywania i analizowania złożonych algorytmów, w których bada się określony (i na ogół skomplikowany) zestaw warunków i w zależności od ich aktualnego stanu wykonuje się jeden ze zdefiniowanych zestawów czynności. Tablica decyzyjna umożliwia opisanie algorytmu w sposób zwięzły i przejrzysty poprzez pokazanie wszystkich branych pod uwagę warunków, rozważanych kombinacji tych warunków, wszystkich przewidywanych czynności i wszystkich zestawów czynności powiązanych z zestawami warunków. Tablice decyzyjne mogą być używane dla opisywania algorytmu zamiast sieci działań, która dla złożonych algorytmów jest z reguły bardzo rozbudowana i trudna do analizowania.

Tablica decyzyjna jest podzielona logicznie na cztery podstawowe pola, które można schematycznie przedstawić w sposób następujący:



Odcinek warunków zawiera specyfikacje wszystkich warunków branych pod uwagę w opisywanym problemie. Każdy warunek jest wyrażeniem logicznym (lub wywołaniem funkcji logicznej) typu "IF", reprezentującym wartość "prawda" lub "fałsz". Dla tzw. warunków rozszerzonych może to być wyrażenie logiczne reprezentujące jedną z wyspecyfikowanych wartości (jest to wtedy odpowiednik logicznych wariantów typu "CASE"). Warunki powinny być zapisane w kolejności priorytetów ich testowania. Mniej istotne, lub zależne od innych warunki powinny być zapisywane po warunkach bardziej znaczących. Do tablicy wpisuje się tylko treść warunku, bez "IF".

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

Odcinek czynności zawiera specyfikacje wszystkich czynności wykonywanych w opisywanym problemie. Specyfikacja czynności może być bezpośrednim zapisem czynności w tablicy (np. podstawienie, przesłanie i tp.), lub wywołaniem procedury zdefiniowanej niezależnie od tablicy decyzyjnej. Może to być również wywołanie innej tablicy decyzyjnej, lub powrót na początek analizowanej aktualnej tablicy (zapętlenie tablicy). Czynności należy podawać w takiej kolejności, w jakiej mają one być wykonywane. W pewnych przypadkach może być konieczne powtórzenie zapisów czynności, aby pokazać różną kolejność ich wykonywania dla różnych reguł.

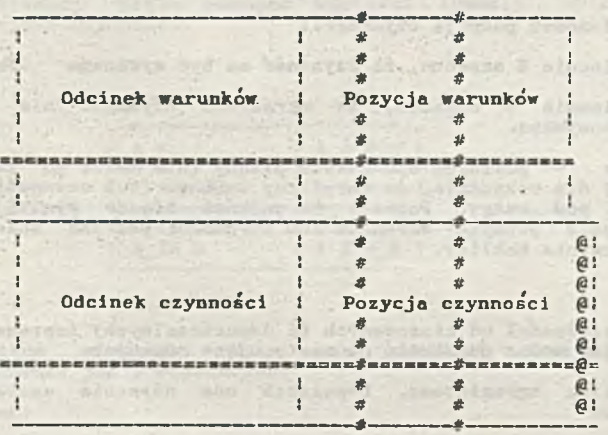
Pozycja czynności zawiera stabelaryzowaną specyfikację wszystkich zestawów czynności odpowiadających poszczególnym zestawom warunków wyspecyfikowanych w pozycji warunków. Każdej specyfikacji zestawu warunków w pozycji warunków musi odpowiadać tylko jedna specyfikacja w pozycji czynności. Czynności wskazane do wykonania w zestawie są wykonywane w kolejności ich zapisania w odcinku czynności (licząc od góry w dół). Oznaczanie zmiennej kolejności wykonania czynności w pozycji czynności nie jest zalecane, gdyż może być mylnie interpretowane.

Wiersz jest to pojedyncza linia pozioma obejmująca odcinek oraz pozycję warunków lub odcinek oraz pozycję czynności. Dla warunków wiersz określa sposób badania aktualnego stanu warunku dla kolejnych pozycji. Dla czynności określa on pozycję, dla których czynność ma być wykonana.

Reguła decyzyjna jest to pojedyncza kolumna pionowa obejmująca odpowiadające sobie: zestaw warunków w pozycji warunków i zestaw czynności w pozycji czynności. Reguła decyzyjna musi definiować jednoznaczny (unikalny) zestaw warunków i odpowiadający mu zestaw czynności. W tablicy decyzyjnej opisuje się tyle reguł decyzyjnych, ile zestawów warunków zostało wzięte pod uwagę w opisywanym algorytmie. Jeśli ilość reguł opisanych w tablicy jest mniejsza niż maksymalna ilość możliwych kombinacji warunków, to dla zagwarantowania kompletności tablicy decyzyjnej należy opisać dodatkową, pomocniczą regułę tzw. regułę "ELSE".

Reguła 'ELSE' nie definiuje pozycji warunków, a jedynie pozycje czynności wykonywanych w przypadku, gdy żadna z kombinacji warunków zdefiniowanych w pozycjach warunków tablicy nie jest spełniona. Reguła ELSE nie musi być podana w każdej tablicy decyzyjnej. Jeśli została podana, to musi być ostatnią regułą opisaną w tablicy. Nie wolno jej podawać, jeśli w pozycji warunków zostały wyspecyfikowane wszystkie możliwe kombinacje warunków rozważanych w opisywanym algorytmie. Jeśli nie zostały wyspecyfikowane wszystkie możliwe kombinacje warunków, to zazwyczaj jest wymagane podanie reguły ELSE.

Strukturę tablicy decyzyjnej można zatem przedstawić schematycznie w sposób następujący:



W przedstawionym powyżej schemacie tablicy decyzyjnej można wyróżnić:

- pola: odcinka warunków, pozycji warunków, odcinka czynności i pozycji czynności
- wiersze (zaznaczone liniami =====). Ilość wierszy zależy od ilości warunków i czynności wyspecyfikowanych w tablicy
- reguły decyzyjne (kolumny występujące tylko po stronie pozycji i zaznaczone liniami ####). Ilość reguł zależy od ilości kombinacji warunków i czynności rozważanych w tablicy.
- kolumnę pozycji czynności odpowiadającą regule ELSE (zaznaczona linią @@@@).

Przy zapisywaniu tablicy decyzyjnej stosuje się następujące oznaczenia:

- w kolumnach pozycji warunków:
  - wpisanie Y lub T oznacza, że określony warunek musi być spełniony ('PRAWDA'), jeśli reguła ma być wybrana
  - wpisanie N oznacza, że określony warunek nie może być spełniony ('FAŁSZ'), jeżeli reguła ma być wybrana
  - wpisanie - oznacza, że stan określonego warunku nie ma znaczenia lub warunek nie jest możliwy do spełnienia
- w kolumnach pozycji czynności:
  - wpisanie X oznacza, że czynność ma być wykonana
  - wpisanie - oznacza, że określona czynność nie ma być wykonywana.

Znak '-' powinien być zawsze pisany (nie można go zastępować spacją) dla wskazania, że określony warunek (lub czynność) został wzięty pod uwagę. Pozwala to uniknąć błędów wynikających z pominięcia pewnych warunków lub czynności podczas układania i analizowania tablicy.

W zależności od stosowanych (i dopuszczalnych) zapisów tablice decyzyjne można podzielić na następujące rodzaje:

Tablica ograniczona. Dopuszcza ona używanie następujących zapisów:

- Warunki muszą być wyrażeniami logicznymi typu "IF" tzn. ich aktualny stan może być tylko 'PRAWDA', 'FAŁSZ' lub 'NIE BADANY' (odpowiednie symbole w pozycji warunków: T, N, -).
- Czynności mogą być wykonywane, lub pomijane (odpowiednie symbole w pozycji czynności: X, -).

Przykład:

a = 10	T T T N
b = a	- - N T
a + b >= x	N T - -
b := b + a	X X X -
a := b	X - - X

Tablica rozszerzona. Dopuszcza ona używanie następujących zapisów:

- Warunki mogą być wyrażeniami logicznymi typu "CASE" tzn. w pozycji warunków można podawać aktualnie braną pod uwagę wartość, a nie stan logiczny. Aktualna wartość jest na ogół cyfrą, lub pojedynczym znakiem. Rzadziej dopuszcza się możliwość podawania jako aktualnej wartości liczb, identyfikatorów danych, lub ciągów znaków (tekstów).

- Można określać kolejność wykonywania czynności w ramach reguły decyzyjnej. W pozycji czynności wpisuje się wtedy symbol '-' dla czynności pomijanych, oraz dowolny znak (na ogół cyfrę) dla czynności wykonywanych. Czynności są wykonywane w kolejności wyznaczonej przez rosnące wartości symboli wpisanych w pozycjach czynności.

Przykład:

a =	1 2 8 -
b =	- - a a
a + b >=	- x - -
b := x + a	2 1 1 -
a := b	1 2 - 1

Tablica mieszana. Może ona zawierać zapisy zarówno ograniczone, jak i rozszerzone.

Przykład:

a = 10	T T T N
b =	- - - a
a + b >= x	N T - -
b := x + a	2 X X -
a := b	1 - - X

Używanie w tablicy decyzyjnej zapisów rozszerzonych lub mieszanych pozwala zazwyczaj zmniejszyć rozmiary tablicy i polepszyć jej czytelność. Utrudnia to jednak analizowanie i weryfikowanie tablicy. Większe jest zatem ryzyko popełnienia podczas opracowywania tablicy błędu lub przeoczenia. Tablice o zapisach rozszerzonych i mieszanych są też trudniejsze do automatycznego tłumaczenia na programy i procedury komputerowe. Dlatego zakres ich stosowania jest ograniczony.

Niektóre warunki opisywane w tablicy decyzyjnej mogą być wzajemnie zależne, lub mogą się wzajemnie wykluczać.

Dwa warunki są zależne, jeżeli stan jednego z nich może mieć wpływ na stan drugiego. Na przykład warunki:

a = 10
a > 7

są zależne, gdyż spełnienie pierwszego z nich automatycznie pociąga za sobą spełnienie drugiego.

Dwa warunki się wykluczają, jeżeli są zależne i spełnienie jednego z nich wyklucza możliwość spełnienia drugiego. Na przykład warunki:

x = 5
x = 7

się wykluczają, gdyż spełnienie jednego z nich wyklucza możliwość spełnienia drugiego.

Wystąpienie w tablicy decyzyjnej warunków zależnych lub wykluczających się nie jest błędem, o ile nie powoduje dwuznaczności reguł (problem dwuznaczności reguł będzie omówiony w dalszej części opracowania). Na ogół wystąpienie warunków zależnych pociąga za sobą zwiększenie rozmiarów tablicy (zwłaszcza zwiększenie ilości warunków w odcinku warunków i związane z tym zwiększenie ilości reguł decyzyjnych).

Warunki zależne występują najczęściej w tablicach decyzyjnych ograniczonych. Tablice rozszerzone i mieszane umożliwiają bowiem opisywanie pozycji warunków wariantowych, a one są najczęstszym powodem powstawania warunków zależnych w tablicy ograniczonej.

Występowanie w tablicy decyzyjnej warunków zależnych lub wykluczających się powinno być odpowiednio zaznaczane. Przy analizowaniu i weryfikowaniu poprawności tablicy nie musi bowiem być z góry wiadome (ani oczywiste), że niektóre warunki są zależne. Znaczenie takie realizuje się zazwyczaj poprzez wpisanie wspólnego identyfikatora w określonym miejscu we wszystkich wierszach odcinka warunków zawierających warunki zależne.

Jeśli nie stosujemy zaznaczania warunków zależnych, a opisujemy tablicę decyzyjną ograniczoną, w której występują warunki wykluczające się, to należy pamiętać o konieczności definiowania warunku w pozycji warunków jako niespełnionego (symbol 'N') w tych regułach, w których nie może on być spełniony. Definiowanie w tym miejscu warunku jako nie badanego (symbol '-') może łatwo doprowadzić do potraktowania reguły jako dwuznacznej (zwłaszcza, jeśli analiza dwuznaczności jest wykonywana automatycznie np. przez program komputerowy).

Na przykład pozycje warunków wykluczających się zapisane w postaci:

```
x = 1        T - -
x = 5        - T -
x = 8        - - T
```

mogą spowodować, że opisane w ten sposób reguły decyzyjne zostaną potraktowane jako dwuznaczne. Przy weryfikacji tablicy nie musi bowiem być z góry wiadome że podane warunki się wykluczają. Dlatego przykładową tablicę decyzyjną "bezpieczniej" jest zapisać w postaci:

```
x = 1        T N N
x = 5        N T N
x = 8        N N T
```

Innym sposobem uniknięcia dwuznaczności może być wspomniane powyżej stosowanie znaczenia warunków zależnych i wykluczających się np.:

```
!        x = 1        T - -
!        x = 5        - T -
!        x = 8        - - T
@        y = 20       - T -
@        y = 50       - - T
```

gdzie znaki '!' i '@' sygnalizują, że zaznaczone w ten sposób warunki należy traktować jako zależne, lub wykluczające się (znaki ! i @ zostały w przykładzie przyjęte dowolnie i nie są to oznaczenia ogólnie stosowane).

Problem ten ma szczególne znaczenie, gdy tablica decyzyjna ograniczona będzie analizowana i weryfikowana automatycznie.

Problem zaznaczania i analizowania warunków zależnych ma również znaczenie w tablicach decyzyjnych rozszerzonych i mieszanych. Nie zawsze jest możliwe rozpisanie w jednym wierszu tablicy rozszerzonej wszystkich wariantów wymagających wzięcia pod uwagę. W takiej sytuacji, pomimo stosowania zapisu rozszerzonego konieczne staje się zdefiniowanie w tablicy warunków zależnych. Na przykład zestaw warunków:

```
x =          1 2 1 4 5
y >          4 5 2 1 1
x + y >      4 8 - 9 -
```

daje w rezultacie zależność warunku 3 od warunków 1 i 2. Weryfikacja takiej tablicy decyzyjnej może wskazać reguły 1 i 3 jako dwuznaczne (wartości x = 1 i y = 5 spełniają obie te reguły).

Podczas analizowania i wykonywania reguł opisanych w tablicy decyzyjnej są stosowane następujące zasady:

- tablica decyzyjna musi być identyfikowana nagłówkiem zawierającym jej nazwę, unikalną w ramach systemu. W nagłówku mogą też być umieszczone pomocnicze informacje opisujące tablice (ilość warunków, czynności i reguł, ilość wierszy zapisu na jeden wiersz tablicy, opcję analizowania tablicy, występowanie reguły ELSE i tp.).

- tablica decyzyjna jest traktowana jako wydzielona procedura (np. podprogram), lub wydzielony fragment sekwencji procedur.

- wejście wykonywania do tablicy decyzyjnej z innych procedur może nastąpić tylko do początku tablicy. Procedury inicjujące są wykonywane tylko raz, po wejściu do tablicy.

- wyjście z tablicy decyzyjnej następuje po wykonaniu ostatniej czynności wskazanej do wykonania w wybranej regule decyzyjnej, o ile nie jest to czynność nakazująca powrót na początek tej samej tablicy, lub o ile wcześniej nie została wykonana czynność bezwarunkowego wyjścia z tablicy.

- podczas analizowania i wykonywania tablicy decyzyjnej realizowane są następujące czynności:

  = wyznaczany jest aktualny stan warunków wyspecyfikowanych w odcinku warunków. Warunki są analizowane w kolejności ich zapisania w odcinku warunków (logiczne JEŚLI warunek-1 ORAZ warunek-2 ORAZ ..... warunek-n).

  = w pozycji warunków jest badana zgodność aktualnego stanu warunków i kolejnych zestawów warunków opisanych w pozycji warunków tablicy. Badane są kolejne kolumny pozycji warunków licząc od lewej strony (logiczne JEŚLI reguła-1 LUB reguła-2 LUB ..... reguła-n). Do wykonania jest wybierana pierwsza reguła, dla której aktualny zestaw warunków jest dokładnie spełniony. Jeśli nie jest spełniony żaden z wyspecyfikowanych zestawów warunków, to wybierana jest reguła ELSE.

  = wszystkie zaznaczone do wykonania czynności w pozycji czynności wybranej reguły są wykonywane w kolejności ich wpisania w odcinku czynności (licząc od góry do dołu), lub w kolejności wynikającej z zapisu rozszerzonego (logiczne 'WYKONAJ .... ORAZ WYKONAJ .... ORAZ WYKONAJ .. ').

Dla ilustracji omówionych spraw został podany przykład prostej tablicy decyzyjnej. Tablica ta algorytm aktualizowania zbioru danych o dostępie sekwencyjnym poprawkami zapisanymi w innym zbiorze o dostępie sekwencyjnym. Każdy rekord poprawki posiada dodatkowy kod identyfikujący typ poprawki. Kod ten jest zapisany w polu rekordu poprawki o nazwie KODPOPR i może mieć jedna w wartości: 'K' (kasowanie). 'D' (dopisanie) i 'A' (aktualizacja). Rekordy zbioru głównego i zbioru poprawek są posortowane według identycznego klucza. Rekordy o takim samym kluczu w zbiorze poprawek są dodatkowo posortowane według kodu poprawki i mają

kolejność: kasowanie, dopisanie, aktualizowanie. Pole zawierające wartość klucza w aktualnym rekordzie zbioru głównego nazywa się KLUCZGŁ. Pole zawierające wartość klucza w aktualnym rekordzie zbioru poprawki nazywa się KLUCZPO. Podczas aktualizowania zbiór główny jest przepisywany do zbioru wyjściowego. Procedury błędu w danych wejściowych, oraz procedury redagowania rekordu wyjściowego z rekordu poprawki są opisane w oddzielnych tablicach decyzyjnych TABDECRED i TABDECBLE (nie pokazanych w przykładzie). Komentarze są ujęte w nawiasy klamrowe { }. Ostatnią regułą w tablicy jest reguła ELSE. Dla uproszczenia przyjęto zapis tablicy ograniczonej. Strzałki zaznaczone po prawej stronie tablicy pokazują sposób przekazywania sterowania przez czynność "powrót na początek tablicy".

```
   { nagłówek }
TABLICA DECYZYJNA aktualizacja;     W=7; C=9; R=8;
   { procedury inicjujące }
- otwarcie zbioru głównego
- otwarcie zbioru poprawek
- utworzenie zbioru wyjściowego
- wczytanie rekordu zbioru głównego
- wczytanie rekordu zbioru poprawek
   { tablica decyzyjna }
```

koniec zbioru głównego	N N N N N T T	< < <
koniec zbioru poprawek	N N N N T N T	^
KLUCZPO > KLUCZGŁ	T N N N - - -	^
KLUCZPO = KLUCZGŁ	- T T N - - -	^
KODPOPR = 'K' {kasowanie}	- T N - - - -	^
KODPOPR = 'D' {dopisanie}	- - - T - T -	^
KODPOPR = 'A' {aktualizacja}	- N T - - - -	^
		^
procedury aktualizacji	- - X - - - - -	^
przesłanie rek. we. na rek. wy.	X - X - X - - -	^
wykonanie tablicy TABDECRED	- - - X - X - -	^
zapis rekordu wyjściowego	X - X X X X - -	^
wczytanie rekordu zb. głównego	X X X - X - - -	^
wczytanie rekordu zb. poprawek	- X X X - X - X	^
powrot na początek tablicy	X.X X X X X - X	> > ^
wykonanie tablicy TABDECBLE	- - - - - - - X	
zamknięcie zbioru wyjściowego	- - - - - - X -	

W podanej tablicy przykładowo trzecia reguła jest interpretowana w sposób następujący:

"JEŻELI NIE koniec zbioru głównego ORAZ NIE koniec zbioru poprawek ORAZ NIE KLUCZPO > KLUCZGŁ ORAZ KLUCZPO = KLUCZGŁ ORAZ NIE KODPOPR = 'K' ORAZ KODPOPR = 'A',
TO
WYKONAJ procedury aktualizacji ORAZ WYKONAJ przesłanie rekordu wejściowego na rekord wyjściowy ORAZ WYKONAJ zapis rekordu wyjściowego ORAZ WYKONAJ wczytanie rekordu zbioru głównego ORAZ WYKONAJ wczytanie rekordu zbioru poprawek ORAZ WYKONAJ powrót na początek tablicy".

Przy zapisywaniu tablicy decyzyjnej można stosować dodatkowe formy zapisu ułatwiające analizowanie i używanie tablicy:

- Na początku tablicy (przed nagłówkiem, lub pomiędzy nagłówkiem a odcinkiem i pozycją warunków) mogą być wyspecyfikowane pomocnicze czynności inicjujące, które należy wykonać przed rozpoczęciem badania i wykonywania tablicy np. zerowanie zmiennych globalnych, otwarcie zbiorów, ustawienie początkowych wartości parametrów i tp. Ewentualna czynność "powrót na początek tablicy" podana w odcinku czynności pomija powtorne wykonanie czynności inicjujących.

- W nagłówku tablicy, lub ponad odcinkiem warunków tablicy należy podać całkowitą ilość warunków, czynności i reguł wyspecyfiowanych w tablicy (w podanej kolejności). Ilości te należy oznaczyć odpowiednio literami np. 'W', 'C', 'R' (Warunki, Czynności, Reguły).

- W razie potrzeby można umieszczać w tablicy dodatkowe komentarze lub objaśnienia. Nie mogą one jednak pogarszać czytelności i przejrzystości tablicy

- Warunki i odpowiadające im pozycje, oraz czynności i odpowiadające im pozycje zapisuje się na ogół w tym samym wierszu. Jeśli tekst definiujący warunek lub czynność jest długi, lub ilość rozważanych w tablicy reguł decyzyjnych jest duża, to warunek (lub czynność) i jego pozycja mogą się nie zmieścić w tym samym wierszu. Możliwe jest wtedy zapisanie tekstu warunku w jednym wierszu, a pozycji warunku w wierszu następnym. Należy jednak unikać takiego zapisu, gdyż na ogół pogarsza on czytelność i przejrzystość tablicy. Najprostsze sposoby uniknięcia zbyt długich zapisów to zastąpienie warunku wywołaniem funkcji logicznej, a czynności wywołaniem procedury.

Tablica decyzyjna może być wykonywana cyklicznie (w pętli). Można to osiągnąć poprzez umieszczenie jako jednej ze zdefiniowanych czynności (na ogół ostatniej czynności wykonywanej w pozycji czynności) powrotu na jej początek.

Tablice decyzyjne mogą być wzajemnie zagnieżdżane. Najłatwiej można to osiągnąć definiując jako jedna z czynności tablicy wykonanie innej tablicy decyzyjnej. Tablica zagnieżdżona musi być wyodrębniona procedurą.

Do tablicy decyzyjnej mogą być wstawiane pomocnicze czynności lub procedury ułatwiające jej testowanie. Najczęściej polegają one na monitorowaniu (tzn. rejestrowaniu i pokazywaniu) aktualnego stanu warunków i wynikającego z tego stanu wyboru reguły decyzyjnej .

Tablice decyzyjne mogą być powiązane między sobą, tzn. ułożone w logiczne sekwencje. Tablice wykonywane sekwencyjnie powinny być ułożone kolejno, lub ostatnią czynnością w poprzedzającej tablicy powinno być przekazanie sterowania do następnej tablicy (tj. czynność "SKOCZ DO TABLICY..."). Wykonywanie następnej tablicy rozpoczyna się od jej początku.

Tablica decyzyjna może być powiązana z innymi algorytmami opisywanymi w analizowanym systemie np. przy pomocy sieci działań za pomocą czynności typu 'SKOCZ DO SIECI DZIAŁAŃ ...'. Identyfikator sieci działań do której następuje wyjście musi być podany w specyfikacji czynności realizującej wyjście. W tablicach decyzyjnych będących wyodrębnionymi procedurami wyjście z tablicy powinno być realizowane za pomocą czynności typu 'EXIT', a nie czynności typu 'GOTO'.

W szczególnych przypadkach stosuje się inne sposoby rozszerzania możliwości tablicy decyzyjnej (zapisywanie reguł pozornych, przekazywanie sterowania z czynności tablicy do jej innej reguły i tp.). Sprawy te nie będą szerzej omawiane.

2. PROBLEMY OPRACOWYWANIA TABLIC DECYZYJNYCH.

Opracowanie tablicy decyzyjnej opisującej badany algorytm można zrealizować w sposób następujący:

1. Zdefiniować wszystkie warunki, których badanie jest konieczne dla realizacji analizowanego algorytmu.
2. Zdefiniowane warunki wpisać w logicznej kolejności w polu odcinka warunków tablicy decyzyjnej.
3. Zdefiniować wszystkie związane z analizowanymi warunkami czynności, na jakie można rozłożyć procedury realizacji analizowanego algorytmu.
4. Zdefiniowane czynności wpisać w polu odcinka czynności.
5. W polu pozycji warunków rozpisać wszystkie możliwe kombinacje warunków wpisanych w odcinku warunków.
6. Analizując kolejne kolumny pozycji warunków zaznaczać w odpowiednich kolumnach pozycji czynności potrzebne zestawy działań. W ten sposób powstaną wszystkie możliwe reguły decyzyjne analizowanego algorytmu.
7. Po ułożeniu w ten sposób tablicy decyzyjnej należy ją przeanalizować pod kątem możliwości połączenia niektórych reguł, oraz pod kątem występienia w niej reguł dwuznacznych i sprzecznych. Problemy te zostaną dokładniej omówione poniżej.

Przedstawiona metoda jest łatwa i niezawodna. Nie jest ona jednak praktyczna. Należy bowiem zauważyć, że ilość kombinacji warunków rozpisywanych tą metodą rośnie wykładniczo (przy podstawie 2) ze wzrostem ilości warunków. Dla 5 warunków należy zatem rozpisać 32 ich kombinacje, ale już np. dla 8 warunków kombinacji będzie 256, a dla 16 warunków jest ich aż 65536. Rozpisanie i przeanalizowanie takiej ilości warunków nie jest praktycznie możliwe. Zazwyczaj nie jest też konieczne rozważanie wszystkich możliwych kombinacji warunków.

Dlatego przy układaniu tablicy decyzyjnej postępuje się zwykle w sposób następujący:

1. Definiuje się wszystkie warunki, których badanie jest konieczne dla realizacji analizowanego algorytmu. Zdefiniowane warunki wpisuje się w logicznej kolejności w polu odcinka warunków tablicy decyzyjnej.

2. Definiuje się wszystkie związane z analizowanymi warunkami czynności, na jakie można rozłożyć procedury realizacji analizowanego algorytmu. Zdefiniowane czynności wpisuje się w polu odcinka czynności tablicy decyzyjnej.

3. W polu pozycji warunków rozpisuje się kolejno kombinacje warunków wymagające rozważenia w analizowanym algorytmie.

4. Analizując kolejne kolumny pozycji warunków zaznacza się w odpowiednich kolumnach pozycji czynności potrzebne zestawy działań.

5. Specyfikacje zestawu czynności dla wszystkich zestawów warunków nie uwzględnionych w tablicy opisuje się w polu pozycji czynności reguły ELSE.

7. Ułożoną tablicę decyzyjną analizuje się kątem możliwości połączenia reguł, oraz pod kątem wystąpienia w niej reguł dwuznacznych i sprzecznych.

Taki uproszczony sposób postępowania jest bardziej praktyczny, ale ma kilka wad:

- stwarza ryzyko przeoczenia (pominięcia) reguły decyzyjnej wymagającej uwzględnienia w tablicy.

- może doprowadzić do zdefiniowania w tablicy zbyt dużej ilości reguł decyzyjnych (kontrolowanie możliwości łączenia reguł jest utrudnione)

- wprowadza do tablicy element nieokreśloności, gdyż użycie reguły ELSE (konieczne dla zdefiniowania czynności związanych ze wszystkimi zestawami warunków nie branymi pod uwagę w tablicy) może spowodować utratę kontroli nad nietypowymi zestawami aktualnych stanów warunków tablicy.

Problemem wymagającym szerszego omówienia jest weryfikacja logicznej poprawności tablicy decyzyjnej. W ramach weryfikacji sprawdza się przede wszystkim możliwość łączenia reguł, oraz występowanie w tablicy reguł dwuznacznych lub sprzecznych.

Dwie reguły decyzyjne mogą być połączone (tzn. zastąpione jedną regułą), jeśli w polu pozycji czynności mają podane identyczne zestawy czynności, a przyporządkowane im w polu pozycji warunków zestawy warunków tylko w jednym wierszu mają parę "T" "N". Reguły takie można zastąpić jedną regułą wstawiając w miejsce pary "T" "N" znak "-" (nie badać).

Przykład:

tablice decyzyjną:

a = 10	T T T N
b = a	T T N N
a + b >= x	N T - T
b := x + a	X X X -
a := b	X X - X

można zastąpić tablicą:

a = 10	T T N
b = a	T N N
a + b >= x	- - T
b := x + a	X X -
a := b	X - X

Występowanie w tablicy decyzyjnej reguł dających się połączyć nie jest błędem, ale niepotrzebnie rozbudowuje tablicę. Dlatego reguły należy łączyć wszędzie tam, gdzie jest to możliwe.

Sposób postępowania przy łączeniu reguł jest następujący:

1. Należy przeanalizować kolejno wszystkie kombinacje zestawów czynności w regułach decyzyjnych i wybrać spośród nich te pary, które mają podane identyczne zestawy czynności.

2. Dla każdej wybranej pary reguł należy porównać zdefiniowane w nich zestawy warunków. Jeśli zestawy te różnią się tylko jedną pozycją warunków 'T' N', to reguły można połączyc.

Podane powyżej zasady dotyczą tablic decyzyjnych ograniczonych. Dla tablic rozszerzonych odnajdowanie reguł dających się połączyć jest na ogół trudniejsze, ale zasady postępowania są podobne.

Dwie reguły decyzyjne są dwuznaczne, jeżeli dla identycznego aktualnego stanu warunków jest spełniona każda z nich. Jeżeli definiują one różne zestawy działań, to są również sprzeczne.

Przykład:

a = 10	T T T N
b > 5	- - N N
a + b >= 12	- T - T
b := x + a	X X X -
a := b	X X - X

Opisane powyżej pary reguł 1 i 2, 1 i 3, oraz 2 i 3 są dwuznaczne (pary 1 i 3, oraz 2 i 3 są rownież sprzeczne), gdyż przykładowy zestaw warunków:

a = 10 b = 2

spełnia jednocześnie reguły decyzyjne 1, 2 i 3.

Wystąpienie w tablicy decyzyjnej reguł dwuznacznych jest poważnym błędem, gdyż przedstawiony w ten sposób w tablicy opis algorytmu jest niejednoznaczny i tablica nie może być pprawnie interpretowana. Dlatego reguły dwuznaczne i sprzeczne muszą być bezwarunkowo wyeliminowane z tablicy decyzyjnej.

Istnieje stosunkowo prosty sposób odnajdowania w tablicy decyzyjnej reguł dwuznacznych i sprzecznych. Należy zauważyć, że dwie reguły decyzyjne będą dwuznaczne, jeżeli nie mają pary 'T' 'N' w co najmniej jednym wierszu pozycji warunków. Dlatego sposób postępowania przy wyszukiwaniu reguł dwuznacznych jest następujący:

1. Należy przeanalizować kolejno wszystkie kombinacje zestawów warunków w regułach decyzyjnych i wybrać spośród nich te, które nie mają w co najmniej jednym wierszu pary 'T' 'N'.

2. Jeśli odnalezione w ten sposób reguły są dwuznaczne (tzn. mają takie same zestawy czynności), to należy jedną z nich usunąć z tablicy.

3. Jeśli odnalezione w ten sposób reguły są sprzeczne (tzn. mają różne zestawy czynności), to należy je ponownie przeanalizować i usunąć sprzeczność np. tworząc parę 'T' 'N' w jednym z wierszy odcinka warunków.

Reguły dwuznaczne występują najczęściej w tablicach decyzyjnych, w których wiele warunków w pozycji warunków nie jest badanych (są zaznaczone symbolem '-'). Dlatego przy opracowywaniu tablicy decyzyjnej należy unikać pomijania badania warunków wszędzie, gdzie tylko jest to możliwe.

Odnalezienie w tablicy decyzyjnej reguł nadających się do połączenia, lub reguł dwuznacznych nie jest sprawą łatwą przy użyciu tradycyjnych technik "ręcznego" analizowania. Wymaga to bowiem starannego przeglądnięcia i przeanalizowania wszystkich kombinacji reguł opisanych w tablicy. Dla tablic średniej i dużej wielkości (tzn. od kilkunastu do kilkudziesięciu warunków i czynności) wymaga to dosyć żmudnego przeglądania i analizowania wielu zależności. Czynności związane z analizowaniem i poprawianiem tablicy decyzyjnej można zalgorytmizować. Dlatego weryfikacje tablic decyzyjnych jest zazwyczaj wykonywane przy pomocy narzędzi wspomagających (np. programów komputerowych).

Przy opracowywaniu tablicy decyzyjnej podstawową sprawą jest umiejętne wydzielenie w analizowanym algorytmie wszystkich elementarnych warunków logicznych wymagających przebadania, oraz zdefiniowanie wszystkich elementarnych czynności, z jakich składają się procedury analizowanego algorytmu. Takie podejście różni się od stosowanego zazwyczaj sposobu sekwencyjnego analizowania algorytmów (np. przy użyciu sieci działań). Dlatego praktyczne stosowanie techniki tablic decyzyjnych wymaga przezwyciężenia pewnych nawyków i zmiany podejścia do analizy algorytmu.

Trudności związane z opracowaniem i weryfikacją tablicy decyzyjnej rosną silnie ze wzrostem ilości opisywanych w tablicy warunków i działań. Dlatego należy dążyć do takiego definiowania warunków i działań, aby tablica decyzyjna realizowała poprawnie założony algorytm, ale nie była przy tym zbyt rozbudowana. W praktyce nie powinno się przekraczać ilości kilku do kilkudziesięciu warunków i kilku do kilkudziesięciu czynności opisywanych w jednej tablicy. Jeśli analizowany algorytm jest skomplikowany i wymaga rozłożenia na większą ilość warunków oraz czynności, to należy próbować "rozpisać" go w kilku powiązanych tablicach decyzyjnych. Modułowy charakter tablicy decyzyjnej wyraźnie sprzyja takim rozwiązaniom. Inna metoda ograniczenia rozmiarów tablicy decyzyjnej może być stosowanie wywołań funkcji logicznych w odcinku warunków, lub wywołań procedur zewnętrznych w odcinku czynności.

Łatwość opracowywania i posługiwania się tablicą decyzyjną zależy w dużej mierze od jej starannego i czytelnego zapisania. Tablice decyzyjne są zazwyczaj zapisywane na standardowych formularzach. Formularze te i sposób ich wypełniania są dostosowane do sposobu dalszego postępowania z tablicą decyzyjną. Jeśli np. tablica stanowi jedynie element dokumentowania algorytmu, to stosowane w niej zapisy odcinków warunków i czynności są określeniami potocznymi. Jeśli tablica stanowi opis procedury realizowanej np. komputerowo, to stosowane w niej zapisy odcinków warunków i czynności są wyrażeniami używanymi w języku programowania, w którym będzie kodowany program realizujący algorytm. W każdym przypadku jest rzeczą bardzo ważną, aby zapisy w tablicy decyzyjnej były czytelne, jednoznaczne, oraz poprawne merytorycznie i formalnie.

Dla ułatwienia praktycznego wykorzystywania tablic decyzyjnych opracowuje się zautomatyzowane metody weryfikacji tablic oraz tłumaczenia ich na kod źródłowy programu realizującego algorytm opisany w tablicy. Na ogół weryfikacja i tłumaczenie tablic jest realizowane przez specjalne oprogramowanie. Używanie tego oprogramowania stwarza dodatkowe możliwości związane z testowaniem merytorycznej poprawności tablicy decyzyjnej w rzeczywistych warunkach. Programy tłumaczące są bowiem z reguły wyposażone w możliwość wbudowywania w kod programu źródłowego dodatkowych procedur umożliwiających obserwowanie i kontrolowanie procesu realizacji algorytmu przez tablice decyzyjną. Można dzięki temu stosunkowo łatwo sprawdzić merytoryczną poprawność opisu algorytmu w opracowanej tablicy decyzyjnej. Realizowanie tych samych czynności środkami tradycyjnymi (przez opracowywanie, kodowanie i testowanie algorytmów sieciowych) jest zawsze trudniejsze i bardziej pracochłonne.

3. CHARAKTERYSTYKA TABLIC DECYZYJNYCH.

Używanie tablic decyzyjnych dla opisywania algorytmów ma wiele zalet. Najważniejsze spośród nich to:

- stosunkowo prosty i czytelny sposób opisu złożonego algorytmu.
- szybka i stosunkowo łatwa weryfikacja poprawności.
- szybkie i łatwe wprowadzanie zmian i uzupełnień.
- czytelne i uniwersalne dokumentowanie algorytmu.
- łatwość przekazywania informacji o skomplikowanych zależnościach logicznych w opisywanym algorytmie.
- możliwość automatyzowania przejścia od zapisu algorytmu do kodu programu realizującego ten algorytm.

Podstawowe wady tablic decyzyjnych to:

- specyficzne podejście do analizowania algorytmu, w poważny sposób różniące się od podejścia tradycyjnego.
- niekonwencjonalny sposób zapisywania algorytmu, wymagający poznania go i zrozumienia.

Pomimo zwięzłości powyższego zestawienia należy mocno podkreślić fakt, że tablice decyzyjne są bardzo wygodnym i skutecznym narzędziem opisywania lagorytmów. O ich zaletach najlepiej można się przekonać stosując je w praktyce. Żadne opisy słowne nie potrafią zastąpić bezpośrednich wniosków wynikających z doświadczeń praktycznych.

4. TŁUMACZENIE TABLIC DECYZYJNYCH NA KOD PROGRAMU.

Praktyczna przydatność tablic dacyzyjnych znacznie wzrasta z chwilą, gdy istnieje możliwość automatycznego przetłumaczenia tablicy na kod źródłowy programu realizującego algorytm opisany w tej tablicy. Istnieją dwa podstawowe sposoby przejścia od kodu tablicy decyzyjnej do programu realizujacego opisany w niej algorytm.

Pierwszy z nich to **program interpretujący**. Dla programów interpretujących tablica decyzyjna musi być zapisana w sposób pozwalający na "wbudowanie" jej w kod źródłowy programu pisanego w określonym języku programowania. Nie jest przy tym istotny wybór języka programowania. Ważny jest taki sposób zapisania tablicy, aby mogła ona być zaakceptowana przez kompilator języka, a następnie zinterpretowana i wykonana przez program wynikowy. Na ogół osiąga się to opisując odcinki warunków i czynności jako stablicowane wywołania odpowiednich funkcji i procedur, a pozycje warunków i czynności jako tablice kodów w obszarze stałych programu. Program interpretujący działa wtedy w ten sposób, że przeglądając stablicowane pozycje warunków wywołuje odpowiednie funkcje logiczne określające aktualny stan warunków. Po ustaleniu tego stanu program interpretujący wyszukuje w stablicowanej pozycji warunków tą regułę, która spełnia aktualny stan warunków. Na tej podstawie wybiera odpowiedni element stablicowanej pozycji czynności i wywołuje stablicowane procedury wskazane do wykonania w wybranej pozycji.

Częścią stałą (na ogół modułami bibliotecznymi) programu interpretującego są procedury: przeglądania stablicowanej pozycji warunków, wywoływania funkcji badających warunki, wyszukiwania spełnionej reguły decyzyjnej, przeglądania stablicowanych pozycji czynności, oraz wywoływania procedur realizujących wskazane do wykonania czynności. Zmienna (tzn. opisywana oddzielnie dla każdego indywidualnego przypadku) jest zawartość odcinków warunków i czynności, oraz zawartość tablic zawierających pozycje warunków i czynności.

Tablica decyzyjna, lub grupa tablic decyzyjnych jest wbudowana w tekst programu źródłowego zawierającego wywołania części stałych programu interpretującego. Funkcje i procedury realizujące badania warunków i wykonywanie czynności są zakodowane jako oddzielne części programu źródłowego. Program źródłowy musi być zakodowany w sposób gwarantujący poprawne przygotowanie, wywołanie i wykonanie tablicy decyzyjnej (lub grupy tablic). W programie źródłowym można również zakodować inne procedury związane z realizowanym algorytmem.

Program interpretujący nie sprawdza merytorycznej poprawności tablicy decyzyjnej, ani innych zakodowanych części programu źródłowego. Poprawność programu musi zapewnić osoba opracowująca go, co w znacznym stopniu ogranicza zalety stosowania tablicy decyzyjnej, oraz utrudnia wykrycie ew. błędów.

Program interpretujący może mieć wbudowane procedury formalnej weryfikacji tablicy (wywoływane np. na początku programu lub przed wejściem do tablicy), oraz procedury monitorowania realizacji algorytmu opisanego w tablicy decyzyjnej.

Części stale programu interpretującego powinny być opracowane w sposób umożliwiający prawidłowe wykonywanie tablic decyzyjnych wzajemnie zagnieżdżonych, oraz umożliwiający prawidłowe przekazywanie sterowania pomiędzy tablicami wzajemnie powiązanymi.

Zapis tablicy decyzyjnej przystosowanej do wymagań programu interpretującego jest na ogół mało czytelny. Forma zapisu tablicy musi bowiem odpowiadać wymaganiom składniowym języka, w którym został opracowany program interpretujący, oraz musi spełniać wymagania dotyczące rozmieszczenia i struktury jej elementów nałożone przez logiczną budowę programu interpretującego. Ograniczenia te z reguły pozostają w sprzeczności z wymogami zwartości i czytelności zapisu tablicy.

Wykonywanie tablicy decyzyjnej przy pomocy programu interpretujacego jest stosunkowo mało efektywne, gdyż konieczne jest wykonywanie wielu pomocniczych czynności związanych z interpretowaniem zapisu tablicy decyzyjnej, przeglądaniem stablicowanych zapisów odcinków i pozycji warunków i reguł i tp.

Dlatego zastosowanie programów interpretujących dla wykonywania tablic decyzyjnych jest ograniczone. Metoda ta jest najczęściej stosowana tam, gdzie nie ma innych możliwości, lub inne możliwości nie mogą być wykorzystane (np. konieczne jest kodowanie na poziomie języka assemblerowego).

Znacznie częściej do weryfikacji i tłumaczenia tablic decyzyjnych są używane tzw. preprocesory. Są to pomocnicze programy, które tłumaczą tablice decyzyjną (umieszczoną jako fragment kodu źródłowego programu) na odpowiedni ciąg instrukcji w języku programowania programu źródłowego. Fragmenty programu źródłowego nie będące elementami zapisu tablic decyzyjnych są przez preprocesor kopiowane bez zmian. Tekst tablicy decyzyjnej jest zatem częścią programu źródłowego, ale nie musi być zapisany zgodnie z wymogami składniowymi języka programowania. Preprocesor zamienia go na poprawny kod źródłowy programu i włącza do programu źródłowego przeznaczonego do kompilacji.

Preprocesory umożliwiają stosowanie znacznie bardziej czytelnych zapisów tablicy decyzyjnej, gdyż nie nakładają ograniczeń wynikających ze składni języka programowania. Ponieważ tablica decyzyjna jest częścią programu źródłowego, więc można jej pewne elementy (np. funkcje lub procedury dla warunków i czynności) kodować poza tekstem tablicy.

Podczas tłumaczenia tablicy decyzyjnej preprocesor wykonuje zazwyczaj jej analizę formalną, wykrywając ewentualne nieprawidłowości (np. istnienie reguł dwuznacznych, sprzecznych, lub nadmiarowych).

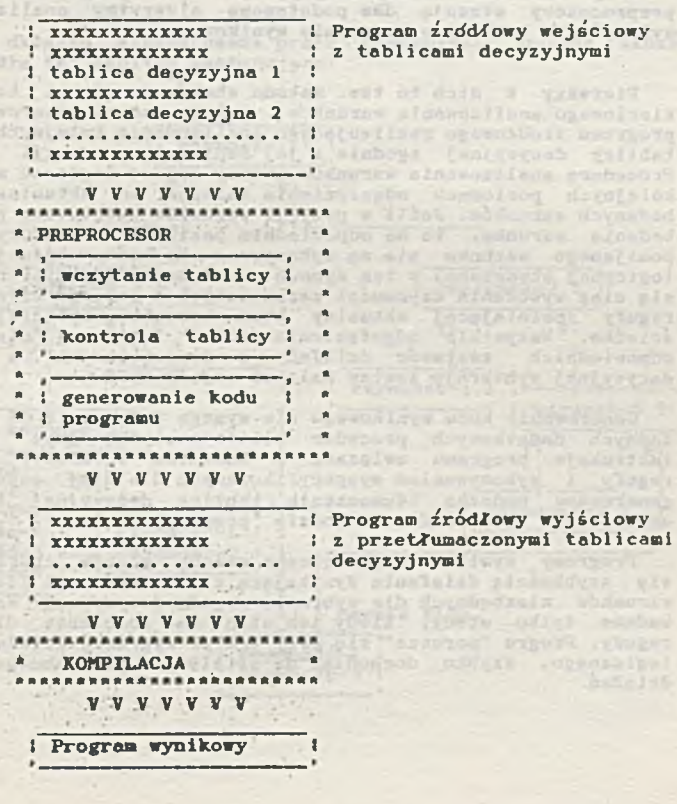
Preprocesor wykonuje tłumaczenie tablicy decyzyjnej w trzech kolejnych fazach wykonywanych cyklicznie, aż do napotkania końca przepisywanego programu źródłowego:

1. Przepisując program źródłowy wyszukuje w nim tablice decyzyjną.

2. Wykonuje analizę i weryfikację formalnej poprawności tablicy decyzyjnej.

3. Jeśli nie wykryje w tablicy decyzyjnej błędów, to generuje do przepisywanego programu źródłowego ciąg instrukcji realizujących algorytm opisany w tablicy decyzyjnej.

Opisany cykl pracy preprocesora można przedstawić schematycznie w sposób następujący:



Zapis tablicy decyzyjnej przeznaczonej do tłumaczenia przez preprocesor zawiera na ogół informacje pomocnicze potrzebne dla prawidłowego interpretowania i tłumaczenia tablicy np. nagłówek i znacznik końca tablicy, opis struktury tablicy,ilość warunków i czynności, ilość reguł, występowanie reguły ELSE, sposób zapisywania tablicy, oznaczenia warunków zależnych, dołączanie procedur monitorowania tablicy i tp.

Stosowanie preprocesorów umożliwia uzyskiwanie znacznie bardziej efektywnych programów wynikowych. Często użytkownik może mieć wpływ na efektywność programu wynikowego, gdyż wiele preprocesorów umożliwia wybranie sposobu tłumaczenia tablicy decyzyjnej, lub wykorzystuje sposoby tłumaczenia najbardziej efektywne dla warunków, i sprzętu na którym będzie pracował program wynikowy.

Przy zamianie tablicy decyzyjnej na kod programu źródłowego preprocesory stosują dwa podstawowe algorytmy analizowania i wykonywania tablicy w programie wynikowym.

Pierwszy z nich to tzw. metoda skoków (nazywana też metodą sieciowego analizowania warunków). Polega ona na generowaniu kodu programu źródłowego realizującego analizowanie kolejnych warunków tablicy decyzyjnej zgodnie z jej zapisami w pozycji warunków. Procedura analizowania warunków tworzy drzewo logiczne mające na kolejnych poziomach odgałęzienia zależne od aktualnego stanu badanych warunków. Jeśli w pozycji warunków zaznaczono pominięcie badania warunku, to na odpowiednim poziomie procedury badania pomijanego warunku nie są wykonywane. Na końcu każdej ścieżki logicznej utworzonej w ten sposób sieci badań logicznych znajduje się ciąg wywołania czynności zaznaczonych w pozycji czynności dla reguły spełniającej aktualny stan warunków wybierających tą ścieżkę. Wszystkie odgałęzienia sieci logicznej nie mające odpowiednich zestawów działań z definiowanych w tablicy decyzyjnej wybierają zestaw działań reguły ELSE.

Generowanie kodu wynikowego nie wymaga dołączania do programu żadnych dodatkowych procedur bibliotecznych, gdyż wszystkie instrukcje programu związane z badaniem warunków, wybraniem reguły i wykonywaniem wyspecyfikowanych dla niej czynności są generowane podczas tłumaczenia tablicy decyzyjnej i zostają umieszczone bezpośrednio w kodzie programu.

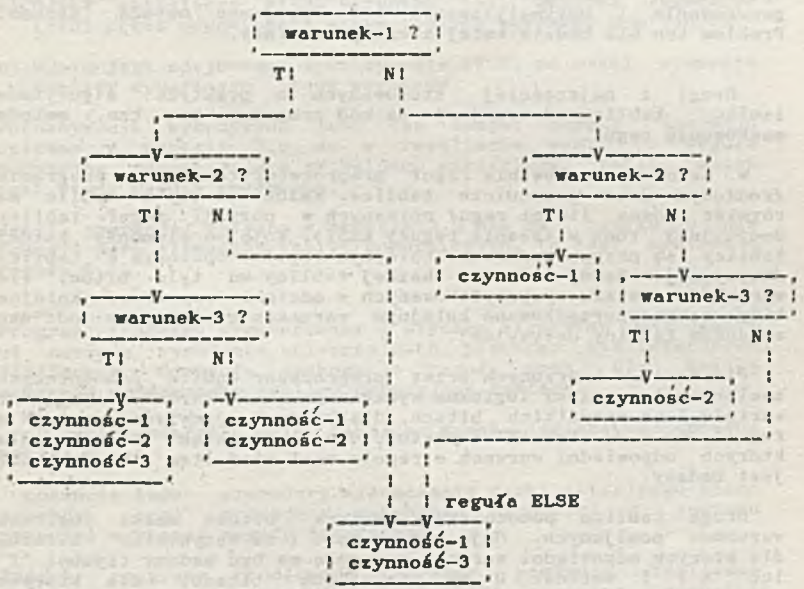
Programy wynikowe wygenerowane metodą skoków charakteryzują się szybkością działania wynikającą z minimalizacji ilości badań warunków niezbędnych dla wybrania reguły decyzyjnej. Warunki są badane tylko wtedy, kiedy ich stan ma znaczenie dla wyboru reguły. Progra "porusza" się przy tym po wybranej ścieżce drzewa logicznego, szybko dochodząc do ustalenia potrzebnego zestawu działań.

Poniżej został podany przykład sieci działań wygenerowanej metodą skoków dla prostego algorytmu opisanego w tablicy decyzyjnej.

Przykładowa tablica decyzyjna ma postać:

warunek-1	T	T	N	N
warunek-2	T	T	T	N
warunek-3	T	N	-	T
czynność-1	X	-	X	-
czynność-2	X	X	-	X
czynność-3	X	X	-	-

Sieć działań wygenerowana przez preprocesor metodą skoków wygląda dla tej tablicy następująco:



Metoda skoków tworzy długi kod programu źródłowego dla każdej tablicy decyzyjnej (występują liczne powtórzenia procedur badania tych samych warunków i wykonywania tych samych czynności). W konsekwencji programy wynikowe wygenerowane tą metodą zajmują zazwyczaj duży obszar pamięci operacyjnej.

Dlatego metodę skoków wybiera się zazwyczaj wtedy, gdy wymagana jest duża szybkość pracy programu wynikowego, a zasoby sprzętowe (zwłaszcza pamięć operacyjna) są dostępne i mogą być wykorzystane bez ograniczeń.

Algorytm generowania przez preprocesor kodu programu metodą skoków jest w praktyce znacznie bardziej skomplikowany, niż mogło by to wynikać z przedstawionego omówienia. Utworzenie prawidłowej sieci logicznej badań warunków wymaga uwzględnienia różnych możliwych wariantów zapisu w pozycjach reguł (zwłaszcza wtedy, kiedy wiele pozycji warunków nie jest badanych). Jeszcze trudniejsze jest poprawne generowanie kodu dla tablic rozszerzonych i mieszanych. Wygenerowana sieć działań powinna przy tym optymalizować przeglądanie warunków, a więc preprocesor musi być wyposażony w możliwości analizowania i wyboru optymalnych rozwiązań. Dlatego istnieje wiele algorytmów generowania i optymalizowania kodu programu metoda skoków. Problem ten nie bedzie tutaj szerzej omawiany.

Drugi z najczęściej stosowanych w praktyce algorytmów zamiany tablicy decyzyjnej na kod programu to tzw. metoda maskowania reguł.

W metodzie maskowania reguł preprocesor tworzy w programie źródłowym dwie pomocnicze tablice. Każda z tych tablic ma rozmiar równy ilości reguł opisanych w pozycji reguł tablicy decyzyjnej (bez wliczania reguły ELSE). Kolejne elementy każdej tablicy są przyporządkowane kolejnym regułom opisanym w tablicy decyzyjnej. Każdy element każdej tablicy ma tyle bitów, ile warunków zostało wyspecyfikowanych w odcinku warunków. Kolejne bity są przyporządkowane kolejnym warunkom zapisanym w odcinku warunków tablicy decyzyjnej.

Pierwsza z tworzonych przez preprocesor tablic pomocniczych zawiera bitowe maski logiczne wymaganego stanu warunków. Mają one wartość 1 na wszystkich bitach, dla których odpowiedni warunek w regule ma mieć stan 'T' i wartość 0 na wszystkich bitach, dla których odpowiedni warunek w regule musi mieć stan 'N' lub nie jest badany.

Druga tablica pomocnicza zawiera bitowe maski logiczne warunków pomijanych. Mają one wartość 1 na wszystkich bitach, dla których odpowiedni warunek w regule ma być badany (symbol 'T' lub 'N') i wartość 0 na wszystkich bitach, dla których odpowiedni warunek w regule nie ma być badany (symbol '-').

Tworzenie tablic masek logicznych ułatwia formalną weryfikację tablicy decyzyjnej, gdyż tablice masek mogą być wykorzystane np. do wyszukiwania reguł dwuznacznych.

Algorytm wyszukiwania reguły decyzyjnej w metodzie maskowania reguł jest nastepujący:

1. Sprawdzane są aktualne stany wszystkich warunków wyspecyfikowanych w odcinku warunków tablicy decyzyjnej.

2. Tworzona jest bitowa maska logiczna aktualnego stanu warunków, zawierająca wartość 1 na wszystkich bitach odpowiadających warunkom, których aktualny stan jest "PRAWDA" i wartość 0 na wszystkich bitach odpowiadających warunkom, których aktualny stan jest "FAŁSZ".

3. Maska aktualnego stanu warunków jest porównywana z kolejnymi elementami tablic masek logicznych utworzonych przez preprocesor. Dla każdej pary kolejnych elementów obu tablic wykonywane są następujące czynności:

   a) Maska aktualnego stanu warunków jest mnożona logicznie (AND) przez odpowiednią maskę tablicy warunków pomijanych

   b) Wynik jest odejmowany symetrycznie (XOR) od maski elementu tablicy wymaganego stanu warunków.

4. Porównywanie wykonywane jest tak długo, dopóki operacja opisana w punkcie 3.b da w rezultacie wynik 0. Reguła decyzyjna odpowiadająca aktualnemu wskaźnikowi tablic masek jest wtedy regułą spełnioną.

5. Jeśli operacja opisana w punkcie 3.b nie da wyniku 0 dla żadnej pary elementów tablic pomocniczych, to jest wybierana reguła ELSE.

Program źródłowy wygenerowany z użyciem algorytmu maskowania reguł zawiera wywołanie uniwersalnych procedur bibliotecznych realizujących operacje maskowania reguł, oraz dla każdej tłumaczonej w programie tablicy decyzyjnej zawiera:

- w obszarze stałych dwie tablice masek logicznych opisane powyżej

- w obszarze kodu: procedury wyznaczania maski aktualnego stanu warunków, oraz procedury wykonania czynności z zestawu czynności wybranej reguły decyzyjnej

Części programu źródłowego zapisane pomiędzy tablicami decyzyjnymi, oraz sekwencje instrukcji zapisane wewnątrz tablicy, ale nie należące do jej struktury (np. instrukcje inicjujące) są przepisywane do tłumaczonego programu bez zmian.

Metodę maskowania reguł ilustruje następujący przykład:

Tablica decyzyjna ma postać:

warunek-1	T T N N
warunek-2	T T T N
warunek-3	T N - T
czynność-1	X - X - X
czynność-2	X X - X -
czynność-3	X X - - X

Bitowa tablica masek warunków pomijanych zawiera elementy:

111, 111, 110, 111

Bitowa tablica masek wymaganego stanu warunków zawiera elementy:

111, 110, 010, 001

Dla przykładowego aktualnego stanu warunków:

warunek-1 = 'FAŁSZ' warunek-2 = 'PRAWDA' warunek-3 = 'PRAWDA'

maska aktualnego stanu warunków będzie miała wartość bitową 011.

Wybieranie reguły będzie wykonywane następująco:

- badanie pierwszych elementów tablic masek:

011 AND 111 = 011; 011 XOR 111 = 100 => reguła 1 nie spełniona

- badanie drugich elementów tablic masek:

011 AND 111 = 011; 011 XOR 110 = 101 => reguła 2 nie spełniona

- badanie trzecich elementów tablic masek:

011 AND 110 = 010; 010 XOR 010 = 000 => reguła 3 spełniona

W ten sposób zostanie wybrana jako spełniona reguła 3.

Stosowanie metody maskowania reguł daje stosunkowo krótki kod źródłowy programu. Dlatego program wynikowy będzie potrzebował stosunkowo mało pamięci operacyjnej. Wadą tej metody jest jednak konieczność badania aktualnego stanu wszystkich warunków opisanych w tablicy decyzyjnej. W przypadku np. tablic o zapisie ograniczonym zawierających warunki wykluczające się prowadzi to do konieczności wykonywania niepotrzebnych badań. Metoda maskowania reguł powinna zatem być stosowana, gdy istnieje ograniczenie dostępnej pamięci operacyjnej, a nie jest wymagana wysoka efektywność programu wynikowego.

5. STOSOWANIE TABLIC DECYZYJNYCH NA MIKROKOMPUTERACH.

Technika tablic decyzyjnych powstała pod koniec lat pięćdziesiątych. Została rozwinięta i znalazła szerokie zastosowanie praktyczne w latach sześćdziesiątych i z początkiem lat siedemdziesiątych. W wielu ośrodkach jest stosowana do dzisiaj.

Najszersze zastosowanie tablice decyzyjne znalazły wszędzie tam, gdzie były dostępne dobre i wygodne w użyciu preprocesory. W polskich warunkach dotyczy to przede wszystkim maszyn serii ODRA 1300, które dysponowały bardzo dobrymi preprocesorami tablic decyzyjnych dla programów pisanych w języku COBOL. Znane są też (opracowane w Polsce) interpretery tablic decyzyjnych dla programów pisanych w języku PLAN maszyn serii ODRA 1300. Niestety krajowy producent maszyn Jednolitego Systemu nie udostępnił oprogramowania ułatwiającego wykorzystywanie tablic decyzyjnych w programach pisanych na ten sprzęt. jest to jedna z głównych przyczyn zaniku zainteresowania i zastosowań tablic decyzyjnych.

Rozpowszechnienie sprzętu mikrokomputerowego nie zmieniło sytuacji na lepsze. Nie jest bowiem dostępne w kraju oprogramowanie dla mikrokomputerów, wspomagające korzystanie z tablic decyzyjnych. Jest to wielka szkoda, gdyż zasoby sprzętowe mikrokomputerów np. klasy IBM PC pozwalają z powodzeniem stosować technikę tablic decyzyjnych. Mikrokomputery tej klasy posiadają na ogół wystarczająco dużą pamięć operacyjną, oraz szybkę i pojemną pamięć dyskową. Wymogi efektywności programów nie są szczególnie istotne w wypadku programów i systemów realizujących typowe przetwarzanie danych na sprzęcie tej klasy. Istnieją zatem bardzo sprzyjające warunki dla stosowania tablic decyzyjnych w systemach mikrokomputerowych.

6. PODSUMOWANIE

Celem niniejszego opracowania było przypomnienie tablic decyzyjnych, które w ostatnich latach zniknęły z pola zainteresowań i zastosowań praktycznych. Poruszone w opracowaniu problemy zostały potraktowane bardzo skrótowo, gdyż nie było ani możliwości, ani potrzeby omawiania ich dokładniej. Nawet tak zwięzłe omówienie pozwala dostrzec bogactwo możliwości wnoszonych przez stosowanie tablic decyzyjnych dla opisywania algorytmów. Tablice decyzyjne wielokrotnie udowodniły w praktyce swoje liczne zalety. Wielu projektantów i programistów korzysta z powodzeniem do dzisiaj z tablic decyzyjnych, które poznali pracując na maszynach serii ODRA 1300. Należy mieć nadzieję, że przypomnienie tablic decyzyjnych zachęci do głębszego przestudiowania problemów i korzyści związanych z ich stosowaniem. Należy też mieć nadzieję, że stosunkowo szybko zostanie zapełniona luka wynikająca z braku na krajowym rynku oprogramowania wspomagającego używanie tablic decyzyjnych w praktyce.

S P I S T R E Ś C I

str.

1. POJĘCIA PODSTAWOWE. 1

2. PROBLEMY OPRACOWYWANIA TABLIC DECYZYJNYCH. 11

3. CHARAKTERYSTYKA TABLIC DECYZYJNYCH. 16

4. TŁUMACZENIE TABLIC DECYZYJNYCH NA KOD PROGRAMU. 17

5. STOSOWANIE TABLIC DECYZYJNYCH NA MIKROKOMPUTERACH. 25

6. PODSUMOWANIE. 25

B I B L I O G R A F I A

1. S.L. Pollack, H.T.Hicks, W.I Harrison
TABLICE DECYZYJNE
wyd. PWN
Warszawa 1975 r

2. FMC ODRA serii 1300 – Oprogramowanie
COBOL. PROGRAMY POMOCNICZE. Wyd. II
wyd. WZE ELWRO nr 137903
Wrocław, marzec 1979 r.

3. A.Daniels, D.Yates
PODTAWY ANALIZY SYSTEMÓW
wyd. PWN
Warszawa 1974 r.

Dr inż. Barbara Łukasik-Makowska
Instytut Informatyki
Akademia Ekonomiczna Wrocław

STANDARYZACJA SYSTEMOW POWIELARNYCH

1. Założenia ogólne

Dynamiczny rozwój mikrokomputerów sprawił, że obok indywidualnych systemów użytkowych, swój renesans przeżywać zaczęły także systemy powielarne. Ich tworzenie jest naturalnym sposobem na zaspokojenie ogromnego "głodu" oprogramowania, na który cierpi obecnie rynek produktów informatycznych. Przygotowanie dobrego systemu użytkowego wymaga czasu, fachowości, narzędzi - musi zatem drogo kosztować [DWOR88]. Konwencja powielarności systemów pozwala te wysokie koszty rozłożyć pomiędzy kilku użytkowników. Aby jednak powielarność była "cnotą" systemów użytkowych, a nie ich "grzechem", jak to bywało za czasów "dużej" informatyki, głównym atrybutem użytkowych systemów powielarnych musi być ich elastyczność. Elastyczność ta umożliwia dopasowanie systemu do [ŁUKA88a]:

- klasy i konfiguracji sprzętu,
- specyfiki obiektu,
- warunków eksploatacji systemu,
- potrzeb rozbudowy systemu.

Pierwszy z wymienionych kierunków dotyczy wszystkich elementów składających się na konfigurację sprzętu, na którym będzie eksploatowany system. Przedmiotem dopasowania mogą być:

- złożoność instalacji - sieciowa lub autonomiczna,
- rodzaj systemu operacyjnego,
- rodzaj karty graficznej,
- wybór monitora barwnego lub monochromatycznego,
- typ drukarki,
- adres i rodzaj urządzeń pamięci zewnętrznej przeznaczonych

do składowania zbiorów.

Wybór poszczególnych elementów i realizacja dopasowania może być udostępniony użytkownikowi lub też pozostawiony wyłącznie w gestii twórcy systemu. Pierwszy wariant pozwala na pełną swobodę operowania systemem przez użytkownika, niezależnie od rodzaju sprzętu i kolejnych zmian w jego konfiguracji. Jednak każdorazowa zmiana konfiguracji, czy przejście na inny egzemplarz komputera mogą wymagać także rekonfigurowania systemu. Wiąże się to z pewnymi dodatkowymi operacjami technologicznymi, które dla użytkownika o małej wiedzy informatycznej mogą być niejasne, a tym samym rekonfigurowanie systemu może doprowadzić do chwilowej lub nawet trwałej nieprawidłowości w działaniu systemu, a nawet do zniszczenia jakichś zbiorów użytkowych. Innym sposobem zapewnienia potrzebnej elastyczności jest generowanie systemu przez producenta (dystrybutora) zgodnie z zapotrzebowaniem zgłoszonym przez użytkownika. W praktyce stosuje się zazwyczaj rozwiązanie kompromisowe, polegające na tym, że wybrane działania, mające charakter "strategiczny" (pierwsze dwa z wymienionych) stanowią przyczynek do tworzenia wersji generowanych przez producenta, natomiast pozostałe, mające aspekt działań podręcznych pozostają w gestii użytkownika [ŁUKA87].

Drugi kierunek działań uelastyczniających pozwala na dopasowanie systemu standardowego do indywidualnych potrzeb, oczekiwań i realiów obiektu, a konkretnie jego systemu informacyjnego. W tej grupie można wymienić następujące mechanizmy dopasowania:

- indywidualizację kodowania i nazywania danych,
- parametryzację wyboru danych do generowania raportów,
- swobodne generowanie raportów wg projektu użytkownika,
- wybór stosownych algorytmów obliczeniowych z asortymentu zawartego w systemie,
- definiowanie, przez użytkownika własnych algorytmów obliczeniowych,
- swobodny wybór poziomu agregacji danych i ich uporządkowania przy prezentacji.

- repetycję agregatów danych w wybranych rekordach (możliwość deklarowania rekordów zmiennej długości).

Oczywiście nie dla wszystkich systemów użytkowych konieczne jest wstawianie do systemu wszystkich, wymienionych mechanizmów. Umieszczenie konkretnego mechanizmu w konkretnym systemie wymaga przeanalizowania obszaru problemowego tego systemu w różnych obiektach i ustalenia zakresu zmienności i jej dynamiki w warunkach praktycznych.

Trzeci z wymienionych kierunków uelastyczniania pozwala na dopasowanie systemu do zmieniających się warunków eksploatacyjnych, w ramach tych mechanizmów istnieją:

- możliwość eksploatowania systemu przez różną liczbę operatorek,
- wybór szerokości i rodzaju papieru (ciągły lub stronnicowany),
- decyzja o emitowaniu informacji na ekran lub papier,
- decyzja o liczbie kopii wydruku (lub repetycji wydruku),
- wybór nośnika do składowania zbiorów (TM lub dyskietka).

Wszystkie te mechanizmy stanowią podręczne wyposażenie systemu i pozwalają użytkownikowi zmieniać realia działania systemu wraz ze zmieniającymi się warunkami, w których odbywa się przetwarzanie. Należy zwrócić uwagę, iż wyspecyfikowane powyżej mechanizmy są w swej istocie bardzo bliskie kierunkowi dopasowania do klasy i konfiguracji sprzętu. W praktycznych warunkach może być zatem dyskusyjnym które z omówionych mechanizmów udostępnione zostaną do bieżącej działalności użytkownika, a które pozostaną wyłącznie do dyspozycji producentów oprogramowania.

Czwarty z kierunków uelastyczniania wynika z potrzeby dopasowania systemu informatycznego do zmieniających się warunków w obiekcie, czyli rozwoju obiektu, zmian w systemie informacyjnym, modyfikacji formalno-prawnych itp. Kierunek ten obejmuje następujące działania:

- rozszerzanie zakresu danych, zmiana struktur danych [OWOC89],
- generowanie podzbioru danych do przeniesienia ich do in-

nego systemu użytkowego (lub pakietu programowego),
- konwersja danych z innego systemu (pakietu) do zbiorów systemu,
- restrukturyzacja globalna kodów danych,
- rozszerzenie zakresu problemowego systemu (dołączenie nowych modułów),
- działania globalne na wszystkich rekordach bazy danych,
- działania globalne na deklarowanej grupie rekordów bazy danych (wraz z mechanizmami swobodnego wyboru takiej grupy).

Należy zwrócić uwagę, iż wymienione w tej grupie mechanizmy określają w praktyce opcje realizacyjne (wersje systemu), które mogą być oferowane użytkownikom. Jest bowiem dyskusyjnym czy w danym obszarze problemowym należy opracować pojedyncze wersje systemów o bardzo bogatej elastyczności, czy też należy opracowywać wiele takich wersji, z których każda będzie przeznaczona dla węższego kręgu użytkowników. I jedno i drugie podejście ma swoje pozytywy i negatywy zarówno dla twórców systemów, jak i dla ich użytkowników, tym więc istotniejszym jest znalezienie rozwiązania kompromisowego, które w możliwie szerokim zakresie uwzględni potrzeby konkretnych użytkowników, a zarazem od strony projektowej i technologicznej stanowić będzie produkt określonej jakości.

Standaryzacja bywa widziana w trzech płaszczyznach, jako:
- standaryzacja systemu, czyli pewnego produktu myśli i działań zespołu twórców,
- standaryzacja projektowania, czyli trybu tworzenia - wiąże się ona zazwyczaj z pewną metodologią projektowania, praktycznie każda metodyka tworzenia systemów niesie ze sobą pewną standaryzację działań projektowych,
- standaryzacja dokumentowania, czyli sposób, zakres, szczegółowość i postać dokumentowania kolejnych działań (etapów, faz, kroków itp.) w procesie tworzenia systemu.

Każda z tych płaszczyzn, to ogromny konglomerat problemów. Najpopularniejsza, podyktowana potrzebami pracy zespołowej jest standaryzacja dokumentowania. Zazwyczaj jest ona śladem pewnych

ustaleń standaryzacyjnych w płaszczyźnie projektowania i związana jest z pewną tradycją i przyzwyczajeniami zespołu twórców systemu. Bywają też sytuacje, gdy zlecający tworzenie systemu (przyszły jego użytkownik) narzuca konwencję i standard dokumentacji, w jakiej oczekuje wykonania powierzonych prac. Zarówno standaryzacja projektowania, jak i dokumentowania były już wielokrotnie przedmiotem wielu dyskusji, ustaleń, prac badawczych itp. A jednocześnie dotychczas niewiele uwagi poświęcono standaryzacji samych systemów.

Konwencja opracowywania powielarnych systemów użytkowych, jako gotowych produktów o określonych walorach użytkowych i gwarantowanej jakości konstrukcyjno-technologicznej (eksploatacyjnej), przeznaczonych do przekazania ich użytkownikom natychmiast po złożeniu zamówienia na określony produkt i możliwych do wdrażania, gdy tylko zespół wdrożeniowy przygotuje obiekt i zespół do tego przedsięwzięcia, wymaga opracowania tych produktów w określonej postaci. Ustalenie takiej postaci jest obecnie przedmiotem penertacji badawczych i eksperymentalnych, chodzi bowiem o pogodzenie nie zawsze spójnych interesów twórców systemów, ich dystrybutorów oraz nabywców - użytkowników. Istnieją jednak pewne ramy ogólne określające wstępnie konwencję, w jakiej powstawać powinny takie produkty - systemy użytkowe, a mianowicie [ŁUKA88b]:

- przedmiotem działania systemu musi być jednoznacznie określony, szczegółowo sformułowany, skończony zakres funkcjonalny,
- oprogramowanie systemu musi być całkowicie zakończone, sprawdzone i zweryfikowane,
- system stanowi zbiór programów i struktur danych, które są jawne dla użytkownika poprzez dostarczaną mu dokumentację eksploatacyjną (struktury są wyspecyfikowane, algorytmy przetwarzania opisane),
- użytkowanie systemu jest możliwe bądź natychmiast po jego zainstalowaniu na komputerze, bądź po założeniu odpowiedniej bazy danych,

- użytkownikami systemu mają być dowolne osoby, niekoniecznie o profesjonalnym profilu informatycznym, które przeszły odpowiednie szkolenie przygotowawcze (w zakresie obsługi komputera oraz użytkowania systemu).

Osiągnięcie tego zamierzenia jest możliwe, jeśli twórcom systemów przedstawione zostaną założenia projektowe o charakterze standardu. Przedmiotem standaryzacji mogą być trzy przekroje tworzonego systemu, a mianowicie:

- merytoryczny,
- konstrukcyjny i
- technologiczny.

Pierwszy dotyczy tworzenia kolejnych systemów użytkowych w ustalonym obszarze problemowym, musi więc być omawiany w odniesieniu do tego obszaru. Dążenie do standaryzacji nie musi jednak oznaczać, że w danym obszarze problemowym powinien zostać określony wyłącznie jeden wariant systemu uznany za standard i optimum.

Komputerowe wspomaganie zarządzania może być bowiem bardzo zróżnicowane, w zależności od potrzeb użytkownika i jego możliwości inwestowania w informatykę. Tym niemniej dla każdego obszaru problemowego można wskazać pewne minimum funkcjonalne, wynikające z gradacji potrzeb użytkowników działających w określonych realiach gospodarczych (regulowanych obowiązującymi przepisami i zarządzeniami). Taki zestaw funkcji podstawowych musi stanowić jądro systemu rozszerzane następnie o inne moduły w ramach dodatkowych potrzeb użytkowników systemu i inwencji projektantów. W ten sposób w danym obszarze merytorycznym mogą powstać różne systemy powielarne, odnoszące się do różnych klas użytkowników oraz systemy o różnym zasięgu funkcjonalnym. Przykładowo mogą istnieć rozłączne systemy kadrowe i płacowe, pomiędzy którymi następować będzie przesyłanie stosownych danych lub w ich miejsce jeden spójny system kadrowo-płacowy. W pierwszym przypadku będą zatem istniały dwa standardy merytoryczne - jeden dla zagadnień kadrowych, a drugi dla płacowych, natomiast w drugim przypadku będzie to jeden kompleksowy standard zawierający w sobie oba pop-

rzednie (jest on zazwyczaj wypadkową obu tych poprzednich standardów) [KEUN87].

Ustalenie standardu merytorycznego może dawać następstwa w konstrukcji i technologii systemu. Z uwagi na fakt, że ustalanie zakresu funkcji podstawowych musi się odnosić do zdefiniowanego obszaru, więc szczegółowe rozważania nad tym problemem zostaną w niniejszym opracowaniu pominięte, na korzyść ustalenia kierunków w jakich winna postępować standaryzacja merytoryczna.

Dwa następne przekroje standaryzacji mają charakter bardziej uniwersalny (niezależny od przedmiotu - obszaru działania systemu) i dotyczą wyłącznie sposobu rozwiązywania zadania projektowego. Istnieje także pewna liczba elementów, wynikających z obszaru problemowego (dziedziny przedmiotowej systemu) i mających wpływ na konstrukcję i technologię, dla przejrzystości wywodów zostaną one omówione w ramach merytorycznego przekroju standaryzacji.

Standaryzacja systemu jako produktu dotyczy nie tylko samych programów, ale także jego dokumentacji eksploatacyjnej, która dla użytkownika stanowi praktycznie nierozerwalny element składowy systemu. Dokumentacja ta, wraz z nośnikami pamięci (obecnie zazwyczaj dyskietkami) stanowi przedmiot transakcji kupna-sprzedaży systemu użytkowego i z racji swojej postaci (wydruku, maszynopisu) jest materialnym świadectwem zawartości magnetycznych nośników pamięci [ŁUKA88e].

2. Konstrukcyjno-technologiczne zalecenia standaryzacyjne

Przedmiotem standaryzacji konstrukcyjno-technologicznej mogą być następujące obszary systemu użytkowego:

1. Struktura systemu.
2. Mechanizmy zabezpieczeń.
3. Dialog komputerowy.
4. Wprowadzanie danych.
5. Zbiory (baza danych) systemu.
6. Wydruki komputerowe.

Pierwszy obszar standaryzacji dotyczy konstrukcji systemu i ma znaczenie nie tylko dla konkretnie projektowanego systemu, ale także całej grupy systemów, które docelowo mają ze sobą współpracować. Właściwa konstrukcja systemu gwarantuje:

- prostotę i łatwość tworzenia systemu (pisania i testowania programów),
- swobodę rozwoju systemu,

a także ma wpływ na:

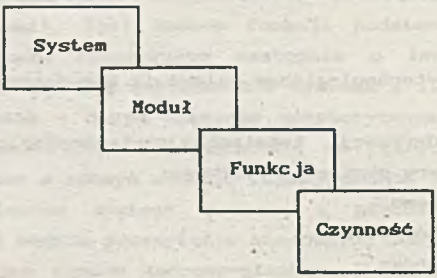
- sprawność przetwarzania (szybkość wykonywania operacji technologicznych),
- wygodę obsługi systemu przez użytkownika.

Standaryzacja tego obszaru nakłada na twórców systemu wymagania, aby:

1.1. System miał regularną konstrukcję hierarchiczną, tzn. w obrębie poszczególnych elementów powinna być porównywalna (analogiczna) liczba poziomów zagłębień.

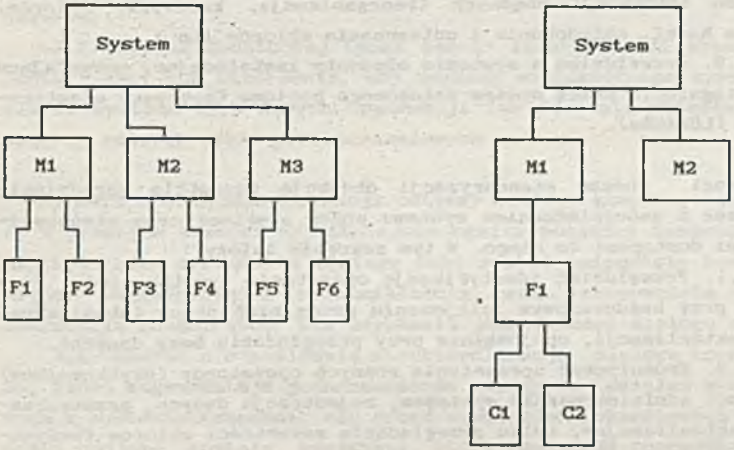
1.2. Na określonym poziomie struktury systemu znalazły się elementy równoważne, w zakresie i złożoności funkcjonalnej.

1.3. Jako zalecaną uważać strukturę czteropoziomową, tj.: system, moduł, funkcja, czynność (rys. 1).



Rys. 1. Elementy struktury konstrukcyjnej systemu

1.4. Nie wydzielać kolejnego poziomu struktury, jeśli elementy wyższego poziomu mają wyłącznie jedną wersję realizacyjną. (rys.2. w przykładzie b, w strukturze zbędny jest element F1).



a) Dobrze b) Zle

Rys. 2. Przykładowe struktury konstrukcyjne systemu

1.5. Przewidzieć następujący ciąg technologiczny działań związanych z wprowadzaniem i/lub aktualizacją danych:

- wprowadzenie (danych),
- kontrola (danych),
- przeglądanie (wprowadzonych danych),
- korekta (wprowadzonych danych),
- ślad w systemie (zapis/aktualizacja).

1.6. "Ukryć" technologię przetwarzania przed użytkownikiem w czasie eksploatacji systemu, tzn. przykładowo zapewnić, aby elementy systemu (np. zbiory) miały nazwy wynikające z ich przeznacze-

nia użytkowego, a nie nazwy technologiczne.

1.7. Przewidzieć w strukturze systemu nie tylko elementy wynikające z zamierzonych działań przetwarzaniowych, ale także elementy zapewniające niezawodną pracę systemu (por. pkt. 2. - standaryzacja mechanizmów zabezpieczeń) oraz gwarantujące eliminację z systemu informacji zbędnych (reorganizacja, kasowanie zbiorów, zmiana haseł, składowanie i odtwarzanie zbiorów itp.).

1.8. Przewidzieć w systemie elementy instalacyjne, pozwalające na osiągnięcie przez system założonego poziomu (zakresu) elastyczności [ŁUKA88a].

Drugi obszar standaryzacji obejmuje wszystkie zagadnienia związane z zabezpieczeniem systemu przed awariami oraz nieupoważnionymi dostępami do niego. W tym zakresie należy:

2.1. Przewidzieć identyfikację operatorów (użytkowników) systemu, przy każdorazowym inicjowaniu przez nich pracy (obowiązkowo przy aktualizacji, opcjonalnie przy przeglądaniu bazy danych).

2.2. Zróżnicować uprawnienia różnych operatorów (użytkowników) systemu: administrowanie systemem, rejestracja danych, przetwarzanie (aktualizacja), tylko przeglądanie zawartości zbiorów (wszystkich lub jedynie wybranych).

2.3. Wprowadzić hasła ochrony dla działań (funkcji), do realizacji których nie są upoważnieni wszyscy operatorzy (użytkownicy) systemu.

2.4. Zapewnić w systemie pozostawienie śladu, kto dokonywał jakich działań na zbiorach (systemie). Może być także w systemie prowadzony rejestr wszystkich inicjacji działań (wszystkich prób wejścia do systemu).

2.5. Objąć dodatkową ochroną wszystkie działania systemu, których skutkiem jest naruszenie aktualnej zawartości bazy danych. Ostateczne podjęcie, przez komputer, takich działań powinno być poprzedzone odpowiednim komunikatem ostrzegawczym, wyświetlanym na ekranie, a w razie potrzeby także sygnalizowanym akustycznie (por. pkt. 3.4.).

2.6. Chronić zbiory systemu poprzez cykliczne, automatyczne kopiowanie ich zawartości. Działania w istotny sposób zmieniające zawartość bazy danych (np. reorganizacja, kasowanie zbiorów transakcyjnych itp.) winny być poprzedzone obowiązkowym składowaniem zbiorów.

2.7. Używać dodatkowej (poza nazwą) identyfikacji zbiorów, w postaci daty ich utworzenia, aby uniknąć wielokrotnego wprowadzania do systemu tych samych transakcji lub łączenia zbiorów dotyczących różnych cykli przetwarzaniowych.

Trzeci obszar standaryzacji dotyczy dialogu komputerowego, który w praktyce stanowi najważniejsze ogniwo pomiędzy komputerem, a użytkownikiem systemu. W dialogu musi zostać osiągnięty kompromis między zwięzłością i zrozumiałością oraz rzeczowością i obszernością komunikatów. Dla uzyskania jednolitości dialogu należy:

3.1. Zadbać o odpowiednią strukturalizację dialogu (por. pkt. 3), która zagwarantuje jednoznaczność następstwa działań wykonywanych w systemie (zapewni, aby przed wykonaniem określonych czynności podjęte zostały wszystkie, które muszą je poprzedzać, np. aktualizacja zbiorów przed obliczeniami).

3.2. Redagować komunikaty systemu użytkowego wyłącznie w języku polskim. Dopuszcza się w wyjątkowych sytuacjach (awarie) pojawienie się na ekranie oryginalnych komunikatów systemu operacyjnego (w jęz. angielskim), lecz wszystkie takie komunikaty winny być spisane i wyjaśnione w dokumentacji eksploatacyjnej systemu.

3.3. Dbać w systemie o zwięzłość, rzeczowość i jednoznaczność komunikatów.

3.4. Przewidzieć w systemie trzy grupy komunikatów:

- rutynowe, informujące użytkownika o podjęciu określonych działań (np. ustaleniu parametrów realizacji czynności),
- błędów, informujących o nieprawidłowych działaniach użytkownika, błędach we wprowadzonych danych itp.,
- ostrzegawcze o dodatkowych (być może przez użytkownika nie przewidzianych) skutkach podjęcia określonych działań (doty-

czy to zwłaszcza działań trudno odwracalnych, jak np. reorganizacja i usuwanie z bazy rekordów nieaktywnych).

3.5. Przewidzieć w systemie dwa poziomy funkcji pomocy (help):
- naturalny, jako komentarz dostępny na ekranie do realizowanych aktualnie działań,
- kontekstowy, jako dodatkowe wyjaśnienia, które w danej chwili (do bieżącego poziomu działań) może uzyskać użytkownik, po wywołaniu funkcji pomocy (naciśnięciu określonego klawisza).

3.6. Rozplanować ekran w taki sposób, aby komunikaty określonej grupy pojawiały się zawsze w tym samym obszarze.

3.7. Zaprojektować w systemie jednolite zasady wyboru funkcji i czynności (działania użytkownika na systemie) - te same klawisze muszą służyć do wyboru tych samych działań (np. klawisze funkcyjne lub alfanumeryczne, por. pkt. 3.10.).

3.8. Przewidzieć możliwość rezygnacji ze wszystkich zainicjowanych działań, niezależnie od złożoności działania (poziomu zagłębienia dialogu) jeszcze przed ostatecznym potwierdzeniem zamiaru ich realizacji oraz przerwania działań już podjętych, bez konieczności awaryjnego wyłączania komputera.

3.9. Standaryzować brzmienie komunikatów, dążąc do tego aby komunikat składał się z części stałej i części zmiennej (sparametryzowanej). Jednocześnie wystąpienie określonej sytuacji (zwłaszcza błędu) musi być zawsze (w różnych fragmentach systemu) sygnalizowane identycznym komunikatem.

3.10. Stosować jednoznakowe symbole komend, używane w dialogu jako dyrektywy działań zlecanych komputerowi przez użytkownika, wyłącznie w ściśle obowiązującym w całym systemie (grupie systemów), unikalnym znaczeniu, tj.:

"e" - wyjście na ekran,
"d" - wyjście na drukarkę,
"i" - inny (rekord),
"k" - korekta (danych, parametrów),
"n" - następna strona,

"p" – powrót,

"w" – wprowadzanie (danych, parametrów),

"y" – wyświetlanie,

"z" – zapisanie (danych, parametrów).

3.11. Zagwarantować odpowiednimi komunikatami śledzenie długotrwałych działań systemu (nie wystarczy jednorazowy komunikat "proszę czekać"), np. przeszukiwania całej bazy danych, oczekiwania na zredagowanie złożonego wydruku, reorganizacji zbiorów itp. W zależności od typu działania, na ekranie muszą się w regularnych odstępach czasu (nie rzadziej niż co 2-3 min) pojawiać komunikaty świadczące o prawidłowym przebiegu przetwarzania (może to być np. odliczanie rekordów "z góry na dół", informacja o kolejnych fazach przetwarzania, gdy użytkownik zna ich docelową liczbę itp.).

3.12. Raportować na ekranie (i ew. wydruku) konsekwencje realizacji działań o znaczeniu strategicznym (np. reorganizacja, aktualizacja itp.) podając liczbę rekordów (zbiorów), której takie działanie dotyczyło (np. liczba skasowanych rekordów).

3.13. Badać stan zbiorów (fakt ich istnienia) i przydatność do przetwarzania przed podjęciem przetwarzania (wyeliminuje to przetwarzanie zbiorów pustych).

Obszar czwarty obejmuje wszystkie elementy związane z gromadzeniem danych i wprowadzaniem ich do komputera. W ramach tego obszaru należy w projekcie systemu użytkowego przestrzegać następujących zasad:

4.1. Przy wprowadzaniu danych, obowiązkowe wypełnianie pola, musi dotyczyć tych wszystkich pól, których zawartość służy do realizacji zestawień rutynowych (np. GUS).

4.2. Przy wprowadzaniu danych system powinien przyjmować wartości domyślne, dla tych wszystkich pól, o których wiadomo, że statystycznie znacząca liczba wystąpień w danych przyjmuje określoną wartość (np. w danych dotyczących zatrudnienia, dla pola "etat" będzie to wartość "pełny etat"). Wartości domyślne powinny być także przyjmowane w takich polach, gdzie istnieją tylko dwa rodza-

je przeciwstawnych wystąpień (np. "Tak" i "Nie" lub dla pola "płeć" wystąpienie "kobieta" i "mężczyzna").

4.3. Nie należy jako wartości domyślnej pola używać znaku spacji (spacji używają zazwyczaj użytkownicy w przypadku braku danych).

4.4. Należy umieścić w systemie wszystkie możliwe kontrole danych wejściowych. Dotyczy to kontroli z punktu widzenia:

- obowiązku wypełnienia pola (musi istnieć blokada wprowadzenia "pustego" pola, którego dotyczy bezwzględny obowiązek wypełnienia por. pkt. 4.1.),
- przedziału wartości dopuszczalnych,
- występowania określonych wartości (por. zasady wykorzystania tablic kodowych dla danych wejściowych, pkt. 4.6.),
- powiązania wartości określonego pola z wartościami innych pól (kontrola logiczna).

4.5. Wartości kontrolne (brzegowe) danych mogą być traktowane jako element uelastyczniający system, tzn. mogą (powinny) być one przewidziane do wprowadzenia w czasie instalacji systemu (przez użytkownika).

4.6. Tablice kodowe wybranych danych powinny być wprowadzane do systemu w czasie jego instalacji.

4.7. W systemie musi istnieć możliwość poprawy wszystkich wprowadzonych danych.

4.8. Dla dokumentów wieloekranowych (wprowadzanych przez wypełnienie kolejnych ekranów) należy przewidzieć kontrole fragmentaryczne, dotyczące fragmentu dokumentu obserwowanego jednorazowo na ekranie (np. sumy kontrolne ekranu, a dopiero na końcu suma kontrolna całego dokumentu).

4.9. Należy w konstrukcji systemu rozróżnić poprawę i aktualizację danych. Aktualizacja musi pozostawiać ślad, tzn. system musi rejestrować dane przed i po aktualizacji, datę jej dokonania, a w systemach dla wielu operatorek także symbol operatorki, która wprowadziła aktualizację. W odniesieniu do danych "strategicznych" w systemie należy także przewidzieć raportowanie dokonanych aktu-

alizacji.

4.10. Wszystkie dane o charakterze daty muszą mieć w systemie identyczny format i postać (nawet wówczas, gdy nie wykorzystuje się pełnej daty, a jedynie jej fragment) RR/MM/DD.

4.11. W momencie inicjacji pracy systemu winien on przyjmować domyślną wartość daty z zegara komputerowego (z możliwością jej zmiany przez użytkownika).

4.12. W systemie musi istnieć możliwość zmiany wartości określonej danej we wszystkich lub w wybranej grupie rekordów, nie tylko w postaci serii kolejnych zmian na pojedynczych rekordach, ale także jako zmiany lub aktualizacji globalnej, dokonywanej łącznie na tej grupie rekordów wskazanych przez użytkownika.

4.13. Dane qasi-stałe (katalogi, normy, cenniki itp.) powinny być do systemu dostarczone w postaci gotowego zbioru na nośniku pamięci (dyskietka), z możliwością wymiany zawartości takiego zbioru, w przypadku konieczności jego aktualizacji.

Piąty obszar standaryzacji obejmuje zbiory gromadzone i przechowywane w systemie, w odniesieniu do nich należy zapewnić:

5.1. Pełną obsługę archiwowania i odzyskiwania bazy danych (jak również tworzenia okresowych kopii wybranych zbiorów, np. transakcyjnych).

5.2. Możliwość łączenia zbiorów tworzonych przez różne operatorki (dotyczyć to powinno nawet systemów jednokomputerowych wówczas, gdy zasadne jest kontrolowanie źródła wprowadzania danych do systemu lub podział kompetencyjny zadań pracowników wobec systemu por. pkt. 6.2.).

5.3. Tworzenie zbiorów archiwalnych (lub jednego łącznego zbioru archiwalnego) z danych usuwanych z bazy podczas reorganizacji.

5.4. Pracę systemu (w zakresie przeglądania i emisji zestawień) na zbiorach archiwalnych.

5.5. Elementarne operacje technologiczne na zbiorach archiwalnych (łączenie, porządkowanie, kopiowanie itp.).

5.6. Tworzenie zbiorów przeznaczonych do przetwarzania w innych systemach. W systemie musi istnieć możliwość wydrukowania (dla celów dokumentacyjnych) pełnej zawartości takiego zbioru.

5.7. Możliwość wprowadzenia do systemu, w postaci zbioru, danych pochodzących z innego systemu. W systemie musi istnieć możliwość wydrukowania pełnej zawartości takiego zbioru, przed jego wykorzystaniem w cyklu przetwarzaniowym (np. przed dokonaniem aktualizacji bazy takimi danymi).

5.8. Samoczynne czyszczenie systemu, tzn. kasowanie wszystkich zbiorów roboczych po zakończeniu danego cyklu przetwarzaniowego (nie należy zaśmiecać dysku przechowując zbiory robocze).

5.9. Zrozumiałe dla użytkownika nazewnictwo zbiorów, zwłaszcza w odniesieniu do tych wszystkich zbiorów, których "obsługę" użytkownik będzie widział/inicjował w ramach dialogu. Niedopuszczalne jest posługiwanie się w komunikatach systemu technologicznymi nazwami zbiorów.

Szósty obszar standaryzacji odnosi się do wydruków komputerowych, a właściwie wyjść z systemu. Dla każdego z wydruków należy przewidzieć:

6.1. Pełną identyfikację wydruku przez podanie w jego nagłówku: symbolu, nazwy, daty opracowania oraz numeracji stron.

6.2. Symbolizację końca poszczególnych stron oraz końca całego wydruku.

6.3. Możliwość restartu już zrealizowanych wydruków.

6.4. Możliwość przerwania wydruku w trakcie jego realizacji, a następnie restartu (wznowienia) lub skasowania.

6.5. Możliwość powtórzenia fragmentu wydruku, przy zachowaniu pierwotnej numeracji stron (ważne zwłaszcza dla długich wydruków).

6.6. Możliwość wyboru klucza uporządkowania danych na wydruku.

6.7. Możliwość swobodnego deklarowania przez użytkownika szerokości (papier szeroki lub wąski) i rodzaju papieru (papier ciągły lub stronnicowany), gęstości wydruku na stronie (pojedynczy skok, półtora skoku, podwójny skok).

6.8. Wyświetlanie na ekranie obowiązujących parametrów wydruku i dokonywanych na nich, przez użytkownika, zmian.

6.9. Domyślne ustawienie przez program parametrów wydruku, z możliwością ich zmiany przez użytkownika (i zapamiętania tych nowych parametrów, aż do kolejnej zmiany).

6.10. Wyraźne rozróżnienie wydruków opracowanych jako "stan na dzień" i wydruków "za okres od... do...".

6.14. Wybór emisji zestawienia na ekran lub na papier. Wybranie wersji ekranowej w odniesieniu do dużych wydruków musi pozwalać na ich przeglądanie na zasadzie arkusza elektronicznego.

6.12. Możliwość generowania zestawień według wymagań użytkownika, który określi dane i ich układ na zestawieniu (oraz czy ma to być lista danych, czy ich liczba lub suma zbiorcza).

6.13. Możliwość określenia przez użytkownika szczegółowości zestawień, czy mają być one w postaci listy, sumy zbiorczej, czy np. sum cząstkowych (wg jakiegoś klucza uporządkowania zestawienia) i sumy zbiorczej itp.

6.14. Wyjście z systemu na zbiór przeznaczony do dalszego przetwarzania z użyciem edytora tekstowego (wyboru pól, które powinny znaleźć się w tym zbiorze powinien dokonywać użytkownik).

6.15. Wyjście z systemu na zbiór przeznaczony do dalszego przetwarzania z użyciem arkusza kalkulacyjnego (wyboru pól, które powinny znaleźć się w tym zbiorze powinien dokonywać użytkownik).

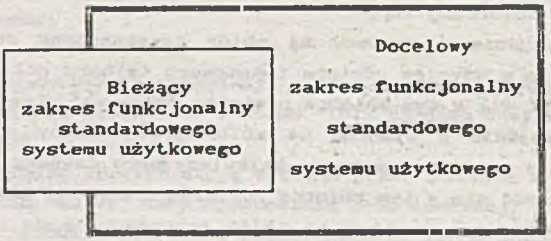
6.16. Wyjście z systemu na zbiór przeznaczony do prezentacji graficznej danych z wykorzystaniem pakietu graficznego (wyboru pól, które powinny znaleźć się w tym zbiorze powinien dokonac użytkownik).

6.17. Możliwość wyemitowania zestawienia, w zaprojektowanej w systemie postaci, ze zbioru archiwalnego, dotyczącego dowolnego z minionych okresów eksploatacji systemu.

6.18. Możliwość zapamiętania zredagowanego zestawienia jako zbioru na nośniku pamięci i nadanie mu nazwy złożonej z nazwy zestawienia i daty emisji (lub innego identyfikatora), dotyczyć to powinno zestawień zbiorczych o znaczeniu "strategicznym".

3. Obszary standaryzacji merytorycznej systemów powielarnych

Standaryzacja systemów użytkowych w przekroju merytorycznym stanowi najtrudniejszą grupę zagadnień. Wiąże się to bowiem z koniecznością penetracji rozległego obszaru oczekiwań i potrzeb klientów - przyszłych użytkowników systemu, czyli przeprowadzenia specjalistycznych badań analitycznych oraz z dysponowaniem ogromnym wyczuciem tendencji rozwojowych danego obszaru problemowego, aby to co obecnie uznane zostało za standard, mogło nim pozostać w możliwie długim okresie czasu, a ponadto aby mogło stanowić podstawę do ustalania nowych standardów (lub uściślania standardów już istniejących), wraz ze zmieniającymi się warunkami i oczekiwaniami (rys. 3). Słowem aby ustalenia standaryzacyjne nie stanowiły blokady przed rozwojem systemów, a wręcz przeciwnie, rozwój taki inspirowały i ułatwiały.



Rys. 3. Rozwój zakresu funkcjonalnego systemu użytkowego

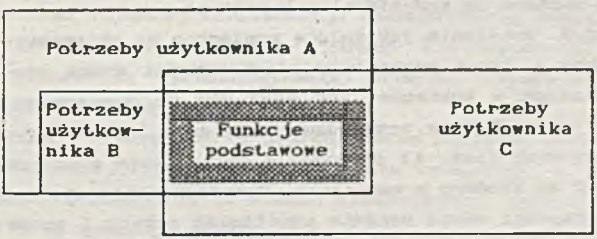
W przekroju merytorycznym standaryzacja może dotyczyć:

1. Zakresu funkcjonalnego systemu.
2. Modularności budowy systemu.
3. Mechanizmów elastyczności umieszczonych w systemie.
4. Cykli przetwarzaniowych obsługiwanych przez system.
5. Wymaganych mechanizmów modernizacji systemu.

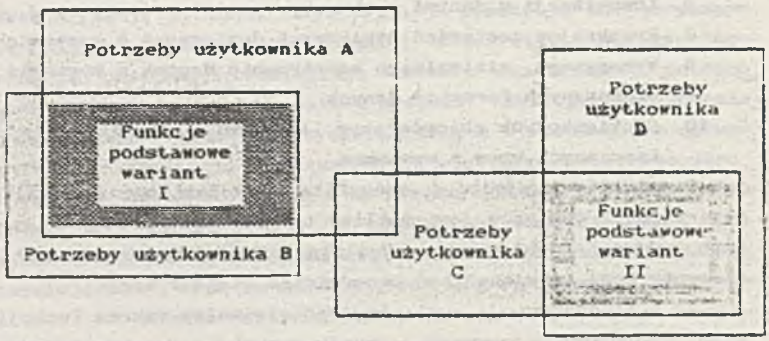
6. Komunikacji z innymi systemami.
7. Przekrojów zestawień wynikowych dostępnych w systemie.
8. Wymaganego, minimalnego asortymentu danych w systemie.
9. Stosowanych formatów danych.
10. Obowiązkowych zbiorów norm, katalogów, cenników itp. dostarczanych wraz z systemem.

Przedmiotowe ujęcie i specyfikacja zawartości dla każdego z wymienionych obszarów jest możliwe tylko w odniesieniu do konkretnego zakresu problemowego. Ogólnie można jedynie wskazać jakie elementy mogą (powinny) być przedmiotem standaryzacji.

Ad.1. Standaryzacja musi określać minimalny zakres funkcji systemu, najbardziej typowych - wynikających z obowiązujących przepisów, dla danej dziedziny przedmiotowej, które bezwzględnie muszą się znaleźć w projektowanym systemie (rys. 4.). Szeroka analiza problemu (dziedziny przedmiotowej) może wskazać, że w tym obszarze może istnieć możliwość (i potrzeba) określenia kilku wariantów różnych zestawów funkcji podstawowych, co wynika z różnorodności obiektów (potrzeby różnych użytkowników są bardzo zróżnicowane, rys. 5.). W takim przypadku istnieje konieczność jednoznacznego wskazania (określenia) klasy użytkowników, dla których ustalony został określony wariant funkcji podstawowych. Sytuacja taka prowadzi do istnienia (konieczności projektowania) kilku standardów systemów (kilku wariantów) dla danej dziedziny przedmiotowej.



Rys. 4. Ustalanie zakresu funkcji podstawowych

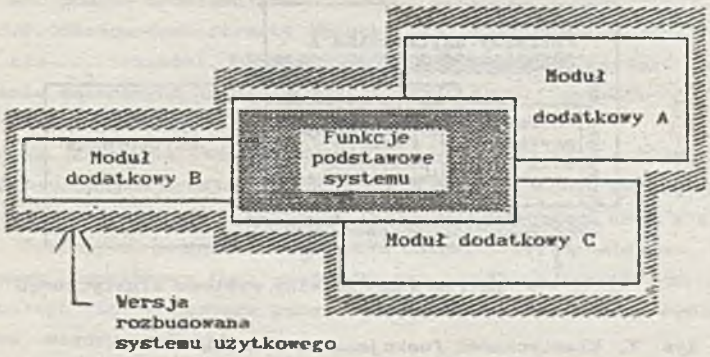


Rys.5. Rozłączne warianty funkcji podstawowych

Ad.2. System standardowy winien mieć budowę modułową, a zatem w ramach standaryzacji należy określić opcjonalne wersje systemu, możliwe do opracowania (w danym obszarze dziedziny przedmiotowej) w postaci zestawu modułów, z których następnie zestawiane będą systemy dla konkretnych użytkowników. Jako moduł podstawowy należy traktować wersję systemu określoną w pkt. 1 (rys. 6.). W przypadku istnienia kilku wariantów modułu podstawowego należy rozważyć celowość ich realizowania w postaci odrębnych systemów lub utworzenie dodatkowych modułów rozszerzających, wyposażonych w dodatkowe mechanizmy elastyczności, które zapewnią szeroką adaptację systemu do indywidualnych potrzeb.

Ad.3. Ustalenie jak dalece rozłączne są potrzeby różnych użytkowników i jakie mechanizmy elastyczności muszą się w związku z tym znaleźć w systemie, aby mógł być on dopasowany do tych potrzeb (rys. 7.). W przypadku określania kilku wariantów funkcji podstawowych (pkt. 1) stosowanie mechanizmów elastyczności należy odnieść do każdego z wariantów. Przewidziane w systemie mechanizmy elastyczności muszą ponadto umożliwiać rozwój i modernizację systemu w czasie jego bieżącej eksploatacji. Mechanizmy te muszą

umożliwiać dokonywanie stosownych zmian bezpośrednio przez użytkownika, bez konieczności ingerowania technologa (programisty). Jako zalecany horyzont czasowy dla tych mechanizmów proponuje się przyjąć 10 lat.

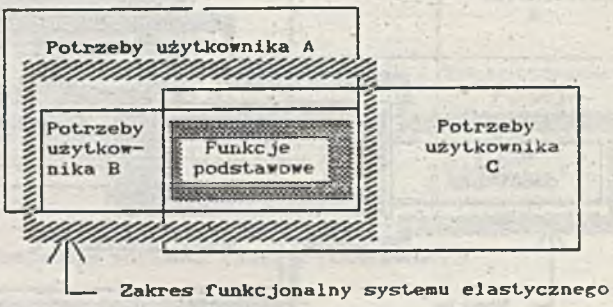


Rys. 6. Modularność budowy systemu standardowego

Ad.4. Określenie cykli przetwarzaniowych, w których system musi pracować, a których istnienie wynika z obowiązujących reguł zarządzania, sprawozdawczości i tradycji. Istotne jest także zwrócenie szczególnej uwagi na cykle o charakterze technologicznym (a nie wyłącznie użytkowym) jak archiwacja i utrzymywanie zbiorów danych. Poza cyklami bieżącej eksploatacji standaryzacją muszą być objęte cykle historyczne (dotyczące danych historycznych), a zwłaszcza długookresowe przechowywanie danych nieaktywnych, regulowane stosownymi przepisami prawnymi.

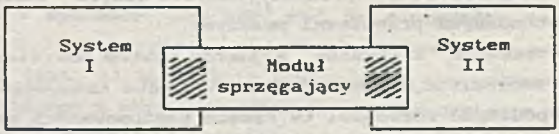
Ad.5. Wytyczenie kierunków, w jakich system powinien podlegać samoczynnej modernizacji. Określenie horyzontu czasowego, w jakim system musi podlegać rozwojowi (w ramach zastosowanych mechanizmów elastyczności) oraz sytuacji granicznych, po przekroczeniu których system wymagać będzie interwencji technologicznej (modyfikacje

programów). Przy ustalaniu założeń projektowych te graniczne parametry winny być jednoznacznie określone i jawne dla użytkowników (podane w dokumentacji eksploatacyjnej systemu).



Rys. 7. Elastyczność funkcjonalna systemu

Ad.6. Ustalenie z jakimi systemami (obszarami problemowymi) oraz programami narzędziowymi, dany system powinien współpracować, ew. jakie moduły powinny być na styku dwu obszarów (mogąc się znaleźć po każdej ze stron) problemowych (rys. 8.). Jaki zakres informacji i w jakich cyklach czasowych powinien być przekazywany z danego systemu do innego oraz jakie dane (ew. kiedy i w jakiej postaci) powinny być otrzymywane z innego systemu (jakie dokumenty wejściowe są zastępcze dla systemu, jeśli powiązanie takie nie istnieje).



Rys. 8. Powiązania między różnymi systemami użytkowymi

Ad.7. Jakie zestawienia są obowiązkowe (wymagane przepisami i zwyczajami), jaki zakres swobody w definiowaniu zestawień należy pozostawić do dyspozycji użytkownika.

Ad.8. Zestaw danych, który musi (z punktu widzenia minimum funkcjonalnego - pkt. 1) znaleźć się w systemie - dla tych danych muszą być podane obowiązujące formaty.

Ad.9. Standardowe formaty danych, dla określonych danych (nie mniej niż .... znaków). Dla danych alfabetycznych określenie zasad ustalania zalecanych formatów danych (pola typu: "nazwisko", "adres", "nazwa ..." itp.).

Ad.10. Z zakresu funkcjonalnego systemu (pkt. 1) i z przyjętego zakresu danych źródłowych wynika jakie dane winny być zgromadzone w zbiorach stałych (quasistałych) dostarczanych wraz z systemem. Dotyczy to wszystkich danych o charakterze normatywnym, katalogowym, cennikowym itp., mających swe źródło w przepisach, zarządzeniach itp. o zasięgu ponadobiektowym (np. branżowe, ogólnokrajowe, międzynarodowe).

4. Standaryzacja dokumentacji eksploatacyjnej dla systemów powielarnych

Dokumentacja eksploatacyjna systemu użytkowego ma za zadanie:

1. Zapoznać użytkownika z zakresem, funkcjonalnością i złożonością systemu, zanim podejmie on decyzję o zakupie i wdrożeniu tego produktu.

2. Stanowić przewodnik dla osób wdrażających system.

3. Być podręcznikiem wspomagającym naukę możliwości, funkcji i zasad obsługi systemu dla jego bezpośrednich użytkowników.

4. Rozstrzygać (pomóc rozwiązać) sytuacje unikalne, które mogą pojawić się w czasie eksploatacji systemu.

5. Stanowić element marketingu, będąc swoją atrakcyjnością i czytelnością dowodem jakości produktu programowego.

Tekst dokumentacji winien być tak opracowany, aby zrozumienie zasad działania systemu nie wymagało konfrontowania szczegółów

opisanych w dokumentacji z funkcjonującym pakietem programów.

Dokumentacja musi zawierać kompendium wiedzy o systemie i być podstawą do całościowego i dokładnego poznania pakietu od strony:

- jego wymagań eksploatacyjnych i (lub) instalacyjnych,
- zakresu realizowanych funkcji,
- szczegółów budowy pakietu, niezbędnych do zrozumienia zasad jego działania,
- zakresu danych gromadzonych i przechowywanych w pakiecie,
- informacji możliwych do uzyskania za pośrednictwem pakietu,
- zakresu prac oraz organizacji procesu wdrożenia pakietu,
- zasad organizacji bieżącej eksploatacji,
- zasad gospodarowania archiwum danych,
- możliwości (zamierzonych kierunków) rozwoju systemu.
- ewentualnych powiązań z innymi pakietami (lub zasad współpracy z innymi pakietami),
- sposobu włączenia systemu do całokształtu działań przetwarzaniowych w obiekcie (w danym obszarze tematycznym).

Założeniem głównym opracowywania gotowych (standardowych) pakietów użytkowych jest ich eksploatacja przez pracowników różnych specjalności zawodowych, którzy nie muszą się legitymować wiedzą informatyczną. Pracownicy ci używają komputera (i pakietu programów) jako narzędzia wspomagającego ich rutynową działalność zawodową. Z tego też powodu pakiet (funkcje pakietu) musi odpowiadać konkretnym potrzebom, związanym z wykonywaniem określonego obszaru prac, a dokumentacja ma pomóc w zrozumieniu działania poszczególnych funkcji i (lub) być zarazem podręcznikiem do nauki obsługi pakietu dla użytkownika. Należy pamiętać o tym, że dla wielu użytkowników kontakt z określonym systemem użytkowym (np. GM, FK itp.) jest pierwszym (a często i jedynym) kontaktem z komputerem.

Dokumentacja nie może więc być pisana w "manierze informatycznej", ale powinna być czytelna i zrozumiała dla osób, których dotychczasowy kontakt z informatyką był minimalny (lub nawet żaden). W tekście należy zatem wyjaśnić jednoznacznie wszystko, co mogłoby być przyczyną nieprawidłowych działań użytkownika wobec

systemu oraz zwrócić uwagę na działania interwencyjne, jakie należy podjąć w przypadku "zgubienia" toku pracy lub nietypowego (nieprawidłowego) działania systemu [ŁUKA88c, ŁUKA88d].

Pożądane jest pokazanie powiązania systemu z praktycznymi potrzebami użytkownika. A zatem w dokumentacji winien się znaleźć model działania danej dziedziny i na jego tle winny być omawiane poszczególne funkcje pakietu, które służą realizacji określonych zadań użytkowych. Model taki mógłby być przedstawiony w postaci schematycznego rysunku.

Prezentacja pakietu powinna więc być dokonana nie od strony jego konstrukcji technologicznej, ale od strony praktycznej funkcjonalności i użyteczności. Wskazane jest ilustrowanie opisu przykładami, stanowiącymi określone zadania użytkowe jakie na bieżąco lub okresowo realizowane są w praktyce zawodowej. Przykładowo dla systemu KADRY takimi zadaniami są: udzielanie pracownikom informacji o dysponowanym przez nich urlopie, wydawanie zaświadczeń o zatrudnieniu, przygotowanie materiałów do przeszeregowań, edycja nowych angaży dla pracowników, w związku z przeszeregowaniem itp.

5. Podsumowanie

Warto zwrócić uwagę, że obecne projektowanie użytkowych systemów powielarnych jest w zasadzie odwróceniem klasycznych reguł nadlerowskiej koncepcji piramidy. Otóż za punkt wyjścia do projektowania przyjmuje się zazwyczaj pewien umowny zakres funkcjonalny uznany arbitralnie za podstawowy (najczęściej na podstawie zgrubnej analizy przeprowadzonej dla niewielkiej liczby kandydatów na użytkowników). Zakres ten stanowi podstawę podjęcia prac realizacyjnych. Następnie "gotowy" system upowszechnia się i wdraża u możliwie wielu użytkowników, a w toku tych wdrożeń i bieżącej pracy systemu (u tych różnych użytkowników) gromadzi się uwagi i oczekiwania, które są kolejno podstawą do opracowania kolejnych wersji systemu lub modułów rozszerzających. Cykl taki powtarza się

wielokrotnie.

Zważywszy jednak, że tworzenie systemów powielarnych to:

- dostępne na rynku produkty dla wielu użytkowników,
- niższe koszty oprogramowania dla pojedynczego użytkownika,
- tendencja do dbałości o jakość produktów,
- szersze możliwości szybkiego rozwoju systemów,
- możliwość wymiany informacji między systemami różnych użytkowników.

a także szereg innych pozytywów, warto chyba zagadnieniom tworzenia systemów powielarnych poświęcić więcej uwagi zarówno od strony teoretycznej (opracowania standardów i metodyki), jak i doskonalenia praktyki (opracowania narzędzi wspomagających tworzenie, prowadzenie atestacji systemów itp.).

Literatura

[DWOR88] J.Dworzecki, B. Łukasik-Makowska, Indywidualne a standardowe systemy informatyczne - przesłanki wyboru strategii tworzenia systemów, w: "Systemy wspomagania decyzji - aspekty metodologiczne" AE Katowice 1988

[KEUN87] P.Keune, A.Bentele, Kompromiss zwischen Standardsoftware und Eigenentwicklung, Online nr 8/1987

[ŁUKA87] B.Łukasik-Makowska, System informatyczny narzędzie użytkownika, INFOKRAK'87 AE i TNOiK Kraków 1987

[ŁUKA88a] B.Łukasik-Makowska, M.Skwarnik, Metodologiczne aspekty tworzenia powielarnych informatycznych systemów zarządzania, INFOGRYF'88 TNOiK i Uniwersytet Szczeciński Kołobrzeg 1988

[ŁUKA88b] B.Łukasik-Makowska, Problemy wdrażania i rozwoju stan-

dardowych systemów informatycznych, SPIS'88 GUS Warszawa 1988

[ŁUKA88c] B.Łukasik-Makowska, Komputerowe stanowisko pracy (1) PT nr 6/88

[ŁUKA88d] B.Łukasik-Makowska, Komputerowe stanowisko pracy (2) PT nr 8/88

[ŁUKA88e] B.Łukasik-Makowska, Smutna konieczność czy słuszny wybór, PT nr 39/88

[DWOC89] M.L. Owoc, Struktury danych w powielarnych systemach informatycznych, II Wiosenna Szkoła Projektowania PTI Szczecin 1989

Jerzy Marcinkiewicz
Instytut Cybernetyki i Informatyki
Uniwersytet Szczecinski

MERISE I REMORA :

METODY BUDOWY SYSTEMOW INFORMACYJNYCH

1. Wprowadzenie

Intensywne prace nad metodami budowy i rozwoju skomputeryzowanych systemow informacyjnych przebiegaja w osrodkach badawczych calego swiata (por. 1,2). Jednym z krajow, w ktorym prowadzi sie intensywne prace w tym zakresie jest Francja. W okresie lat osiemdziesiatych opracowano tam szereg interesujacych rozwiazan - przy czym do najwazniejszych nalezy zaliczyc metody : REMORA, MERISE i AXIAL. W opracowaniu tym zostana przedstawione metody MERISE I REMORA. Metoda AXIAL zostala opracowana przez filie IBM we Francji . Stanowi ona przyklad tradycyjnego podejscia do budowy zastosowan informatycznych.

Wiekszosc obecnie opracowywanych metod budowy i rozwoju systemow informacyjnych zawiera cztery podstawowe skladniki:

- **sposob postepowania** w procesie budowy i rozwoju, zawierajacy strukture procesu (podzial na etapy), opis czynnosci na kazdym etapie (podetapie) ze wskazaniem sposobu ich wykonywania, oraz punkty kontrolne zwiazane z decyzjami zleceniodawcy (uzytkownika);

- **model systemu informacyjnego**, dostarczajacy zestawu pojec do formalnego wyrazenia struktury i funkcjonowania fragmentu rzeczywistosci obslugiwanej przez system informacyjny, jak rowniez projekowanej struktury i funkcjonowania samego systemu informacyjnego;

- **jezyk opisu systemu informacyjnego**, umozliwiajacy sformalizowany opis pojec wystepujacych w modelu systemu informacyjnego, interpretowalny komputerowo;

- narzedzia komputerowego wspomagania budowy i rozwoju systemow informacyjnych. Wspomaganie to moze obejmowac skomputeryzowany opis projektowanego systemu (innymi slowy dokumentowanie prac), automatyzacje czynnosci projektowych (programowych) na poszczegolnych etapach budowy systemu , zarzadzanie organizacja (harmonogramem) prac nad systemem oraz przydzielonymi zasobami .

W ukladzie tych czterech skladnikow zostana ponizej przedstawione metody: MERISE i REMORA.

2.Metoda MERISE.

2.1 Podstawowe cechy metody.

Metoda ta zostala opracowana na zlecenie Ministerstwa Przemyslu Francji z mysla o szerokim, praktycznym wykorzystywaniu w osrodkach projektowo-programowych. I faktycznie - obecnie znaczny procent prac nad zastosowaniami (szacuje sie nawet do 50%) jest realizowany w mniejszym lub wiekszym stopniu w oparciu o te metode.

MERISE przyjmuje zalozenie, ze w kazdej instytucji traktowanej jako system, mozna wyroznic trzy podsystemy: podsystem wykonawczy, podsystem zarzadzajacy oraz podsystem informacyjny. Zadaniem tego ostatniego jest:

-przyjmowanie z podsystemu wykonawczego danych, opisujacych zachodzace tam zjawiska,

- przyjmowanie informacji decyzyjnych z podsystemu zarzadzajacego, dotyczacych funkcjonowania podsystemu wykonawczego jak i samego systemu informacyjnego,

- przygotowanie i przesylanie danych wymaganych przez podsystem zarzadzajacy,

- przetwarzanie informacji decyzyjnych naplywajacych z podsystemu zarzadzania - na elementarne dane decyzyjne dotyczace dzialania podsystemu wykonawczego oraz przekazanie ich do odbiorcow.

W metodzie MERISE zaklada sie, ze proces budowy i rozwoju systemu odbywa sie w trzech cyklach:

- zycia systemu informacyjnego.
- abstrakcji,
- decyzji.

Metoda obejmuje pelny **cykl zycia systemu informacyjnego**, a w tym:

- projektowanie systemu , ktorego wynikiem jest szczegolowy opis rozwiazan funkcjonalnych, informacyjnych i technicznych,

- realizacje, polegajaca na opracowaniu (wyprodukowaniu) programow i instrukcji uzytkowania zaprojektowanych rozwiazan,

- utrzymywanie systemu, ktorego celem jest ciagla adaptacja systemu do zmieniajacego sie otoczenia.

Caly cykl zycia jest tu podzielony na etapy, a w ramach nich wyroznia sie zadania przewidziane do wykonania przez zespol projektowo-programowy.

Cykl abstrakcji dotyczy przechodzenia w procesie budowy systemu informacyjnego od sformulowania celow, przez opis ogolny, po szczegoly realizacyjne. Uwzglednianie tego wymiaru w procesie budowy jest niezbedne przy rozwiazywaniu duzych problemow, na przyklad system informacyjny przedsiebiorstwa. Praktycznie dokonuje sie tego poprzez wyodrebnienie poziomow abstrakcji opisu - poczynajac od poziomu najbardziej ogolnego. MERISE wyroznia trzy poziomy abstrakcji:

a) **Poziom konceptualny:** umozliwia on dokonanie opisu klas obiektow i regul ich zachowania sie, ktore wydaja sie istotne dla realizacji celow instytucji, zdefiniowanych przez decydentow. Na poziomie tym system jest opisywany w ujeciu statycznym i dynamicznym. Abstrahuje sie tu od jakichkolwiek ograniczen technicznych przyszlego rozwiazania;

b) **Poziom logiczny (organizacyjny** - z punktu widzenia procedur przetwarzania): umozliwia on uwzglednienie natury zasobow, w oparciu o ktore system bedzie realizowany. Zasobami tymi sa: ludzie, system komputerowy, oprogramowanie narzedziowe (na przyklad system zarzadzania baza danych).

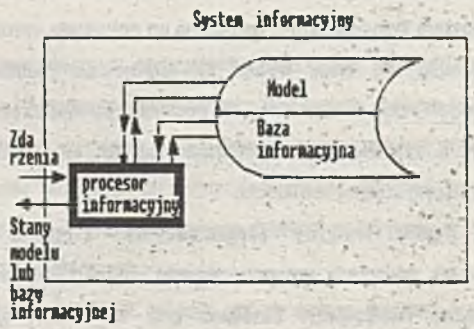
c) Poziom fizyczny (operacyjny - z punktu widzenia procedur przetwarzania): jest on wynikiem decyzji projektowo-programowych, okreslajacych fizyczna realizacje struktur danych (aspekt statyczny) oraz programow (aspekt dynamiczny).

Cykl decyzji obejmuje zespol wyborow, ktore winien byc dokonywany przez zleceniodawce (uzytkownika) i kierownictwo prac projektowo-programowych - w ciagu cyklu zycia systemu informacyjnego. Decyzje te dotycza struktury systemu, jego funkcjonowania, sposobu jego wdrazania, zasad eksploatacji i konserwacji.

Autorzy metody wyraznie podkreslaja potrzebe jednoczesnego ujmowania aspektu statycznego i dynamicznego systemu informacyjnego - we wszystkich fazach budowy systemu. Prezentacje szczegolowych rozwiazan w metodzie rozpoczniemy od przyjetego modelu systemu informacyjnego.

2.2 Model systemu informacyjnego.

Metoda zaklada, ze na architekture wewnetrzna systemu informacyjnego skladaja sie trzy elementy: model systemu, baza informacyjna oraz procesor informacyjny (por. rys. 1). Baza informacyjna zawiera dane rzeczywiste, przy czym ich sposob prezentacji jest okreslony przez model systemu. Z kolei procesor informacyjny przetwarza dane o rzeczywistosci, naplywajace do procesora w postaci zdarzen.



Rys. 1. Ogolny model systemu informacyjnego wg MERISE.

Jak juz wspomniano wyzej, model systemu metody MERISE obejmuje pojecia umozliwiajace formalna reprezentacje aspektu dynamicznego i statycznego systemu informacyjnego.

Do modelowania **aspektu dynamicznego** wykorzystuje sie trzy podstawowe pojecia:

- **zdarzenie,** bedace stwierdzeniem zajscia faktu w obszarze opisywanym przez system informacyjny, w jego otoczeniu lub w samym systemie informacyjnym;

- **operacja,** definiowana jako jedna lub zespol akcji realizowanych przez procesor informacyjny - w reakcji na okreslone zdarzenie;

- **synchronizacja ,** definiowana jako warunek wstepny, ktory musi byc spelniony zeby operacja mogla byc zainicjowana.

Wystapienie kazdego z wymienionych pojec nalezy do okreslonego typu, a wiec do typu zdarzenia, operacji lub synchronizacji. Kazdy typ posiada wyrozniajace go charakterystyki.

Typ zdarzenia jest definiowany za pomoca nastepujacych charakterystyk elementarnych:

- typu komendy, okreslajacej akcje powodowane typem zdarzenia,

- komunikatu, ktory zawiera opis poszczegolnych typow wlasciwosci (cech) powiazanych ze zdarzeniem.

Poza powyzszymi, typy zdarzen posiadaja rowniez charakterystyki generalne dotyczace zbioru wystapien tego typu zdarzenia:

- **pojemnosc** typu zdarzenia : jest to maksymalna liczba wystapien typu operacji ktore procesor informacyjny moze zaakceptowac;

- **czestotliwosc** pojawiania sie wystapien typu zdarzenia.

Typ operacji jest definiowany przez:

- typy zdarzen wewnetrznych lub wewnetrznych, ktore sa emitowane w wyniku wykonania tej operacji. Wygenerowanie przez operacje zdarzenia podlegaja regulom

emisji. Sa to stwierdzenia logiczne ktorych prawdziwosc jest badana po wykonaniu operacji;

- **akcje,** ktora sie sklada z elementarnych dzialan (dołaczenie, usuniecie, wyszukiwanie) na bazie informacyjnej. Akcja moze byc wyrazona za pomoca elementarnych struktur programowania strukturalnego;

- **trwanie operacji,** wyrazone w przyjetej jednostce czasu.

Z kolei typ **synchronizacji,** okreslajacy warunki uruchomienia jednej lub kilku operacji jest charakteryzowany przez:

- **typy zdarzen** wewnetrznych i zewnetrznych, uczestniczacych w typie synchronizacji;

- **stwierdzenie logiczne,** odnoszace sie do typow zdarzen bioracych udział w synchronizacji; przybycie zdarzenia moze spowodowac fakt, ze stwierdzenie staje sie prawda - wowczas synchronizacja jest uruchamiana;

- **warunki lokalne,** powiazane ze stwierdzeniem logicznym; pozwalaja one okreslic, ktore z wystapien zdarzenia obecnych w procesorze informacyjnym powinny wziasc udział w wystapieniu (realizacji) synchronizacji;

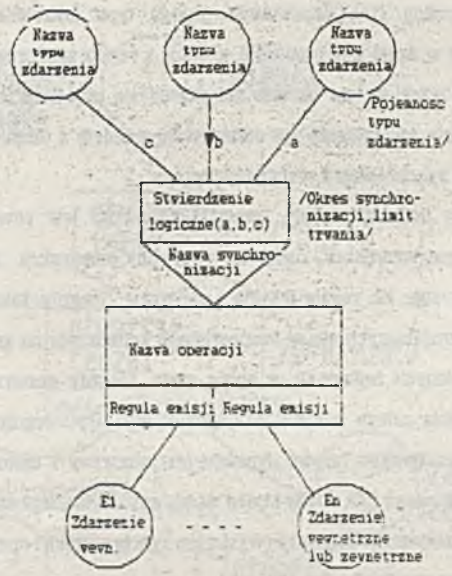
- **limit trwania** synchronizacji: typ synchronizacji moze miec okreslony limit czasu oczekiwania - to znaczy jesli podczas odcinka czasu zainicjowanego przybyciem pierwszego zdarzenia nalezacego do typu synchronizacji, stwierdzenie logiczne nie stanie sie prawda, to synchronizacja jest calkowicie anulowana.

Powyzsze pojecie posiadaja w metodzie MERISE reprezentacje graficzna, na przyklad typ zdarzenia jest wyrazony okregiem. Opis dynamiki systemu stanowi w efekcie graf zorientowany, ktorego wezlami sa:

- zdarzenia,

- oraz operacje powiazane z synchronizacja.

Sposob graficznej prezentacji pojec opisujacych dynamike w metodzie MERISE prezentuje rys. 2.



Rys. 2. Sposob graficznej prezentacji dynamiki w metodzie MERISE

Opisanie funkcjonowania systemu informacyjnego za pomoca modelu dynamiki MERISE wymaga wprowadzenia kilku dodatkowych, niezbednych pojec. **Stan modelu** jest okreslany za pomoca zetonow przyporzadkowanych poszczegolnym typom zdarzen. **Zeton** obrazuje wystapienie typu zdarzenia. Przyklad modelu dynamiki, ze wskazaniem zetonow powiazanych z typami zdarzen, przedstawia rys. 3. Jak latwo zauwazyc ten sposob prezentacji dynamiki jest zaczerpniety bezposrednio z techniki sieci Petriego /2/.

Z kolei **udział typu zdarzenia w typie synchronizacji** - to liczba wystapien danego typu zdarzenia w realizacji danego typu synchronizacji. **Liczba kardynalna typu zdarzenia w** typie operacji to liczba identycznych wystapien typu zdarzenia, emitowanych przez dana operacje.

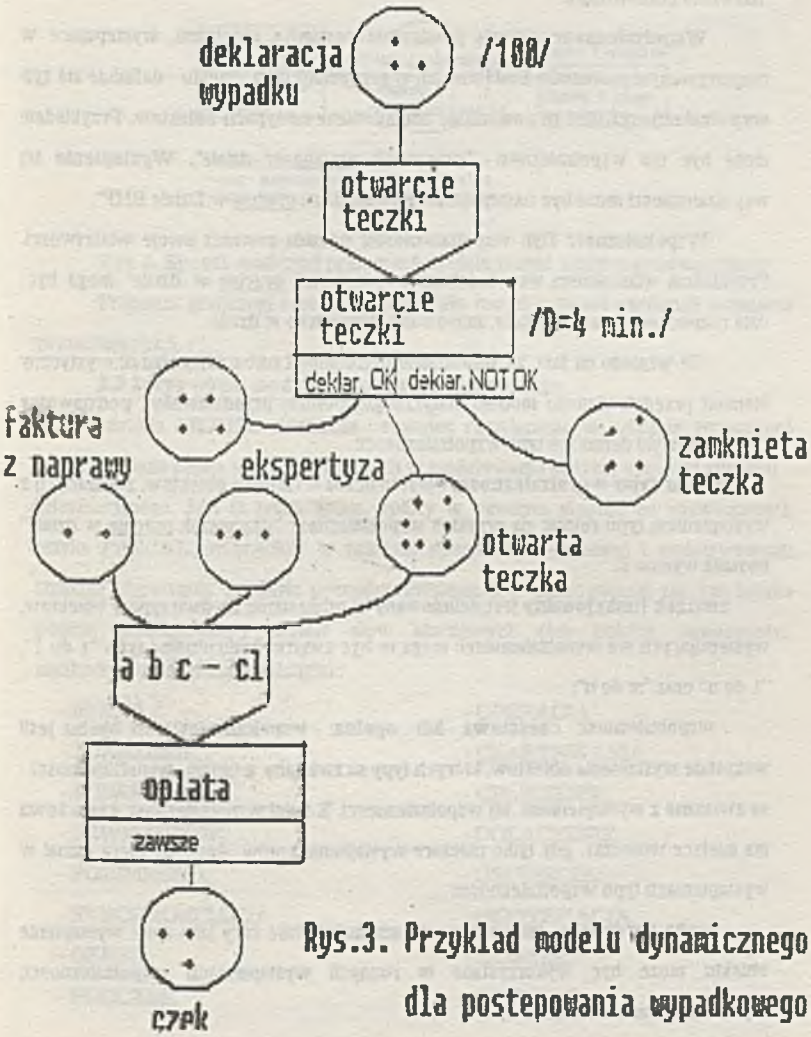
Limit czasu udzialu typu zdarzenia w typie **synchronizacji** - okresla sie jako maksymalny czas oczekiwania danego typu zdarzenia na inne zdarzenia wystepujace w danej synchronizacji, zgodnie z regula synchronizacji (stwierdzeniem logicznym). Jezeli w tym okresie nie pojawi sie inne zdarzenie zwiazane z dana synchronizacja, to zdarzenie jest definitywnie usuniete z udzialu w synchronizacji, a wiec proces synchronizacji zostaje zawieszony.

Przy definicji modelu dynamiki konieczne jest rowniez stosowanie tak zwanych regul weryfikacji, stanowiacych reguly konstrukcji modelu. W metodzie MERISE stosuje sie reguly lokalne i generalne. **Reguly lokalne** odnosza sie do elementow znajdujacych sie w bezposrednim sasiedztwie na grafie i sa definiowane jedynie na danych zawartych w opisie grafu. **Reguly generalne** odnosza sie do funkcjonowania calego modelu. Przykladem tego typu reguly moze byc definicja **modelu wlasciwego:** "model dynamiki jest wlasciwy i umozliwiajacy powrot do stanu poczatkowego EO, wtedy i tylko wtedy gdy dla kazdego stanu E, poczynajac od EO, istnieje sekwencja aktywacji (wystapien synchronizacji i operacji i zdarzen), ktora prowadzi do stanu EO."

Metoda opisu **aspektu statycznego** w MERISE opiera sie na modelu ER "Entity- Relationship" (Obiekt-Wspolzaleznosc), zaproponowanym przez P. P. Chena /4/ i obecnie powszechnie wykorzystywanym i rozwijanym w metodach budowy baz danych i systemow informacyjnych.

Punktem wyjscia przy definiowaniu wykorzystywanych tu pojec jest **obiekt.** Jest abstrakcyjny lub fizyczny element analizowanego problemu. Obiekty, ktore maja wspolne wlasciwosci (na przyklad pracownicy przedsiebiorstwa) naleza do **typu obiektu** , na przyklad "pracownik". Typ obiektu ma swoje wystapienia, ktore posiadaja zawsze ten sam zestaw wlasciwosci, a jednoczesnie musza byc niepowtarzalne ze wzgledu na przyjmowane wartosci wlasciwosci.

Typ **wlasciwosci** jest uogolnionym wyrazeniem tej samej wlasciwosci, posiadanej przez wszystkie wystapienia danego typu obiektu (na przyklad



Rys.3. Przyklad modelu dynamicznego dla postepowania wypadkowego.

nazwisko pracownika).

Wspolzaleznosc okresla powiazanie pomiedzy obiektami, wystepujace w rozpatrywanym problemie. Podobnie jak w przypadku typu obiektu - definiuje sie **typ wspolzaleznosci.** Jest to powiazanie zdefiniowane na typach obiektow. Przykladem moze byc typ wspolzaleznosci "pracownik pracuje w dziale". **Wystapienie** tej wspolzaleznosci moze byc nastepujace: "Kowalski Jan pracuje w Dziale BHP".

Wspolzaleznosc (typ wspolzaleznosci) posiada rowniez swoje wlasciwosci. Przykladem wlasciwosci wspolzaleznosci "pracownik pracuje w dziale" moga byc: data rozpoczecia pracy w dziale, zajmowane stanowisko w dziale.

Ze wzgledu na fakt, ze wspolzaleznosc stanowi najbardziej charakterystyczny element przedstawianego modelu statycznego, ponizej przedstawimy podstawowe charakterystyki definiujace typy wspolzaleznosci:

- **wymiar typu wspolzaleznosci** - jest to liczba wystapien obiektow, związanych z wystapieniem typu relacji; na przykład wspolzaleznosc "pracownik pracuje w dziale" posiada wymiar 2;

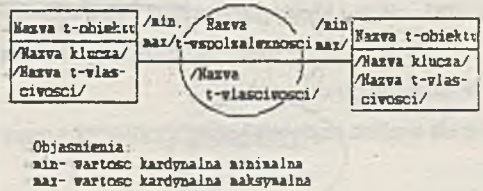
- **zwiazek funkcjonalny** jest definiowany w odniesieniu do dwu typow obiektow, wystepujacych we wspolzaleznosci; moga to byc związki funkcjonalne typu "1 do 1", "1 do n" oraz "m do n";

- wspolzaleznosc **czesciowa** lub **ogolna:** wspolzaleznosc jest ogolna jesli wszystkie wystapienia obiektow, ktorych typy sa związane z typem wspolzaleznosci - sa związane z wystapieniami tej wspolzaleznosci. Z kolei wspolzaleznosc **czesciowa** ma miejsce wowczas, gdy tylko niektore wystapienia typow obiektow biora udział w wystapieniach typu wspolzaleznosci;

- **liczba kardynalna minimalna** - to minimalna ilosc razy, jaka dane wystapienie obiektu moze byc wykorzystane w roznych wystapieniach wspolzaleznosci; najczestsze wartosci to 0 lub 1;

- **liczba kardynalna maksymalna** - to maksymalna ilosc razy jaka dane wystapienie obiektu moze byc wykorzystane w typie wspolzaleznosci.

Metoda MERISE proponuje graficzny sposob prezentacji pojec opisujacych model statyki. Wykorzystywane schematy graficzne przedstawiono na rys. 4.



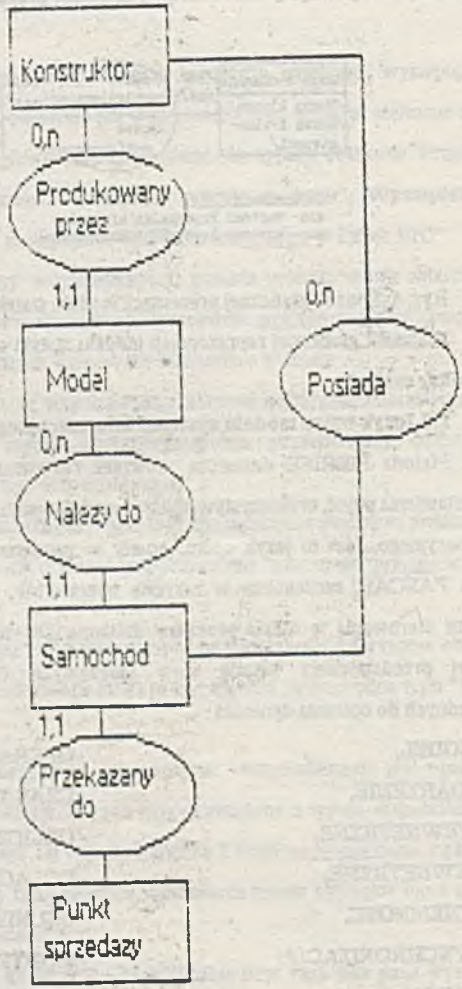
Rys. 4. Sposob graficznej prezentacji modelu statyki systemu informacyjnego

Fragment graficznej reprezentacji modelu statyki systemu ewidencji pojazdow prezentuje rys.5.

2.3 Jezyk opisu modelu systemu informacyjnego.

Metoda MERISE dostarcza rowniez rozwiazania w zakresie formalnego przedstawienia pojec, wykorzystywanych w modelowaniu statyki i dynamiki systemu informacyjnego. Jest to jezyk opisu, oparty w pewnym stopniu na rozwiazaniach jezyka PASCAL, szczegolnie w zakresie operatorow, oznaczen i podstawowych struktur sterowania w opisie procedur dzialania. Dla zilustrowania zakresu jezyka ponizej przedstawiamy zestaw slow kluczowych (ich polskie tlumaczenia), niezbednych do opisania dynamiki :

- MODEL,
- ZDARZENIE,
- WEWNETRZNE,
- ZEWNETRZNE,
- POJEMNOSC,
- SYNCHRONIZACJA
- OKRES,
- PODCZAS,
- OPERACJA
- CZAS TRWANIA,
- OBLICZENIE,
- DOLACZENIE,
- USUNIECIE,
- MODYFIKACJA,
- ZAWSZE,
- TO (WIEC),



Rys. 5. Fragment graficznej prezentacji modelu statyki systemu

- JESLI,
- JESLI NIE.
- Z,
-DLA KAZDEGO.

Ponizej przedstawiamy wybrane przyklady opisu typu zdarzenia i synchronizacji za pomoca jezyka MERISE.

MODEL : Biuro rejestracji pojazdow

ZDARZENIE zewn-1 ZEWNETRZNE : Pozwolenie dzialania dla producenta c

SYNCHRONIZACJA : Poczatek dzialalnosci producenta

JESLI zewn-1 I zewn-2

Z c.zewn-1= c.zewn-2

OPERACJA 01 DOLACZENIE (Producent c jest w stanie aktywnosci)

JESLI Producent c

WTEDY wewn-1

INACZEJ wewn-2

Z kolei do formalnego opisania aspektu statycznego systemu, wykorzystuje sie nastepujace slowa kluczowe:

- MODEL,
- WYMIAR,
- TYP-OBIEKTU
- KOLEKCJA.
- IDENTYFIKATOR,
- LICZBA-KARDYNALNA.
- OPIS,
- FUNKCJA,
- TYP-RELACJI,
- NA.

Poszczegolne pojecia sa przedstawiane w opisie w nastepujacej kolejnosci : identyfikacja modelu, obiekty, wspolzaleznosci miedzy nimi. Ponizej przedstawiamy przykladowy opis wspolzaleznosci produkowany przez, oraz zwiazanych z nia obiektow:

MODEL Biuro rejestracji pojazdow

TYP-OBIEKTU Producent

IDENTYFIKATOR nr producenta

OPIS nr producenta

stan aktywnosci

TYP-OBIEKTU model pojazdu

IDENTYFIKATOR nr modelu

OPIS nr modelu

zuzycie paliwa

TYP-RELACJI produkowany przez

WYMIAR 2

KOLEKCJA producent

model pojazdu

LICZBA-KARDYNALNA producent 0,n

model-pojazdu 1,1

Jak wynika to z powyzszego przykladu, slowo kluczowe OPIS sluzy po przedstawienia zestawu wlasciwosci (atrybutow) poszczegolnych obiektow.

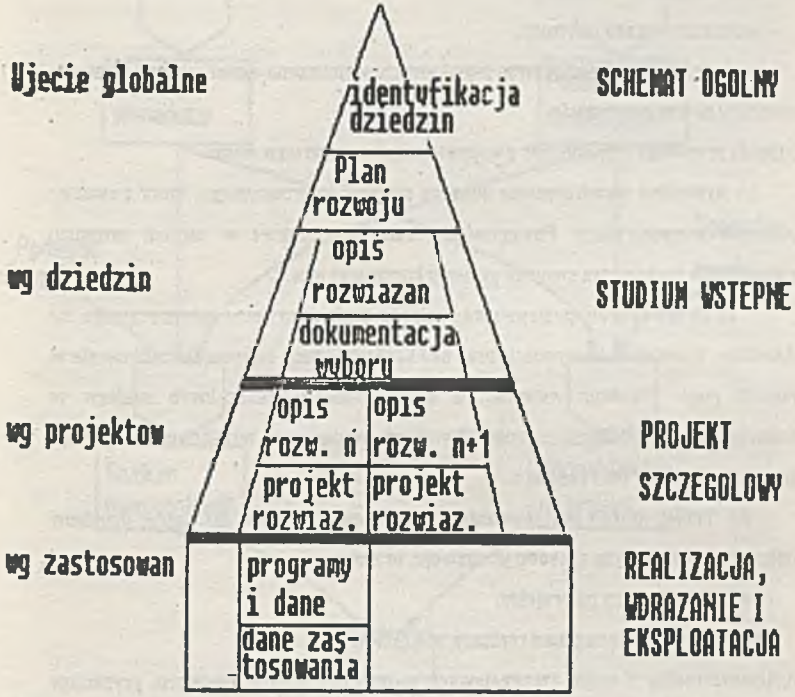
Zaproponowany przez tworcow MERISE jezyk jest pomyslany jako potencjalne narzedzie formalnego opisu modelu konceptuálnego systemu informacyjnego - dla potrzeb komputerowych narzedzi metody MERISE.

2.4 Sposoby postepowania projektowego

Podobnie jak inne metody budowy i rozwoju systemow informzacyjnych, MERISE proponuje konstruktorom i uzytkownikom skomputeryzowanego systemu:

- podzial cyklu zycia systemu na etapy i podetapy,
- konkretne zadania do wykonana, zwiazane z etapem (podetapem),
- glowne, typowe scenariusze postepowania,
- rozmieszczenie glownych decyzji zwiazanych z budowa i eksploatacja systemu.

Podzial procesu rozwoju na etapy i uzyskiwane wyniki przedstawia rys. 6. Ponizej zostana przedstawione poszczegolne etapy.



Rys. 6. Etapy w rozwoju systemu informacyjnego wg MERISE

Schemat ogolny - a dokladniej jego sporzadzenie stanowi pierwszy etap prac. Jego celem jest okreslenie zakresu prac oraz srodkow niezbednych do realizacji systemu, uwzgledniajac:

- komputeryzowany obszar
- calokształt potrzeb instytucji,
- kolejnosc realizacji zastosowan czastkowych, wymuszona ograniczonymi srodkami instytucji na komputeryzacje.

Metoda proponuje trzy sposoby postepowania w ramach tego etapu:

1) Arbitralne wyodrebnienie obszaru systemu informacyjnego, który powinien podlegac komputeryzacji. Postepowanie takie jest mozliwe w ramach instytucji swiadomych niedomagan swojego systemu informacyjnego.

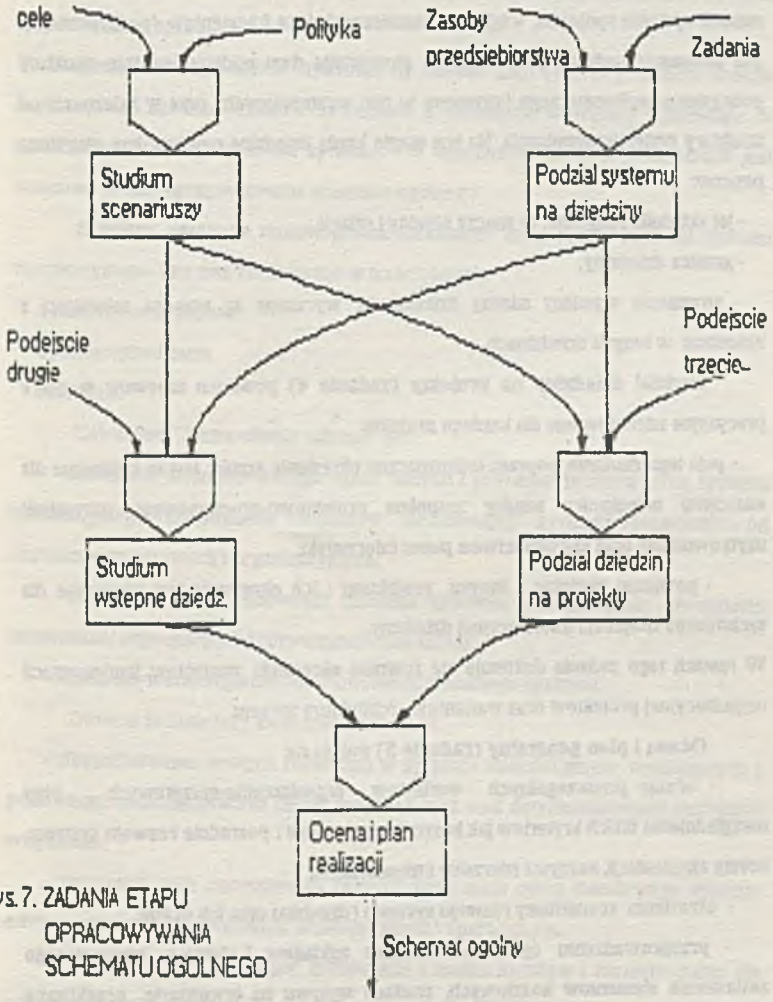
2) Drugi sposob postepowania zaklada podział systemu informacyjnego na dziedziny, a nastepnie kontynuacje prac nad kazda dziedzina odbywa sie rownolegle w ramach etapu "Studium wstepne". Z kolei wyniki poszczególnych studiów sa konsolidowane i podlegaja ocenie. Wynikiem ostatecznym jest **schemat ogolny**, zawierajacy ogolny plan realizacji.

3) Trzeci sposob postepowania jest podobny, z tym ze dla kazdej dziedziny dokonuje sie okreslenia sposobu jej rozwoju, to znaczy:

- podziału dziedziny na projekty,
- wstepnego harmonogramu realizacji projektów.

Kolejnosc realizacji zadan szczegolowych w drugim i trzecim podejsciu prezentuje rys. 7. Ponizej przedstawiamy krotkie charakterystyki poszczególnych zadan szczegolowych.

Studium scenariuszy (zadanie 1) dotyczy przewidywanych kierunków rozwoju jednostki organizacyjnej, dla której przewiduje sie modernizacje systemu informacyjnego. Rózne typy scenariuszy strategicznych powinny zapewnic okreslenie przyszlych celów i zadan instytucji oraz wynikajacych z tego wymagan (potrzeb) informacyjnych.



Rys. 7. ZADANIA ETAPU OPRACOWYWANIA SCHEMATU OGÓLNEGO

Podział systemu na dziedziny (zadanie 2) umożliwia na wyodrębnienie w ramach systemu spójnych, względnie niezależnych jego fragmentów (podsystemów). Dla dokonania podziału proponuje się stosowanie dwu podejść: według struktury podsystemu wykonawczego (strumieni w nim występujących) oraz w zależności od struktury systemu zarządzania. Na tym etapie każda dziedzina powinna być określona poprzez:

- jej składniki statyczne, to znaczy obiekty i relacje,
- granice dziedziny,
- strumienie wymiany między dziedzinami, wyrażone za pomocą zależności z obiektami w innych dziedzinach.

Podział dziedziny na projekty (zadanie 4) powinien zapewnić w miarę precyzyjne zdefiniowanie dla każdego projektu:

- pola jego działania, poprzez jednoznaczne określenie granic; jest to konieczne dla właściwej współpracy między zespołem projektowo-programowym, przyszłymi użytkownikami oraz kierownictwem pionu informatyki;
- powiązań projektu z innymi projektami ; ich określenie jest konieczne dla zachowania spójności informacyjnej dziedziny.

W ramach tego zadania dokonuje się również określenia wariantów implementacji organizacyjnej projektów oraz wariantów architektury sprzętu.

Ocena i plan generalny (zadanie 5) polega na:

- ocenie poszczególnych wariantów organizacyjno-sprzętowych , przy uwzględnieniu takich kryteriów jak koszty bezpośrednie i pośrednie rozwoju systemu, koszty eksploatacji, korzyści mierzalne i niemierzalne;
- określeniu scenariuszy rozwoju systemu (dziedzin) oraz ich ocenie,
- przeprowadzeniu ogólnego rachunku nakładów i efektów, zawierającego zestawienie elementów kosztowych, studium wpływu na organizację, oczekiwane efekty wymierne i niewymierne oraz studium wykonalności;

- sporządzeniu planu realizacji projektów, a w tym: planu wykorzystania służby informatycznej, zlecania prac na zewnątrz, przygotowania kadry informatycznej oraz wyposażania w sprzęt.

W drugim sposobie podejścia wykonuje się zadania 1,2,3 i 5 ; w podejściu trzecim wykonuje się zadania: 1,2,3,4 i 5. Zadanie 3 (**Studium wstępne**) stanowiące w zasadzie kolejny etap rozwoju systemu - w wymienionych dwu podejściach jest włączony w fazę opracowywania schematu ogólnego.

Studium wstępne stanowi jednak zasadniczo drugi etap w rozwoju systemu informacyjnego. Jest ono realizowane w trzech fazach:

1) Zestawienie faktów.

2) Konceptualizacja.

3) Ocena.

Celem fazy "**Zestawienie faktów**" jest:

- dokonanie sformalizowanego opisu danych i procedur przetwarzania systemu informacyjnego - na poziomie elementów "niezmiennych" systemu, niezależnych od rozwiązań sprzętowych i organizacyjnych;

- określenie nieprawidłowości działania systemu w odniesieniu do systemu zarządzania, organizacji jak i rozwiązań technicznych;

- określenie wąskich gardeł funkcjonowania aktualnego systemu.

Główne zadania fazy **konceptualizacji** to:

- sformalizowanie nowych rozwiązań w systemie informacyjnym, wynikających z ponownego przeanalizowania celów instytucji oraz wad dotychczasowych rozwiązań w systemie,

- wyrażenie tych rozwiązań na różnych poziomach opisu docelowego systemu informacyjnego: konceptualnym, organizacyjnym i operacyjnym.

Zadaniem fazy **Oceny** jest zestawienie i analiza kosztów i harmonogramu dla poszczególnych wariantów rozwiązania.

Na fazę "**zestawienie faktów**" składa się realizacja następujących zadań:

1) Zdefiniowanie zadan nowego systemu informacyjnego.

2) Modelowanie systemu informacyjnego z punktu widzenia przetwarzania danych

3) Modelowanie systemu informacyjnego z punktu widzenia danych.

4) Synteza i hierarchizacja procedur przetwarzania oraz wybor reprezentatywnego podzespolu procedur w rozwiazaniu.

5) Zdefiniowanie modelu danych dla tworzonego rozwiązania.

6) Aprobowanie diagnozy sytuacji przez dyrekcje.

W fazie **konceptualizacji** dokonuje sie realizacji nastepujacych zadan:

7) Budowa nowego modelu przetwarzania w ujeciu konceptualnym.

8) Budowa modelu przetwarzania w ujeciu organizacyjnym.

9) Weryfikacja konceptualnego modelu danych.

10) Przeksztalcenie konceptualnego modelu danych na ogolny model logiczny (sieciowy).

11) Oszacowanie wystapien elementow modelu danych oraz czestotliwosci realizacji procedur.

12) Budowa fizycznego modelu danych.

13) Budowa operacyjnego modelu przetwarzania (projekty ekranow, zestawien wynikowych oraz grafow ich powiazan w wybranych procedurach).

14) Aprobowanie modelu systemu przez dyrekcje.

Ostatnia faza Studium wstepnego - **ocena** - jest realizowana poprzez nastepujace zadania:

15) Opracowanie scenariusza realizacji rozwiazania.

16) Wycena kosztow.

17) Ocena rozwiazania.

18) Aprobowanie oceny przez dyrekcje.

19) Przygotowanie dokumentu zawierajacego synteze proponowanego rozwiazania.

Etap **Projektowania szczegolowego** jest realizowany dla poszczegolnych projektow. Jego celem jest takie uszczegolowienie modeli: **konceptualnego i organizacyjnego przetwarzania**, zeby uzyskac:

- projekty kolekcji danych, zwiazanych ze zdarzeniami (projekty zestawien wynikowych, ekranow, itp.),
- algorytmy kontroli danych i innych procedur przetwarzania,
- modele zewnetrzne danych, zwiazane z poszczegolnymi procedurami.

Rys. 8 prezentuje zadania, skladajace sie na ten etap, w postaci organigramu.

Etap **Realizacji** ma na celu wytworzenie zespolu programow, realizujacych procedury zdefiniowane w poprzednich etapach. Przebiega on w dwu podetapach:

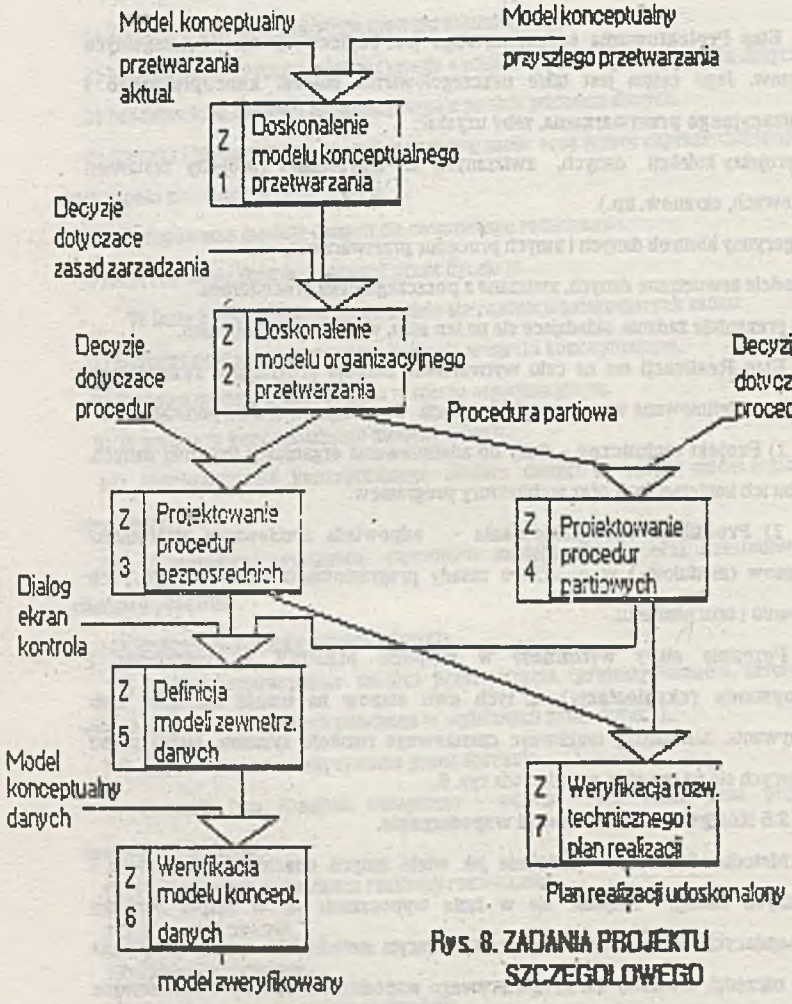
1) **Projekt techniczny** - sluzy do zdefiniowania organizacji fizycznej danych sposobu ich implementacji oraz architektury programow.

2) **Produkcja oprogramowania** - odpowiada zbudowaniu algorytmow programow (modulow) w oparciu o zasady programowania strukturalnego , ich kodowaniu i uruchamianiu.

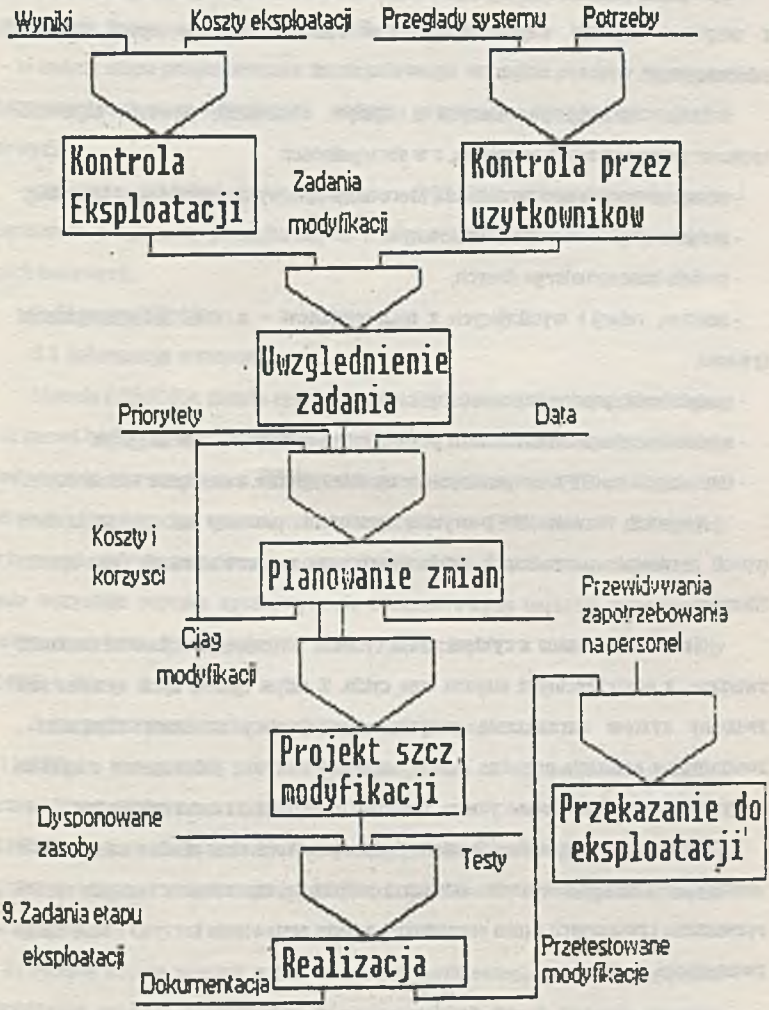
Pozostale etapy wyrozniane w metodzie MERISE to **wdrazanie i utrzymywanie (eksploatacja)**. Z tych dwu etapow na uwage zasluguje etap utrzymywania, zakladajacy mozliwosc czastkowego rozwoju systemu. Uklad zadan skladajacych sie na ten etap, przedstawia rys. 9.

2.6 Komputerowe narzedzia wspomagania.

Metoda MERISE - podobnie jak wiele innych obecnie uzytkowanych i rozwijanych metod, znajduje sie w fazie wyposazania jej w zespol narzedzi wspomagajacych. W opracowaniu przedstawiajacym metode /5/ , autorzy proponuja zestaw narzedzi, wlasciwy dla komputerowego wspomagania metody. Proponowane narzedzia sa przedstawione w ukladzie trzech cykli rozwoju systemu: **abstrakcji, zycia systemu** oraz **decyzyjnego**.



Rys. 8. ZADANIA PROJEKTU SZCZEGÓŁOWEGO



Rys. 9. Zadania etapu eksploatacji

- narzedzie typu edytor tekstu - niezbedny do przygotowania roznego typu opisow i dokumentacji,

- oprogramowanie umozliwiajace kommikowame sie roznych grup uzytkownikow, z zespolem narzedzi wspomagajacych, w roznych jezykach opisu systemu informacyjnego.

Grupa narzedzi powiazanych z **cyklem abstrakcji**, powinna zapewnic spojnosc proponowanych rozwiazan, a w szczegolnosci:

- niesprzecznosc regul zarzadzania, ktore moga byc wyrazone w logice formalnej,

- zestawu obiektow i wspolzaleznosci,

- modelu konceptualnego danych,

- zdarzen, relacji i wynikajacych z nich rezultatow - a wiec modelu dynamiki systemu,

- modelu konceptualnego procedur przetwarzania,

- modeli danych i przetwarzania na poziomie organizacyjnym i operacyjnym,

- tlumaczenia modeli konceptualnych na modele logiczne, a nastepnie standardowe.

Narzedzia rozwiazujace powyzsze zagadnienia powinny sie opierac na dwu typach rozwiazan: **narzedziach graficznych** oraz na rozwiazaniach typu system Ekspert.

Narzedzia zwiazane z **cyklem zycia** systemu wspomagaja czynnosci (zadania) zwiazane z poszczegolnymi etapami tego cyklu. Z całym cyklem zycia systemu jest zwiazany **system zarzadzania projektem**, obejmujacy strukture czynnosci , zwiazanych z realizacja projektu. Jest to narzedzie zwiazane jednoczesnie z **cyklem decyzyjnym** systemu informacyjnego. Poza tym proponuje sie narzedzia dla faz:

- w ramach **tworzenia schematu ogolnego** - konieczne zbudowanie narzedzi umozliwiajacych wyodrebnienie dziedzin i projektow, zapamietania roznych modeli rozwiazania i zwiazanych z nimi scenariuszy, a takze zestawienia korzysci i strat z nimi zwiazanych;

- w ramach studium wstępnego - przewiduje się narzędzia konstrukcji poszczególnych modeli zredukowanych do reprezentatywnego podzespołu, uzyskania danych ekonomicznych charakteryzujących warianty rozwiązania (koszty, czas projektowania itp.), symulacji funkcjonowania rozwiązań;

- w trakcie etapu projektowania szczegółowego narzędzia powinny zapewniać konstruowanie schematu bazy danych , stuktury programów, przygotowanie danych testowych;

- na etapie produkcji oprogramowania - MERISE przewiduje stosowanie generatorów programów proceduralnych i nieproceduralnych, oraz generatorów danych testowych.

3. Metoda REMO

3.1 Informacje wstępne.

Metoda REMORA została opracowana w środowisku uniwersyteckim Paryża, przez zespół badawczy kierowany przez profesor C. Rolland. Pierwsza pełna wersja metody została zaprezentowana na kongresie grupy roboczej 8.1 IFIP, zajmującej się problemami baz danych i systemów informacyjnych , w 1982 roku. W następnych latach metoda ta była rozwijana - poprzez doskonalenie sposobu modelowania, a przede wszystkim poprzez opracowywanie komputerowych narzędzi wspomagania rozwoju systemów informacyjnych, ścisle powiązanych z założeniami metody REMORA /3/.

A oto podstawowe cechy charakterystyczne metody:

1) Zakłada ona podział procesu rozwoju systemu na trzy fazy: konceptualną, logiczną i fizyczną (implementacji).

2) Zakłada ona modelowanie i projektowanie systemu przy uwzględnianiu aspektu statycznego i dynamicznego we wzajemnym ich powiązaniu, we wszystkich fazach budowy systemu.

3) Metoda kładzie większy nacisk na sposób modelowania niż na szczegółowe, rygorystyczne przepisy postępowania w poszczególnych fazach rozwoju systemu.

Wynika to z faktu, ze metoda ta jest wynikiem prac badawczych w srodowisku uniwersyteckim.

4) Modelowanie systemu informacyjnego opiera sie w tej metodzie na zasadzie **elementarnosci** opisu zjawisk oraz na zasadzie **odzwierciedlania wszelkich zmian** zachodzacych w obserwowanej p iszczyznie rzeczywistosci.

5) Przyjeta metoda modelowania opiera sie rowniez na **relacyjnym modelu danych** oraz na modelu **"Entity-Relationship"** (Obiekt- Wspolzaleznosc), zaproponowanym przez P.P. Chena /1/.

Powyzsze cechy metody zostana szerzej omowione w nastepnych punktach. Metode REMORA przedstawimy w takim samym ukladzie jak metode MERISE: model, jezyk, sposob postepowania i narzedzia wspomagajace

3.2 Model systemu informacyjnego.

Przyjety tu sposob modelowania informacyjnego rzeczywistosci opiera sie na trzech podstawowych pojeciach: **obiekt, zdarzenie i operacja**. Umozliwiaja one modelowanie tak aspektu statycznego jak i dynamicznego systemu. Podobnie jak w innych metodach - stosuje sie tu zasade abstrakcji, stosujac pojecia **kategorii** obiektu, zdarzenia lub operacji.

Modelowanie to przebiega w dwu fazach:

- tworzenie **modelu opisowego**;
- konstruowanie **sformalizowanego modelu konceptualnego**.

Model opisowy stanowi bezposrednie wyrazenie rzeczywistosci za pomoca dlku podstawowych pojec. **Modelowanie aspektu statycznego** opiera sie na pojeciu **obiekt**. Jest to trwajacy w czasie, rzeczywisty lub abstrakcyjny skladnik organizacji . Obiekty roznia sie miedzy soba cechami, zwanymi tez **wlasciwosciami**. Pomiedzy obiektami moga wystepowac **powiazania (wspolzaleznosci).**

Klasie wystapien obiektow tej samej natury (na przyklad pracownikow) odpowiada typ obiektu, okreslany skrotowo jako **t-obiekt**. W podobny sposob powstaly pojecia **t-powiazanie** oraz **t-wlasciwosc**. Mozna powiedziec, ze t-obiekt

definiuje zbior obiektow tej samej natury. Podobnie jest z t-powiazaniem (zbior wystapien powiazan tej samej natury) oraz z t-wlasciwoscia (zbior wartosci wlasnosci tej samej natury). Jak latwo zauwazyc, zastosowano tu pojecia wywodzace sie z modelu ER "Entity-Relationship".

Modelowanie aspektu dynamicznego dokonuje zapomoca pojec operacja i zdarzenie. **Operacje** definiuje sie jako akcje, ktora moze byc wykonana w organizacji w sposob wyizolowany, i ktora modyfikuje stan obiektow. Przykladem moze byc operacja wysylki towaru z magazynu, obliczenie zarobku dla Kowalskiego. Kazda operacja uczestniczy w sposob bezposredni lub posredni w realizacji celow organizacji, poprzez kreowanie obiektow rzeczywistych (na przyklad produkty) lub modyfikowanie stanow obiektow abstrakcyjnych (na przyklad zmiana stanu konta).

Podobnie jak obiekty - operacje sa charakteryzowane przez wlasciwosci nastepujacych rodzajow:

- **czasowo-przestrzenne**, pozwalajace okreslic operacje w czasie i przestrzeni,
- **powiazane z zadaniami operacji**, a wiec precyzujace funkcje organizacji, do ktorych ona nalezy,
- **wlasciwosci specjalne**, definiujace dokladnie sama akcje oraz warunki jej wystapienia, przy czym najbardziej charakterystyczna wlasciwoscia tego typu jest **tekst operacji.**

Do zdefiniowania zaleznosci pomiedzy operacja i obiektami, ktore ona modyfikuje - wykorzystuje sie specjalny typ powiazania: **MODYFIKUJE.**

Podobnie jak w przypadku obiektow - kazdej klasie operacji tego samego rodzaju odpowiada typ operacji, okreslany skrotem **"t-operacja"**. Wszystkie powiazania typu MODYFIKUJE, wiazace obiekty nalezace do jednej lub kilku klas, oraz operacje nalezace do jednej klasy - nalezą do **klasy powiazania** MODYFIKUJE.

Zdarzenie jest definiowane w metodzie jako specyficzna zmiana stanu jednego lub kilku obiektow. Przykladem moze byc poczatek roku 1989 (zmiana w obiekcie kalendarz) lub przybycie zamowienia (zmiana obiektu zamowienie). Zdarzenia inicju

operacje w sposob warunkowy lub bezwarunkowy. Podobnie jak obiekt i operacja - zdarzenia posiadaja rowniez wlasciwosci, na przyklad data i miejsce wystapienia.

Dla zdefiniowania zaleznosci pomiedzy zdarzeniami a obiektami i operacjami wprowadzona dwa rodzaje zaleznosci : KONSTATUJE oraz WYZWALA.

Powiazanie **KONSTATUJE** pomiedzy zdarzeniem i obiektem odpowiada specyficznej zmianie stanu tego obiektu. Nie kazda zmiana obiektow odpowiada takiemu powiazaniu. Zmiana ta jest opisana przez predykat zdefiniowany na wlasciwosciach obiektow, na przyklad "jesli stan magazynowy < normatyw".

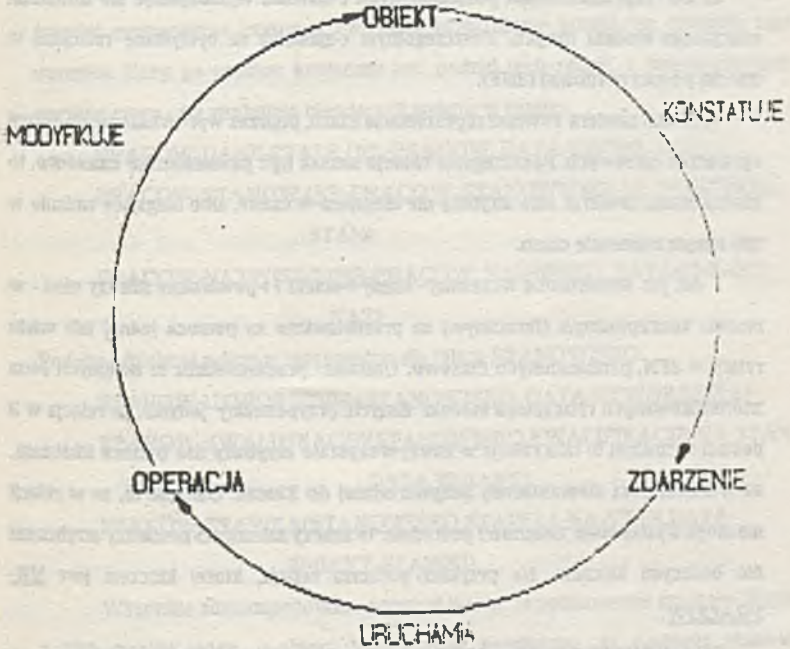
Zaleznosci pomiedzy zdarzeniami i operacjami opisuja powiazania typu WYZWALA. Uruchomienie operacji moze byc bezwarunkowe lub warunkowe . Operacja warunkowa wymaga **warunku uruchomienia.** Jest to wyrazenie okreslajace stan systemu, aby operacja mogla byc uruchomiona. Uruchomienie operacji moze byc wielokrotne. Jedna z wlasciwosci powiazania **WYZWALA** jest wiec **wspolczynnik uruchomienia.** Definiuje on zbior obiektow na ktorych operacja powinna byc wykonana.

Klase zdarzenia definiuje typ zdarzenia, opisywany jako **t -zdarzenie.** Powiazanie klasy zdarzenia z klasa lub klasami obiektow definiuje **klase powiazania** KONSTATUJE. Podobnie definiuje sie klase powiazania WYZWALA. Zaleznosci pomiedzy trzema zjawiskami (obiekt, zdarzenie, operacja) i trzema podstawowymi typami powiazan przestawia rys. 10.

Przedstawiony wyzej model opisowy stanowi zrodlo dla zbudowania **modelu konceptualnego,** opartego na zespole przyjetych regul i ograniczen formalnych. Przyjeto tu nastepujace zalozenia modelu:

1) Opiera sie on na trzech pojeciach: **C-OBIEKT, C-OPERACJA** oraz **C-ZDARZENIE.** Zaleznosci pomiedzy tymi kategoriami a pojeciami modelu opisowego sa nastepujace:

- kazdy t-obiekt jest reprezentowany przez jeden lub wiele c-obiektow
- kazda t-operacja oraz t-powiazanie MODYFIKUJE jest reprezentowane przez



Rys. 10. PODSTAWOWE POJĘCIA MODELU REMORY

jedna lub kilka c-operacji,

- kazde t-zdarzenie oraz t-powiazania typu KONSTATUJE i WYZWALA sa reprezentowane przez jedno lub kilka c-zdarzen.

2) Model przyjmuje zasade reprezentowania t-zjawisk w formie elementarnej, to znaczy niepodzielnej.

3) Do reprezentowania poszczegolnych c-zjawisk, wykorzystuje sie formalizm relacyjnego modelu danych. Poszczegolnym c-zjawiska sa opisywane relacjami w trzeciej postaci normalnej (3NF).

4) Model zawiera rowniez reprezentacje czasu, poprzez wprowadzenie do relacji ograniczen czasowych. Poszczegolne relacje musza byc permanentne czasowo, to znaczy musza zawierac albo atrybuty nie ulegajace w czasie, albo ulegajace zmianie w tym samym momencie czasu.

Jak juz stwierdzono wczesniej - kazdy t-obiekt i t-powiazanie miedzy nimi - w modelu konceptualnym (formalnym) sa przedstawiane za pomoca jednej lub wielu relacji w 3FN, permanentnych czasowo. Unikajac przedstawiania tu kolejnych form znormalizowanych relacyjnego modelu danych, przypomnimy jedynie, ze relacja w 3 postaci normalnej to taka relacja w ktorej wszystkie atrybuty nie bedace kluczem, sa w zaleznosci elementarnej bezposredniej do klucza. Oznacza to, ze w relacji nie moga wystepowac zaleznosci posrednie, to znaczy zaleznosci pomiedzy atrybutami nie bedacymi kluczem. Na przyklad ponizsza relacja, ktorej kluczem jest NR-PRACOW:

PRACOWNIK (NR-PRACOW, NAZWISKO, DATA-URODZ, PLEC,
STANOWISKO, KWALIFIKACJE-NA-STAN, STAWKA-NA-STAN)

nie jest w 3FN, ponieważ atrybuty KWALIFIKACJE-NA-STAN (kwalifikacje wymagane na stanowisku) oraz STAWKA-NA-STAN (stawka placy przysługująca na stanowisku) sa w zaleznosci posredniej od atrybutu STANOWISKO. Zeby uzyskac relacje w 3FN, konieczny jest podział powyższej relacji na dwie inne:

PRACOWNIK (NR-PRACOW, NAZWISKO, DATA-URODZ, PLEC,

STANOWISKO)

STANOWISKO(STANOWISKO, KWALIFIKACJE-NA-STAN, STAWKA-NA- STAN)

Dodatkowym wymogiem formalnym Remory jest, aby relacje w 3FN były rowniez permanentne czasowo. Powyzsze przykladowe relacje nie spelniaja tego warunku. Zeby go uzyskac konieczny jest podzial tych relacji, z wprowadzeniem atrybutu czasu - dla atrybutow ulegajacych zmianie w czasie:

PRACOW-DANE-STALE (NR-PRACOW, DATA-URODZ, PLEC)

PRACOW-STANOW (NR-PRACOW, STANOWISKO, DATA-OBJECIA-STAN)

PRACOW-NAZWISKO (NR-PRACOW, NAZWISKO, DATA-ZMIANY-NAZ)

Podobne dzialanie nalezy przeprowadzic dla relacji STANOWISKO:

STANOW-UTWORZENIE (STANOWISKO, DATA-UTWORZENIA)

STANOW-KWALIFIKACJE(STANOWISKO,KWALIFIKACJE-NA-STAN DATA-ZMIANY)

STANOW-STAWKA(STANOWISKO,STAWKA-NA-STAN,DATA-ZMIANY-STAWKI)

Wszystkie zdekomponowane powyzej relacje sa permanentne czasowo. Kazda z nich opisuje jeden c-obiekt. Mozna wiec powiedziec, ze **c-obiekt stanowi reprezentacje okreslonego aspektu czasowego t-obiektu.** Jednemu t-obiektowi z modelu opisowego odpowiada wiec najczesciej wiele c-obiektow, z ktorych kazdy jest opisywany relacja w 3FN, permanentna czasowo.

Podobny tok rozumowania stosuje sie przy normalizacji t-operacji i przeksztalcaniu ich na c-operacje. Przy konstrukcji c-operacji stosuje sie dwa typy ograniczen:

- **ograniczenia strukturalne**, ktore pozwalaja na reprezentacje zespolu t-obiektow i t-operacji w sposob minimalizujacy liczbe powiazan miedzy nimi i eliminujący

redundancje w reprezentacji. Uzyskuje sie to poprzez zastosowanie w c-operacjach elementarnosci podobnej jak w c-obiektach. Jedna c-operacja moze wiec modyfikowac tylko jeden c-obiekt;

- **ograniczenia kompletnosci,** ktore pozwalaja projektantowi opisac regule zachowania, odpowiadajaca kazdej c-operacji.

C-operacja jest powiazana z tylko jednym c-obiektem poprzez wykonanie operacji w danym momencie. Wykonanie to modyfikuje stan tylko jednego obiektu (wystapienia obiektu). Z powyzszego wynika ze c-operacja reprezentuje typ elementarnej operacji systemu informacyjnego.

Kazda c-operacja jest opisywana za pomoca kilku relacji (schematow relacji) typu c-operacji. Na przyklad c-operacja "aktualizacja stanu zapasu" moze byc opisana nastepujacymi relacjami:

AKTUAL-ZAPAS1 (AKTUAL-ZAPAS, PRODUKT, DATA-KREOW-OPER.)

AKTUAL-ZAPAS2 (AKTUAL-ZAPAS, TEKST-OPER, DATA-MOD-TEKSTU)

AKTUAL-ZAPAS3 (AKTUAL-ZAPAS, DATA-WYK-OPER, IDEN-PRODUKT)

Relacja AKTUAL-ZAPAS1 opisuje stale wlasciwosci (atrybuty) operacji, to znaczy date utworzenia operacji w systemie (DATA-KREOW-OPER) oraz nazwe c-obiektu, ktora ta operacja modyfikuje.

Relacja AKTUAL-ZAPAS2 okresla czynnosci skladajace sie na operacje, poprzez okreslenie nazwy tekstu operacji (TEKST-OPER), oraz date modyfikacji tekstu operacji (DATA-MOD-TEKSTU).

Relacja AKTUAL-ZAPAS3 definiuje kolejne wystapienia c-operacji, poprzez okreslenie daty jej wykonania (DATA-WYK-OPER) oraz identyfikatora c-obieku , na ktorym operacja zostala wykonana; w naszym przypadku jest to identyfikator materialu (IDEN-PRODUKT).

Dla opisania c-operacji konieczne jest wiec uzycie powyzszych trzech form schematow c-operacji. Kazdy z nich jest w 3FN permanentnej czasowo.

Pojecie **c-zdarzenia** jest definiowane w odniesieniu do c-obiektow i c-operacji, uwzgledniajac:

- **ograniczenia strukturalne**, ktore wymagaja aby c-zdarzenie bylo w zwiazku tylko z jednym c-obiektem i conajmniej z jedna c-operacja,
- **ograniczenia kompletnosci**, ktore wiaza z kazdym c-zdarzeniem tekst predykatu zmiany stanu c-obiektu, warunki oraz wspolczynniki uruchomienia operacji.

Poniewaz c-obiekt reprezentuje typ elementarnego stanu systemu informacyjnego, a c-operacja elementarna operacja w systemie - **c-zdarzenie** definiuje sie jako **szczegolny i elementarny typ zmiany** w systemie informacyjnym, **powodujacy przeksztalcenie elementarnego stanu systemu, poprzez wykonanie zespolu elementarnych operacji.** Wystapienie c-zdarzenia ma miejsce wowczas , gdy stwierdzi sie wystapienie specjalnego typu zmiany w c-obiekcie powiazanym z c-zdarzeniem.

C-zdarzenie powoduje uruchomienie conajmniej jednej c-operacji. uruchomienie to moze byc bezwarunkowe lub warunkowe. W przypadku **c-operacji warunkowej**, jej uruchomienie jest uzaleznione od spelnienia **warunku uruchomienia**. Uruchomienie c-operacji moze tez oznaczac jej wielokrotne wykonanie na wielu wystapieniach c-obiektu. Zbior tych wystapien okresla **wspolczynnik uruchomienia.**

Wyrazenie c-zdarzenia w ujeciu relacyjnym dokonuje sie rowniez za pomoca kilku form relacji w 3FN, permanentnych czasowo. Przedstawimy to na przykladzie zdarzenia "zalamanie sie zapasu materialowego":

ZAL-ZAPASU1 (ZAL-ZAPASU, PRODUKT, ZAMAWIANIE, DATA-WPROW-ZDARZ)

ZAL-ZAPASU2 (ZAL-ZAPASU, DATA-MOD-PRED, PRED-ZAL)

ZAL-ZAPASU3 (ZAL-ZAPASU, ZAMAWIANIE, DATA-MOD-WAR-WSP.

WAR-ZAMAW,WSP-ZAMAW)

ZAL-ZAPASU4 (ZAL-ZAPASU, DATA-ZAL-ZAPASU, IDEN-PRODUKTU;

ZAL-ZAPASU5 (ZAL-ZAPASU, DATA-ZAL-ZAPASU, IDEN-ZAM,

DATA-URUCH-ZAM).

Relacja ZAL-ZAPASU1 opisuje stale wlasciwosci c-zdarzenia: jego nazwe (ZAL-ZAPASU), obiekt, ktorego zmiana powoduje konstatacje zdarzenia (PRODUKT), uruchamiana operacje (ZAMAWIANIE), date wprowadzenia c-zdarzenia do systemu (DATA-WPROW-ZDARZ).

Relacja ZAL-ZAPASU2 okresla predykat definiujacy warunek wystapienia zadrzenia (PRED-ZAL) - podaje jego nazwe, oraz date modyfikacji predykatu (DATA-MOD-PRED).

Relacja ZAL-ZAPASU3 okresla nazwe warunku uruchomienia operacji (WAR-ZAMAW), wspolczynnika uruchomienia (WSP-ZAMAW) oraz date zmiany tych wielkosci (DATA-MOD-WAR-WSP).

Relacja ZAL-ZAPASU4 okresla date poszczegolnych wystapien zdarzenia (DATA-ZAL-ZAPASU) oraz identyfikator obiektu (w naszym przypadku produktu), w oparciu o ktory stwierdzono wystapienie zdarzenia (IDEN-PRODUKTU).

Relacja ZAL-ZAPASU5 okresla dla kazdego wystapienia zdarzenia ZAL-ZAPASU : identyfikator wystapienia operacji zamawiania (IDEN-ZAM) oraz date uruchomienia wystapienia tej c-operacji (DATA-URUCH-ZAM).

Produktem modelowania konceptualnego jest tak zwany **schemat konceptualny**, skladajacy sie z dwu czesci:

- **subschematu statycznego**, stanowiacego zestaw relacji opisujacych (definiujacych) wszystkie c-obiekty i c-powiazania wyroznione w modelowanym problemie,

- **subschematu dynamicznego**, stanowiacego zestaw relacji opisujacych wszystkie c-operacje i c-zdarzenia w modelowanym problemie.

Powyzszy schemat jest uzupelniany opisem slownym regul integralnosci modelu, stanowiacych te informacje o budowie i funkcjonowaniu systemu, ktore nie mozna zawrzec w relacjach.

Poza wyrazeniem modelu systemu za pomoca za pomoca specjalnego jezyka - Metoda REMORA proponuje rowniez **graficzne narzedzia opisu modelu konceptualnego**. Ich zadaniem jest ulatwienie tworzenia i analizy modelu konceptualnego. Stanowia one narzedzia pracy projektanta w procesie tworzenia modelu. Daja one tez mozliwosc syntetycznej prezentacji zawartosci modelu.

Graficzna prezentacja subschematu statycznego opiera sie grafie elementarnych zaleznosci funkcjonalnych wystepujacych w poszczegolnych relacjach modelu. Sposob reprezentacji graficznej c-obiektow i c-powiazan przedstawia rys. 11:

- c-obiekt jest wyrazany za pomoca malego grafu umieszczonego w prostokacie; nazwa c-obiektu jest umieszczona w lewym gornym rogu;

- c-klasa (odpowiadajaca zespolowi c-obiektow opisujacych t-obiekt modelu opisowego) jest przedstawiana za pomoca prostokatu, obejmujacego nalezace do niej c-obiekty;

- c-powiazanie jest przedstawiane za pomoca linii laczacych c-klasy.

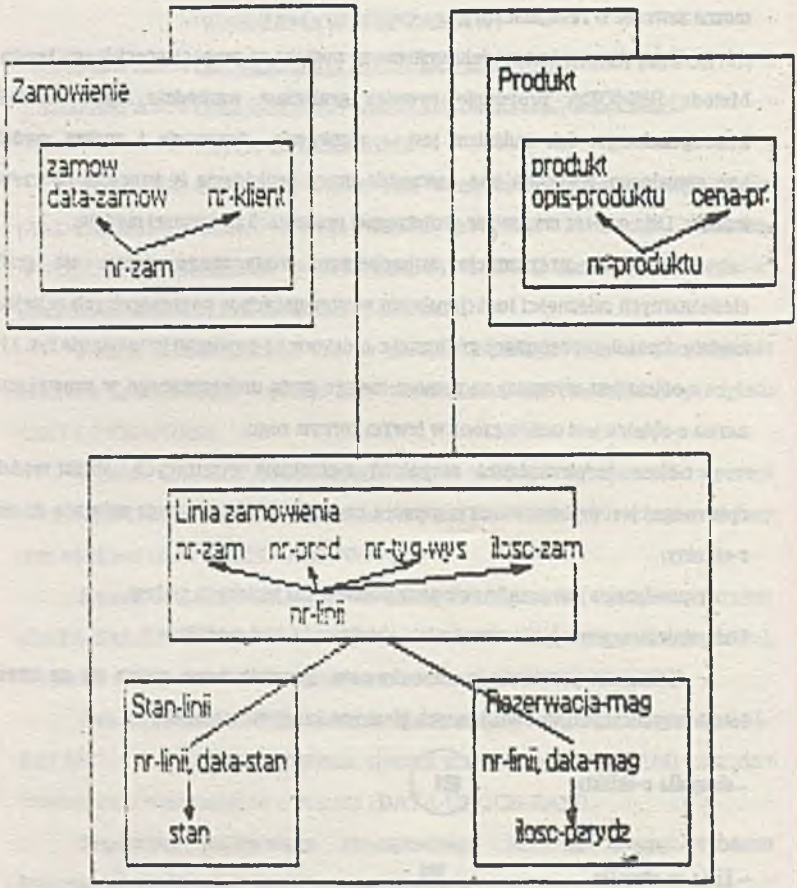
Uzupelnieniem grafu jest zestawienie c-obiektow i c-klas modelu.

Graficzna prezentacja subschematu dynamicznego opiera sie na trzech formach graficznych, dpowiadajacych glownym kategoriom modelu:

- okrag dla c-obiektu

- linia ze strzalka dla c-operacji

- trojkat dla c-zdarzenia



Rys. 11. PRZYKLAD GRAFICZNEJ PREZENTACJI SUBSCHEMATU DYNAMICZNEGO

Trzy podstawowe powiazania pomiedzy tymi formami odpowiadaja trzem glownym powiazaniom miedzy kategoriami modelu:

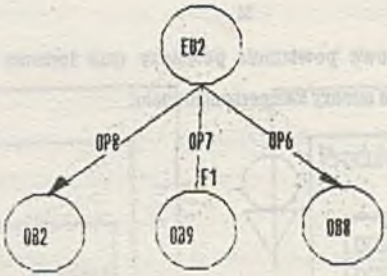
- powiazanie KONSTATUJE 

- powiazanie URUCHAMIA 

- powiazanie MODYFIKUJE 

Powyzsze formy graficzne sluza do konstruowania grafu, przedstawiajacego sekwencje nastepujacych zjawisk: zmiana stanu obiektu, konstatacja zdarzenia, uruchomienie operacji, modyfikacja obiektu za pomoca operacji, itd. Modyfikacja stanow obiektu moze spowodowac wystapienie zdarzenia , oczywiscie jezeli zostanie spelniony warunek opisany w predykacie powiazanym z danym typem zdarzenia. Przyklad takiego grafu zbudowanego dla problemu "Realizacja zamowien", przedstawia rys 12.

Podstawowa jednostka skladowa tak skonstruowanej struktury dynamicznej jest **cykl dynamiczny**, ktorego przyklad jest przedstawiony ponizej.



Cykl dynamiczny sklada sie on z sekwencji nastepujacych zjawisk: zdarzenie, uruchamiane przez nie operacje oraz modyfikowane obiekty. Graf sklada sie z ciagu nastepujacych po sobie cykli.

Dla kompletnosci opisu modelu, metoda proponuje uzupelnienie grafu zestawami opisu nastepujacych wielkosci zwiaznych ze struktura dynamiczna grafu:

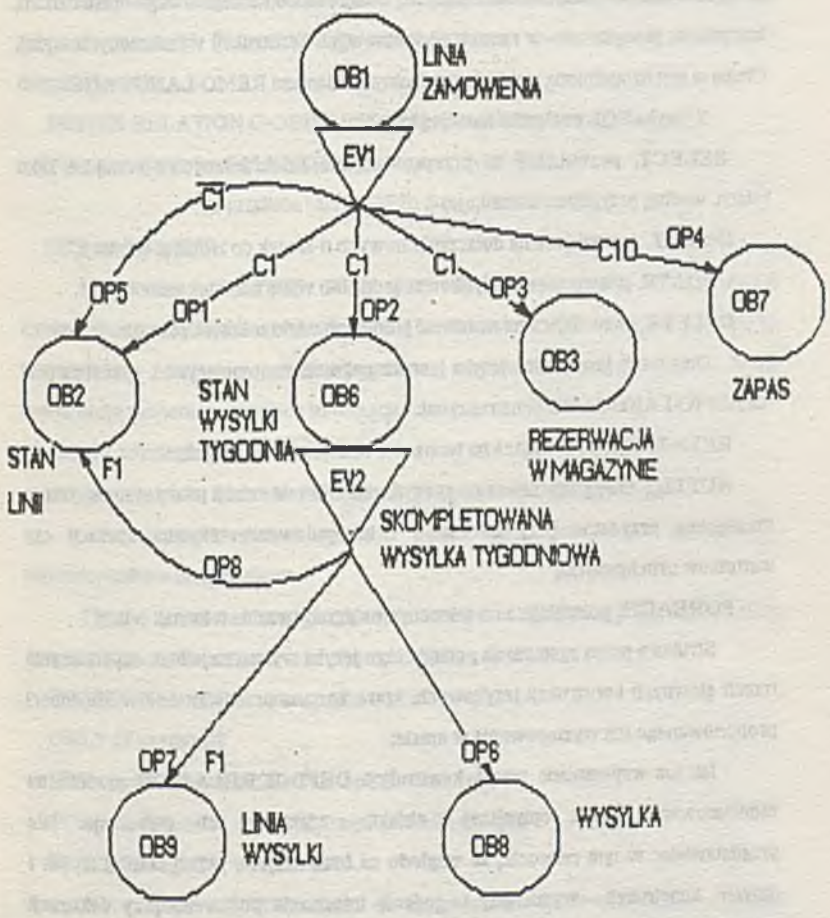
- c-operacji, przedstawiajacych opis czynnosci zwiazanych z operacja (tekst operacji)
- c-zdarzen, opisujacych slownie dane zdarzenie,
- predykatow c-zdarzen, podajacych dokladna ich definicje.
- warunkow uruchomienia operacji, podajacych dla kazdej nazwy warunku - tekst warunku.
- wspolczynnikow uruchomienia operacji, okreslajacych zbior obiektow, na ktorych operacja ma byc wykonywana.

3.3 Jezyk opisu systemu informacyjnego.

Tworcy metody REMORA proponuja jej uzytkownikom jezyk REMO-LANGUAGE. Sluzy on do formalnego opisu tekstowego modelu systemu informacyjnego oraz do ewentualnego jego komputerowego przetwarzania.

Jezyk ten sklada sie z dwu czesci:

1) Pierwsza zawiera konstrukcje jezykowe pozwalajace na opis poszczegolnych elementow modelu to znaczy: relacji (konstrukcja DEFINE RELATION), cykli



Rys. 12 PRZYKŁAD GRAFICZNEJ PREZENTACJI SUBSCHEMATU DYNAMIKI

dynamicznych (konstrukcja DEFINE EVT) oraz innych elementow takich jak tekst operacji, predykatu zdarzenia itp. (konstrukcja ASSERT).

2) Druga sklada sie z wybranych zdan (klauzul) jezyka SQL, zaprojektowanego do manipulowania na relacyjnej bazie danych. Sluzy ona do szczegolowego opisu dzialan, warunkow, predykatow - w ramach podstawowych konstrukcji wymienionych wyzej. Grupa ta jest uzupelniona rowniez oryginalnymi zdaniami REMO-LANGUAGE.

Z jezyka SQL pochodza nastepujace zdania:

SELECT, pozwalajace na przeprowadzenie selekcji danych z jednej lub kilku relacji, według przyjetych warunkow;

- **INSERT**, pozwalajace na dolaczanie nowych n-krotek do istniejacej relacji,
- **UPDATE**, pozwalajace modyfikowac jedna lub wiele n-krotek relacji:
- **DELETE**, pozwalajace na usuniecie jednej lub wielu n-krotek relacji.

Omawiana grupa zdan jezyka jest uzupelniona nastepujacymi , specyficznymi dla REMO-LANGUAGE konstrukcjami:

- **RELATION** , pozwalajaca na tworzenie relacji tymczasowych;
- **NUPLET OF**, pozwalajaca na specyfikacje n-krotki relacji jako zmiennej; jest to szczegolnie przydatne przy tworzeniu i manipulowaniu tekstami operacji czy warunkow uruchomienia;
- **FOREACH**, pozwalajaca na sekwencyjne przetwarzanie n-krotek relacji.

Strukure opisu systemu za pomoca tego jezyka wyznacza jednak uzycie trzech glownych konstrukcji jezykowych, ktore zostana przedstawione w kolejnosci proponowanego ich wystepowania w opisie;

Jak juz wspomniano wyzej, konstrukcja **DEFINE RELATION** pozwala na zadeklarowanie relacji opisujacej c-obiekt, c-zdarzenie lub- c-operacje. Nie przedstawiajac w tym referacie, ze wzgledu na brak miejsca, pelnej skladni tej jak i innych konstrukcji - wymienimy tu jedynie informacje podawane przy deklaracji relacji:

- nazwa i typ relacji (c-obiekt, c-operacja, c-zdarzenie),

- nazwa c-klasy,

- format relacji (w ramach c-klasy),

- specyfikacja klucza i atrybutow niekluczowych, lacznie z okresleniem ich dziedzin.

Przykladem uzycia tej konstrukcji jest opis c-obiektu "linia zamowienia", ze schematu na rys..12. Wszystkie uzyte slowa kluczowe sa dla odroznienia napisane duzymi literami:

DEFINE RELATION C-OBJET PERM C-CLASSE linia-zamowienia

linia-zamowienia (nr-linii : INTEGER, nr-zam : INTEGER,

nr-artykulu : INTEGER, il-zam : INTEGER)

KEY (nr-linii) END

Konstrukcja **DEFINE EVT** sluzy do zadeklarowania jednego cyklu dynamicznego schematu konceptualnego. Jej format jest tak zbudowany, ze pozwala na bezposrednie przejscie z prezentacji graficznej na opis w omawianym jezyku. Jedna konstrukcja zawiera nastepujace informacje:

- nazwe zdarzenia inicjujacego,

- nazwe predykatu zdarzenia i obiektu, na ktorym jest on zdefiniowany,

- nazwy uruchamianych operacji, oraz zwiazanych z nimi warunkow i wspolczynnikow uruchomienia.

Ponizej przedstawiamy przyklad zastosowania konstrukcji do opisu cyklu dynamicznego, zwiazanego ze zdarzeniem EV2, ze schematu na rys. 12.

DEFINE EVT ev2 ON stan-wysylki-tygodniowej

ONLY IF predykat2

FOR wspolczynnik1 EXECUTE op8 ON stan-linii

FOR wspolczynnik1 EXECUTE op7 ON linia-wysylki

EXECUTE op6 ON wysylka END.

Konstrukcja **ASSERT** sluzy do uzupelnienia opisu elementow systemu wprowadzonych w zdaniach DEFINE RELATION i DEFINE EVT, to znaczy:

- ograniczen integralnosciowych, dotyczacych jednego lub wielu c-obiektow;

- opisu predykatow, tekstow operacji oraz warunkow i wspolczynnikow uruchomienia operacji.

Kazda z deklaracji sklada sie zasadniczo z dwu czesci:

- okreslenia nazwy i typu deklarowanej wielkosci.

- oraz z jej tekstu.

Ponizej podajemy przyklad deklaracji warunku C1 "zapas produktu jest wystarczajacy", powodujacy uruchomienie odpowiedniej operacji:

```
ASSERT c1  COND : BOOL-FUNCTION : funkcja-c1(VAR s NUPLET OF
linia-zamowienia)
BEGIN
SELECT INTO n NUPLET OF zapas
*FROM zapas
WHERE nr-produktu = s. nr-produktu
AND data-stanu-zapasu LAST
IF n. stan-zapasu >= s. ilosc-zamowiona
THEN funkcja-c1 = <<TRUE>>
ELSE funkcja-c1=.<<FALSE>>
END
END
```

Opis systemu wyrazony w przedstawianym jezyku sklada sie z trzech czesci, odpowiadajacych trzem glownym konstrukcjom jezyka:

- czesci opisujacej relacje wszystkich klas,

- czesci opisujace poszczegolnej cykle dynamiczne (uzycie konstrukcji DEFINE EVT),

- czesci zawierajacej dodatkowe deklaracje modelu (uzywajac konstrukcji ASSERT).

3.4 Tok postepowania w metodzie.

Jak juz wspomniano, podstawowa zasada metody REMORA jest podział całego procesu rozwoju na trzy fazy: konceptualna, logiczna i implementacji (fizyczna). Przedstawia ona szczegolowy sposob postepowania dla dwu faz rozwoju systemu informacyjnego: fazy konceptualnej i logicznej.

REMORA koncentruje sie na modelowaniu struktury informacyjnej problemu. Z tego wzgledu metoda ta moze wydawac sie niekompletna.

Faza **modelowania konceptualnego** sklada sie z czterech etapow.

Etap I polega na analizie obszaru rzeczywistosci, majacej na celu identyfikacje glownych zjawisk wystepujacych w problemie: obiektow, zaleznosci miedzy nimi, zdarzen i operacji. Wynikiem tych prac jest **model opisowy**, stanowiacy zestawienie t-obiektow, t-powiazan (model aspektu statycznego) oraz t-operacji i t-zdarzen (model dynamiki).

Punktem wyjscia dla **Etapu II** jest model opisowy. Etap ten polega na **formalizacji opisu aspektu statycznego i dynamicznego** zgodnie z zasadami konceptualizacji przedstawionymi w p. 3.2. Kolejnosc czynnosci w tym etapie jest nastepujaca:

- normalizacja t-obiektow i t-powiazan (aspektu statycznego),
- normalizacja t-operacji i t-zdarzen (aspektu dynamicznego).
- wstepna weryfikacja zgodnosci obu aspektow.

Etap III obejmuje dwa typy weryfikacji schematu konceptualnego:

1) Weryfikacje zgodnosci z regulami integralnosci (spojnosci) modelu.

2) Weryfikacje schematu z potrzebami informacyjnymi uzytkownikow.

Wynikiem powyzszych etapow jest schemat konceptualny w postaci graficznej, wykazow c-zjawisk, oraz wykazow specjalnych opisow (na przyklad predykatow).

Etap IV to wyrazenie schematu w jezyku REMO-LANGUAGE.

Faza logiczna rozwoju systemu powinna realizowac dwa cele:

- konstruowac strukture danych dla potrzeb zastosowanego SZBD, prezentujaca zgrupowania danych oraz sciezki nawigowania miedzy nimi,

- konstruowac spojny zespol transakcji, wykorzystujacy w sposob optymalny strukture danych.

Pierwszy etap tej fazy ma na celu zaprojektowanie **schematu logicznego danych.** Odbywa sie ono w dwu krokach:

1) Pierwszy krok to konstrukcja **schematu logicznego standardowego.** Nie odpowiada on zadnemu konkretnemu SZBD, i ze wzgledu na swa ogolnosc moze on sluzyc jako punkt wyjscia do latwego zaprojektowania schematow logicznych dla roznych SZBD. Opiera sie on na pojeciach: typ rekordu, typ funkcji wejscia, typ funkcji dostepu. W procesie konstrukcji tego schematu wystepuja dwa podkroki:

- tworzenia **schematu logicznego standardowego maksymalnego,** ujmujacego cala zawartosc semantyczna schematu konceptualnego statycznego,
- tworzenia **schematu logicznego standardowego zaadaptowanego,** w ktorym moga byc dodawane lub usuwane rekordy, dokonywane ich fuzje, dodawane lub usuwane funkcje dostepu. Wprowadzane zmiany moga wynikac z potrzeby zaspokojenia zadan informacyjnych uzytkownikow oraz z potrzeb zwiazanych z aktualizacja bazy danych.

2) Drugi krok to **tworzenie schematu logicznego specyficznego** dla SZBD wybranego do realizacji systemu informacyjnego.

Drugi etap fazy logicznej to projektowanie **schematu logicznego transakcji.** Jego celem jest uzyskanie projektu procedur programowych, zaadaptowanych do warunkow konkretnego SZBD.

Etap ten sklada sie z dwu krokow:

1) Konstruowanie **schematu transakcyjnego minimalnego.** Punktem wyjscia jest tu subschemat konceptualny dynamiczny, ktory w oparciu o reguly interpretacji jest przeksztalcany na struktury transakcyjne przewidziane w metodzie. Remora proponuje uogolniony model prezentacji struktur transakcyjnych, opierajac sie na pojeciach **transakcji** (sekwencja atomicznych akcji, niepodzielna w ich wykonaniu) i

wyzwalacza (predykat definiujacy warunek uruchomienia transakcji). Kazda struktura moze sie skladac z kilku typow struktur, zbliżonych do podstawowych konstrukcji programowania strukturalnego.

2) Konstruowanie **schematu transakcyjnego zaadaptowanego.** Powstaje on w oparciu o schemat transakcyjny minimalny, przy uwzglednieniu:

- ograniczen konkretnego SZBD,
- warunkow funkcjonowania systemu, narzuconych przez uzytkownikow.

Przy konstruowaniu tego schematu wykorzystuje sie zespol regul tworzacych, wsrod ktorych mozna wyroznic reguły grupowania transakcji, przetwarzania zdarzen zewnetrznych i programowania transakcji.

W efekcie schemat transakcyjny zaadaptowany stanowi projekt programow uzytkowych (w warunkach bazy danych), lacznie z opisem struktur sterowania w programie - w oparciu o pseudojezyk programowania.

3.5 Narzedzia komputerowego wspomagania metody.

Po kilku latach prac badawczych nad sposobami komputerowego wspomagania metody , oraz po opracowaniu szeregu czastkowych rozwiazan prototypowych - w 1988 roku opracowano projekt systemu kompleksowego wspomagania systemow informacyjnych RUBIS.

RUBIS opiera sie na opisie systemu informacyjnego przedstawionej w formie relacyjnej bazy danych. Bedzie on wyposazony w trzy interfejsy komunikacji z projektantem:

- interfejs graficzny, wykorzystujacy miedzy innymi ikony, odpowiadajace elementom opisu,
- System Ekspert, pracujacy na opisie systemu informacyjnego, w oparciu o podzbior jezyka naturalnego,
- interfejs oparty na technice menu i jezyku Remory.

System RUBIS umozliwia wiec tworzenie i rozwijanie opisu, weryfikacje jego spojnosci oraz wykorzystanie opisu w trakcie modelowania konceptualnego i

logicznego. System ten jest rowniez wyposazony w elementy umozliwiajace uzyskanie prototypu rozwiazania i przeprowadzenie testow jego funkcjonowania. Sa to: **Monitor zastosowan** (rozpoznaje i interpretuje zdarzenia zewnetrzne), **Procesor zdarzen** (rozpoznaje i obsluguje zdarzenia wewnetrzne) i **Procesor czasu** (ustala zdarzenia czasowe).

4. Podsumowanie

Przedstawione metody - na pozor rozne - posiadaja jednak wiele punktow wspolnych, poniewaz wywodza sie z jednego kierunku badawczego - badan nad metodami rozwoju systemow informacyjnych opartych na technice baz danych.

A oto ich cechy wspolne:

1) Obie metody podkreslaja potrzebe uwzgledniania obu aspektow systemu informacyjnego - statycznego i dynamicznego - od poczatku procesu jego rozwoju.

2) Obie metody zakladaja podzial calego procesu rozwoju systemu informacyjnego na trzy glowne fazy: konceptualna,logiczna i fizyczna.

3) Opis rzeczywistosci w obu metodach opiera sie na modelu "Entity-Relationship", zaproponowanym przez P.P. Chena.

4) Dla obu metod opracowuje sie narzedzia wspomagajace, ktorych glownymi elementami sa:

- komputerowy opis systemu, w oparciu o przyjety w metodzie model,
- System Ekspert, korzystajacy z opisu i rozwijajacy go.

Przedstawione tu metody sa reprezentantami nowej grupy metod rozwoju systemow informacyjnych, ktore klada nacisk na modelowanie konceptualne obu aspektow systemu , pod katem zastosowania techniki baz danych.

Bibliografia

1.Chen P.P.:The Entity Relationship Model - Toward a Unified View of Data, ACM Transactions on Database Systems, vol. 1, n° 3/1976.

2. Matheron J.P.: Comprendre Merise - Outils conceptuels et organisationnels: Eyrolles 1988.

3. Rolland C. and all: The RUBIS System, paper for Congres IFIP WG 8.1, London, 1988.

4. Rolland C., Foucaut O., Benci G.: Conception des systemes d'information - La méthode REMORA, Eyrolles, Paris 1988.

5.Rolland C., Richard Ch.: The REMORA methodology for information systems design and management, in: T.W. Olle, H.G. Sol, A.A. Verijn-Stuart (editors): "Information Systems Design Methodologies: A Comparatve Review", North Holland Publishing Company,1982.

6. Tardieu H., Rochfeld A.,Colletti R.: La méthode MERISE - principes et outils, Editions d'Organisation, 1983.

Wojciech Olejniczak
Ireneusz Szydłowski
Instytut Cybernetyki Ekonomicznej
i Informatyki Uniwersytetu Szczecińskiego

GLOBALNA METODA PROJEKTOWANIA SYSTEMÓW INFORMATYCZNYCH

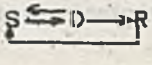
1. Wprowadzenie - założenie wyjściowe

Podejście globalne, systemowe, kompleksowe do projektowania systemów informacyjnych ma dwa aspekty. Po pierwsze rozwiązuje "wszystko" w SI i projektuje "wszystko". Oznacza to, że obejmuje całość (cały system projektowany i cały proces projektowania). Tworzenie systemu informacyjnego ma charakter działania innowacyjnego o niestandardowym przebiegu. Przebieg ten można określić jako jedność przeplatających się faz: fazy projektowania (sensu stricto) i fazy wytwarzania - produkcji unikatowego rozwiązania systemowego, poprzez dominującą tu rolę projektowania oprogramowania i programowania - takiego do jakiego się przyzwyczailiśmy (z kodowaniem włącznie). Czyli prace projektowe i wytwórcze są organicznie powiązane i trwają od chwili pojawienia się i wyartykułowania potrzeby (inspiracji) do końca, czyli do sukcesu lub porażki, z malejącą tendencją projektowania.

Co daje globalne projektowanie?

Po pierwsze likwiduje jednostronne i wąskie pojmowanie

systemu informacyjnego. Projektuje się przez nie system: "dla komputera", "dla użytkownika" i "dla projektanta". Po drugie przyjmuje rozszerzoną, a właściwie m e t a f o r m u ł ę projektowania: "struktura - działanie - rozwój" (S-D-R), w której mieści się i pamiętane podejście Chena E-R i podejście O-A-R, czy np. Z-P-O, czy wreszcie inne formuły. Metaformuła wchłania i rozwija te podejścia; zakładając pewną strukturę, następnie jej działanie (funkcjonowanie) i rozwój działania, który może zmienić strukturę. Metaformuła S-D-R nie określa kolejności dwóch pierwszych członów i może przebiegać alternatywnie:



taki właśnie jak to zaznaczono na powyższym rysunku.

Globalne podejście do projektowania, które my tu i teraz prezentujemy, i którego poniekąd jesteśmy twórcami i zwolennikami, zakłada s t r u k t u r a l i z a c j ę z zadanymi czterema i piątym punktami wejścia w system projektowany jak niżej:

⟶struktura funkcjonalna (SF),	
⟶struktura informacyjna (SI),	
⟶struktura przestrzenna (SP),	+ ⟶struktura organizacyjna (SO)
⟶struktura techniczna (ST),	

PLANSZA DEFINIUJĄCA STRUKTURY

Włączenie do tej czwórki struktur, piątej struktury (organizacyjnej) jest pewnym gestem pod adresem użytkownika,

skłania go do projektowania partycypacyjnego. Sama preparacja projektowa struktury organizacyjnej jest sugestią, a nie ingerencją w strukturę obiektu.

Przyjmujemy, że proces tworzenia systemu informacyjnego przebiegają w trzech stadiach:

- i n s p i r a c j a - zawiera globalny opis problemu projektowego, punktem wyjścia jest potrzeba zmiany stanu istniejącego,
- s t r u k t u r a l i z a c j a - pozwala widzieć całość, całość w szczegółach, szczegóły w całości, czyli daje całość i spektrum zarazem,
- s c a l a n i e s y s t e m u - jego konsolidacja i strojenie, ogólnie prace porządkowe, iluminacja rozwiązania.

Podejście globalne do projektowania, zakłada także, że współczesne systemy są realizowane przez system projektujący. Ten ostatni jest "wyposażony" w ogólną metodologię projektowania systemów i metodologie szczegółowe projektowania struktur[1)], czyli dysponuje określonym zasobem wiedzy. System projektujący jest też wyposażony w odpowiednie komputerowe narzędzia wspomagające tworzenie systemów informacyjnych (typ CAD/CAM) Innymi słowy mamy do czynienia z warsztatem twórczym i wytwórczym, w którym merytorycznie odpowiedni ludzie realizują procesy tworzenia systemu informacyjnego, korzystając z odpowiednich narzędzi wspomagających. Narzędzia te mogą być ustawione "pod proces" projektowania, lub używane autonomicznie.

Sam p r o c e s p r o j e k t o w a n i a. Istnieją w praktyce silne tendencje, naciski do ścisłego zdeterminowania przebiegu tego procesu, określenia jego jedynego początku i jedynego końca, chronologicznego ułożenia kolejności

poszczególnych etapów. Otóż niczego takiego nasze podejście globalne nie proponuje. Wprost przeciwnie. Proces projektowania jest chaotyczny, którego modelem jest sieć stochastyczna. Ma (praktycznie) dowolnie wiele wejść i tyleż samo wyjść. Projektanci poruszają się po tej sieci, korzystając z wszystkich możliwości figur szachowych. Jednak, aby to stwierdzenie (i zezwolenie) nie było zbyt obrazoburcze, spostrzegamy, że nawet niekonwencjonalni szachiści (a więc geniusze) mają swoje ulubione konwencjonalne ruchy.

Także tok niniejszego wykładu, można powiedzieć, że dalej wykorzysta tę dowolność i rozpocznie się od dowolnego punktu startu, że będzie to punkt ulubiony, pozostanie naszą tajemnicą.

Otóż wydawać by się mogło, że projektowanie systemu informacyjnego zaczyna się od struktury funkcjonalnej i dalej biegnie kaskadowo po kolejnych strukturach. My jednak zaczniemy od struktury informacyjnej, skupiając się na następujących problemach:

- główna struktura problemu (formuła projektowania struktury),
- implementacja komputerowa,
- wskazówki metodologiczne.

Ponadto w opisie problematyki projektowania struktury informacyjnej umieścimy wszystkie charakterystyczne i istotne dla podejścia globalnego elementy, które będą też występowały w projektowaniu pozostałych struktur. Te elementy odpowiednio oznaczymy w trakcie wykładu i będą one miały szczególne znaczenie dla implementacji komputerowej. Te elementy zdecydują też o kształcie implementacji, która zapewni spójność, nieredundantność, kompresję podejścia.

2. Formuła projektowania struktury informacyjnej

W pierwszej kolejności wyróżniamy dwa typy systemu, który projektujemy:

1° to s y s t e m i s t n i e j ą c y, rzeczywiście lub też może być o wykoncypowanej postaci. System ten może być poznany i opisany według ustalonej konwencji specyfikacji;

2° to s y s t e m k r e o w a n y, będący "podwójnym modelem" typ istniejącego i tego przyszłego, który ma ten istniejący zastąpić. Jest to wyrażenie wiedzy projektanta w pewnym formalizmie o istniejącym stanie rzeczy, tak aby możliwa była komputerowa implementacja.

Na marginesie tego podziału dwie uwagi wyjaśniające. System istniejący w procesie projektowania p r z e c h o d z i s t o p n i o w o w system kreowany. Celem do uzyskania jest bowiem system drugi, a nie rozróżnienie między systemem pierwszym i drugim.

Sam proces projektowania przebiega w kilku fazach:

-faza pierwsza to r o z p o z n a n i e p r z e z s t r u k t u r a l i z a c j ę,

-faza druga to k r e a c j a i u s t r u k t u r a l i z o w a n i e zbiorów informacyjnych,

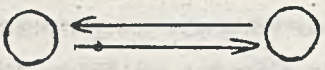
-faza trzecia to k r e a c j a d z i a ł a n i a.

W fazie pierwszej następuje rozpoznanie, badanie i strukturalizacja zbiorów zewnętrznych, branych z systemu istniejącego. Posiłkujemy się już formułą obiekt: relacja: cecha. Zbiory zewnętrzne w tej fazie przekształcane są w k a t e g o r i e o b i e k t ó w, w sposób pokazany na rysunku.

W wyniku tej fazy przyjmujemy pięć kategorii obiektów: użytkowników (tworzą krąg zewnętrzny i należą do struktury

organizacyjnej SI), wejścia, zbiory bazy danych, wyjścia i procesy (tworzą właściwą strukturę informacyjną). Dla każdego obiektu wg. kategorii znamy jego cechy i powiązania. W tym celu stosujemy specjalne mechanizmy rozpoznawania, którym przypisujemy następujący szablon:

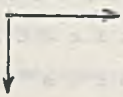
- dla powiązań technologicznych:



- dla powiązań hierarchicznych:

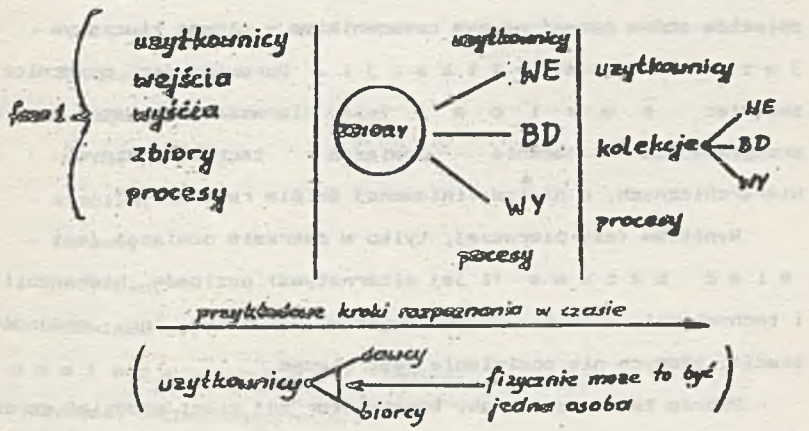
między dwoma obiektami, z tym że dla powiązań technologicznych (procesu przetwarzania informacji) powiązanie dotyczy różnych kategorii obiektów,[2)] a dla powiązań hierarchicznych - w ramach tej samej kategorii. Możemy stosować dwa alternatywne podejścia:

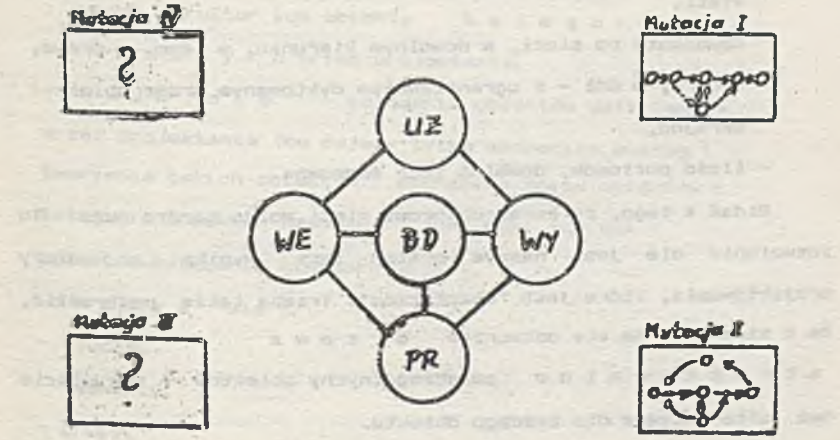
- z góry do dołu:



- z dołu do góry:

Powiązania technologiczne i hierarchiczne są podstawowymi relacjami, które rozpoznajemy stosując schemat powiązań.


faza 1
użytkownicy
wejścia
wyjścia
zbiory
procesy
użytkownicy
zbiory
WE
BD
WY
procesy
użytkownicy
kolekcje
WE
BD
WY
procesy
przykładowe kroki rozpoznania w czasie
użytkownicy
dawcy
biorcy
fizycznie może to być jedna osoba


Mutacja IV
?
Mutacja I
UŻ
WE
BD
WY
PR
Mutacja III
?
Mutacja II

Ilość różnych mutacji schematu zależy od konwencji i inwencji, a zmieniając mechanicznie kierunek strzałek otrzymujemy alternatywne podejścia. Każde powiązanie między kategoriami obiektów można nazwać pewnym czasownikiem - słowem kluczowym j ę z y k a s p e c y f i k a c j i . Wprowadzając pomocnicze pojęcie: p o z i o m , faza pierwsza wzbogaca się o przeplatające badania powiązań technologicznych i hierarchicznych, o nie zdefiniowanej ściśle regule STOP.

Wynikiem fazy pierwszej, tylko w zakresie powiązań jest s i e ć b a z o w a (i jej alternatywa) poziomów hierarchii i technologii. Sieć ta podlega rygorom formalnym (np. spójność sieci), których nie spełnienie jest błędem.

Dobrze też by było, aby konstruktor tej sieci wiedział co mu wolno:

- wejście w sieć, w dowolnym obiekcie, na dowolnym poziomie, czyli w dowolnym miejscu sieci - ściślej w dowolnym węźle sieci,
- chodzenie po sieci, w dowolnym kierunku, w lewo, w prawo, w górę, w dół - z ograniczeniem dyktowanym przez obiekty skrajne,
- ilość poziomów, dowolna lecz sensowna.

Widać z tego, że konstruktorowi sieci wolno bardzo dużo. To zezwolenie nie jest naszym gestem lecz wynika z natury projektowania, które jest "chaotyczne". Trzeba także podkreślić, że z sieci tej da się odtworzyć d r z e w a s t r u k t u r a l n e poszczególnychy obiektów - oczywiście dwa alternatywne dla każdego obiektu.

Problem cech obiektów, określenie obiektów przez ich cechy realizujemy dodając relacje typu:

- " m a z a s t o s o w a n i e " (baza normatywna

projektowania):

- standard,
- normatyw,
- wzór,
- literatura,
- dokument,
- norma,
- szablon,
- format,
- przykład,
- rozwiązanie,

- "posiada cechy" (dotyczy obiektów i powiązań technologicznych):
  - ilościowe (mierzalne),
  - jakościowe (opisowe),
- "przypisuje się" - głównie obiekty z innych struktur lub obiekty kategorii kreowanych przez projektanta,
- "kreuje się" - kategorie obiektów definiowane przez projektanta (tu dajemy tylko mechanizm obsług i tworzenia takich obiektów); mechanizm opisu obiektów z kategorii obiektów kreowanych jest taki sam jak : obiektów kategorii standardowych,
- "oznacza się":
  - klucze,
  - segmenty,
  - grupy,
- "wyróżnia się":
  - zbiory,
  - podzbiory,
  - sety.

Dwa ostatnie typy wyprzedzają tok rozważań.

Ponadto czas już powiedzieć o konieczności uwzględnienia procesów kierowania projektem, w postaci dwóch mechanizmów:

- kontrolki,
- przewodnika.

Kontrolka pokazuje stan zaawansowania projektu w czasie, w kosztach itp. oraz związuje projektantów z systemem.

Przewodnik pokazuje "następne ruchy" jakie projektant może wykonać. Podpowiada dalsze działania, dając projektantowi możliwość ich wyboru.

Dotychczas dawaliśmy "podgląd" w proces projektowania systemu (o strukturze bądź co bądź labiryntu) od strony wejścia. Natomiast trzeba zdać sprawę, że cały czas trwają "w środku" procesy selekcji informacji, jej przeglądanie, sortowanie, kontrolowanie, zliczanie, klasyfikowanie, itd. W pewnej mierze zdajemy sobie sprawę z konieczności utworzenia oprogramowania metodologicznego projektowania i z konieczności skorzystania tego oprogramowania. W systemie wspomagającym oprogramowanie to nazywa się b a n k i e m m e t o d p r o j e k t o w y c h, a dla odróżnienia dane będziemy gromadzić w b a z i e d a n y c h p r o j e k t o w y c h.

Choć można powiedzieć, że dane i metody są zorientowane na poszczególne struktury, to można korzystać z m e t o d w s p ó l n y c h. Przykładowo dla klasyfikacji i organizacji obiektów w różnych kategoriach można ustalić metody charakterystyczne dla projektowania s t r u k t u r y f u n k c j o n a l n e j i o r g a n i z a c y j n e j, korzystając z mechanizmów systemu wspomagającego. W każdym razie faza pierwsza pozwala osiągać pewną syntezę systemu strukturowego w z b i o r z e w y n i k ó w.

Faza druga to jakby uszczegółowienie pierwszej, w kierunku

tworzenia podstaw systemu kreowanego. Pojawiają się tu nowe kategorie obiektów, charakterystyczne dla systemu informatycznego. Kategorie te mogą nie występować w systemie rzeczywistym, w każdym bądź razie w jakiejś formalnej postaci. Podstawową kategorią obiektów jest tu r e k o r d .

Rekordy "powstają" na ostatnim poziomie badania powiązań technologicznych i hierarchicznych. Są związane z kolekcjami danych - wchodzą w ich skład. Ważną cechą rekordów jest ich s t a t u s (rekordy wejściowe, bazy danych, rekordy wyjściowe ect).

Kreacja i ustrukturalizowanie zbiorów w tej fazie występuje jak już wspomniano na najniższym poziomie uzyskanym w fazie pierwszej. Otrzymujemy tu e l e m e n t a r n y p r z e p ł y w s t r u m i e n i t e c h n o l o g i c z n y c h w systemie. Celem zaś jest uzyskanie, jeśli nie schematu logicznego bazy danych, to przynajmniej wszystkich niezbędnych danych do konstrukcji takiego schematu i podschematów.

Stosujemy następujące mechanizmy kreacji i strukturalizacji zbiorów, zapisane w postaci relacji:

- podział rekordu na d a n e (e l e m e n t a r n e)[3]. lub określenie rekordów przez dane - w zależności czy preferuje się podejście "z dołu do góry" czy "z góry do dołu",
- podziały pomocnicze w postaci relacji "oznaczenie": kluczy, segmentów grup i "wyróżnienia" zbiorów i podzbiorów,
- określenie z w i ą z k ó w f u n k c j o n a l n y c h (powiązań) między rekordami różnych typów,
- wprowadzenie danych "technicznych" do rekordów, a w tym nadanie im statusu.

W rezultacie operujemy ośmioma kategoriami obiekt4w w tej

fazie:

- rekordy,
- dane,
- klucze,
- segmenty,
- grupy,
- zbiory,
- podzbiory,
- sety,

oraz nowym typem związków - powiązania funkcjonalne.

Każdą nową kategorię można dodatkowo określić poprzez mechanizm opisany w fazie pierwszej, tj. poprzez sześć relacji plus mechanizm kontrolki i przewodnika. Także i tu możemy uzyskać pogłębione powiązania z innymi strukturami oraz dokonać selekcji, klasyfikacji itd. Jednak wynikiem podstawowym jest schemat logiczny i dalej ewentualnie schemat fizyczny bazy danych - oczywiście jeszcze niekompletne, przed nami bowiem cała faza trzecia.

Faza trzecia wypełnia całkowicie drugi człon formuły S - D - R i jest poświęcona projektowaniu organizacji działania wyróżnionych elementów strukturowych. Precyzyjne określenie działania jest możliwe po zdefiniowaniu elementarnych obiektów informacyjnych i określeniu struktury ich powiązań - szczególnie funkcjonalnych.

Istnieć mogą różne konwencje i formuły opisu działania. W przypadku omawianego podejścia stosujemy trzy kategorie pojęć:

- elementarny obiekt uogólniony,
- operacja,
- zdarzenie.

Dwa pierwsze pojęcia zostały zdefiniowane w fazie pierwszej i drugiej - są doprowadzone do elementarnej, niepodzielnej

postaci. Zdarzenia są tu nowym pojęciem, które formalnie możemy zidentyfikować wcześniej według następującej klasyfikacji:

PLANSZA DLA ZDARZEŃ

Między tymi kategoriami pojęć zachodzą następujące typy związków:

- modyfikuje,
- stwierdza,
- wyzwala.

Są ściśle określone reguły identyfikacji obiektów i związków między nimi. Pełne zrozumienie działania wymaga wprowadzenia specjalnej kategorii obiektu, który można określić mianem k a l e n d a r z a z z e g a r e m oraz wprowadzenie pojęć pomocniczych:

- predykatów,
- warunków,
- wskaźników.

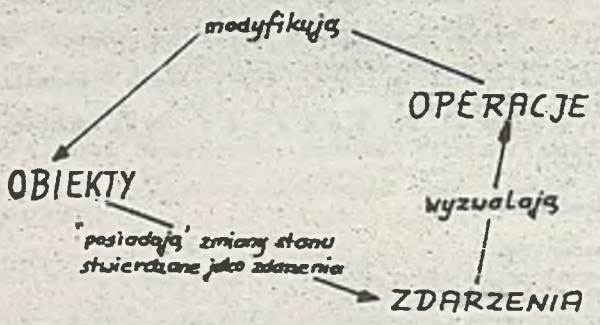
Pojęcie ZDARZENIA pozwala reprezentować zmiany OBIEKTÓW, które wywołują efektywne wykonanie operacji. Tak samo jak obiekty, również zdarzenia posiadają własności pozwalające wytwarzać je w czasie i przestrzeni oraz identyfikator, aby je odróżniać.

Wyróżnić można kilka rodzajów zdarzeń:

- czasowe - ich zajście jest skutkiem naturalnego upływu czasu,
- związane ze zmianami stanów obiektów,
- losowe (te nas nie interesują, gdyż ich przyczyny i efektynie są skończone).

Ponadto, każde zdarzenie posiada PREDYKAT, określający warunek jego wystąpienia. Pojęcie OPERACJI pozwala na modelowanie akcji zachodzących w rzeczywistości – jest działaniem, które może być wykonane przez dany czynnik, w danym miejscu i w danym momencie. Operacja jest powiązana z jednym OBIEKTEM, wykonanie wystąpienia operacji modyfikuje jedno wystąpienie danego typu obiektu. Oprócz nazwy i identyfikatora, OPERACJA dodatkowo posiada NOTATKĘ, której treść określa sposób, według którego operacja modyfikuje własności obiektu oraz WARUNKI określające możliwość zajścia danej operacji.

Przedstawione pojęcia służą do budowy sieci działań według następującej formuły:



Należy wyjaśnić, że nie każda zmiana stanu obiektu jest zdarzeniem. Dane zdarzenie może wywołać wiele różnych operacji. Dana operacja modyfikuje tylko jedno wystąpienie obiektu. Operacja może mieć charakter iteracyjny (modyfikować wiele wystąpień danego obiektu); określone jest to przez WSKAŹNIK OPERACJI. Dopiero teraz można wyjaśnić szczegółowe reguły działania.

Po pierwsze wprowadzimy standardową kategorię obiektów o nazwie KALENDARZ (Z ZEGAREM). Obiekty tej kategorii ("podziałka"

kalendarza) służą różnym celom, a w określeniu działania dla odczytu czasu i stwierdzenia stanu obiektów odczytuje się stan początkowy i stan końcowy. Pewne zmiany stanów obiektów mogą być zdarzeniami. Zdarzeniami jednak są tylko te stwierdzone zmiany stanów obiektów, które wyzwalają operacje. Sam upływ czasu może powodować zmiany stanu obiektów, które stwierdzono jako zdarzenie (wywołujące operacje).

Po drugie określa się następujące zasady spójności sieci działania:

- z każdym obiektem może być związanych wiele zdarzeń (1:n), lub wiele operacji (1:n),
- z każdym zdarzeniem może być związany jeden predykat, który określa możliwość wystąpienia zdarzenia (1:1) lub (1:0),
- wyzwalanie operacji może być związane z wieloma warunkami (1:n) lub (1:0),
- operacje (wyzwalane przez dane zdarzenie) modyfikuje tylko jedno wystąpienie obiektu (1:1),
- operacja może mieć charakter iteracyjny, krotność operacji jest określona wskaźnikiem operacji (1:1). Wskaźnik określa podzbiór wystąpień obiektu, na którym ma być wykonana dana operacja,
- każde pojęcie, każdy związek w sieci działania, może być objaśnione dodatkowym tekstem (1:1).

Pełen cykl dynamiczny sieci działań obejmuje w kolejności zbiór:

- START,
- obiekt,
- zdarzenie (ewentualnie z predykatem),
- operacja (ewentualnie z warunkami),
- obiekt,

- STOP.

Po trzecie można wyjaśnić związek tej fazy projektowania z fazą pierwszą i drugą przez przedstawienie zasad tworzenia operacji i obiektów:

- obiektami są rekordy, osobliwe rekordy bazy danych -- spełniają formalne wymagania,
- procesy są oddzielone od operacji, ale są od nich przypisywane na poziomie elementarnym,
- ostatni poziom hierarchii procesów umożliwia jednoznaczne przypisanie im przynajmniej jednego rekordu wejściowego i jednego rekordu wyjściowego,
- jest to równoznaczne z tym, że przepływ może być określany także w ramach obiektów tej samej kategorii,
- operacja jest takim kwantem procesu, który posiada dokładnie jeden rekord, na którym działa.

Faza trzecia - działanie, akceptuje wszystkie ustalenia faz poprzednich, wykorzystuje z nich potrzebne elementy i ustalenia. Formuła fazy trzeciej jest też taka, że może ona istnieć samodzielnie i może być też początkiem (wejściem) do projektowania systemu. Gwoli sprawiedliwości każda z poprzednich faz ma też tę właściwość.

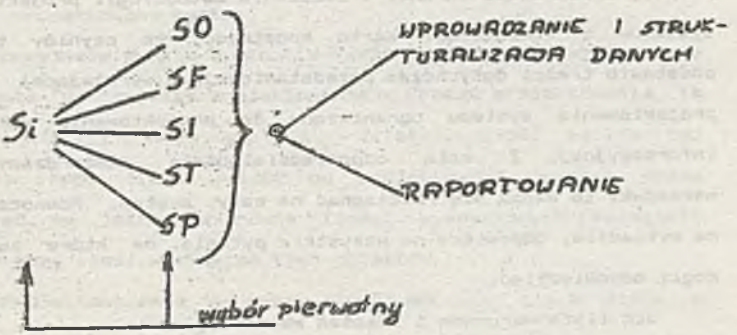
3. Implementacja komputerowa projektowania struktury informacyjnej

Implementacja komputerowa ma na celu stworzenie systemu wspomaganego projektowania SI - stosownie do przedstawionego sformułowania problemu. System wspomaganego projektowania należy widzieć dla wszystkich struktur. Oznacza to, że wszystkie mechanizmy systemu (ilustrowane dla struktury informacyjnej) są możliwe (wspólne) przy projektowaniu pozostałych struktur.

Generalny szkic systemu zakłada:

- wybór systemu projektowanego,
- wybór projektowanej struktury,

i dalej albo wprowadzanie i strukturalizowanie danych, albo raportowanie. Szkic ten ilustruje poniższy rysunek:



SWP działa w oparciu o główne elementy:

- baza danych projektowych,
- moduł wprowadzania i strukturalizacji danych (wg zdefiniowanych mechanizmów),
- moduł raportowania,
- bank metod projektowych,
- baza normatywna i metodologiczna projektowania.

System wspomaganego projektowania i jego główne elementy n i e r o z r ó ż n i a j ą s t r u k t u r . Zasoby systemu są też wspólne dla wszystkich struktur. Wspólne są też zasady (procedury) wprowadzania, strukturalizacji i raportowania. Rozróżnianie struktur następuje przez projektanta, przez formaty wejścia i przez metodologię. Bliższa ilustracja systemu w referacie [1] i poprzez prezentację na komputerze.

4. Wskazówki metodologiczne projektowania

Niniejsze wskazówki metodologiczne należy czytać na zasadzie uzupełnienia z referatem [1]. W obecnym zaś przyjmujemy pewne punkty, określające warunki i założenia metodologii projektowania systemów z komputerem. Warto spostrzec, że czynimy to na podstawie treści dotychczas przedstawionej, a wyrażającej zakres projektowania systemu ograniczony do projektowania struktury informacyjnej. Z całą odpowiedzialnością stwierdzamy, że wskazówki te dadzą się rozciągnąć na cały system. Równocześnie, na wykładzie, odpowiemy na wszystkie pytania, na które będziemy mogli odpowiedzieć.

Oto lista warunków i założeń metodologicznych:

1° Warunkiem projektowania systemu (z komputerem) jest posiadanie odpowiednio sformalizowanej metodologii projektowania - ogólnej i szczegółowej.

2° Projektowanie jest przetwarzaniem danych i wiedzy. W zależności od dojrzałości metodologicznej można tym przetwarzaniem "obdzielić" człowieka i komputer. Proporcje tego podziału pozostają kwestią otwartą. Eliminacja człowieka z projektowania jest głupotą.

3° Dzisiaj każdy (pod warunkiem 1) może stworzyć własny system projektowania typu CAD/CAM.

4° Projektowanie jest zmienną. Stan tej zmiennej obserwujemy na tle trzech zmiennych pomocniczych:

- czasu,
- kosztów,
- wzorca,

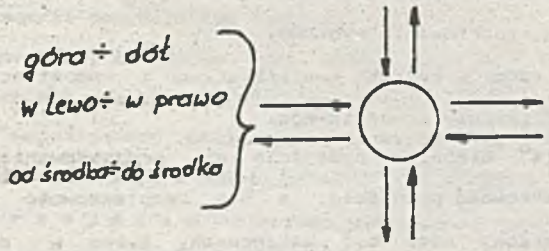
jako podstawowych elementów kierowania projektem (baza odniesienia). Ta baza odniesienia jest z m i e n n ą norma sterowania projektem. Norma sterowania jest wmontowana do systemu

wspomaganego projektowania, jednak jej pewna część zawsze pozostanie poza tym systemem.

5° Pod warunkiem 1, projektowanie jest procesem chaotycznym. Oznacza to, że istnieje dowolna ilość wejść do projektu, w dowolnym jego miejscu. Dowolna to nie nieskończona. Dowolność określa projektowany system.

6° Przyjęcie punktu 5 oznacza także swobodę poruszania się projektanta "po procesie projektowania". Proces projektowania to a l t e r n a t y w n a s i e ć działań. Ilość węzłów tej sieci jest równa ilości wejść do projektu. Z grubsza można powiedzieć, że jest ona równa ilości wyznaczonych kategorii obiektów razy ilość wystąpień tych obiektów.

7° Projektant może (po wykonaniu 5) poruszać się w sieci w dowolnym kierunku:



Może zmieniać swój sposób chodzenia po sieci, zakreślać swój teren, wskakiwać (poprzez 5) w różne miejsca - aż do spełnienia rozwiązania.

8° Alternatywna sieć działań ma początkowo charakter eksplozyjny - potem implozyjny.

9° Każdy "ruch" projektanta w sieci działań w każdym kierunku powoduje, że można wygenerować "ruch" przeciwstawny - automatycznie. Swobodę ruchów ograniczają: granice systemu, granice kategorii obiektów i o g r a n i c z e n i a (i n t e l e k t u a l n e) kierownika.

10° Projektowanie "od prawidłowego początku - do prawidłowego końca" jest szczególnym przypadkiem chaosu w projektowaniu.

11° Punkty od 5 do 10 są niczym innym jak wskazówkami dla kierowania projektem. Kierowanie projektem, posługując się s t r u k t u r a l i z a c j a , wyprowadza projektowanie z odchłani chaosu w porządek. Czyniąc w ten sposób działanie sprawnym, skutecznym, wydajniejszym. Kierownikowi też pomaga zaprzyjaźniony komputer.

12° Kierownik jest s t a n d a r d o w ą k a t e g o r i ą systemu wspomaganego projektowania.

13° Posługiwanie się systemem wspomaganego projektowania to interaktywne seanse:

- wprowadzania danych,
- restrukturalizacji danych,
- raportowania wyników,

przez ludzi o różnych kwalifikacjach i kompetencjach tworzących całość zespołu projektowego.

14° Globalne podejście do projektowania SI zakłada kompleksowość podejścia, a nie kompleksowość systemu. System kompleksowy może być ustanowiony tylko w drodze rozwoju. Kompleksowość podejścia to odejście od jednostronności. Zapewnienie, że system jest realizowany:

- pod użytkownika,
- pod komputer,
- pod projektanta,

jest zapewnione przez ustanowienie początkowe czterech (i piątej) struktur.

15° Każdy system (a zatem i każdy projekt) ma te struktury - ich ignorowanie (pomijanie jakiejś struktury) daje rozwiązanie niepełne i jednostronne.

5. Projektowanie w pozostałych strukturach

5.1. Uwagi ogólne

Projektowanie w pozostałych strukturach przejmuje w całości mechanizmy opisu i strukturalizacji - na wykorzystywanie ich mogą być jednak położone różne akcenty. "Dołożenie" nowej struktury pozwala na stwierdzenie, że w banku metod mogą być w s p ó l n e metody wykorzystywane w projektowaniu struktur. Mogą być metody specyficzne.

Pamiętajmy też, że każda struktura to odmienny punkt widzenia na ten sam system. Ten odmienny punkt widzenia przejawia się w typach kategorii obiektów, choć mogą być znowu kategorie nazwowo wspólne lecz rozpatrywane w innym kontekście. Zatem o treści takiej kategorii decyduje też kontekst.

Wyróżnione struktury to tylko inny sposób poznania tego samego systemu. Ich byt nie jest i z o l o w a n y. Ustanowione już w systemie wspomaganego projektowania mechanizmy dają możliwość swobodnego przejścia z jednej srtuktury do drugiej. Ta r e l a c j a p r z e j ś c i a[4)] nie jest normatywnie ustalona - wynika z potrzeby i metodologii projektowania.

W niniejszym wykładzie o projektowaniu powiemy tylko to co niezbędne, tzn.:

- zdefiniujemy struktury,
- określimy kategorie obiektów,
- określimy problem projektowy,
- ewentualnie wskażemy na powiązania preferowane.

Natomiast w toku samego wykładu (także poza nim - na specjalne życzenie publiczności) prezentowane będą rozwiązania, które oferować będzie nasz system wspomaganego projektowania.

5.2. Projektowanie struktury funkcjonalnej systemu

Definicja: Struktura funkcjonalna jest systemem zadań przetwarzania danych.

Jest to najkrótsza definicja tej struktury, jaką udało się wymyśleć. Jeśli wsłuchamy się w nią wnikliwie — to uznamy ją za bardzo dobrą. Budowanie struktury funkcjonalnej to projektowanie systemu pod użytkownika. Struktura funkcjonalna odzwierciedla cel, treść i z a d a n i a przetwarzania danych, przez które cel i treść systemu się realizują. Dla projektowania zadania są najważniejsze - stanowią ten element, którym projektant może manipulować. Cel i treść muszą pozostać niezmienione (jako zbiór wymagań użytkownika). Mogą co najwyżej zostać ulepszone. Projektant może dać więcej niż chce użytkownik.

Określmy główne kategorie obiektów występujące w projektowaniu struktury funkcjonalnej i ich wzajemne związki. Po pierwsze z a d a n i e p r z e t w a r z a n i a d a n y c h, które traktujemy jako pojęcie pierwotne. Zadanie jest określone przez nazwę i repertuar cech, które temu zadaniu mogą być przypisane. C e c h y są drugą kategorią obiektów mającą zastosowanie także w projektowaniu struktury funkcjonalnej. W projektowaniu SF rozróżniamy cechy ilościowe i jakościowe. Zadanie przetwarzania danych ma (systemie rzeczywistym) swego wykonawcę lub wykonawców. W y k o n a w c a z a d a n i a jest trzecią kategorią obiektów SF. Wykonawca ma też cechy. Zadania są agregowane i dezagregowane. Rozpoznanie następuje przez strukturalizację.

Treść tak rozumianego zadania nie jest sformalizowana. Artykułuje ją w zasadzie użytkownik. Zadanie ma charakter postulatywny: "należy w systemie robić to, to i to". Oczywiście owo "to, to i to" realizuje się poprzez pewien proces. Każde

zadanie ma swój p r o c e s , który dla nas jest następną kategorią obiektów. Procesy określają j a k zadania są realizowane, jakimi sposobami. Opis procesów powinien być bardziej ścisły, ale bez przesady.

Ściśle już definiuje zadania i dalej procesy a l g o r y t m. Algorytmy to nowa kategoria obiektów struktury funkcjonalnej. Algorytm może "obsługiwać" kilka zadań oraz jedno zadanie może mieć alternatywne (lub zapasowe) algorytmy.

Wyróżnione kategorie "zagnieżdżają się" w sobie a w obrębie tej samej kategorii łączą się między sobą tworząc s i e ć s t r u k t u r y f u n k c j o n a l n e j . Tym co łączy zadanie, procesy, algorytmy w sieci struktury funkcjonalnej są k a n a ł y p r z e s y ł a n i a d a n y c h . Kanały i dane (które "płyną" kanałami) są kategoriami obiektów SF. Celem zabiegów jest optymalny podział sieci struktury funkcjonalnej na p o d s y s t e m y f u n k c j o n a l n e ; optymalny ze względu na przyjętą strategię podziału i sytuację.

Przyjmuje się dwie strategie podziału:

- podział ze względu na podobieństwo obiektów,
- podział ze względu na powiązania obiektów.

Reguły decyzyjne podziałów odpowiednio brzmią:

1^{o} Łącz w podsystemy funkcjonalne zadania (alternatywnie procesy lub algorytmy) najbardziej do siebie podobne.

2^{o} Łącz w podsystemy funkcjonalne zadania (alternatywnie procesy lub algorytmy) najbardziej ze sobą powiązane - minimalizując powiązania między podsystemami.

Sytuacje podaje się przez stopień ustrukturalizowania:

1^{o} Sytuacja dobrze ustrukturalizowana: znana jest ilość podsystemów funkcjonalnych i ilość w nich zadań (procesów, algorytmów).

2^{o} sytuacja średnio ustrukturalizowana: znana jest tylko

ilość podsystemów.,

3° sytuacja słabo ustrukturalizowana: nic nie jest znane a priorii.

Istnieje też conajmniej kilka przypadków szczególnych, które z powodzeniem udaje się nam rozwiązać.

Uwagi szczegółowe:

1° Projektant określa tożsamość (lub związek): wykonawca - użytkownik (w SI).

2° Projektant określa tożsamość (lub związek) : kanał - powiązanie technologiczne (w SI).

3° Projektant ustala podział danych na kolekcje i dalszy sposób egzystencji danych (z SF).

4° Jak wyżej w odniesieniu do procesów i algorytmów.

5° Projektant ma perspektywę wykorzystania metod stosowanych w projektowaniu SF do projektowania innych struktur systemu informatycznego.

6° Najważniejsza jest analiza wyników i ich wykorzystanie jako podstawy w dalszym projektowaniu. W wyniku bowiem uzyskaliśmy podział kompleksu projektowego na elementy łatwiejsze do opanowania.

5.3. Projektowanie struktury technicznej systemu

Definicja: Struktura techniczna jest systemem liczącym systemu informatycznego.

Ustalenie: Kategorią standardową struktury technicznej jest urządzenie.

Techniczna realizacja systemu stwarza w pewnych przypadkach w projektowaniu struktury technicznej sytuację, która jest dla projektanta niestandardowa - absolutnie. W szczególności istniejące tu problemy z zakresu projektowania techniki wymagają

angażowania specjalistycznych zespołów np. z dziedziny informatyki technicznej, teleinformatyki. Te specjalistyczne zagadnienia zostają wyłączone z wykładu - są dla nas niestandardowe - absolutnie. Mimo to pozostaje otwarte pytanie, jakie problemy w projektowaniu struktury technicznej SI roztrzyga projektant ?

Samo pojęcie struktury technicznej można potocznie określić jako "techniczne środki realizacji systemu informacyjnego". Bliżej zaś, struktura techniczna systemu jest zbiorem wzajemnie połączonych urządzeń przetwarzania informacji, mogących w oparciu o wyposażenie programowe wykonywać zadania systemu informacyjnego poprzez realizację logicznych iarytmetycznych operacji na danych. Wśród operacji na danych wyróżniać się będzie:

- identyfikację, jako określenie danych na podstawie analizy znaków lub kontekstu,
- kodowanie i dekodowanie, jako zmianę znaków na sygnały i odwrotnie,
- przetwarzanie, jako zmianę jednego rodzaju danych na inny zgodnie z określonymi regułami,
- przesyłanie, jako zmianę współrzędnych przestrzennych danych,
- przechowywanie, jako zmianę współrzędnych czasowych danych.

Operacje te wykonywane są przez techniczne środki realizacji systemu dzięki oprogramowaniu, które możemy podzielić na :

- programy systemowe (techniczne),
- programy użytkowe.

Programy systemowe są przypisane wypełnianym przez sprzęt funkcjom. Natomiast oprogramowanie użytkowe reprezentuje specyficzne funkcje systemu informacyjnego.

Projektowanie tak określonej struktury technicznej jest w

istocie rzeczy o r g a n i z o w a n i e m s y s t e m u l i c z ą c e g o , w którym na czoło wybijają się dwa powiązane zagadnienia:

- przedstawienie u k ł a d u danych wejściowych do projektowania o odpowiedniej strukturze,
- problem wyboru konfiguracji systemu liczącego.

Układ danych wejściowych z jednej strony winien określać zapotrzebowanie systemu na "moc obliczeniową", a z drugiej prezentować charakterystyki środków technicznych. Jest więc to bilans, będący punktem wyjścia do ustaleń projektowych, w którym:

- "lewą" stronę stanowi opis zadań przetwarzania danych wykonany w określony sposób,
- stosowny opis sprzętu jest "prawą" stroną tego bilansu.

Adekwatne dostosowanie systemu liczącego do zadań jest rozwiązaniem problemu projektowania struktury technicznej sustemu. To dostosowanie polaega na wyborze konfiguracji systemu liczącego.

5.4. Projektowanie struktury przestrzennej systemu

Definicja: Struktura przestrzenna jest systemem komunikacji systemu informatycznego.

Projektowanie struktury przestrzennej jest w istocie tworzeniem s y s t e m u k o m u n i k a c y j n e g o , gdyż chodzi tu o optymalne rozmieszczenie elementów systemu w środowisku i zapewnienie połączeń między nimi, które są niczym innym, jak drogami przesyłania danych. Wynikiem rozmieszczenia jest sieć, której węzłami mogą być:

- źródła danych,
- ujęcia danych,
- punkty koncentracji danych,

- punkty przechowywania danych,
- punkty przetwarzania danych,
- punkty dyspozycji danych,
- ujścia danych,
- punkty wykorzystywania danych.

Punkty te są połączone kanałami przesyłania danych. Punkty i kanały traktujemy jako kategorie obiektów struktury przestrzennej.

Punktami, w których system poprzez strukturę przestrzenną, styka się z obiektem - m i e j s c a m i s t y k u są dwa rodzaje punktów:

- źródła danych,
- punkty wykorzystywania danych.

Kojarzyć je możemy z komórkami s t r u k t u r y o r g a n i z a c y j n e j lub z użytkownikami systemu poprzez r e l a c j e przejścia. Ź r ó d ł a d a n y c h są stanowiskami emisji danych źródłowych dla systemu przez obiekt lub jego otoczenie, które następnie będą w określony sposób, kanałami przesyłania danych dostarczane do systemu.

P u n k t y w y k o r z y s t y w a n i a d a n y c h są natomiast punktami, do których dane w postaci przetworzonej mają być dostarczone. Zarówno źródła danych, jak i punkty wykorzystywania są komórkami organizacyjnymi użytkownika lub otoczenia i nie wchodzą w skład systemu. Wyodrębnienie tych punktów jest potrzebne dla projektowania "warstwy nośnej" systemu. Ponadto jest to konsekwencja wcześniejszych ustaleń w strukturze funkcjonalnej i informacyjnej, wyraziście określająca granice systemu. Wynika stąd, że źródła danych i punkty wykorzystywania są u s t a l o n e .

Każdy z wyróżnionych punktów charakteryzuje się za pomocą pewnych wielkości identyfikacyjnych (cech). W szczególności będą

to współrzędna przestrzeni (np. siatka kilometrowa, numery komórek organizacyjnych, numery pokojów itd.) i różne miary intensywności danych, np. ilość danych znormalizowanych na wejściu lub wyjściu w określonej jednostce czasu.

Problem projektowy polega na wyborze optymalnej struktury przestrzennej dla danego systemu, poprzez odpowiednie r o z m i e s z c z e n i e jego obiektów w środowisku. Działanie systemu projektującego przebiegać może wg następujących etapów:

- identyfikacja źródeł danych i punktów wykorzystania;
- wyodrębnienie pozostałych punktów,
- "zagnieżdżenie" w/w punktach stosownych obiektów innych struktur (relacje przejścia, przenikania, przypisania),
- ustalenie połączeń,
- optymalizacja struktury.

Jest to w zasadzie problem wyznaczania układu komunikacyjnego dla danego systemu, zasadniczo rozwiązywalny metodami programowania całkowitoliczbowego. Jednakże odrębne projektowania struktury przestrzennej, w oderwaniu od całego systemu, nie jest dopuszczalne. Konieczne jest podejście integracyjne - łączne z pozostałymi strukturami.

5.4. Projektowanie struktury organizacyjnej w systemie

O projektowaniu struktury organizacyjnej opowiemy w czasie wykładu.

6. Zakończenie

Niniejsze opracowanie należy traktować na zasadzie notatek z wykładu. Sam wykład będzie rozwijał pewne elementy treściowe, a inne mogą być pominięte.

Literatura:

[1] L. Szydłowski: Metoda wspomaganego projektowania systemów informatycznych, Druga Wiosenna Szkoła PTI, Świnoujście 1989

Przypisy:

1) Ogólna metodologia projektowania definiuje m. in. założenie struktur, ich współdziałanie, klasy itd.

2) W obrębie wyróżnionych pięciu kategorii obiektów, potem zrezygnujemy z tego ograniczenia.

3) Dana elementarna jest następną standardową kategorią obiektów.

4) Relacja przejścia umożliwia łączenie obiektów w różnych strukturach. Jej konkretna treść może być przypisaniem, tożsamością, stawaniem się, połączeniem.

METODY TWORZENIA SYSTEMOW EKSPERTOWYCH

Andrzej Kubacki, Marek Valenta, Kazimierz Leśniak, Andrzej Potępa
Instytut Informatyki AGH w Krakowie

1. WSTĘP

Próby wykorzystanie komputera do przetwarzania danych symbolicznych oraz do rozwiązywania problemów rozpoczęły się pod koniec lat piecdziesiątych naszego wieku [Charniak 85]. Termin sztuczna inteligencja (artificial intelligence - AI) pochodzi z pracy "Steps Toward Artificial Intelligence", która M. Minsky opublikował w 1961 roku. Idea nie jest wytworem naszych czasów. Legenda o Pigmalionie, Frankenstein - Shelley, Golem - Capka czy automat Kempelena to wyrazy ludzkiego dążenia do stworzenia myślącej maszyny. Ale dopiero pojawienie się komputera otworzyło nowe perspektywy - fantazja uzyskała narzędzie, które może ją urzeczywistnić.

1.1. SIECI NEURONOWE

W latach 50. podjęto probę stworzenia myślącego systemu komputerowego poprzez symulowanie pracy mózgu. Wyróżniającym się efektem tych prac był system PERCEPTRON imitujący pracę siatkówki oka, a umożliwiający rozpoznawanie pewnej klasy obrazów.

Mała wiedza o budowie i pracy mózgu, ogromna ilość tak komorek nerwowych w nim zawartych jak i połączeń miedzy nimi, spowodowały krytykę sieci neuronowych i zmniejszenie (znaczne) zainteresowania tym tematem. Ostatnio jednak ta idea wydaje się znów interesować naukowców.

1.2. WYSZUKIWANIE HEURYSTYCZNE

A. Newell oraz H.Simon (Carnegie - Mellon University) oraz ich dzieło GPS (General Problem Solver) to przedstawiciele nowego kierunku badań AI. W ich pracach myślenie człowieka zostało przedstawione jako manipulacje symbolami (porównywanie, wyszukiwanie, modyfikowanie), a rozwiązywanie problemu jako wyszukiwanie go spośród zbioru możliwych rozwiązań sterowane poprzez użycie reguł heurystycznych [Forsyth 84], [Shirai 84].

GPS radził sobie dobrze z małymi problemami, w których istniały dobrze określone reguły postępowania i mała przestrzeń wyszukiwania.

1.3. REPREZENTACJA WIEDZY

GPS nie umożliwiał jednak rozwiązywania problemów, z którymi człowiek spotyka się w praktyce. E.Feigenbaum (Stanford) zaproponował nowe podejście. Ponieważ próby stworzenia systemów, które byłyby na tyle uniwersalne, aby przy ich pomocy rozwiązać każdy problem, nie dawały rezultatów, zaproponował ograniczenie obszaru ich stosowalności. Kosztem uniwersalności, uzyskano rzeczywistą użyteczność.

Tak narodziły się systemy ekspertowe (SE), którym początek dał DENDRAL, a których standardowym przedstawicielem jest MYCIN - system do diagnozowania infekcji bakteryjnych [Shortliffe 76]. Wiedza systemu o dziedzinie której on dotyczy została oddzielona od mechanizmów wnioskowania co ,w połączeniu z ograniczonym obszarem zastosowania , okazało się podstawą komercyjnego sukcesu.

1.4. PERSPEKTYWY

Systemy ekspertowe są efektem prac lat siedemdziesiątych w ramach AI. W latach osiemdziesiątych ciężar badań przeniósł się na problemy związane z automatycznym uczeniem się.D.Lenat (Stanford) stworzył system uczący się o nazwie EURISKO, który automatycznie rozszerza zbiór reguł heurestycznych, którymi się posługuje.

Tak jak rozpoznawanie obrazów czy robotyka wywodzące się z AI, tak i SE wydają się rozpoczynać życie niezależnie od głównych nurtów AI.

Perspektywy rozwoju SE skupiają się w następujących kierunkach:

- Rozwój zastosowań komercyjnych,
- Zastosowanie architektur wieloprocesorowych,
- Automatyczne uczenie się.

2. SYSTEMY EKSPERTOWE

Burzliwy rozwój SE datuje się od prac Feigenbauma i stworzonego przez niego systemu DENDRAL (chemia, spektrografia). Od tego czasu powstało wiele innych systemów, które z powodzeniem znalazły praktyczne zastosowanie i przynoszą wiele wymiernych korzyści.

2.1 DEFINICJA SE

Systemy ekspertowe są systemami komputerowymi opartymi na wiedzy, umożliwiającymi rozwiązywanie problemów z wąskiej dziedziny ich zastosowania z biegłością eksperta [Sell 85].

Podkreślenia wymagają 3 aspekty powyższej definicji stanowiące istotę SE:

a. Wiedza - rozumiana jako reguły postępowania, fakty, wierzenia, hipotezy, heurystyki z uwzględnieniem jej niepełności i niepewności [Hayes-Roth 83], [Fox 86].

b. Dziedzina zastosowania; SE nie posiada wiedzy ogólnej. Jego wiedza dotyczy tylko wąskiej dobrze określonej w swych ramach dziedziny. Przyczyny tego faktu są dyskutowane m.in. w [Hayes-Roth 83], [Sell 85].

c. Przeznaczenie; zadaniem SE jest rozwiązywanie lub pomoc w rozwiązywaniu problemów - z biegłością specjalisty w danej dziedzinie.

Dodatkowe cechy charakteryzujące SE i decydujące o jego użyteczności to m.in. [Hayes-Roth 83]:

- umiejętność rozwiązywania trudnych problemów,
- interakcyjny sposób korzystania z systemu,
- umiejętność objaśniania toku rozumowania,
- dopasowanie czasu trwania ekspertyzy do potrzeb,
- implementacja umożliwiająca łatwą rozbudowę i modyfikację,
- "przyjazny" człowiekowi interfejs.

Na szczególną uwagę zasługuje umiejętność objaśniania toku rozumowania. Jest to w systemach komputerowych nowość, o brzemiennych, pozytywnych, skutkach psychologicznych wynikających z uwiarygodnienia wyników uzyskanych za pomocą komputera poprzez danie możliwości prześledzenia jego pracy i porównania toku rozumowania systemu z tokiem myślenia użytkownika (lub stwierdzeniu sensowności rozumowania).

2.2. KLASYFIKACJA SYSTEMÓW EKSPERTOWYCH

Możliwe są różne klasyfikacje SE. Najczęściej spotyka się klasyfikacje oparte o:

1. dziedzinę zastosowania SE :
   - medycyna (MYCIN, PUFF, VM, INTERNIST,..),
   - chemia (DENDRAL, SECS, MOLGEN),
   - geologia (PROSPECTOR, Dipmeter Advisor),
   - elektronika (NA, SL, SOPHIE),
   - matematyka (AM),
   - obronność,
   - itd.
2. rodzaj strategii użytej do rozwiązywania problemu [Haeys-Roth 83]:
   - interpretacja; określenie rodzaju zaistniałej sytuacji na podstawie danych sensorycznych,
   - przewidywanie; wnioskowanie o możliwych konsekwencjach danej sytuacji,
   - diagnostyka; wnioskowanie o nieprawidłowym funkcjonowaniu obiektów na podstawie obserwacji,

- projektowanie;	konfigurowanie obiektów zgodnie z określonymi zasadami,
- planowanie;	projektowanie akcji, określanie kolejności wykonywania czynności,
- śledzenie;	porównywanie obserwacji w celu określenia "nieprawidłowości"
- odpluskwianie;	określanie środków zaradczych w celu usunięcia nieprawidłowości,
- poprawianie;	wykonanie czynności umożliwiających wykorzystanie środków zaradczych,
- instruowanie;	diagnostyka, odpluskwianie i poprawianie zachowania ucznia w procesie nauczania,
- sterowanie;	interpretacja, przewidywanie, poprawianie i śledzenie zachowania systemu.

Inne, rzadziej spotykanie klasyfikacje opierają się na rozmiarach systemu [King 85] , rodzaju zastosowanych mechanizmów wnioskowania, sposobu reprezentacji wiedzy itp.

1.2.3 ARCHITEKTURA SE

W architekturze SE można wyróżnić 5 podstawowych modułów:

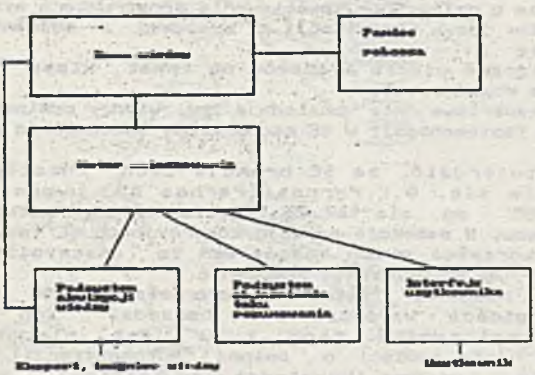
1. baza wiedzy (knowledge base) - zawierająca wiedzę z dziedziny wiedzy zastosowania SE (fakty, reguły, wyniki pośrednie toku rozumowania - pamięć robocza (working memory)).

2. motor wnioskowania (inference engine) - zawierający strategie wnioskowania i umożliwiający sterowanie procesem rozwiązywania problemu,

3. podsystem akwizycji (knowledge acquisition subsystem) - zespół środków umożliwiających automatyzację procesu pozyskiwania wiedzy z jej źródeł i transpozycję do postaci czytelnej dla systemu komputerowego.

4. podsystem objaśniający (explantation subsystem) - umożliwiający prezentację użytkownikowi SE sposobu rozumowania systemu,

5. interfejs użytkownika (user interface) - środki umożliwiające komunikację użytkownika z SE (menu, język komunikacji, grafika itp.).



Rys. 1. Architektura SE

Cecha charakterystyczna SE jest oddzielenie wiedzy zawartej w bazie wiedzy od motoru wnioskowania. Zapewnia to łatwą modyfikację wiedzy systemu i stanowi główną podstawę powstawania narzędzi do budowy SE (shell) takich jak EMYCIN, M.1, TIMM itd.

2.4 PODSTAWOWE OGRANICZENIA TECHNOLOGII SE

Ograniczenia SE wynikają przede wszystkim z trzech czynników. Po pierwsze, wiedza na temat procesów myślenia i inteligencji jest zbyt mała. Po drugie, zbyt niedoskonałe i nieefektywne są metody komputerowej reprezentacji pojęć i mechanizmów związanych z rozwiązywaniem problemów przez człowieka. Po trzecie struktura wiedzy ludzkiej jest niezwykle złożona i słabo poznana.

Ograniczenia SE są następujące [Hayes-Roth 83], [Waterman 85]:

- mały obszar dziedziny zastosowania,

- brak adekwatnych środków do wyrażania faktów i relacji (szczególnie czasowych i przestrzennych),
- mała wiedza o dziedzinie zastosowania konstruktora systemu,
- nienaturalny język komunikacji z systemem i objaśnianie toku rozumowania
- niedostateczna wiedza systemów na temat klasy problemów jakie mogą rozwiązywać,
- systemy ekspertowe nie posiadają tzw. wiedzy ogólnej,
- trudności implementacji w SE mechanizmów samouczenia się.

Można stwierdzić, że SE brakuje cech inteligentnego zachowywania się. D.L.Parnas [Parnas 85] uważa, że AI w kontekście SE ma się tak do inteligencji jak sztuczny kwiat do prawdziwego. W aspekcie rozwoju komercyjnych SE (ok. 500 firm w USA) i korzyści z ich zastosowań ta uszczypliwa uwaga Parnasa nie ma praktycznego znaczenia.

Inne niemniej istotne ograniczenia SE wynikają z niedoskonałości współczesnych narzędzi ich budowy. Głównym ograniczeniem tego typu jest niedostateczne wspomaganie (nie mówiąc o pełnej automatyzacji) procesów pozyskiwania wiedzy (knowledge acquisition), który ma podstawowy wpływ na jakość systemu z użytkowego punktu widzenia.

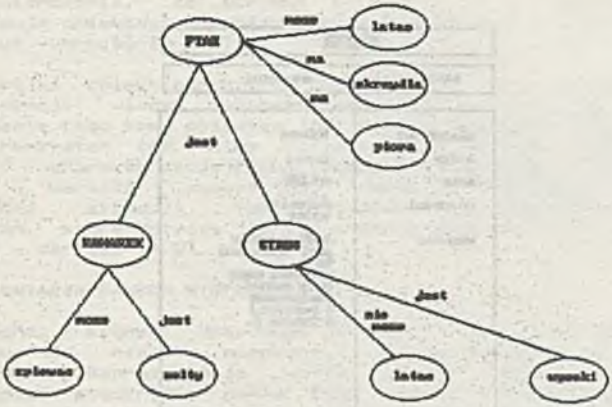
Poważne trudności wiążą się z testowaniem i walidacją wiedzy systemu ekspertowego. Istniejące rozwiązania dostępne w narzędziach budowy SE ciągle można uważać za niedoskonałe.

3. REPREZENTACJA WIEDZY

Dla potrzeb SE wiedzę z danego obszaru i jego zastosowania należy odpowiednio reprezentować w bazie wiedzy. Należy zwrócić uwagę na fakt, że dotychczas wszystkie przedstawione poniżej sposoby reprezentacji wiedzy są niedoskonałe.

3.1. SIECI SEMANTYCZNE

Najstarszą i najbardziej ogólną metodą reprezentacji wiedzy wywodzącą się z psychologii [Lindsay 84] są sieci semantyczne (semantic network)[Nilsson 80], [Shirai 86], [Charniak 85].Sieć semantyczna jest zbiorem obiektów nazywanych węzłami (nodes) połączonych za pomocą skierowanych krawędzi (links).



Rys 2. Przykład sieci semantycznej

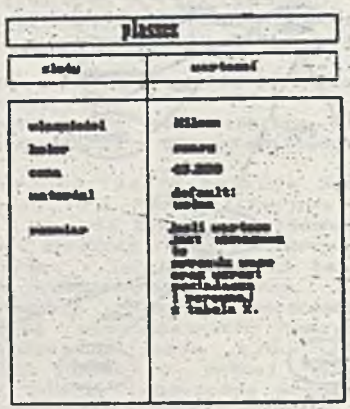
Węzły używane są do reprezentcji obiektów i opisów. Obiekty reprezentują fizyczne rzeczy (domy, ludzie, drzewa), wydarzenia, czynności, lub pojęcia abstrakcyjne. Opisy dostarczają dodatkowej informacji o obiektach.

Krawędzie reprezentują dowolne relacje między obiektami i opisami. Podstawowymi relacjami są:
- SS (relacja zawierania),
- EL (relacja przynależności),
- EQ (relacja równoważności).

Metody reprezentacji wiedzy w postaci sieci semantycznej są bardzo elastyczne. Dzięki temu nowe węzły lub krawędzie mogą być łatwo dodane do sieci.Charakterystyczne dla sieci semantycznych jest tzw. dziedziczenie cech. Właściwość ta polega na tym, że węzeł dziedziczy cechy innego węzła z nim związanego.

3.2 RAMY (frame)

Rama (M.Minsky) stanowi kompletny opis obiektu zawarty w tzw. slotach. Sloty mogą zawierać informacje defaultowe przypisane obiektowi, wskaźniki (pointers) do innych ramek zbiory reguł lub procedury.



Rys. 3. Przykład ramy

Defaultowa zawartość slotu ramki zakłada, że jeśli nie zostanie "uzyskana przez system" sprzeczna z nią informacja, to należy ją przyjąć jako obowiązującą. Wartości defaultowe są użyteczne w przypadkach, gdy reprezentacja dotyczy wiedzy z małą ilością wyjątków.

Dołączenie proceduralne (procedural attachment) jest pewną metodą uzyskania wartości slotu. W przypadku tym slot zawiera procedurę (np. wzór matematyczny) determinującą sposób obliczania jego wartości. Instrukcja natomiast może powodować dołączenie wartości z innych slotów lub ramek. Slot może także zawierać regułę wnioskowania typu IF - THEN.

Głównymi ograniczeniami stosowania ramek jest złożoność implementacji modułu motoru wnioskowania, który wnioskuje w oparciu o taką reprezentację wiedzy.

3.3. STWIERDZENIA i REGUŁY

Jednym z głównych elementów bazy wiedzy są zbiory stwierdzeń (faktów) opisujących to, co zaszło lub zachodzi w rzeczywistości. Dotyczą one zdarzeń, zjawisk, objawów, czynów, określonych stanów rzeczy itp. [Cholewa 87].

Stwierdzenie, ze atrybut o nazwie A i wartości W przysługuje obiektowi O zapisywane jest w postaci trójki obiekt - atrybut -wartość (O-A-W).

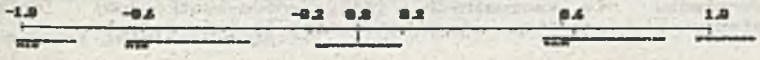
Trójka obiekt-atrybut-wartość (O-A-W) jest schematem reprezentacji wiedzy zastosowanym m.in. w systemie MYCIN. W schemacie tego typu obiektem jest dowolny obiekt fizyczny (np. dom, tranzystor itp.) albo pojęcie (np. bramka logiczna). Atrybuty charakteryzują cechy związane z obiektami (rozmiar, kolor). Wartość specyfikuje nazwę atrybutu w rozważanej sytuacji (atrybut=kolor, wartość=czerwony). Dla zdań, które dotyczą jednego obiektu stosuje się parę atrybut - wartość (A-W).

REPREZENTACJA DANYCH NIEPEWNYCH

Dane, którymi dysponuje ekspert przy rozwiązywaniu problemu są często niepewne, nieprecyzyjne, a nawet przeciwstawne. Reprezentacja wiedzy w SE musi ten fakt uwzględniać stadtrójka O-A-W jest często rozszerzona o dodatkowy element tak zwany współczynnik pewności (CF - certainty factor).

CF reprezentuje stopień pewności wartości przypisanej danemu atrybutowi obiektu. W zdaniu: Kanarek jest prawdopodobnie żółty słowo "prawdopodobnie" określa stopień pewności z jakim określamy kolor kanarka.

Jednym ze sposobów reprezentacji CF, jest sposób przyjęty w systemie MYCIN. Obrazuje to rysunek 4.



Rys. 4. Stopień pewności reprezentowany przez wartości CF w systemie MYCIN

Innym interesującym sposobem reprezentacji niepewności wiedzy jest wykorzystanie teorii zbiorów rozmytych (System SPERIL).

Tematowi niepewności w rozwiązywaniu problemów i związanych z tym zagadnieniem poświęcono wiele prac m.in. [Hunt 84], [Takeguchi 84].

REGUŁY

Najczęściej spotykaną metodą reprezentacji wiedzy są reguły. Metoda ta opiera się na użyciu zdań
IF (warunek) THEN (akcja).

Na przykład:
1) if the patient was an insulator befor 1965, then the patient directly handled asbestos.
2) if the patient directly handled asbestos and the patient was exposed in confined spaces, then the patient had a severe exposure.

Przesłanki złożone są z jednego lub wielu prostych zdań logicznych. Zdania te mogą być połączone zarówno operatorem and jak i or. Przesłanka określa warunki, których spełnienie pozwala na uznanie konkluzji za prawdziwą. Warunki w regułach określone są dla trójek O-A-W.

Systemy oparte na regułach (RBS - Rule Based System) charakteryzują się m. inn. [Hayes-Roth 85]
- odwzorowywaniem wiedzy dziedzinowej w reguły if - then,
- ich możliwości zwiększają się proporcjonalnie do zwiększania ilości reguł,
- mogą rozwiązywać różnorodne problemy,
- łatwo do nich konstruować motor wnioskowania.

Podstawowe ograniczenie stosowania RBS jest związane z ilością reguł potrzebnych do ekspertyzy. Jak podaje F.Hayes-Roth [Fox 86]:
- interesująca demonstracja rozwiązywania problemu wymaga ok. 50 reguł,
- profesjonalny poziom ekspertyzy osiąga się przy ok. 10.000 regułach,
- laureat nagrody Nobla posługuje się przy rozwiązywaniu problemów regułami w liczbie ok. 100 000.

Drugim ważnym ograniczeniem stosowania RBS jest trudność w wyrażaniu nie kauzualnych (przyczynowo-skutkowych) relacji miedzy obiektami (tzw. wiedzy głębokiej [Coombs 84]).

3.4 METAWIEDZA

Mowiąc najbardziej ogólnie metawiedza jest wiedzą o wiedzy, a rozważana w kontekście SE regułowych jest wiedzą o wiedzy zawartej w tym systemie wyrażoną za pomocą metareguł. Przykładami metareguł mogą być [Hayes - Roth 83] :
- preferuj reguły eksperta a nie "nowicjusza",
- ten syetem nic nie wie o fizyce i chemii,
- reguły o obiekcie x są niewiele mowiące, używaj ich w ostateczności.

Wykorzystanie metawiedzy (metareguł) pozwala na [Hayses-Roth 83]:
- sterowanie lokalizacją i selekcją reguł do wykorzystania przez SE,
- dołączenie informacji o zasięgu kompetencji SE i strukturze wiedzy dziedzinowej zastosowania,
- ocenę reguł pod względem ich przydatności w danym przypadku, co zwiększa możliwość objaśniania rozumowania,
- wykrywanie błędów działania systemu.

Metawiedzę dysponują m.in. takie systemy jak MECHO, CRYSTALIS, EURISKO [Lauviere 83].

4. MOTOR WNIOSKOWANIA

Motor wnioskowania SE spełnia dwie podstawowe funkcje:

1. bada istniejące fakty oraz reguły i dodaje do bazy wiedzy nowe fakty jeśli takie wystepują,
2. decyduje o porządku w jakim przebiega proces rozumowania systemu.

Podstawowy schemat działania motoru wnioskowania dla systemow regułowych składa się z trzech etapów:

I etap: detekcja -określenie które reguły i jakie fakty są stosowne i możliwe do wykorzystania w aktualnej fazie rozumowania.
II etap: wybór -decydowanie, która z dających się zastosować reguł będzie wykorzystana (rozwiązanie konfliktu).
III etap: dedukcja -wykonanie akcji (konkluzji reguły),aktualizacja bazy wiedzy.

4.1 WNIOSKOWANIE

MODUS PONENS, MODUS TOLLENS

Najczęściej wykorzystywaną strategią wnioskowania w SE jest modus ponens, opisana przez Arystotelesa.

Jej schemat jest nastepujący:

ze zdania if p then q

jeśli p

wnioskujemy, że q.

Inna prostą reguła wnioskowania stosowana w SE jest reguła o schemacie:

ze zdania if p then q

jeśli ¬p

wnioskujemy, że ¬q.

Reguła ta nosi łacińską nazwę modus tollens.

Z użycia modus tollens lub modus ponens w SE wynikają dwie implikacje:

1. ponieważ reguły są proste, rozumowanie systemu jest łatwe do zrozumienia,
2. zastosowanie modus ponens lub modus tollens nie pozwala na osiągnięcie wszystkich prawidłowych wniosków, jakie dla danego stanu wiedzy systemu są możliwe do osiągnięcia (w inny sposób).

REZOLUCJA

Rezolucja (J. Robinson, 1965 rok) jest sposobem rozumowania pozwalającym stwierdzić czy nowy fakt jest prawdziwy przy danym zbiorze reguł.

Idea rezolucji jest następująca:

przyjmijmy, że mamy dwie klauzule (alternatywy literałów)

(1) P1 V P2 V P3 V ... V Pn

(2) ¬P1 V Q1 V Q2 V ...V Qm

przy założeniu że P i Q są różne.

Zauważmy, że jedna z klauzul zawiera literał, który jest w negacji w drugiej klauzuli.

Z tych dwóch klauzul rodzicow (parent) możemy wnioskować nową klauzulę nazywaną rezolwentą klauzul wyjściowych. Rezolwentę uzyskuje się przez stworzenie alternatywy literałów składowych rodziców i eliminacji pary P i ¬P . W oparciu o tę ideę może działać motor wnioskowania. Załóżmy, że w pewnym systemie rozwiązywania problemów mamy zbiór formuł S na podstwie którego chcemy sprawdzić czy dana formuła (cel rozumowania) W daje się wywieść ze zbioru S. Schemat postępowania jest następujący [Nilsson 80]:

1. tworzymy zaprzeczenie W (¬W)
2. dodajemy ¬W do S
3. rozszerzony zbiór S przekształcamy na zbiór klauzul (alternatyw)
4. używamy rezolucji, aż do osiągnięcia sprzeczności, co reprezentuje klauzula pusta NIL.

Osiągnięta sprzeczność dowodzi, że dana klauzula jest prawdziwa.

Metoda rezolucji posiada 2 podstawowe wady:

1. duża złożoność kombinatoryczna
2. nienaturalność (człowiek rzadko rozumuje przez redukcje do sprzeczności).

Zaletą metody jest jednak łatwość jej implementacji.

REGUŁA BAYESA

Aby stwierdzić, w kategoriach rachunku prawdopodobieństwa, jak prawdopodobna jest dana choroba, na podstawie informacji o zaistnieniu określonego związanego z nią symptomu możemy użyć prawdopodobieństwa warunkowego:

P(choroba/symptom)

Na przykład jeśli wiemy, że wśród ludzi o żółtej skórze 19% ma żółtaczkę możemy to zapisać jako:

P(żółtaczka/żółta skora) = 0.19

Prawdopodobieństwo warunkowe należy odróżnić od prawdopodobieństwa niewarunkowego. Zapis:

P(choroba)

oznacza prawdopodobieństwo choroby bez względu na symptomy. Prawdopodobieństwo bezwarunkowe nazywa się często prawdopodobieństwo a'priori, a prawdopodobieństwo warunkowe prawdopodobieństwem a'posteriori.
Nie możemy prawdopodobieństwa warunkowego używać wprost, ponieważ nigdy nie znamy faktów typu:

wśród i-tej liczby ludzi z żółtą skórą, j-ta ilość ma żółtaczkę, bo nie mamy możliwości określenia prawdopodobieństwa wystąpienia choroby przy danych symptomach.
Jednakże można określić jak wiele osób z daną chorobą wykazuje dany symptom. Stąd wiemy, że np.

P(obrzęk źrenic / żółtaczka) = 0.01
P(krwawy mocz / żółtaczka) = 0.60
P(żółta skora / żółtaczka) = 0.90

To spostrzeżenie jest ważne, ponieważ jeśli znamy powyższe wartości a ponadto prawdopodobieństwa chorób i symptomów, to możemy określić warunkowe prawdopodobieństwo choroby na podstawie danych symptomów w oparciu o regułę Bayesa:

$$P(d/s) = \frac{P(d)\,P(s/d)}{P(s)}$$

Ograniczenia tej metody są następujące:

- trudności w ustalaniu prawdopodobieństw i ich duża liczba
- założenie o rozłączności zdarzeń
- trudności w wypadku wystąpienia wielu chorób jednocześnie.

Mimo ograniczeń metoda ta ma wielu zwolenników (Mc Dermott), a

mechanizmy wyszukiwania oparte na niej zastosowano w systemach PROSPECTOR, MYCIN, CADUCEUS [Charniak 85].

4.2 STEROWANIE

Sposoby wnioskowania nie dają odpowiedzi na pytanie jak prowadzić rozumowanie. Dlatego aby zbudować motor wnioskowania należy określić strategię rozumowania. Strategia musi udzielić odpowiedzi na następujące pytania:

1. w jaki sposób rozpocząć proces rozumowania (od czego zacząć)
2. którą regułę wybrać do wykonania w danej chwili (w szczególności rozwiązać konflikt gdy możliwe jest zastosowanie wielu reguł).

Poniżej przedstawione są typowe metody sterowania procesem wnioskowania dla systemów , dla których wiedza reprezentowana jest w postaci reguł.

Ponieważ problem sterowania procesem rozumowania jest bardzo złożony, dlatego omówione zostaną wyłącznie mechanizmy odnoszące się do systemów regułowych.

ROZUMOWANIE W PRZÓD

Rozumowanie w przód unaocznia sens podziału bazy wiedzy na pamięć stałą i roboczą. Pamięć robocza zawiera fakty ustalone w procesie rozumowania. Cykl rozumowania rozpoczyna się od porównywania przesłanek reguł zawartych w bazie wiedzy z zawartością pamięci roboczej. Kiedy dla danej reguły okaże się że przesłanki są znane, jej konkluzja umieszczana jest w pamięci roboczej i cykl się powtarza. Rozumowanie w przód jest często określane jako cykl rozpoznanie-wykonanie (recognize - act).

ROZUMOWANIE W TYŁ

Ogólny schemat rozumowania w tył jest następujący:
Jeśli celem rozumowania SE jest ustalenie faktu C znajduje się reguły dla których C jest konkluzją. Następnie dla danej reguły analizuje się przesłanki. Jeśli któraś nie jest znana, staje się ona nowym celem rozumowania a cały cykl się powtarza do momentu zweryfikowania wszystkich przesłanek reguły o konkluzji C. W momencie, gdy linia tak prowadzonego rozumowania załamie się, system pyta użytkownika o brakujące mu informacje.

5. TWORZENIE SE

Budowa systemu ekspertowego może rokować, w zależności od dziedziny jego zastosowania, różne perspektywy. Pierwszym i podstawowym staje się pytanie "czy dla rozwiązania danego problemu zastosowanie SE jest właściwe". Odpowiedź na to pytanie nie jest prosta. Ponieważ nie ma ogólnej recepty, można jedynie sformułować pewne wskazówki, które można streścić w zdaniu: decyzja o budowie systemu ekspertowego może być rozważana tylko wtedy, gdy jego budowa jest możliwa, usprawiedliwiona a technologia SE właściwa dla danej dziedziny potencjalnego zastosowania [Waterman 1986].

BUDOWA SE JEST MOŻLIWA

Budowa SE jest możliwa wtedy, gdy spełnione są następujące warunki:
- zadanie nie wymaga wiedzy ogólnej i
- zadanie wymaga tylko zdolności umysłowych i
- ekspert może objaśnić metody rozwiązywania problemów i
- istnieją autentyczni eksperci w danej dziedzinie wiedzy i
- eksperci zgadzają się co do rozwiązań i
- zadanie nie jest zbyt trudne i
- zadanie nie jest zbyt mało zrozumiałe.

BUDOWA SE JEST "USPRAWIEDLIWIONA"

Budowę SE można uważać za usprawiedliwioną, kiedy spełniony jest jeden z następujących warunków:

- rozwiązanie zadania przyniesie zysk dla instytucji wdrażającej lub
- ekspert w danej dziedzinie jest trudno dostępny lub
- ekspertyzy są potrzebne w wielu miejscach lub
- środowisko, w którym wymagane jest rozwiązywanie problemów jest dla człowieka szkodliwe(reaktory atomowe, stacje kosmiczne, środowisko toksyczne itp.)

TECHNOLOGIA SE JEST WŁAŚCIWA

Aby zastosowanie SE było decyzją uzasadnioną ,dziedzina wiedzy musi charakteryzować się pewnymi właściwościami do których należą :

- zadanie wymaga przetwarzania symbolicznego i
- zadanie daje się efektywnie rozwiązać tylko metodami heurystycznymi i
- zadanie nie jest zbyt łatwe i
- zadanie ma praktyczną wartość i
- zadanie ma rozmiary nie przekraczające możliwości realizacji technicznej.

5. POZYSKIWANIE WIEDZY

Pojecie "systemy ekspertowe" zostało użyte do określenia systemów, które posługuja się znaczna ilościa informacji pobranej od eksperta na temat konkretnej dziedziny, w celu rozwiązywania problemów z tej dziedziny. Ze względu na szczególna role wiedzy w tych systemach nazywane sa one również "systemami z baza wiedzy". Zbiór czynności zmierzajacych do zebrania wiedzy z danej dziedziny oraz przedstawienie jej w przyjętej dla systemu formie, określa się procesem POZYSKIWANIA WIEDZY.

Za potencjalne źródła tej wiedzy uznaje się [Sell 85],[Hayes-Roth 83] :

- ekspertów,
- literaturę,
- bazy danych,
- doświadczenia osobiste,
- dane empiryczne.

Kompletna baza wiedzy powinna się opierać na wiedzy pochodzacej ze wszystkich w/w. źródeł. Najłatwiej gromadzi się i odwzorowuje wiedzę z tych dziedzin, w których nie ma znaczenia to czy pochodzi ona z literatury, czy od eksperta. W przypadku dziedziny, w której znaczaca role odgrywa praktyczna znajomość tematu i doświadczenie napotyka się na duże trudności. A właśnie takie dziedziny stały się polem działania systemów ekspertowych. Podstawowym źródłem wiedzy jest w nich ekspert, który przy rozwiązywaniu problemów przeważnie działa intuicyjnie i nie jest w stanie opisać formalnie metodologii swego postępowania. Z reguły człowiek nie zdaje sobie sprawy z ilości posiadanej wiedzy, z zależności pomiędzy jej elementami oraz z tego, których z nich używa na danym etapie wnioskowania. Wiedza wykorzystywana przez eksperta jest zbiorem specjalistycznych faktów, procedur oraz reguł osadzajacych, które są trudno przenoszalne do struktur danych tworzacych bazę wiedzy i algorytmów wnioskujących stanowiacych motor wnioskowania systemu ekspertowego.

W trudnym procesie przeniesienia wiedzy eksperta do systemu nieodzowny jest inżynier wiedzy. Jednym z najtrudniejszych, a zarazem najważniejszych aspektów pracy inżyniera wiedzy jest naprowadzanie eksperta na strukturalizację dziedziny wiedzy oraz identyfikację i formalizację głównych nurtów tej dziedziny.

Dla pozyskiwania wiedzy A.Kidd [Kidd 86] proponuje nastepujace 4 techniki :

a. Wywiad informacyjny - jest to technika polegajaca na rozmowie z ekspertem. Wywiad powinien być przeprowadzony w sposób zaplanowany, pytania do eksperta odpowiednio wybrane i tak postawione, by z odpowiedzi eksperta można było wywnioskować sposób jego rozumowania. Ta technika daje szerokie pojęcie o samej wiedzy i powiązaniach miedzy jej składowymi.

b. Protokół - jest to rejestrowanie na bieżąco pracy eksperta przy rozwiązywaniu przykładowych problemów. Inżynier wiedzy powinien skłonić eksperta do objaśniania kroków postępowania, by później móc przeanalizować strategię jako nie-ekspert).

c. Studium obserwacyjne - jest to obserwacja pracy eksperta przy rozwiązywaniu problemu lub konsultacji z klientem w jego naturalnym środowisku. Metoda ta jest szczególnie użyteczna w określeniu roli "klienta" w procesie rozwiązywania problemu, charakteru dialogu ekspert - "klient", kolejności wykonywania czynności przez eksperta, prędkości rozwiązywania problemu oraz związku prędkości z określonym przypadkiem.

d. Indukcja automatyczna - może mieć miejsce, gdy na podstawie dużego zbioru przykładów z dziedziny system sam znajduje prawidłowości kierujące tymi przykładami. W tej technice wymagane jest posiadanie bazy danych przechowującej udokumentowane przypadki. Metoda ta jest zalecana tam, gdzie ludzka ekspertyza jest bardzo słabo rozwinięta lub jest wyłącznie intuicyjna.

Powyższe cztery techniki są wystarczające dla celów pozyskiwania wiedzy i metodologii postępowania eksperta przy rozwiązywaniu problemów. By uzyskać jak najlepsze rezultaty zalecane jest stosowanie wszystkich czterech technik, bo informacje uzyskane w wyniku ich zastosowania będą z pewnością się zazębiać.

4.2. ETAPOWANIE PROCESU PROJEKTOWANIA SYSTEMU EKSPERTOWEGO

W systemach ekspertowych nie można rozpatrywać procesu pozyskiwania wiedzy w oderwaniu od procesu konstrukcji tych systemów. Wyznacznikami tego procesu są, kolejne etapy metodyki tworzenia systemów ekspertowych. Wynika z niej, jak zostanie to przedstawione poniżej, że istnieją dwa zupełnie różne cele, które należy zrealizować w procesie pozyskiwania wiedzy. Pierwszy z nich osiągnięty ma być podczas META - AKWIZYCJI WIEDZY i osiągnięcie go doprowadza do pełnej , ostatecznej konceptualizacji systemu dziedzinowego-zastosowań : realizacja drugiego celu następuje w trakcie AKWIZYCJI WIEDZY , która obejmuje wszelkie działania zmierzające do "wyposażenia" bazy wiedzy systemu w poprawną i kompletną wiedzę z zakresu objętego obszarem zastosowania tego systemu.

Pozyskiwanie wiedzy przewija się przez wszystkie kolejne etapy budowy systemów ekspertowych. Wielu autorów w podobny sposób definiuje i charakteryzuje te etapy :

- postawienie problemu,
- budowa systemu prototypowego (in. systemu I-szej generacji),
- rozwój systemu prototypowego i jego testowanie,
- powstanie systemu użytkowego (in. II-giej generacji),
- szacowanie i testowanie systemu użytkowego.

Niejasną może się wydawać rola Meta Akwizycji Wiedzy (MAW) w przypadku gdy dla tworzenia systemu dziedzinowego korzysta się

z narzędzi w postaci powłok (schell) systemów ekspertowych. Ale jeżeli projektant systemu ma do wyboru kilka systemów powłokowych lub dysponuje narzędziami nowej generacji charakteryzującymi się dużą ekspresją języka definiowania reprezentacji wiedzy i wnioskowania to podjęcie decyzji odnośnie wyboru najistotniejszych elementów systemu następuje całkowicie w oparciu o fakty stwierdzone podczas MAW . Nowe metodyki tworzenia systemów ekspertowych zakładają częste sięganie po systemy powłokowe jeśli nie dla realizacji systemu użytkowego to przynajmniej dla realizacji systemów prototypowych .

4.3 META AKWIZYCJA WIEDZY I JEJ ETAPY

W niniejszym opracowaniu ze względu na wagę i stopień złożoności problemu uwaga skoncentrowana zostanie na pozyskiwaniu wiedzy określanym jako Meta Akwizycja Wiedzy. Czynności z nią związane powtarzane są w cyklu tworzenia systemu , dwukrotnie na różnych poziomach szczegółowości i w oparciu o zróżnicowany poziom wiedzy "wyjściowej". Tworzenie bowiem systemu użytkowego (II-giej generacji) bazuje w bardzo dużym stopniu na doświadczeniach uzyskanych z testów systemu prototypowego (I-szej generacji). A z dotychczasowych doświadczeń wynika także, że MAW ma zasadnicze znaczenie dla jakości przyszłego systemu i decyduje o jego przyszłej użyteczności.

Etapami Meta - Akwizycji Wiedzy są kolejno: identyfikacja, konceptualizacja, formalizacja i testowanie.

I. Etap identyfikacji

W tym pierwszym etapie inżynier wiedzy powinien poznać charakterystykę najważniejszych aspektów problemu, a są nimi:

a) Identyfikowanie problemu; - odbywa się przez zapoznanie się z literaturą na temat problemu oraz wymianę poglądów z ekspertem na temat różnych płaszczyzn podejścia do zagadnienia, jego definicji, charakterystyk i podproblemów. Celem tych czynności jest scharakteryzowanie problemu i obejmowanych przez niego struktur wiedzy .

W tej fazie najistotniejszymi elementami do ustalenia pozostają:

- klasa problemów jaką będzie system rozwiązywał?
- jak można te problemy scharakteryzować lub zdefiniować?
- jakie są istotne podproblemy oraz podział zdarzeń między nimi?
- jakie są dane?
- jakie są istotne określenia językowe i ich wzajemne powiązania?
- jaką postać przyjmuje rozwiązanie?
- jaka jest natura wiedzy leżącej u podstaw rozwiązań stawianych przez eksperta?
- jakie sytuacje utrudniają rozwiązanie problemu i jak

wpływają na przeprowadzanie ekspertyzy?

Ścisła współpraca inżyniera z ekspertem polega na tym, że ekspert daje opisowe wyjaśnienia problemów, objaśnia jak je rozwiązuje i z jakiej korzysta wiedzy, a inżynier wiedzy te wyjaśnienia dokumentuje i analizuje.

b) Identyfikacja celów jakie powinien osiągnąć system; - należy określić, jaką przewidywaną rolę ma system do spełnienia w środowisku zastosowania.

c) Identyfikacja zasobów; - typowymi zasobami są : źródła wiedzy, czas, udogodnienia programistyczne i pieniądze. Wszystkie one potrzebne są do gromadzenia wiedzy, implementacji systemu i testowania go. Ekspert i inżynier muszą wykorzystać różne źródła w celu otrzymania wiedzy odpowiedniej w budowie systemu ekspertowego. Dla eksperta będą to jego ostatnie doświadczenia w rozwiązywaniu problemów, książki oraz przykłady problemów i ich rozwiązań. Z kolei źródłem dla inżyniera wiedzy są doświadczenia w pracy z problemami analogicznymi, wiedza na temat metod i sposobów reprezentacji oraz narzędzi budowy systemów ekspertowych. Jeżeli chodzi o czas, to jest on zasobem krytycznym .

II. Etap konceptualizacji (budowy konceptu).

W etapie konceptualizacji zostają explicite określone kluczowe koncepcje oraz relacje, które zostały zasygnalizowane podczas identyfikacji. Inżynier wiedzy powinien próbować zakwalifikować różne ekspertyzy eksperta pod pewne szersze typy wiedzy uznane wcześniej za reprezentatywne. W uzupełnieniu do tego trzeba zwrócić uwagę na słownictwo, jakie stosuje ekspert, próbować określić podstawowe strategie jego postępowania przy rozwiązywaniu problemów i inne mechanizmy organizacyjne. Podczas tego etapu inżynier wiedzy powinien przemyśleć, jak będzie można sformalizować wiedzę tzn. jakie architektury będą się najlepiej reprezentować.

W tym etapie inżynier wiedzy musi znaleźć odpowiedź na następujące pytania :
- jakie fakty ekspert ustala jako pierwsze?
- jakie pytania ekspert zadaje jako pierwsze?
- czy ekspert stawia wstępne hipotezy?, jeśli tak, to jakie pytania służą do ich potwierdzania?
- w jakiej kolejności ekspert wykonuje podgrupy zadań?
- jakie typy danych są dostępne?
- co jest dane z góry, a co unioskowane?
- czy można zidentyfikować hipotezy częściowe, które są nagminnie używane?
- jak są powiązane obiekty w dziedzinie?
- czy można narysować diagram hierarchii relacji, opisać go, określić zawieranie się relacji?

- jakie procesy są zaangażowane w rozwiązywanie problemu?
- jaki jest przepływ informacji?
- czy można zidentyfikować i oddzielić wiedzę użytą do rozwiązywania problemu od wiedzy użytej do potwierdzania rozwiązania?

Inżynier wiedzy powinien jednak powstrzymać się jeszcze od wyboru konkretnych struktur reprezentacji oraz narzędzi pomocniczych.

III. Etap formalizacji

Etap formalizacji obejmuje przeniesienie wiodących koncepcji, podproblemów oraz charakterystyk przepływu informacji wyłonionych w procesie konceptualizacji na bardziej formalną reprezentację, opartą na różnorodnych narzędziach inżynierii wiedzy oraz obejmuje zapewnienie tej reprezentacji kompletności i integralności. Inżynier wiedzy zaczyna odgrywać rolę bardziej aktywną i zapoznaje eksperta z istniejącymi metodami reprezentacji, które mogłyby pasować do danego problemu. Wynikiem tego etapu jest zbiór specyfikacji opisujących możliwe reprezentacje problemu za pomocą wybranych narzędzi.

Trzy najistotniejsze czynniki analizowane w etapie formalizacji to :
- przestrzeń możliwych hipotez,
- model leżący u podstaw procesu tworzenia rozwiązania,
- charakterystyki danych wchodzących do i wychodzących z systemu.

By zrozumieć strukturę przestrzeni hipotez należy sformalizować koncepcje i określić jak one się łączą przy tworzeniu hipotezy. Koncepcje określają charakter przestrzeni, mówią czy jest ona skończona, czy zawiera wyspecyfikowane klasy, czy hipotezy powinny być rozpatrywane w kolejności hierarchicznej, czy hipotezy pośrednie i końcowe są pewne, czy niepewne, czy będą potrzebne różne poziomy abstrakcji.

Analiza modelu procesu tworzenia rozwiązania przez eksperta może wnieść dużo nowego na temat powiązań i dodatkowych informacji w zależności od rodzaju modelu. Ustalenie charakteru danych pomaga w znalezieniu powiązań między hipotezami opierającymi się bezpośrednio na danych a hipotezami wyższego rzędu oraz określeniu jak te hipotezy mają się do zadań procesu wnioskowania. Stwierdza się czy powiązanie danych z hipotezami jest typu przyczynowego, definicyjnego czy korelacyjnego.

Wynikiem etapu formalizacji jest specyfikacja budowy bazy wiedzy.

IV. Etap testowania

Ten etap w kontekście MAW ma za zadanie oszacować system i formy reprezentacji, które wykorzystuje, by znaleźć słabe punkty bazy wiedzy i struktury wnioskowania . System należy testować na wielorakich przykładach. Przykłady powinny być wybrane zgodnie z sugestią eksperta i tak postawione,

by uwypuklały poprawności oraz niedociągnięcia i błędy systemu. Elementami, które mogą mieć wpływ na złą pracę systemu i które należy bezwzględnie przetestować są charakterystyki wejścia/wyjścia, elementy wiedzy i strategie kontroli.

Testowanie przebiega najczęściej w dwóch płaszczyznach :

* weryfikacji - sprawdzeniu, czy kod komputerowy jest napisany bez błędów. Ten rodzaj testowania jest stosunkowo prosty. Opiera się na standardowych metodach kompilacji i może być przeprowadzony, gdy baza wiedzy jest tworzona lub w końcowej fazie testowania.

* walidacji - sprawdzeniu, czy znaczenie i zawartość elementów wiedzy spełniają określone wymogi dopasowania do dziedziny i do rzeczywistego środowiska pracy systemu ekspertowego.

W aspekcie testowania systemu dla potrzeb MAW interesującym dla nas będzie proces walidacji systemu . Wyczerpująco szczegółowe krokiprowadzenia walidacji zanalizowane zostały w [Marcot 87].

Mogło by się wydawać ,że powyższa charakterystyka etapow MAW odnosi się do procesu tworzenia prototypu. Ostatni jednak scharakteryzowany etap testowania-walidacji systemu dostarcza projektantowi-inżynierowi wiedzy oraz ekspertowi wiele dodatkowego materiału , który staje się wraz z wiedzą nagromadzoną w dotychczasowym procesie MAW podstawą do zaprojektowania w oparciu o nowy cykl MAW ostatecznej-użytkowej wersji systemu.

6. NARZĘDZIA DO BUDOWY SYSTEMÓW EKSPERTOWYCH

Tworzenie systemu ekspertowego jest procesem wymagającym stosowania specyficznych metod i technik inżynierii oprogramowania. Istotną rolę odgrywa wybór odpowiednich narzędzi programowych umożliwiających i wspomagających implementację systemu.

6.1 KLASYFIKACJA NARZĘDZI DO BUDOWY SE

Najczęściej narzędzia do budowy systemów ekspertowych klasyfikuje się do jednej z trzech grup [Cholewa 87]:

- języki programowania,
- systemy szkieletowe,
- metasystemy (zintegrowane środowiska narzędziowe).

Zastosowanie do implementacji SE tradycyjnych języków programowania takich, jak FORTRAN, BASIC, czy PASCAL jest możliwe ale trudne, a przede wszystkim uciążliwe. Dlatego w dziedzinie sztucznej inteligencji stosuje się najczęściej języki umożliwiające programiście skupienie uwagi na problemie projektowym w większym stopniu niż na detale implementacyjne. Języki te cechują się następującymi właściwościami [Matthews 87]:

- deklaratywny sposób reprezentacji pojęć i klas obiektów,
- nieproceduralne, zorientowane na przetwarzanie wiedzy mechanizmy sterowania przebiegiem programu (rozumowanie wprzód

i wstecz , rezolucja, przekazywanie komunikatów),
- mechanizmy abstrachowania (procedury związane z klasami obiektów),
- mechanizmy oddzielające programistę od niskiego poziomu implementacji (przetwarzanie danych symbolicznych, automatyczne zarządzanie pamięcią, automatyczne porównywanie z wzorcem, dziedziczenie).

Obecnie do implementacji SE oraz dla innych zastosowań w dziedzinie sztucznej inteligencji stosuje się takie języki programowania jak: OPS5, PROLOG, Common LISP, C++.

Najbardziej popularnym językiem przetwarzania reguł typu IF(warunek)THEN(akcja) jest OPS5 [Browston 86]. Opiera się on o mechanizmy rozumowania wprzód i szybki algorytm porównywania ze wzorcem (Fast Rete Pattern Matching Algorithm). Najbardziej znane implementacja SE w języku OPS5 jest system konfigurowania mikrokomputerów VAX o nazwie R1/XCON.

PROLOG [Kluźniak 83] jest językiem opartym o logikę Horna i teorię rezolucji. Naturalne w tym języku są mechanizmy rozumowania w tył. Język ten popularność zawdzięcza japońskiemu projektowi "piątej generacji", w którym ma być zastosowany. Ze względu na małą elastyczność oraz nieczytelność dłuższych programów jego zastosowanie maleje.

Common LISP stał się standardem języka wykorzystywanego do zastosowań sztucznej inteligencji na terenie U.S.A. Jest to duży język przetwarzania symbolicznego, którego istotą są struktury listowe. O możliwościach tego języka świadczy np. fakt że posiada on około 700 wbudowanych funkcji.

C++ [Stroustrup 86] jest obiektowo-zorientowanym rozszerzeniem języka programowania C. Pozwala on między innymi na przekazywanie komunikatów oraz wykorzystywanie mechanizmów dziedziczenia hierarchicznego. Język ten nie ma możliwości deklarowania symbolicznych typów danych oraz automatycznego zarządzania pamięcią.

Szkieletowe systemy ekspertowe opierają się na paradygmacie niezależności wiedzy od innych elementów architektury SE takich jak motor wnioskowania, moduł objaśniania toku rozumowania czy moduł komunikacji z użytkownikiem. Innymi słowy tylko baza wiedzy jest zależna od dziedziny zastosowania systemu, a pozostałe elementy są w dużej mierze niezależne od zakresu zastosowań. Budowa SE w oparciu o system szkieletowy sprowadza się do opracowania bazy wiedzy dla danej dziedziny zastosowania.

Idea ta zakłada, że forma reprezentacji wiedzy , sposób wnioskowania i inne elementy systemu są dla danego zastosowania odpowiednie, stąd możliwość wpływu na nie programisty jest niewielka. Jest to spore ograniczenie tych systemów, które jednak dla wielu zastosowań nie ma istotnego znaczenia.

Jednym z najbardziej znanych systemów szkieletowych jest EMYCIN , który jest dziedzinowo niezależną wersją systemu ekspertowego MYCIN służącego do diagnostyki medycznej. EMYCIN jest narzędziem odpowiednim przede wszystkim do rozwiązywania problemów dedukcyjnych. Wiedza w systemie jest reprezentowana za pomocą reguł produkcji IF (warunek) THEN (akcja). Warunek jest wyrażeniem boolowskim złożonym z predykatów obiekt - atrybut - wartość. Wiedza jest zorganizowana w drzewo kontekstowe, które elementy wiedzy dziedzinowej porządkuje w prostą hierarchię umożliwiającą stosowanie pewnych mechanizmów dziedziczenia wiedzy. EMYCIN zapewnia reprezentację wiedzy niepewnej poprzez tzw. współczynniki pewności związane z przesłankami i akcjami reguł i wykorzystywane przez mechanizm wnioskujący, który w tym systemie prowadzi rozumowanie wstecz.

Ograniczeń szkieletowych systemów ekspertowych nie mają metasystemy (zintegrowane środowiska narzędziowe). Umożliwiają one swobodny wybór zarówno sposobu reprezentacji wiedzy, mechanizmów wnioskowania jak i innych elementów budowanego systemu takich jak sposób komunikacji z użytkownikiem, objaśnianie toku rozumowania itd. Zarówno systemy szkieletowe jak i (częściej) metasystemy oferują programiście wiele innych udogodnień takich jak: edytory wiedzy, moduły wspomagania akwizycji wiedzy, testowanie poprawności bazy wiedzy, kształtowanie sposobu dialogu z użytkownikiem, łączenie programów napisanych w innych językach programowania itp.

Typowym przedstawicielem metasystemów jest system KEE (Knowledge Enginnering Environment) firmy IntelliCorp. W systemie tym użytkownik może stosować takie sposoby reprezentacji jak: ramy, obiekty, reguły IF-THEN. Niepewność wiedzy może być wyrażana przez współczynniki pewności lub probabilistycznie. Sposób rozumowania jest definiowany przez użytkownika. Dodatkowo w systemie dostępne są: edytor wiedzy wspomagający akwizycję, duża gama narzędzi do testowania działania budowanego w oparciu o KEE systemu ekspertowego. Sposób dialogu z użytkownikiem zależy od konstruktora systemu, który może stosować menu, okna, grafikę i tekst.

Zestawienie możliwości systemów szkieletowych i metasystemów można znaleźć w pracach [Waterman 86]. [Freedman 87],[Harmon 85].

6.2. WYMAGANIA STAWIANE NARZĘDZIOM DO BUDOWY SYSTEMÓW EKPERTOWYCH

Wymagania stawiane narzędziom do budowy systemów ekspertowych wynikają przede wszystkim z :

- dziedziny zastosowania,
- potrzeb na każdym z etapów projektowania i budowy systemu,
- rodzaju użytkownika (student, ekspert, inżynier wiedzy, programista) [Citrenbaum 87].

Wymagania dotyczą takich zagadnień jak : inżynieria wiedzy, reprezentacja wiedzy, wnioskowanie systemu, interfejs z użytkownikiem i efektywność implementacji.

Wymagania dotyczące inżynierii wiedzy to:
- wspomaganie pozyskiwania wiedzy i wprowadzania jej do systemu,
- umożliwienie śledzenia powiązań między elementami bazy wiedzy,
- automatyczne uczenie się systemu z przykładów,
- umożliwienie śledzenia działania systemu,
- automatyczna walidacja bazy wiedzy (spójność, niesprzeczność, kompletność itp.).

Wymagania dotyczące reprezentacji wiedzy to:
- pomoc w wyborze odpowiedniej reprezentacji wiedzy dla danej dziedziny zastosowania systemu,
- umożliwienie zastosowania różnych reprezentacji wiedzy,
- umożliwienie zmiany reprezentacji wiedzy poprzez automatyczne translacje.

Wymagania dotyczące wnioskowania systemu to:
- umożliwienie zastosowania różnych metod wnioskowania,
- umożliwienie modyfikacji mechanizmów wnioskowania,
- zapewnienie mechanizmów wnioskowania niepewnego.

Wymagania dotyczące interfejsu z użytkownikiem:
- umożliwienie stosowania różnych sposobów komunikacji z użytkownikiem SE (menu, okna, dialog naturalny, grafika),
- objaśnianie toku rozumowania systemu (WHY, HOW, HELP).

Wymagania dotyczące implementacji:
- dysponowanie bogatymi środkami testowania systemu i sygnalizacji błędów ,
- umożliwienie łatwego przenoszenia systemu na różne typy komputerów,
- kompatybilność z innymi narzędziami do budowy SE oraz możliwość współpracy z bazami danych, programami typu VISICALC itd.,
- generowanie efektywnego kodu wynikowego.

Dostępne na rynku oprogramowanie narzędziowe (szczególnie metasystemy) spełnia większość przedstawionych wymagań. Stwierdzenie to , niestety, nie odnosi się do konkretnego produktu a ma charakter ogólny.

Bogata gama możliwości jakimi cechują się narzędzia stawia duże wymagania konstruktorowi systemu; wymaga gruntownej znajomości samego narzędzia i problematyki systemów ekspertowych.
Niebagatelny jest koszt zaawansowanych narzędzi do budowy SE kształtujący się w granicach od 50 tys. do 100 tys. dolarów.

7. CHARAKTERYSTYKA WYBRANYCH NARZĘDZI I SYSTEMÓW

W ramach prac prowadzonych w Instytucie Informatyki AGH powstały praktyczne realizacje kilku systemow o charakterze systemów szkieletowych. Należą do nich m.in.: system BAYEX (statystyczny) oraz system MEX (regułowy). Oba te systemy zostały wykorzystane do stworzenia systemów ekspertowych dotyczących zagadnień diagnostyki medycznej. Było to możliwe dzięki współpracy z lekarzami Instytutu Pediatrii AM w Krakowie. Systemy BAYEX i MEX charakteryzuje się różnymi możliwościami reprezentacji wiedzy oraz algorytmami wnioskowania. Badania różnorodnych mechanizmów reprezentacji wiedzy i wnioskowania wiązały się z pracami nad implementacją systemu dziedzinowego ICTERUS, w którym wykorzystywane są niejednorodne techniki reprezentacji wiedzy i wnioskowania.

Wszystkie systemy zaprojektowano i zaimplementowano z dbałością o maksymalną wygodę użytkownika, wyrażającą się w "przyjaznym" interfejsie użytkownik-system, typu rozwijalne menu (typu pull-down) oraz okna. Każdy z tych systemów posiada także funkcje pomocnicze typu HELP, dokładnie objaśniające użytkownika o funkcjach użytkowych systemu i instruujące o sposobie korzystania z niego.

Cenną cechą charakterystyczną wymienionych systemów, jest ich umiejętność zapamiętywania przebiegu konsultacji oraz objaśniania toku rozumowania. Jest to szczególnie ważne przy zastosowaniach dydaktycznych tych systemów.

Dla realizacji systemów BAYEX i MEX powstały także dodatkowe narzędzia programowe w postaci specjalizowanych edytorów i analizatorów baz wiedzy.

System BAYEX i MEX zaimplementowano w języku C z przeznaczeniem na mikrokomputery klasy PC XT/AT pracujące pod kontrolą systemu operacyjnego DOS 3.0 oraz OS/2 i pamięcią operacyjną 640 KB. Systemy pracować mogą z monitorami CGA, HERCULES oraz EGA (jakkolwiek ze względu na czytelność i "przyjazność" interfejsu użytkownika najlepsze efekty uzyskać można przy pracy z monitorem kolorowym typu EGA).

7.1 BAYEX - CHARAKTERYSTYKA SYSTEMU

Reprezentacja wiedzy

System BAYEX jest systemem diagnostycznym. Wiedza w systemie reprezentowana jest w postaci liczb - miar zależności miedzy interesującymi użytkownika zjawiskami. Umożliwia to opatrzenie każdej diagnozy stawianej przez system odpowiadającym jej prawdopodobieństwem zajścia.

W związku z powyższym, baza wiedzy dla systemu BAYEX zawiera zestaw objawów oraz zestaw diagnoz,które powiązane są ze sobą parami prawdopodobieństw: zajścia objawu gdy diagnoza jest prawdziwa oraz zajścia objawu gdy diagnoza jest fałszywe.

Konstrukcja takiej bazy wiedzy, jeśli dysponuje się odpowiednim materiałem statystycznym z danej dziedziny, jest właściwie mechaniczna, a zatem, zupełnie niekłopotliwa.

Strategie sterowania systemu

Motor wnioskowania systemu oparty jest na twierdzeniu Bayesa o prawdopodobieństwie warunkowym zdarzeń.

System, oprócz stałej bazy wiedzy, zawiera wiedzę ogólną (tzw. metawiedzę) czyli kilka-kilkanaście pytań pozwalających usprawnić proces właściwego diagnozowania poprzez uściślenie kontekstu. Pytanie o objawy stałej bazy wiedzy stawiane są w oparciu o ich współczynniki "wartości diagnostycznej" (entropie danego objawu) czyli są to pytania, na które odpowiedź usuwa maksymalną ilość niepewności systemu co do możliwych diagnoz. Współczynniki te są modyfikowane na bieżąco w trakcie prowadzenia dialogu z użytkownikiem.

Własności wyjaśniania toku rozumowania

System umożliwia, w trakcie diagnozowania, przeglądanie listy diagnoz wraz z aktualnie odpowiadającymi im prawdopodobieństwami zajścia. Umożliwia on wtedy również wywołanie szeroko rozumianego komentarza do dowolnej diagnozy.

Dodatkowa cecha systemu jest możliwość zapamiętania całego procesu diagnozowania w nazwanym zbiorze tekstowym i w dowolnym momencie odtwarzania go na ekranie lub wyprowadzenie na drukarkę.

Interfejs użytkownika

BAYEX porozumiewa się z użytkownikiem przy pomocy rozwijalnych menu nie wymagając od niego żadnych innych umiejętności poza umiejętnością uruchomienia programu głównego (opis możliwych czynności w ostatniej linii ekranu). Dla wygody użytkownika ofiaruje Mu, dostępną na każde żądanie opcję HELP będącą krótkim opisem czynności niezbędnych do obsługi systemu.

Aplikacja dziedzinowa systemu BAYEX

System BAYEX posłużył do implementacji systemu ekspertowego diagnostyki medycznej.

Dziedzina zastosowania, obejmuje rzadkie choroby wrodzone manifestujące się wadami budowy twarzoczaszki. Częstość występowania wad wrodzonych u człowieka jest oceniana na około 2-3%. Wiele zespołów dysmorficznych, nawet specjaliście w tej dziedzinie, jest znane na podstawie jedynie opisów literaturowych. Jednostkowe obserwacje poszczególnych zespołów wynikają z bardzo niskich częstości ich występowania. Należy jednak pamiętać, że przy założeniu, że 10% dzieci korzysta z porad

lekarskich, pacjenci z wrodzonymi zespołami w większości znajdują się w tej grupie co powoduje, że pediatra ogólny może mieć praktycznie codziennie styczność z którymś z zespołów dyzmorficznych.

W systemie komputerowym DIADYS zebrano 70 zespołów wad wrodzonych, których wspólną cechą są zaburzenia budowy twarzoczaszki.

7.2 MEX - CHARAKTERYSTYKA SYSTEMU

Reprezentacja wiedzy

Reprezentacja wiedzy w systemie MEX oparta jest o reguły wnioskowania typu "jeżeli (przesłanka) to (wniosek)" (ang. if (clause) then (conclusion)). Baza wiedzy ma postać zbioru tekstowego i zawiera definicję cech i reguł. Język opisu bazy wiedzy jest zbliżony do naturalnej postaci języka polskiego. Dzięki możliwości formułowania pytań w języku polskim oraz formie komunikacji systemu z użytkownikiem, MEX ułatwia inżynierowi wiedzy takie skonstruowanie dialogu z użytkownikiem, że nie będzie od niego wymagana określona wiedza informatyczna poza umiejętnością uruchomienia systemu i korzystania z niego w zakresie prowadzenia diagnozowania. Zasady budowy bazy wiedzy umożliwiają określenie więcej niż jednego celu konsultacji dla danej bazy. Umożliwia to rozwiązywanie różnych problemów, których sposoby rozwiązywania zawarte są w jednej bazie wiedzy (jeden zbiór bazy wiedzy).

Motor wnioskowania

Motor wnioskowania relizuje proces wnioskowania według algorytmu - wnioskowanie wstecz z powrotami.

Składają się na niego:

a) Procedura realizująca właściwy proces wnioskowania. Procedura ta korzystając z informacji zawartych w stałej bazie wiedzy, prowadzi wnioskowanie według przyjętego algorytmu (wnioskowanie wstecz z powrotami), tworząc dynamicznie chwilową bazę wiedzy, która zawiera fakty ustalane na bieżąco przez system, oraz informacje o tym w jaki sposób zostały one ustalone.

b) Procedury dostępu do przechowywanych w pamięci struktur bazy wiedzy. Procedury te umożliwiają dostęp do informacji zawartych w stałej i chwilowej bazie wiedzy.

c) Procedury współpracy z użytkownikiem i modułem obsługi pytań. Umożliwiają one systemowi prowadzenie dialogu z użytkownikiem na drodze wyboru z menu odpowiednich funkcji dostępnych na tym etapie. Także wybór odpowiedzi odbywa się

tą drogą tzn. użytkownik wybiera z listy możliwych odpowiedzi tę, którą uznaje za właściwą, lub "Nie wiem" jeżeli odpowiada to znanym przez niego faktom.

Własności wyjaśniania toku rozumowania

System MEX jest wyposażony w programowe mechanizmy wyjaśniania toku rozumowania. Umożliwiają one użytkownikowi w trakcie diagnozowania określenie stanu tego procesu oraz prześledzenie drogi, jaką system do tego stanu doszedł.

System umożliwia użytkownikowi uzyskanie wyjaśnienia DLACZEGO MEX zadaje dane pytanie, CO wie tzn. jakie fakty zostały do tej pory ustalone oraz JAK została ustalona wartość danej cechy .

Interfejs użytkownika

Cały proces współpracy MEX-a z użytkownikiem systemu odbywa się za pomocą rozwijanych wielopoziomowych menu (ang. pull-down menu) oraz okien (ang. windows). Wszystkie funkcje usługowe MEX-a dostępne są przez wybór odpowiedniej pozycji bieżącego menu.

Interfejs inżyniera wiedzy

Jeżeli użytkownikiem MEX-a jest inżynier wiedzy to może on korzystać z funkcji, które są przeznaczone w zasadzie wyłącznie dla niego. Są to funkcje analizy bazy wiedzy oraz jej walidacji oraz właściwości wyjaśnienia toku rozumowania systemu . Ponadto elementem wspomagającym pracę inżyniera wiedzy jest wbudowany w system parser , który dla danej bazy wiedzy przeprowadza ścisłą analizę syntaktyczną oraz wstępną analizę semantyczną. Parser bazy wiedzy został skonstruowany w oparciu o oprogramowanie powstałe w II AGH [Macowicz 88].

Analizator bazy wiedzy

Procedury tego modułu realizują weryfikację i walidację bazy wiedzy. Pozwalają one również na przeglądanie struktur bazy wiedzy przechowywanych w pamięci komputera. Składają się na niego:

a) Procedura przeglądania listy cech zdefiniowanych w bazie wiedzy. Umożliwia ona użytkownikowi przeglądanie tej części stałej bazy wiedzy, która zawiera definicje cech. Informacje o cechach zawierają skrócone nazwę cechy, pełne nazwę cechy - tłumaczenie, oraz pytanie o cechę jeżeli zostało ono zdefiniowane w źródłowej bazie wiedzy.

b) Procedura przeglądania listy reguł zdefiniowanych w bazie wiedzy. Umożliwia ona użytkownikowi przeglądanie tej części stałej bazy wiedzy, która definiuje reguły wnioskowania.

c) Procedura przeglądania listy celów wyodrębnionej z listy cech. Umożliwia ona wyświetlenie listy tych cech, które w źródłowej bazie wiedzy zostały zdefiniowane jako cele.

d) Procedura analizująca bazę wiedzy pod kątem wykrywania zapętleń w liście reguł. Procedura ta relizuje znaczące dla systemu funkcje walidacji bazy wiedzy, polegające na symulowaniu możliwych przebiegów wnioskowania prowadzonego przez system dla wczytanej bazy wiedzy i wykrywaniu faktów świadczących o tym, że rozumowanie wraca do pewnego stanu, który już wcześniej został w trakcie symulacji osiągnięty. Jeżeli zapętlenia występują w bieżącej bazie wiedzy to wypisywane są sekwencje reguł, które do takiego stanu mogą doprowadzić.

Aplikacja dziedzinowa systemu MEX

Na bazie systemu MEX został stworzony ekspertowy system diagnostyki medycznej DIACOL.

DIACOL jest systemem ekspertowym wspomagania diagnostyki różnicowej chorób tkanki łącznej u dzieci. Choroby te występują u dzieci z częstością określoną współczynnikiem chorobowości około 25 na 100000. Najczęstsze z nich to reumatoidalne młodzieńcze zapalenie stawów, mogące przebiegać pod co najmniej 4 postaciami: choroba reumatyczna, toczeń rumieniowaty, zapalenie skórno-mięśniowe.

Każdy lekarz pediatra jest świadomy trudności diagnostycznych związanych z rozpoznaniem choroby, której manifestacje są początkowo niespójne objawowo, trudne w interpretacji i niespecyficzne.

7.3 ROZWÓJ SYSTEMÓW BAYEX I MEX

Korzystając z doświadczeń zaimplementowanych systemów wersji systemów BAYEX i MEX, mając na uwadze stworzenie narzędzia bardziej ogólnego, w II ŚGH tworzony jest prototyp nowego systemu. Charakteryzuje się on niejednorodnym sposobem reprezentacji wiedzy (reguły, prawdopodobieństwa, ramy) oraz konsekwentnie bardziej złożonym mechanizmem wnioskowania. Cechy użytkowe systemu (interfejs inżyniera wiedzy, interfejs użytkownika, objaśnianie toku rozumowania) w stosunku do systemów wyżej omówionych zostały nieco rozwinięte. System szkieletowy przeznaczony jest do tworzenia dziedzinowych systemów diagnostycznych z uwzględnieniem większej złożoności procesu diagnostycznego.

Proces diagnostyczny prototypu systemu ekspertowego składa się z kilku etapów. Każdy etap pracy systemu odpowiada

określonej fazie rozumowania diagnostycznego analizuje specyficzne dla każdej z faz cele z uwzględnieniem (tam gdzie to możliwe) metod wnioskowania na poziomie jaki wynika z kompromisu między wiedzą o procesach myślowych człowieka, a możliwościami ich efektywnej komputerowej reprezentacji. Ma to kilka pozytywnych konsekwencji:

- użytkownik "rozumie" działanie systemu ,
- realizacja odpowiedzi na pytanie WHY i HOW może lepiej odpowiadać strukturze wiedzy dziedzinowej,
- akwizycja wiedzy osłabia problem jej adekwatności w przyjętym modelu konceptualnym wiedzy systemu ekspertowego ,
- system jest bardziej dydaktyczny.

Etapy procesu diagnostycznego systemu prototypu ekspertowego ICTERUS są następujące:

0. Dialog wstępny (dotyczący objawów elementarnych),
I. Wyodrębnienie zespołów objawów (agregatów),
II. Ustalenie hipotez wstępnych,
III. Ustalenie zawężonych hipotez wstępnych.

0. DIALOG WSTĘPNY

Wyróżnienie etapu wstępnego wynika z częstego faktu posiadania przez użytkownika systemu dużej ilości danych dotyczących danych elementarnych (szczegółowych).

Idea etapu wstępnego jest umożliwienie wykorzystania tych danych w systemie ekspertowym bez konieczności żmudnego ich wprowadzania w trybie konwersacyjnym po rozpoczęciu pracy użytkownika z systemem. (Jednym z elementów systemu ekspertowego /docelowego/ będzie baza danych zawierająca informacje o pacjencie gromadzone sukcesywnie przed rozpoczęciem ekspertyzy).

I. WYODREBNIENIE ZESPOŁÓW OBJAWÓW - SYNDROMÓW

Po uzyskaniu podstawowych informacji o obiekcie, w postaci objawów szczegółowych określonych w etapie 0 jako przesłanki diagnostyczne, ustalenie agregatów stanowi w postępowaniu eksperta kolejny etap diagnozowania. Stanowi on też zasadniczy element opisu procesu diagnostycznego prowadzonego w celu dydaktycznym.

Objawy podstawowe wchodzące w skład agregatów charakteryzują się częstością występowania w agregacie, mogą także wchodzić w skład kilku agregatów. Rozpoznanie agregatu pozwala posługiwać się nim w dalszej diagnostyce jako charakterystycznym objawem, niezależnym od jej stopnia zaawansowania.

Moduł systemu ekspertowego realizujący cel I-go etapu wykorzystuje bazę wiedzy i dane o przesłankach diagnostycznych znanych systemowi z etapu wstępnego, bez dialogu z użytkownikiem. Stwierdzenie obecności agregatu następuje w oparciu o zastosowanie rachunku prawdopodobieństwa (reguły Bayes'a)

Sposób reprezentacji wiedzy stałej tego modułu odpowiada funkcjonalnie sposobowi określania agregatów przez eksperta.

Wynikiem działania modułu jest ustalenie występowania zespołów objawów, agregatów i wpisanie ich do chwilowej bazy wiedzy na równi z innymi objawami.

II. USTALENIE HIPOTEZ WSTĘPNYCH - AKTYWNYCH

Każda z możliwych diagnoz systemu określamy mianem hipotezy. Etap drugi i trzeci mają za zadanie wśród możliwych hipotez (zawartych w stałej bazie wiedzy) wyróżnić te, które należałoby szczegółowo rozpatrzeć na etapie diagnozowania końcowego. Na podstawie znanych systemowi przesłanek diagnostycznych (ustalonych w trakcie etapu wstępnego) oraz stwierdzonych (w etapie pierwszym) agregatów, w etapie drugim ustalana zostaje lista hipotez aktywnych. W ten sposób, podobnie do postępowania eksperta, uwaga systemu skoncentrowana zostaje na pewnej ilości diagnoz, które należy rozpatrywać w dalszym postępowaniu diagnostycznym. Warunki przyjęcia "diagnozy" do grona hipotez aktywnych, określone są w stałej bazie wiedzy. Podstawa kwalifikacji diagnozy do grona hipotez aktywnych jest spełnienie conajmniej jednego z warunków uaktywniających przypisanych każdej z diagnoz w stałej bazie wiedzy.

Etap ustalania listy hipotez aktywnych przebiega bez aktywnego udziału użytkownika, czyli że system korzysta w nim z danych zebranych w etapie 0 oraz wniosków dotyczących występowania agregatów ustalonych w etapie I.

Lista hipotez wstępnych uporządkowana zostaje ostatecznie według prawdopodobieństwa a'priorycznego występowania diagnozy , ustalonego na podstawie badań statystycznych i zadanego systemowi w bazie wiedzy.

III. USTALENIE ZAWĘŻONYCH HIPOTEZ WSTĘPNYCH

Celem tego etapu jest, w oparciu o dotychczasowe ustalenia systemu oraz odpowiedzi użytkownika, udzielane na szczegółowe pytania systemu, zadawane w tym etapie diagnozowania, wyodrębnienie ze zbioru hipotez wstępnych najbardziej prawdopodobnych. Pytania te odnoszą się do objawów spełniających dwa warunki:

- są specyficzne dla wykluczenia lub potwierdzenia aktywnych hipotez,
- uzyskanie przez użytkownika informacji, celem odpowiedzenia na pytanie systemu , nie wymaga skomplikowanych i czasochłonnych zabiegów.

Za zawężoną hipotezę wstępną uważana jest każda ta hipoteza, aktywna, która w tym etapie wnioskowania systemu zostanie:

- potwierdzona lub
- nie anulowana.

Każdemu warunkowi uaktywniającemu diagnozę odpowiadają w bazie wiedzy dwa zbiory warunków:

- warunki anulujące,
- warunki potwierdzające.

Warunki te mają postać złożonego wyrażenia logicznego (reguły). Jego argumenty stanowią przesłanki diagnostyczne i agregaty (na tym etapie znane już systemowi) oraz objawy , o które zapyta system użytkownika.

Kolejno system sprawdza warunki anulujące i potwierdzające (w tej właśnie kolejności) związane ze wszystkimi spełnionymi w etapie drugim warunkami uaktywniającymi diagnozy. Wynikiem tego sprawdzenia jest ostateczne potwierdzenie, niepotwierdzenie, lub anulowanie "hipotezy" .

Na liście zawężonych hipotez wstępnych motor wnioskowania umieści kolejno hipotezy potwierdzone, a dalej aktywne, nie aktywne, anulowane i wykluczone.

Prototyp systemu wykorzystywany jest obecnie do tworzenia systemu wspomagania diagnostyki żółtaczek niemowlęcych (system ICTERUS). Jak wykazały prowadzone w ramach CPBR 11.9 prace schemat wnioskowania systemu prototypowego może być ściśle dopasowany do schematu rozumowania lekarza diagnosty. Oczywiście nie jest to jedyne możliwe zastosowanie systemu szkieletowego. Inne jego zastosowania zależeć będą od współpracy inżynierów wiedzy i specjalistów ekspertów z różnych dziedzin wymagających diagnozowania.

LITERATURA:

[Brownston 86]
L. Brownston, R. Farrell, E. Kant, N. Martin,
"Programming Expert Systems in OPS5. An Introduction to Rule-Based Programming",
Addison-Wesley, 1986.

[Charniak 85]
E. Charniak, D. McDermott,
"Introduction to Artificial Intelligence",
Addison-Wesley, 1985.

[Citrenbaum 87]
Citrenbaum R., Geissman J.R., Shultz R.,
"Selecting a Shell".
AI Expert, Vol. 2, No 9,1987

[Cholewa 87]
Cholewa W., Pedrycz W.,
"Systemy doradcze".
Politechnika Śląska , skrypt uczelniany nr 1447, 1987

[Coombs 84]
M.J. Coombs /ed./,
"Developments in Expert Systems",
Academic Press, 1984.

[Forsyth 84]
R. Forsyth /ed./,
"Expert Systems. Principle and case studies",
Chapman and Hall, 1984.

[Fox 86]
J. Fox /ed./,
"Expert Systems - State of the Art Report 12:7",
Pergamon Infotech, 1986.

[Freedman 87]
Freedman R.,
"27 - Product Wrap-up: Evaluating Shells".
AI Expert, Vol. 2, No 9, 1987

[Harmon 85]
P. Harmon, D. King,
"Expert Systems. Artificial Intelligence in Businnes",
J.Willey & Sons, 1985.

[Hayes-Roth 83]
F. Hayes-Roth /ed./,
"Building Expert System",
Addison-Wesley, 1983.

[Hayes-Roth 85]
F.Hayes-Roth,
"Rule-Based Systems",
Comm. of the ACM. Vol.28. No.9, 1985, pp.921-931.

[Hunt 84]
R.M. Hunt, W.B. Rouse,
"A Fuzzy Rule-Based Model of Human Problem Solving",
IEEE Trans. SMC, Vol.SMC-14, No.1, 1984, pp.112.

[Kidd 86]
A. Kidd, M. Welbank,
"Knowledge acquisition",
in: J. Fox /ed./, Expert Systems-State of the Art
Report 12:7, 1986.

[Kluźniak 83]
Kluźniak F., Szpakowicz S.,
"PROLOG".
WNT, 1983

[Lauviere 83]
J.L. Lauviere,
"Knowledge Representation and Use", Part.1, Part.2.
Technology and Science of Informatics, Vol.1, No.2, 1983.

[Lindsay 84]
P.H. Lindsay, D.A. Norman,
"Procesy przetwarzania informacji u człowieka".
PWN, Warszawa 1984.

[Macowicz 88]
M.Macowicz.
"Praktyczny generator parserów dla gramatyk klasy LL(1)".
Praca dyplomowa zrealizowana w Instytucie Informatyki AGH, 1988

[Marcot 87]
B. Marcot,
"Testing your Knowledge Base".
AI Expert, Vol.2, No.8, 1987.

[Matthews 87]
Matthews M. H.,
"Maintenance & Language Choice",
AI Expert, Vol. 2, No 9,1987

[Nilsson 80]
N.J. Nilsson,
"Principles of Artificiel Intelligence",
Tioga, 1980.

[Parnas 85]
D.L. Parnas,
"Software Aspects of Strategis Defense Systems",
Comm. of the ACM, Vol.28. No.12, 1985. pp.1326.

[Sell 85]
P.S. Sell,
"Expert Systems - A Practical Introduction",
Halsted Press Book, 1985.

[Shirai 86]
Y. Shirai, T. Tsujii,
"Artificial Intelligence. Concepts, Techniques,
Applications",
John Wiley Sons, 1984 (rep. 1986).

[Stroustrup 86]
Stroustrup B.,
"The C++ Programming Language". Addison -Wesley, 1986

[Shortliffe 76]
E.H. Shortliffe,
"Computer-Based Medical Consultations: MYCIN",
Elsevier, New York 1976.

[Takeguchi 84]
T. Takeguchi, H. Akashi,
"Analysis of Decision Under Risk with Incomplete Knowledge",
IEEE Trans. SMC, Vol.SMC-14, No.4, 1984.

[Waterman 86]
D.A. Waterman,
"A Guide to Expert Systems",
Addison-Wesley, 1986.

Zdzisław Semółka

METODA WZORCA STANÓW W PROJEKTOWANIU SYSTEMÓW INFORMACYJNYCH

1. Wprowadzenie

Nowoczesne techniki projektowania systemów informacyjnych rozwinęły i udoskonaliły szeroki wachlarz narzędzi komputerowego wspomagania procesów projektowania. Głównym nurt tworzą tutaj narzędzia /zintegrowane pakiety użytkowe, oprogramowanie narzędziowe, generatory programów/ ułatwiające rozwiązywanie szczegółowych zadań projektowych od strony formalnej i technologicznej. Trudno natomiast wskazać zadowalające narzędzia wspomagające projektanta w zasadniczej fazie budowy systemu to jest w fazie modelowania konceptualnego.

Pogłębianiu dysproporcji w rozwoju narzędzi komputerowego wspomagania procesów projektowania systemów informacyjnych sprzyja utrzymująca się w naszym kraju tendencja wykorzystywania sprzętu komputerowego i mikrokomputerowego do płytkiego przetwarzania danych masowych w kolejnych wersjach systemów dziedzinowych[1]. Systemy te są szybkie a utrzymywane a jednocześnie dobrze rozpoznane. Ich realizacja nie wymaga przeprowadzenia pogłębionej analizy systemu informacyjnego przedsiębiorstwa ani dysponowania modelem jego funkcjonowania i rozwoju. Prowadzi to, jak słusznie zauważa W. Flakiewicz do podtrzymywania tendencji marginesowego wykorzystywania potencjalnych moż-

[1] Zob. W. Flakiewicz: Ograniczenia zastosowań mikrokomputerów w zarządzaniu. W: Aktualne problemy zastosowań informatyki w zarządzaniu, INFOKRAK 87, Kraków-Krynica 1987, s. 222.

limości informatyki i fali kolejnych rozczarowań i krytyk[1].

Zmiana tej sytuacji zależy przede wszystkim od skali rzeczywistych przeobrażeń funkcjonalnych i strukturalnych w modelu gospodarki kraju. Stąd bowiem wynika zarówno zapotrzebowanie na jakość, rodzaj i zakres rozwiązań informatycznych jak i dynamika rozwoju infrastruktury informatyki. Chodzi tu między innymi o zaplecze i produkcję nowoczesnego sprzętu komputerowego oraz oprogramowania, rozwój systemów łączności umożliwiających budowę sieci komputerowych /lokalnych i międzynarodowych/ rozwój systemów kształcenia kadr przyszłych użytkowników.

W sferze zagadnień metodologicznych, wyznaczających efektywne kierunki zastosowań informatyki, zasadniczego znaczenia nabierają problemy budowy narzędzi wspomagających procesy projektowania systemów w fazie modelowania konceptualnego. Wiąże się to m.in. z postępem prac nad sztuczną inteligencją i ich owocami w postaci systemów ekspertowych, które już osiągnęły znaczący sukces komercyjny[2]. Proponowana tutaj metoda wzorca stanów stanowić może jedno z narzędzi budowy architektury logicznej systemu informacyjnego z bazą wiedzy wspomagającej procesy sterowania ekonomicznego nowoczesnych przedsiębiorstw.

2. Wzorzec stanów

W świetle teorii sterowania, punktem wyjścia w projektowaniu systemu informacyjnego jest systemowa analiza problemu funkcjonowania i rozwoju przedsiębiorstwa w niestabilnym oto-

---

[1] Ibidem.

[2] Zob. J. Fox /ed./: Expert Systems. State of the Art Report, Pergamon Infotech Ltd., 1986.

czenia. Wyniki tej analizy tworzą podstawę do ustalenia wzorca /modelu, paradygmatu/ wyrażającego ideał funkcjonowania i rozwoju przedsiębiorstwa. Zgodnie z powszechnie obowiązującym w przyrodzie stereotypem, procesy sterowania są realizowane na zasadzie sprzężenia do wzorca. Wzorzec ten podlega stałym modyfikacjom /ewolucji/ stosownie do zmian w warunkach otoczenia. Procesy sterowania są realizowane w oparciu o wyniki porównywania /pomiaru i oceny/ rzeczywistości z kolejnymi ustaleniami wzorca, pełniącego tutaj funkcję zmiennej normy układu regulacji[1].

W przypadku przedsiębiorstwa, wzorzec wyrażający jego ideał funkcjonowania i rozwoju przedstawia sobą zbiór optymalnych wariantów realizacji zadań lub celów wynikających z relacji popyt-podaż oraz warunków działania przedsiębiorstwa. Warianty optymalne realizacji zadań są tutaj rozumiane jako wynik takiego doboru potencjału wykonawczego do zadań /określonych skalą i strukturą popytu/, przy którym wykorzystane najlepsze z dostępnych metody i modele optymalizacyjne; zarówno matematyczne jak i heurystyczne, przy zastosowaniu techniki komputerowej zapewniającej generowanie wyników w żądanym czasie.

Teoretyczną podstawę generowania zbioru wariantów tworzących zmienną normę układu regulacji stanowi twierdzenie[2], że dla każdego systemu działania istnieje zbiór rozwiązań efektywnych, przy których poprawa jednej z trzech wielkości

---

[1] O. Lange: Cybernetyka, Książka t. 7, PWE, Warszawa 1977, s. 128 i dalsze.

[2] A. Gomółka: Sterowanie ekonomiczne w zapleczu technicznym przedsiębiorstwa transportu samochodowego. W: Matematyczne podstawy teorii systemów transportowych. IBS-PAN, Jabłonna 1980.

sterujących procesami tj. natężenia, jakości lub kosztu powoduje pogorszenie wartości co najmniej jednej z dwu pozostałych.

Podejście takie pozwala sprowadzić problem optymalizacji funkcjonowania przedsiębiorstwa /i budowy wzorca/ do poszukiwania optimum Pareto dla wyróżnionych trzech zmiennych wyznaczających składowe wektora stanu systemu sterowanego.

Wyróżnione kryteria sterowania spełniają warunek funkcji celu, tworzą całościową ocenę przebiegu procesów; oceny ich wykonawców wewnątrz przedsiębiorstwa oraz oceny wyników działania przedsiębiorstwa w otoczeniu. Stanowi to podstawę tworzenia konstrukcji pętli sprzężeń zwrotnych systemu z otoczeniem i sprzężeń wewnątrzsystemowych.

Zbiór wariantów realizacji zadań tworzących wzorzec stanów powinien spełniać warunek

$$W^T = \alpha I^T + \beta J^T + \gamma K^T = \text{const} \; ; \quad I^T > 0, J^T > 0, K^T > 0$$

gdzie

W^T - wartość oceny działania systemu w przedziale czasu T określonym na zbiorze zadań,

I^T, J^T, K^T - wartości ocen natężenia, jakości i kosztu procesów realizowanych w systemie,

α, β, γ - współczynniki wagowe kryteriów sterowania.

Równanie to wyznacza przestrzeń rozwiązań efektywnych, określoną technicznymi warunkami funkcjonowania przedsiębiorstwa w formie zbioru relacji między głównymi składowymi charakterystyk procesów.

3. System informacyjny z wzorcem stanów

Podstawę budowy wzorca stanów tworzą zbiory danych o potencjale wykonawczym przedsiębiorstwa i zadaniach. Procesy analizy stanów składników potencjału wykonawczego, analizy zadań wpływających do systemu oraz doboru potencjału wykonawczego do zadań wykonują w systemie tzw. analizatory stanów[1]. Analizator stanu przedstawia sobą utrzymywany w pamięci komputera zbiór programów i modeli /algorytmów/ optymalizacyjnych służących do syntezy danych z różnych źródeł.

Procesy syntezy danych są ukierunkowane w tym sensie, że obejmują swym zakresem dane tylko o tych składnikach systemu i takich ich właściwościach, które bezpośrednio wpływają na jakość, natężenie lub koszt realizacji zadań.

Analizatory stanów pracują w oparciu o informatyczne podsystemy dziedzinowe, które zapewniają dane do aktualizacji i modyfikacji wzorca stanów.

W zbiorze analizatorów stanów rolę wiodącą /nadrzędną/ spełnia analizator stanu zadań. Analizator ten ustala czas oraz potencjał wykonawczy niezbędny do realizacji zadania zgodnie z oczekiwaniem zleceniodawcy. Następnie za pomocą analizatorów stanów potencjału wykonawczego określa możliwości realizacyjne zadania. Proces kojarzenia potencjału wykonawczego z zadaniami przebiega przy wykorzystaniu matematyczno-ekonomicznych modeli optymalizacyjnych oraz zgodnie z przyjętymi w systemie priorytetami. Wyróżnia się tutaj dwie klasy prio-

[1] Zob. Z. Gomółka: Koncepcja sterowania żeglugą nieregularną. W: Problemy budowy systemu informatycznego zarządzania przedsiębiorstwem żeglugi nieregularnej. Prace Naukowe Politechniki Szczecińskiej, Nr 199, Szczecin 1982, s. 80-83.

rytetów decydujących o kolejności przydziału potencjału:

- priorytety quasi-stałe wynikające z przyjętej strategii sterowania,
- priorytety doraźne /nadrzędne/ stosowane w sytuacjach wyjątkowych /np. stan klęski żywiołowej/.

W wyniku prac analitycznych tworzony jest układ wariantów realizacji zadań ze wskazaniem

a/ zadań, które w żądanym czasie nie będą zrealizowane z braku potencjału wykonawczego. Analizator określa brakujące składniki potencjału oraz najbliższy termin w którym dane zadania będą mogły być wykonane,

b/ składników potencjału przedsiębiorstwa, który nie został zaangażowany do wykonawstwa zgłoszonych zadań.

Z chwilą zaakceptowania wariantów realizacji zadań przez decydentów reprezentujących system zarządzania przedsiębiorstwem, analizatory dokonują szczegółowej rezerwacji poszczególnych składników potencjału wykonawczego.

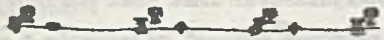
Zadania wraz z realizacją określonym potencjałem wykonawczym przedstawiają sobą wszystkie pożądane stany działania przedsiębiorstwa. Wszystkie te pozwala określić możliwe do osiągnięcia w danych warunkach wartości głównych parametrów działania przedsiębiorstwa w postaci wartości, jakości i kosztów procesów realizacyjnych oraz oceniać wielkość spodziewanych dochodów z wykonania zadań lub dochodu utraconego w wyniku nie podjęcia ich realizacji.

Podstawą do bieżącej aktualizacji i modyfikacji danych /zbiorów o wyspecjalizowanych podsystemach działalności/ dane o rzeczywistym przebiegu realizowanych zadań. Dane te analizator stanu zadań porównuje z danymi i na tej podstawie

oblicza wartości odchyleń procesów rzeczywistych od zadanych wzorcem:

$$V^2 = \alpha \Delta x^2 + \beta \Delta \dot{x}^2 + \gamma \Delta \ddot{x}^2$$

gdzie:



gdzie

V^2 - wartość oceny rezultatów sterowania,

α, β, γ - współczynniki wagowe kryterium sterowania

Δ - wartość odchyleń procesów rzeczywistych i wzorcowych

Proponowana konstrukcja miernika zapobiega tendencjom poprawy rezultatów działania drogą obniżenia jakości lub podwyżki kosztów wytwarzania.

Współczynniki wagowe kryterium sterowania można w konkretnych warunkach działania przedsiębiorstwa ustalać przy wykorzystaniu symulacji komputerowej i metod programowania heurystycznego lub bardziej wyrafinowanych metod sterowania nieliniowego, stochastycznego i optymalnego, wykorzystywanych współcześnie w automatyce[1].

Sterowanie można uznać za optymalne, gdy efekty tego sterowania są co najmniej równe ustalonym we wzorcu; gdy $V^2 \geq 0$.

Wzorzec podlega modyfikacji w każdym przypadku istotnych zmian w rzeczywistych warunkach realizacji zadań /np. zmarie urządzeń/ lub gdy wyniki uzyskiwane faktycznie wykazują trwałą tendencję odchyleń /dodatnich lub ujemnych/ od wzorca. Pomimo to wzorzec podlegać będzie modyfikacjom wynikającym z rozwoju

[1] Zob. Y. Takahshi, M. Rabins, D. Auslander: Sterowanie i systemy dynamiczne, WNT, Warszawa 1976, s. 458-667.

bazy wiedzy systemu i sprowadzania innowacji technicznych i organizacyjnych.

4. Uwagi końcowe

Proponowany sposób podejścia do projektowania i realizacji systemu informacyjnego przedsiębiorstwa może mieć miejsce w sytuacji, gdy spełnione są zasadnicze warunki sterowania w sferze makroekonomicznej i rachunek ekonomiczny stanowi rzeczywiste a nie tylko formalne narzędzie wyboru rozwiązań efektywnych[1].

System informacyjny w zarysie, zapewnia kierownictwu pełny i szybki przegląd sytuacji przedsiębiorstwa, stałą obserwację istotnych zmian sytuacji, zdolność przewidywania skutków bieżąco podejmowanych decyzji oraz skuteczne i racjonalne reagowanie na zakłócenia.

Realizacja tej klasy systemów wymaga spełnienia szeregu wymagań jakościowych zarówno w sferze technicznej jak i programowej. Wymagane tutaj czasy realizacji zadań zapewnić można wykorzystując współczesne wielofunkcyjne sieci komputerowe i techniki programowania symulacyjnego.

Problemem otwartym jest doskonalenie bazy wiedzy systemu. Chodzi przy tym nie tylko o dobór i właściwą selekcję matematyczno-ekonomicznych modeli optymalizacyjnych zapewniających efektywny dobór potencjału wykonawczego do zadań. Istotne znaczenie mają bowiem tutaj modele i wiedza ekspertów wykorzystywane w procesach generowania informacji o aktualnych i przewidywanych stanach wielorakich składników potencjału wykonawczego.

---

[1] Zob. J. Goździński: Sterowanie i planowanie. Ujęcia systemowe, PWE, Warszawa 1982, s. 210 i nast.

Jakość informacji o stanach systemu i stanach jego bieżącego i dalszego otoczenia oraz czas generowania tych informacji stanowią zasadnicze kryteria w poszukiwaniu rozwiązań optymalizacyjnych tej klasy systemów.

Literatura

1 Flakiewicz W.: Ograniczenia zastosowań mikrokomputerów w zarządzaniu. W: Aktualne problemy zastosowań informatyki w zarządzaniu. INFOKRAK 87, TNOiK, Kraków-Krynica 1987

2 Fox J. /ed./. Expert Systems. State of the Art Report. Pergamon. Inf. Lim., England 1986

3 Gomółka Z.: Koncepcja sterowania żeglugą nieregularną. W: Problemy budowy systemu informatycznego zarządzania przedsiębiorstwem żeglugi nieregularnej. Prace Naukowe Politechniki Szczecińskiej, Nr 199, Szczecin 1982.

4 Gomółka Z.: Sterowanie ekonomiczne w zapleczu technicznym przedsiębiorstwa transportu samochodowego. W: Matematyczne podstawy teorii systemów transportowych. IBS-PAN, Jabłonna 1980.

5 Gościński J.: Sterowanie i planowanie. Ujęcie systemowe. PWE, Warszawa 1982

6 Takahahi Y., Rabins M., Auslander D.: Sterowanie i systemy dynamiczne. WNT, Warszawa 1976.

O DEFINIOWANIU SYSTEMU INFORMATYCZNEGO

dr Jacek Irlik
Instytut Systemów Sterowania
ul.Armii Czerwonej 101
40-157 Katowice

Niejednokrotnie już miałem okazję prowadzić lub brać udział w dyskusjach, w których stawiana była kwestia zdefiniowania systemu informatycznego, który ma być przedmiotem opracowania dla określonego użytkownika.
W dyskusjach tych prezentowałem stanowisko, że koniecznie należy dążyć do uzyskania specyfikacji systemu jako materiału, który powinien być - jako konieczny - podstawą przystąpienia do prac projektowo-programistycznych.
Wielu uczestników wspomnianych dyskusji prezentowało pogląd, że opracowywanie takiej definicji jest albo niemożliwe albo niecelowe albo jedno i drugie.
Jest oczywiste, że uzyskanie takiej specyfikacji systemu, która mogłaby być dobrą podstawą rozpoczęcia prac projektowych jest zadaniem trudnym.
Użytkownicy nie wiedzą na ogół, jak przedstawić swoje problemy, aby było to czytelne dla informatyka oraz przekładalne na specyfikację funkcji systemu. Informatycy mają trudności w zrozumieniu istoty problemów, które rozwiązywać ma przyszły system informatyczny. Wynika z tego, że tworzenie takiej specyfikacji jest zadaniem które,musi być rozwiązywane przez obie te grupy wspólnie, w drodze procesu iteracyjnego.

W literaturze (np. M.M.Lehman, L.A.Bellady ; Program Evolution, AP 1985), zwraca się uwagę na to, jakie znaczenie ma opracowanie i udokumentowanie specyfikacji systemu w całym cyklu życia oprogramowania. Specyfikacja taka jest dokumentem, który - obok innych - ewoluuje wraz z oprogramowaniem aż do chwili wycofania go z eksploatacji. Nie można negować potrzeby opracowywania takiej specyfikacji tylko dlatego (a takie są moim zdaniem istotne przesłanki wspomnianych wyżej poglądów), że nie umiemy tego robić.
Źle opracowane specyfikacje, które istotnie mało lub nic nie dają w późniejszych pracach, nie mogą być traktowane jako dowód, że nie są one potrzebne w ogóle.

Co powinna zawierać i jak powinna wyglądać specyfikacja, która może być podstawą opracowania projektu systemu ? Nie wchodząc w zagadnienia metod jej uzyskania chciałbym przedstawić w niniejszym referacie wymagania dotyczące jej zawartości i formy, które ujęte są w Standardzie ANSI/IEEE nr 830-1984, dotyczącym opracowania dokumentu p.n. "Software Requirements Specification" (SRS). Należy zwrócić uwagę na fakt, że zgodnie z ANSI opracowanie, udokumentowanie i archiwowanie SRS jest jednym z istotnych elementów technologii wytwarzania oprogramowania i jest warunkiem koniecznym zapewnienia należytej jego jakości.
Co na temat SRS mówi Standard ANSI/IEEE nr 830-1984 ?
W uwagach ogólnych na temat SRS stwierdza się przede wszystkim, że dokument ten powinien być :

1) jednoznaczny,

2) zupełny,

3) weryfikowalny,

4) spójny,

5) modyfikowalny,

6) łatwy do prześledzenia,

7) możliwy do wykorzystania w fazie tworzenia, eksploatowania i pielęgnacji oprogramowania.

ad 1) Przez jednoznaczność SRS należy rozumieć, że każde wymaganie, które jest w niej zapisane ma tylko jedną interpretację. Dotyczy to zwłaszcza przypadków, w których wymagania zapisane są w języku naturalnym stwarzającym możliwość zapisania zdań interpretowalnych wieloznacznie. Postulat ten jest więc oczywistym, ze względu na cel SRS postulatem precyzyjnego operowania językiem, w którym ma być napisana. Można bowiem podać wiele przykładów wymagań pozornie jedynie jednoznacznych.

ad 2) Kompletność SRS oznacza, że zapisane są w niej wszystkie istotne wymagania dotyczące:

- funkcji, które zapewnić ma oprogramowanie,
- zewnętrznych charakterystyk przetwarzania,
- postulatów w stosunku do rozwiązań projektowych,
- interfejsów ze środowiskiem software'owym oraz z użytkownikiem.

Kompletna SRS powinna również zawierać definicje odpowiedzi systemu na wszystkie możliwe klasy

danych wejściowych dla wszystkich możliwych klas sytuacji. Oznacza to oczywiście konieczność zdefiniowania odpowiedzi systemu również w przypadkach, kiedy dane są nieprawidłowe.

Kompletna SRS powinna zawierać wszystkie części, które przewidziane są w niej zgodnie z przyjętym standardem. Jeżeli jakiś element standardu nie ma w przypadku konkretnej SRS zastosowania, należy umieścić informację o tym fakcie wraz z wyjaśnieniem niestosowalności.

W kompletnej SRS nie powinny mieć miejsca frazy "do późniejszego ustalenia". Jeżeli jakieś konkretne wymaganie nie może być określone w chwili tworzenia SRS, należy określić sposób, w jaki ustalenia to będzie dokonane.

Oczywistym wymaganiem w stosunku do kompletnej SRS jest pełna i jednoznaczna numeracja jej fragmentów oraz jednoznaczne odwołania w tekście do rysunków, tablic, diagramów itp.

ad 3) Weryfikowalność SRS oznacza, że dla każdego wymagania w niej umieszczonego będzie istniała możliwość jednoznacznego sprawdzenia, że produkt finalny wymaganie to spełnia.

Przykładami wymagań niezweryfikowalnych są:

- "oprogramowanie powinno być przyjazne dla użytkownika",
- "żaden program nie może wejść w nieskończoną pętlę".

- "odpowiedź będzie uzyskana zwykle nie później niż po 10 sekundach".

Wymagania, które nie są weryfikowalne powinny być sprowadzone do postaci weryfikowalnej poprzez odpowiednie sprecyzowanie - lub usunięte z SRS. Jeżeli określone wymaganie jest istotne, a nie może być sprecyzowane w chwili definiowania SRS, należy określić, w której fazie tworzenia produktu zostanie ono sprecyzowane.

ad 4) Spójność SRS oznacza, że żadne z wymagań umieszczonych w SRS nie może w jakimkolwiek stopniu prowadzić do wniosku będącego w sprzeczności z którymkolwiek z innych wymagań. W przypadku większej liczby wymagań i większej objętości otrzymanie spójnej SRS nie jest wcale takie oczywiste. Wspomniane sprzeczności mogą najczęściej dotyczyć definicji reakcji systemu, formatów wyprowadzanych danych, kolejności działań, które mają być wykonywane itp.

ad 5) Modyfikowalność SRS wiąże się głównie z redakcją dokumentu. Dotyczy to w szczególności spraw związanych z przejrzystością, właściwą numeracją części i poszczególnych wymagań oraz z unikaniem redundancji w zapisie tych wymagań. Unikanie redundancji ma takie znaczenie, że w przypadku konieczności zmiany określonego wymagania łatwo doprowadzić do powstania niespójności w SRS będącej wynikiem takiej zmiany (w kolejnej wersji SRS).

ad 6) Postulat, aby SRS była łatwa do prześledzenia oznacza, że możliwe jest łatwe ustalenie:

- z jakich wcześniejszych ustaleń, powodów lub dokumentów wynika każde z wymagań w niej zawartych,
- jakie elementy później powstających dokumentów będą się odwoływać do określonego z wymagań.

ad 7) Znaczenie SRS w późniejszych fazach tworzenia, eksploatacji i pielęgnacji oprogramowania stwarza postulaty, aby SRS zawierała informacje o szczególnym charakterze niektórych wymagań.

Są to informacje o tym, że np:

- nieprawidłowe wykonanie określonych funkcji może spowodować określone zagrożenia lub szkody,
- określone funkcje są niezbędne jedynie np. w początkowym okresie eksploatacji,
- określone funkcje będą zaimplementowane przez włączenie do systemu już istniejących produktów,

itp.

Wymienione wyżej siedem postulatów wyraża łącznie intencję, aby SRS stanowiła jednoznaczny dokument będący podstawą wytworzenia i zweryfikowania produktu finalnego. Dokument ten, aby sprostać wymienionym postulatom, musi być dokumentem o dość rozbudowanej strukturze.

Omawiany Standard ANSI/IEEE nr 830-1984 przedstawia kilka wariantów struktury SRS, z których jeden, wybrany dla przykładu, powinien zawierać następujące elementy:

1. Część wstępna.
   1.1. Opis znaczenia danej SRS.
   1.2. Opis ogólny zastosowania produktu finalnego.
   1.3. Definicje użytych pojęć,użyte akronimy i skróty.
   1.4. Odwołania do innych standardów i literatury.
   1.5. Przegląd zawartości SRS.

2. Część ogólna.
   2.1. Znaczenie produktu finalnego w szerszym kontekście systemowym.
   2.2. Funkcje produktu.
   2.3. Charakterystyka użytkowa.
   2.4. Ogólne wymagania.
   2.5. Ogólne założenia.

3. Wymagania szczegółowe.
   3.1. Wymagania funkcjonalne.
      3.1.1. Wymaganie 1.
         3.1.1.1. Wstęp.
         3.1.1.2. Dane wejściowe.
         3.1.1.3. Przetwarzanie.
         3.1.1.4. Dane wyjściowe.
      3.1.2. Wymaganie 2.
      :
      :
      3.1.n. Wymaganie n.
   3.2. Wymagania dla interfejsów zewnętrznych.
      3.2.1. Interfejsy z użytkownikiem.

3.2.2. Interfejsy z hardwarem.

3.2.3. Interfejsy software'owe.

3.2.4. Interfejsy sieciowe.

3.3. Wymagania dotyczące charakterystyk przetwarzania.

3.4. Postulaty w stosunku do projektu.

3.4.1. Zgodność ze standardami.

3.4.2. Ograniczenia hardware'owe.

3.4.3. Inne.

3.5. Cechy szczególne.

3.5.1. Zabezpieczenia.

3.5.2. Warunki pielęgnacji.

3.5.3. Inne.

3.6. Inne wymagania.

3.6.1. Bazy danych.

3.6.2. Rodzaje operacji systemowych.

3.6.3. Instalowanie i adaptowanie.

3.6.4. Inne.

Postulaty dotyczące struktury i zawartości nie nie rozstrzygają oczywiście kwestii, która w tworzeniu SRS jest najistotniejsza. Jak w konkretnym przypadku formułować konkretne wymagania ?

Określenie tego nie może być przedmiotem standardu dotyczącego zawartości jakiejkolwiek dokumentacji. Zastosowanie określonej metody wyspecyfikowania konkretnego wymagania wynikać musi ze stanu wiedzy technicznej oraz umiejętności zawodowych autorów SRS.

Na początku niniejszego referatu wspomniałem o tym, że

opracowania, udokumentowanie i archiwowania SRS jest jednym z istotnych elementów technologii wytwarzania oprogramowania i jest warunkiem uznanym za konieczny w zapewnieniu należytej jakości produktu finalnego.

W szczególności procedura badań jakościowych przewiduje weryfikacje dokumentacji powstającej w kolejnych fazach fazach wytwarzania oprogramowania.Weryfikacja i walidacja dokumentacji fazy kolejnej dokonywana jest ze względu na zawartość dokumentacji fazy poprzedniej, a dokonywane zmiany rzutują "wstecz" i prowadzą w efekcie do wielokrotnych rewizji SRS,która tym samym jest dokumentem żywym przez cały okres życia oprogramowania.

Jacek Irlik

Koncepcja dwupoziomowego systemu generacji oprogramowania dla lokalnych sieci komputerowych*

Leszek Kotulski

Katedra Informatyki
Uniwersytetu Jagiellońskiego
ul. M. Kopernika 27
31-501 Kraków

W miarę wzrostu złożoności oprogramowania wzrasta zapotrzebowanie na środki wspomagające etapy jego projektowania, implementacji, weryfikacji i validacji. Do mile widzianych cech takiego oprogramowania należą: przenaszalność (portability), łatwość jego pielęgnacji (maintenance) i modyfikacji.

Zaznaczające się w ostatnich latach tendencja do odchodzenia od wielkich specjalizowanych maszyn cyfrowych w kierunku dystrybucji obliczeń (zarówno w sensie funkcjonalnym jak i geograficznym) na mniejsze specjalizowane podsystemy czyni tworzenie systemów wspomagających etap tworzenia i weryfikacji oprogramowania dla lokalnych sieci komputerowych nie tylko celowym, ale wprost koniecznym.

Co prawda Nicolaus Wirth w artykule "Towards a discipline of Real Time Programming" [29] dzieląc programy (ze względu na nakład pracy przy tworzeniu i pielęgnacji oprogramowania) na trzy klasy: programów sekwencyjnych, programów współbieżnych i programów czasu rzeczywistego, nie uwzględnia oprogramowania lokanych sieci komputerowych, tym niemniej trudności, które występują podczas tworzenia i pielęgnacji oprogramowanie lokalnych sieci komputerowych są co najmniej porównywalne z występującymi w programach czasu rzeczywistego. Wynikają one przede wszystkim z konieczności synchronizacji działających w czasie rzeczywistym programów współbieżnych. Należy zwrócić uwagę, że od projektanta oprogramowania dla lokalnej sieci komputerowej wymaga się ponadto dokładnej znajomości różnorodnego sprzętu tworzącego sieć nie tylko z poziomu asemblera ale nawet własności elektronicznych.

Wagę tego zagadnienia doceniło już kilka instytucji takich jak ISO, ECMA, IEEE, Xerox. Efektem ich pracy jest wprowadzenie zestawu standardów obsługi lokalnych sieci komputerowych, które definiują protokoły obsługi będące ciągiem zleceń definiujących zarówno sprzętowe jak i programowe reguły postępowania.

Wprowadzenie standardu 7-warstwowego modelu oprogramowania współpracującego w sieci (ISO OSI Reference Model for open systems interconnections [*]) oraz standardów najpowszechniej stosowanych organizacji fizycznej realizacji połączeń międzywęzłowych (ETHERNET [22], Cambridge Ring [27]) znacznie uprościło implementację oprogramowania w lokalnych sieciach komputerowych. Koniecznym wydaje się jednak podjęcie dalszych prac mających na celu zmniejszenie nakładu pracy poniesionego na zaimplementowanie oprogramowanie zgodnego ze standardami.

Jednym z możliwych kierunków badań byłyby na pewno próby zdefiniowania komputerowo wspomaganych metod formalnej weryfikacji zgodności oprogramowania z omawianymi powyżej standardami LAN. Niestety aktualnie zaawansowanie badań nad automatycznym dowodzeniem poprawności oprogramowania ze względu na złożoność obliczeniową nie przekroczyło praktycznie granicy małych przykładowych programów sekwencyjnych. W dalszym ciągu trwają też prace nad normalizacją standardów oraz przedstawieniem już ustalonych standardów w notacji formalnej (oficjalna wersja podawana jest niestety w języku naturalnym, co powoduje powstawanie niejednoznaczności w ich rozumieniu). Nie wróży to niestety szybkiego rozwiązania problemów związanych z weryfikacją złożonego oprogramowania pracującego w czasie rzeczywistym.

Odmienną propozycję rozwiązania problemu zmniejszenia nakładu pracy programisty implementującego oprogramowanie dla lokalnych sieci komputerowych jest próba ukrycia standardów fizycznej organizacji sieci oraz najniższych warstw modelu ISO jako procedur standardowych, rozkazów języka współbieżnego czy ekstrakodów jądra wspomagającego implementację takiego języka. Jest to propozycja niezwykle atrakcyjna dla programisty implementującego oprogramowanie lokalnej sieci komputerowej, gdyż nie tylko zwalnia go z obowiązku zapewnienia zgodności "wyprodukowanego oprogramowania" z zadanymi standardami ale wprost czyni ich znajomość zbędną. Niestety próby wykorzystania istniejących języków programowania współbieżnego w celu implementacji oprogramowania lokalnych sieci komputerowych nie zakończyły się znaczącym sukcesem [17]. Można wymienić dwa główne powody:

- większość ze szerzej stosowanych języków programowania współbieżnego (Concurrent Pascal [1], Modula [28]) zakładała architekturę ze wspólną pamięcią, nie nadaje się więc dla opisu oprogramowania w LAN. Nieliczne języki programowania współbieżnego w systemach z rozproszonymi zasobami albo nie wypracowały sobie jeszcze zadowalających mechanizmów komunikacji i synchronizacji (CHILL [30], *MOD [2], Concurrent C [26]) albo nie

wyszły jeszcze poza stadium próbnych implementacji (CSP [7]).

- żaden z szerzej stosowanych języków programowania współbieżnego nie jest dostosowany do specyfiki lokalnych sieci komputerowych gdzie konfiguracja sprzętowa ulega dosyć częstym zmianom, nie jest więc możliwym opisać oprogramowania dla lokalnych sieci komputerowych w sztywno zdefiniowanych na etapie kompilacji ramach programu współbieżnego. W szczególności nie jest jeszcze rozwiązany
  - problem dynamicznej alokacji, tzn. możliwości wyznaczania położenia poszczególnych programów bez konieczności rekompilacji programu współbieżnego.
  - problem multi-kompilacji w przypadku, różnorodnej architektury sprzętowej poszczególnych węzłów obliczeniowych kompilator musi generować kody poszczególnych modułów programowych w kilku wersjach docelowych. Obecne kompilatory generują cały kod dla jednej określonej maszyny cyfrowej.

Zakładając wielce prawdopodobne pojawienie się w najbliższym czasie kompilatorów pełnej wersji języka ADA [24] również na komputerach osobistych (istnieje kompilator sekwencyjnej wersji języka ADA - JANUS), to niestety również on nie jest dostosowany do specyfiki lokalnych sieci komputerowych. Ponadto okazało się, że pierwsze pojawiające się kompilatory przeznaczone na mikrokomputery (pomijając ich niezwykle wysoki koszt) posiadają zabezpieczenia sprzętowe, które uniemożliwiają jakąkolwiek modyfikację ich środowiska w kierunku przystosowania ich do środowiska rozproszonego.

Zasygnalizowane powyżej niedogodności stały się powodem podjęcia prac nad systemem wspomagania implementacji oprogramowania współbieżnego w systemach rozproszonych COSID (COncurrent Software Implementation for Distributed systems).

Podstawową ideą tego systemu jest dwupoziomowy model generacji oprogramowania współbieżnego: zwany umownie poziomem kompilacji i poziomem alokacji.

Na poziomie alokacji algorytmy opisujące działanie poszczególnych modułów programowych (zwane dalej wzorcami) są definiowane. Badana jest również kompletność zbioru wzorców opisujących dany system oraz poprawność przekazywania parametrów przy komunikacji pomiędzy modułami wygenerowanymi ze zdefiniowanych wcześniej wzorców.

Na poziomie alokacji z danego wzorca generowana jest wymagana liczba modułów programowych, które zostają alokowane do wskazanych węzłów obliczeniowych. Od języka opisu wzorców wymaga się by używał on mechanizmów synchronizacji

komunikacji niezależnych logicznie od wzajemnego położenia komunikujących się modułów, co pozwala uniknąć dodatkowych ograniczeń na sposób dystrybucji modułów. Również na etapie alokacji ustalane są wzajemne powiązania pomiędzy modułami, z tym że poszczególne powiązania muszą być ustalone zgodnie ze schematem narzuconym przez wzorzec, tj. dany moduł może skorzystać z punktu wejścia oferowanego przez inny moduł pod warunkiem, że ten moduł został wygenerowany ze wskazanego wzorca.

Z punktu widzenia architektury logicznej systemu COSID poziom kompilacji wspomagany jest przez dwa moduły:

- kompilator języka COLNET, pozwalający opisać algorytmy poszczególnych wzorców,
- moduł manipulacji opisem środowiska (EDMM - Environment Description Module Manipulator), który utrzymuje opis generowanego systemu współbieżnego, w celu umożliwienia oddzielnej kompilacji poszczególnych wzorców.

Program źródłowy w języku COLNET pozwala na zdefiniowanie tylko jednego wzorca źródłowego jednostki głównej tzn. procesu albo managera (odpowiedzialnego za synchronizację pracy procesów). Danymi dla kompilatora są zarówno program źródłowy jak i opis środowiska konstruowanego systemu. Po bezbłędnej kompilacji programu opis ten jest wzbogacony o opisy wzorców wynikowych wygenerowanych przez kompilator (w przypadku managera, kilka wzorców wynikowych jest zwykle generowanych, jako że pewne jednostki podrzędne zdefiniowane wewnątrz managera uzyskują samodzielność jako moduły wynikowe np. moduły pośredniczące (interface)). Pewne błędy kompilacji mogą wynikać z niezgodności programu z opisem środowiska. W przypadku gdy winę za to ponosi opis środowiska konfliktowe wzorce muszą być poprawione i ponownie skompilowane, co może spowodować konieczność poprawy i rekompilacji dalszych wzorców. Wynika stąd, że moduł manipulacji opisem środowiska musi zagwarantować możliwość sprawdzenia poprawności relacji "między-wzorcowych" oraz dysponować mini językiem komunikacji z operatorem nadzorującym etap kompilacji wzorców dla danego amu współbieżnego.

Poziom alokacji w systemie COSID wspomagany jest przez kilka modułów. Najważniejszym z nich jest moduł sterujący alokacją, który pozwala zaalokować nowe moduły oraz utrzymuje informację o aktualnym stanie systemu (tzn. ilość i rodzaj już wygenerowanych modułów oraz rozdzaj połączeń pomiędzy nimi). Bada on również kompletność wygenerowanego systemu tzn. czy wszystkie moduły do których istnieją odwołania zostały już zaalokowane.

Wszytkie niezbędne informacje dotyczące wzorców, z których poszczególne moduły wynikowe mają być wykreowane, pobierane są z opisu środowiska systemu założonego przez EDMM, a informacje o strukturze dynamicznej utrzymywane w oparciu specjalnie zdefiniowaną gramatykę grafową (dokładniej patrz [4,5,6]). Do komunikacji z użytkownikiem dokonującym alokacji zdefiniowane są dwa języki komend: jeden bazujący na bezpośrednich rozkazach nakazujących zaalokowanie poszczególnych modułów, a drugi konwesacyjny prowadzący "dialog" z użytkownikiem oparty o graficzną reprezentację alokowanego systemu.
Moduł jądra systemu operacyjnego jest odpowiedzialny za realizację komend wydawanych przez moduł sterujący alokacją (np. załaduj moduł programowy do pamięci komputera n poczęwszy od adresu xx), udostępnienie maszyny wirtualnej realizującej połączenia komunikacyjne pomiędzy węzłami obliczeniowymi oraz podział czasu w ramach jednego węzła obliczeniowego.

W podejściu proponowanym przez system COSID termin "implementacja oprogramowania" jest rozumiany w szerszym sensie jako tworzenie, weryfikacja, testowanie, optymalizacja oraz pielęgnacja oprogramowania. Opowiany powyżej dwupoziomowy model systemu generacji oprogramowania współbieżnego powinien być rozszerzony o dalsze poziomy. W chwili obecnej prowadzone są już intensywne badania teoretyczne nad współbieżnym debugerem dla języka COLMET, który wspomaga etap testowania oprogramowania oraz systemem weryfikacji.
Problemy optymalizacji w systemach współbieżnych dotyczą dwóch zagadnień: optymalizacji kodu sekwencyjnego (które zostało już szczegółowo zbadane dla języków sekwencyjnych) oraz optymalizacji dystrybucji obliczeń. W przypadku środowiska rozproszonego stajemy przed problemem znalezienia kompromisu pomiędzy dwoma przeciwstawnymi tendencjami: maksymalnej dystrybucji (jeżeli każdy z modułów programowych będzie dysponował własnym procesorem to nie będzie strat czasu związanych z oczekiwaniem na jego przydział) oraz minimalnej dystrybucji (współpracujące ze sobą moduły powinny być alokowane w tych samych węzłach obliczeniowych aby uniknąć strat związanych z oczekiwaniem na realizację transmisji w sieci komputerowej).
Trudności ze znalezieniem takiego kompromisu wzrastają ponadto z powodu:

- braku możliwości pierwszej tendencji (w lokalnych sieciach komputerowych współbieżnie pracujących modułów programowych powinno być jest niżwęzłów obliczeniowych sieci, a niektóre z tych programów są ściśle związane z zasobami oferowanymi przez węzeł obliczeniowy),
- braku możliwości w pełni wiarygodnego empirycznego porównania efektywności systemu (nawet w przypadku zapewnienia identycznego zestawu

danych wejściowych ze wszystkich węzłów, co może być już niewykonalne, dwa przebiegi mogą mieć różną efektywność z powodu wzajemnych zależności czasowych).

W przypadku systemu COSID dzięki stosowanemu dwupoziomowemu podejściu do generacji oprogramowania współbieżnego możliwa jest zmiana lokacji poszczególnych modułów programowych bez ingerencji w zdefiniowane na poziomie kompilacji definicje algorytmów. Możliwa jest więc optymalizacja bazująca na intuicji projektanta systemu współbieżnego. Wobec naszkicowanych powyżej trudności pozostawienie tych decyzji jedynie w rękach projektanta wydaje się rozwiązaniem niewystarczającym. Podjęto więc badania mające na celu przygotowanie systemów samouczących mających na celu:

- przygotowanie na podstawie statycznej analizy własności danego systemu współbieżnego oraz zgromadzonej wiedzy na temat systemów generowanych na podstawie wzorców definiowanych z tego samego opisu środowiska propozycji optymalnej alokacji na danych zestawie sprzętowym.
- opracowanie na podstawie informacji o przebiegu pracy systemu (uzyskiwanych od jądra), zgromadzonej wiedzy dotychczasowej pracy tego i "podobnych" systemów propozycji poprawienia efektywności systemu.

Możliwości systemu COSID w fazie alokacji, testowania czy optymalizacji rozproszonego programu współbieżnego są szczególnie interesujące dla projektanta systemu, poszczególne moduły programowe realizują jednak programiści, dla których istotne będą przede wszystkim własności języka COLNET. Postaramy się pokrótce naszkicować to zagadnienie.

Podjęcie problematyki weryfikacji ma pewnien związek z przyjętą definicją mechanizmów komunikacji i synchronizacji języka COLNET. W celu wykorzystania bogatego dorobku teoretycznego dotyczącego weryfikacji oprogramowania współbieżnego przy pomocy konstrukcji monitora, manager będący podstawowym mechanizmem komunikacji i synchronizacji w języku COLNET stanowi uogólnienie koncepcji monitora. Przyjęto więc, że manager jest to pewna struktura danych i zbiór operacji, które mogą być wykonywane na tej strukturze. Dla umożliwienia pracy managera w rozproszonym środowisku, należało przede wszystkim określić sposób realizacji zdalnego wywołania tych operacji. Zaproponowane rozwiązanie w postaci możliwości definiowania "modułu pośredniczącego" w wymianie informacji pomiędzy procesem a managerem (ewentualnie managerami w przypadku zagnieżdżonych wołań menagerów) wydaje się być jednym z podstawowych osiągnięć teoretycznych realizowanych badań [14]. Komunikacja pomiędzy modułem pośredniczącym (interface) a managerem odbywa się przy pomocy wymiany komunikatów, co pozwala na łatwą implementację proponowanego mechanizmu w systemach z rozproszoną pamięcią.

Moduł pośredniczący przekodowując wywołanie procedury na kommikat w rzeczywistości wykonuje część pracy standardowo wykonywanej przez poziom transportowy lokanych sieci komputerowych [9], pozwala żywić nadzieję na dużą efektywnóść proponowanego mechanizmu. Prosta obsługa w postaci zakodowania parametrów wywołania do postaci kommikatu, czynności związane z wysłaniem komunikatu do wskazanego managera i odebraniem kommikatu zawierającego odpowiedź, oraz rozkodowaniem tego kommikatu zwana jest standardową obsługą zdalnego wywołania i nie musi być definiowana. Zdefiniowanie własnej programowej specyfikacji obsługi zdalnego wywołania stwarza (poza umożliwieniem współpracy procesu i managera lokowanych w różnych węzłach komputerowych) następujące możliwości, które wydają się być godnymi uwagi:

- umożliwienie realizacji funkcji rozgłaszania (broadcast) tzn. pozwala by kilka managerów tego samego typu obsługiwało współ bieżnie jeden proces,
- pozwala na połączenie różnych typów komputerów bez rozważania ich sprzętowych własności,
- nawet jeżeli moduł pośredniczący dla danej procedury managera nie jest nigdy programowo definiowany (a komunikacja odbywa się przy pomocy standardowego modułu pośredniczącego) to koncepcja interface'u ułatwia wyjaśnienie semantyki mechanizmu organizującego komunikację,
- umożliwia dynamiczną alokację, ponieważ lokalizacja poszczególnych modułów może być zmieniana bez konieczności rekompilacji pozostałych.

Jedną z niedogodności monitora jest brak możliwości zawieszenia procesu w oczekiwaniu na alternatywę lub konjunkcję zdarzeń elementarnych. Jak już wcześniej zwrócił na to uwagę J.L. Keedy [11] uniemożliwia to zaimplementowanie przy pomocy monitora takich funkcji jak odwieszanie procesu po prawidłowym zakończeniu transmisji albo upłynięciu czasu oczekiwania albo w przypadku sygnału o błędzie. Koniecznością stało się więc zmodyfikowanie operacji Delay [10] poprzez rozszerzenie jej argumentu do wyrażenia kolejkowego. Pod tym pojęciem rozumieć będziemy wyrażenie powstałe z nazwy kolejki lub kilku nazw połączonych konstruktorami or, and, ror lub rep oraz odpowiednio powiązanych nawiasami. Znaczenie dwóch pierwszych konstruktorów jest analogiczne jak operatorów boolowskich or i and tzn. proces będzie ponownie obsługiwany przez manager gdy zostanie odwieszony z jednej z kolejek połączonych konstruktorem or lub wszystkich kolejek połączonych konstruktorem and. Pozostałe specjalnie zdefiniowane w języku COLNET konstruktory pozwalają:

- zawieszać proces w oczekiwaniu na alternatywę wzajemnie wykluczających się zbiorów warunków (ror),
- automatycznie podjąć dalszę obsługę w zależności od przyczyny

odwieszenia (tj. bez konieczności badania wspólnej struktury danych managera) (rep),

- oczekiwać na spełnienie k-krotnego zajścia warunku elementarnego związanego z daną kolejką (rep).

Kolejną istotną modyfikację klasycznego monitora, jest sposób implementacji zagnieżdżonych wołań managerów. Dyskusja na ten temat, którą sprowokował A. Lister [21] została szczegółowo opisana w [12,18]. Jednocześnie dokonano w nich wstępnej analizy proponowanych rozwiązań sugerując (odmiennie niż w klasycznym rozwiązaniu) zwalnianie blokady managera w sytuacji gdy wywołuje on procedurę innego managera. Duży wpływ na ten wybór miał niewątpliwie fakt opracowania metody wykluczania błędów uwarunkowanych czasowo [16], które przy powyższym rozwiązaniu mogłyby się pojawić.

Zarówno w klasycznej postaci jak i jego licznych modyfikacjach monitor nie rozwiązuje w zadowalający sposób problemu "czytelnicy-pisarze". Albo zmuszenie jesteśmy do rezygnacji z kontroli poprawności użycia zasobu (ufając, że procesy dobrze go użyją) albo kontrolując jego użycie poprzez udostępnianie jedynie poprzez dobrze zdefiniowanych procedur umożliwiających jego wykorzystanie musimy zrezygnować ze współbieżnej pracy "czytelników". Celowym wydaje się więc rozszerzanie struktury managera (poza procedurami) o nowe komponenty zwane akcjami, które wywoływane przez procedury managera, będą mogły pracować współbieżnie z nimi wykorzystując "wspólny zasób" chroniony przez manager. W celu wykluczenia możliwości powstania błędów uwarunkowanych czasowo spowodowanych współbieżną pracą na wspólnym zasobie wprowadzono podział struktury danych na wspólną (modyfikowalną wyłącznie przez procedury managera) i zdeponowaną (wykorzystywaną wyłącznie przez akcje) oraz wyróżniono akcje z "biernym" (tylko odczyt) i "czynnym" (potencjalna modyfikacja) dostępem do struktury danych (patrz [19]).

Omówione powyżej składniki managera (tj. procedury wejściowe, moduły pośredniczące (interface) oraz akcje) mają charakter maszynowo niezależny. Oznacza to, że ich kod źródłowy nie musi być modyfikowany w przypadku zmiany typu komputera, przy oczywistym założeniu, że na nowym komputerze został zaimplementowany identyczny kompilator.

Istnieje jednak wiele zastosowań np. obsługa przerwań czy kontrola urządzeń zewnętrznych, w których stosowanie mechanizmów maszynowo niezależnych jest niemożliwe lub nie da zadowalających rezultatów. Wprowadzono więc możliwość definiowania sprzętowo zależnych algorytmów w ramach sprzętowego modułu pośredniczącego oraz sprzętowej akcji. Są one uogólnieniem koncepcji akcji i modułu pośredniczącego w tym sensie, że akcja

sprzętowa realizuje swoje operacje również na strukturze danych której adresy są maszynowo zależne, a sprzętowa moduł pośredniczący odbiera informację o potrzebie wywołania procedury managera nie od procesu lecz od warstwy sprzętowej. Podstawową funkcją obydwu z nich jest zdefiniowanie łącza z częścią maszynowo niezależną reprezentowaną przez procedurę managera. W tym celu posiadają one wspólny nagłówek dla wszystkich wariantów danego algorytmu i podczas alokacji jeden z opisanych algorytmów zostanie użyty. Wybór ten jest szczególnie łatwy do efektywnej realizacji dzięki założeniu, że poszczególne komponenty managera (moduł główny, akcje, moduły pośredniczące, akcje sprzętowe oraz sprzętowe moduły pośredniczące) są z punktu widzenia alokatora samodzielnymi modułami.

Zasygnalizowany model wielopoziomowego systemu wspomagającego generację oprogramowania współbieżnego w środowisku rozproszonym wydaje się szczególnie użyteczny w przypadku lokalnych sieci komputerowych.

Zasada że jednostkę kompilacji stanowi jednostka typu manager lub process, czyli że system współbieżny na poziomie źródłowym opisuje zbiór programów a nie jeden wielojednostkowy program, pozwala by na etapie kompilacji moduły wynikowe były przygotowane do pracy na różnych typach komputerów. Generowany system współbieżny może więc być wykonywany na heterogenicznej lokalnej sieci komputerowej.

Cenną zaletę powyższej propozycji wydaje się również ukrycie przed programistę organizacji warstwy transportowej LAN, co niewątpliwie zmniejsza nakład jego pracy. Inne zalety powyższego rozwiązania wynikają z pewnych cech charakterystycznych języka COLNET (np. tryb rozgłaszania w interface [14], możliwość dynamicznego kreowania i kasowania modułów zwanych akcjami w celu zorganizowania współbieżnego dostępu do zmiennych kontrolowanych przez manager [19]), będą one mogły być jednak należycie ocenione po bardziej szczegółowym omówieniu możliwości oferowanych przez język COLNET oraz system alokacji.

Każda z propozycji nowych języków czy metod programowania w naturalny sposób rodzi pytanie o jej użyteczność i zakres zastosowań.

Połączenie koncepcji akcji, tj. możliwości współbieżnego "biernego" dostępu do struktur danych kontrolowanych przez manager oraz koncepcji modułu pośredniczącego (w szczególności trybu rozgłaszania) wydaje się stwarzać ciekawe możliwości aplikacyjne w rozproszonych bazach danych [13]. Teoretyczne rozważania wskazują użyteczność systemu COSID dla przetwarzania informacji w czasie rzeczywistym.

7. Literature.

1. Brinch Hansen P.: The programming language Concurrent Pascal, IEEE Trans. on Soft. Eng. 1, 2, (1975)
2. Cook, R.P.:*MOD-A language for Distributed Programming. IEEE Trans SE, Nov 80, pp.563-571.
3. Crookes D., Elder J.W.G.: An Experiment in Language for Distributed Systems. Software Practice and Experience, vol. 14(10), 957-971, 1984.
4. Flasiński M.: Parsing of edNLC- graph grammars for scene analisys. Pattern Recognition, 1988, no 3.
5. Flasiński M.,Hliniak G.: Realization of the system controlling a distributed software allocation. Przyjęte do druku w Zeszytach Naukowych UJ.
6. Flasiński M., Kotulski L.: On the use of graph grammar for the control of a distributed software allocation. Przyjęte do druku w The Computer Journal.
7. Hoare C.A.R.: Monitors: an operating system structuring concept. Comm of A.C.M. 1974, 17, 10, pp. 549-557.
8. Hoare C.A.R.: Communicating Sequential Processes. Comm of A.C.M. 1978, 21, 8, pp. 666-677.
9. ISO 'Information processing system - Open System Interconnection - Basic Reference Model', International Standard ISO/OSI 7498 (1983).
10. Karczmarczuk J., Kotulski L.: Process queuing on composite conditions and the use of monitors. Elektrotechnika, vol 6 (1987), no 3, pp.284-292.
11. Keedy J.L.: On Structuring Operating Systems for Monitors. The Australian Computer Journal vol, 10, no 1 (February 1978), pp. 23-27.
12. Kotulski L.: About the semantic nested monitor calls. Sigplan Notice, vol. 22 (1987), no 4, pp 80-82.
13. Kotulski L.: Distributed data base implementation with help of COLNET language. Internal raport of the Dept. of Comp. Sci. Jagiellonian University.
14. Kotulski L.: Interface procedure concept - extension of a monitor concept for distributed environment.Elektrotechnika, tom 7 (1988), z 2., pp. 197-216.
15. Kotulski L.: Preliminary Demands for the Support of Implementation Distributed Software System. Przyjęte do druku w kwartalniku Elektrotechnika.
16. Kotulski L.: Time Dependent Errors Exclusion Method in Monitor Structures. Zeszyty Naukowe, 1987, z. 3, pp. 39-47.
17. Kotulski L.: A two level approach to the implementation LANs software. Złożone w Computer Networks and ISDN systems.
18. Kotulski L.: Nested monitor calls implementation problem. Przyjęte do druku w Zeszytach Naukowych UJ.
19. Kotulski L., Noszner P.: Akcja komponet umożliwiający równoległą pracę wewnątrz monitora. Przyjęta do druku w Podstawach Sterowania.
20. Kotulski L., Onderka Z., Zabolotny A.:The manager and its cooperation with the hardware in the local area networks. Elektrotechnika, vol 6 (1987), no 3, pp.293-300.
21. Lister A.M.: The problem of Nested Monitor Calls. Operating Systems Review, vol 11, no 3, pp, 5-7.
22. Metcalfe R.M., Boggs D.R.: Ethernet: distributed packet switching for local area networks. Comm. of A.C.M. 1976, 19, 8.
23. Mitchell J.G et al: Mesa Language Manual. Report CSL-79-3, Xerox Corporation, Palo Alto Research Center, California 1979.
24. Reference Manual for Ada Programming Language. ANSI/MIL-STD-1815A-1983
25. Sharpe W.R., Cash A.R.: Cambridge Ring 82 - interface specification. UK Science and Engeneering Research Counsil, Sept. 1982
26. Tsujno Y., Ando M., Araki T, and Tokura, N: Concurrent C:A programming Language for Distributed multiprocessor System, S, P&E vol, 14, pp, 1061-1078 (1984).
27. Wilkes M.V., Wheeler D.J.: The Cambridge Digital Communication Ring. Local Area Communication Networks Symposium, Boston, May 1979.
28. Wirth N.:Modula - A programming language for modular multiprogramming. Software Practice and Experience, vol 7, 37-52 (1977).
29. Wirth N.: Towards a Discipline of Real Time Programming. Comm. of ACM, vol 20, no 8(August 1977), pp. 577-583
30. Z200 Recomendation CCITT HIGH LEVEL LANGUAGE (CHILL), CCITT, Geneva 1981

Dr inż. Mieczysław Lech Owoc
Akademia Ekonomiczna
we Wrocławiu

STRUKTURY DANYCH W SYSTEMACH POWIELARNYCH

1. Wstęp

Idea oprogramowania powielarnego, czyli takiego, które może być zastosowane w zróżnicowanych warunkach przetwarzania (odmienność użytkowników, odrębność procedur systemu zarządzania oraz informacyjnego itp.) absorbuje nie tylko dystrybutorów środków informatycznych. Pozostaje ona ciągle także w kręgu zainteresowań zespołów badawczo-wdrożeniowych, które zdają sobie sprawę ze złożoności lecz i zarazem z "intratności" takich produktów informatycznych. Prace nad tym problemem spowodowały utworzenie pośredniej (pomiędzy oprogramowaniem podstawowym i użytkowym) warstwy nazwanej oprogramowaniem narzędziowym. Ze względu jednak na dużą nieoznaczoność struktur informacyjnych i funkcjonalnych przeróżnych systemów informacyjnych niezmiernie trudno jest określić zasady prostego ich odwzorowywania za pośrednictwem odpowiednich kategorii programowych. Stąd też konieczność zdekomponowania całego problemu (por. [MAKO89], [SKWA89]) na względnie odrębne zagadnienia.

W niniejszym referacie zostanie przedstawiony pogląd na temat istoty i uwarunkowań realizacyjnych "datalogicznej" części systemu, a mianowicie aspekty dotyczące właściwości i pewnych zasad realizacyjnych struktur danych w omawianej klasie systemów. Rozważania obejmują sytuację bardzo różne poczawszy od różnorodności wynikających z klas obiektów i relacji zachodzącymi między nimi (przekrój rodzajowy struktur danych) aż do zastosowania wyszukanych konfiguracji sprzętowo-programowych. W części początkowej zostanie przedstawiona klasyfikacja rodzajowa struktur danych

istotna z punktu widzenia omawianego problemu. Stanowi to kanwę rozważanych w dalszych częściach referatu rozwiązań dotyczących konstruowania takich sktruktur oraz ich implementowania.

2. Rodzaje struktur danych

Rozwój metod tworzenia systemów informatycznych jest ściśle związany z możliwościami formułowania przeróżnych zależności pomiędzy danymi przetwarzanymi w systemie. Oznacza to, że jedną z barier ograniczających płynny rozwój systemu informatycznego czy też jego zastosowanie w odmiennych warunkach, stanowią szeroko rozumiane struktury danych. Konieczność formułowania struktur bardzo złożonych, a przy tym łatwo modyfikowalnych, jest zasadą uwzględnianą zarówno przez autorów oprogramowania narzędziowego jak i użytkowego. Spróbujmy oszacować stan dokonań w tym zakresie.

Struktury danych reprezentują - jak wiadomo - pewien wycinkowy, a zarazem abstrakcyjny, model rzeczywistości (zob. np. [WIRTH80]). Za ich pośrednictwem odwzorowywane są pewne wybrane właściwości obiektów, relacje między nimi - wreszcie w sposób deklaratywny można wyrażać określone procedury związane z tymi obiektami. Ważne jest zatem to, że odzwierciedlają one te cechy obiektów, które są ważne z punktu widzenia rozwiązywanego problemu w danej chwili i w przyszłości mogą ulegać istotnym modyfikacjom. Abstrakcyjność struktur danych polega na tym, że stanowią one odwzorowanie umowne zgodnie z konwencją języka opisu odpowiednich atrybutów obiektów.

Języki służące do opisu struktur danych umożliwiają na praktycznie nieograniczone deklaracje złożonych i powtarzalnych układów danych. Przygotowane są do tego specjalizowane - a nawet uniwersalne - języki programowania (zob. [NICHS0] s. 195 i nast.), w równym stopniu co oprogramowanie narzędziowe zorientowane na całościowe rozwiązywanie pewnych klas zadań (np. systemy zarządzania bazą danych, arkusze elektroniczne czy pakiety zintegrowane). Reprezentowane są w ramach tego oprogramowania przeróżne struktury od klasycznych już typów: tablic, zapisów, list aż

po struktury złożone, które swój pierwowzór mają w modelach baz danych sdacznie z relacyjnym. Szczególnie w obrębie tych ostatnich można mówić o strukturach zintegrowanych, bowiem tworzone są zarówno podstawy do łatwego ich rozwoju jak i zalecane są deklaracje odpowiednich węzłów integralnościowych.

Przed sformułowaniem ogólnych wniosków dotyczących przydatności różnych rodzajów struktur danych w systemach powiększalnych należałoby przypomnieć ich klasyfikację . W niniejszym referacie ograniczono się do tych podziałów , które są znaczące dla omawianego zagadnienia, (nie powtarzając szeroko znanych klasyfikacji - zob. np. [TURS71] s. 32 i nast. oraz [WIRTH80]. Wyrazić je można następująco:

- stopień sformalizowania,
- poziom zmienności,
- zakres merytoryczny,
- środowisko programowe.

Przy prezentacji rodzaju struktur dla ww kryteriów, zastosowano dwustanowe rozgraniczenia struktur danych, co wydaje się wystarczające dla postawionego problemu.

Z punktu widzenia pierwszego z przygotowanych kryteriów chodzi o zasadę definiowania i używania danych czyli dotyczy ono sposobu widzenia określonych danych. Nie wnikając w przeróżne możliwości programowe,ogół tych struktur można podzielić na dwie zasadnicze:

a) sformatowane, tzn. takie, które charakteryzują się bardzo zdeterminowanymi powiązaniami poszczególnych danych elementarnych w obrębie całej struktury wraz z podaniem szczegółowych atrybutów (klasa pola, zakres wartości itp.). Istnieje wiele klasycznych już sposobów formalizowania.

b) swobodne, które oznaczają dowolny sposób traktowania poszczególnych danych: w skrajnym przypadku wiązanie atrybutów danych z konkretnymi wartościami jest odraczane dopiero do momentu korzystania z nich z możliwością ich wielorakiej interpretacji.

Kryterium drugie - poziom zmienności różnicuje struktury

danych w aspekcie potencjalnych modyfikacji, którym mogą podlegać. Z tego punktu widzenia omawiane obiekty można podzielić na:

a) stałe, które w procesie rozwoju systemu zachowują swoją pierwotną postać (format, otoczenie programowe, itp.); znamionują one te fragmenty układu danych systemu, które praktycznie zachowują swoją ważność w przeróżnych warunkach funkcjonowania,

b) zmienne, oznaczające możliwość istotnych aktualizacji w procesie dostosowywania do odmiennych procedur użytkownika i jego warunków organizacyjno-technicznych.

Trzecie z przytoczonych kryteriów - zakres merytoryczny - ma istotne znaczenie w przypadku wyraźnej struktury tzw. systemów agendowych. Część struktur danych wykazuje przynależność do określonej dziedziny, natomiast pewien fragment danych ma znaczenie ogólniejsze. To uzasadnia podział struktur na (por. [KOWA86]):

a) systemowe, które są związane z realizacją procedur typowych dla konkretnej agendy,

b) metasystemowe, wyrażające powiązania między danymi na poziomie więcej niż jednej dziedziny.

Ostatnie ze sformułowanych kryteriów, wynika z technologiczności konstrukcji systemu informatycznego czyli z konieczności szeroko rozumianej obsługi przez odpowiednie środowisko programowe. Chodzi przy tym zarówno o proces ich konstruowania (definiowanie struktur) jak i korzystania z nich (specjalistyczne instrukcje programowe). Ogół tych struktur można podzielić na obsługiwane w ramach tzw. oprogramowania:

a) podstawowego, przez które rozumiemy klasyczne języki programowania, z pełną gamą instrukcji deklarowania struktur,

b) narzędziowego, czyli umożliwiającego bazowanie na określonych modelach danych i w kompleksowy sposób obsługującego odpowiednie struktury (systemy zarządzania bazą danych, arkusze elektroniczne, systemy przetwarzania tekstów

itd.).

Na zakończenie tej uproszczonej klasyfikacji należy zadać pytanie dotyczące ogólnej oceny wskazanych struktur z punktu widzenia ich przydatności w systemach powielarnych. Z pewnością struktury wymienione na drugich miejscach w ramach tej klasyfikacji są bardziej odpowiednie dla omawianej klasy systemów. Równocześnie można stwierdzić, że przyjęcie rozwiązań zawierających struktury specyfikowane jako pierwsze - istotnie ogranicza możliwości rozwojowe systemu.

3. Standaryzacja struktur danych

Przedstawiona w części poprzedniej klasyfikacja struktur danych miała na celu specyfikację istotnych z punktu widzenia powielarności systemu, właściwości układów informacyjnych. Aby ten pożądany układ danych osiągnąć, należy w procesie tworzenia systemu zadbać o odpowiednią zawartość merytoryczną i formalną zbiorów danych. Pożądane dla systemów powielarnych struktury powstają w wyniku realizacji odpowiedniej metody postępowania, którą można umownie określić jako tzw. standaryzację struktur danych. Innymi słowy standaryzacja strukur danych polega na takim zdefiniowaniu układu informacyjnego, który jest charakterystyczny dla typowej agendy systemu informacyjnego zapewniając zarazem jego dopasowanie (w ograniczonym zakresie) do przeciętnych użytkowników systemu powielarnego. Ujmując rzecz bardziej formalnie, za struktury standardowe należy uważać takie, które obsługują zasadnicze potrzeby danej klasy użytkowników; struktury takie powinny mieć znamiona optymalności, z punktu widzenia wykonywanych na nich algorytmów (por. [BAN87] s. 81 i nast.).

Proces standaryzacji struktur danych systemu powielarnego można realizować w kilku etapach. Mają one na celu kolejno:

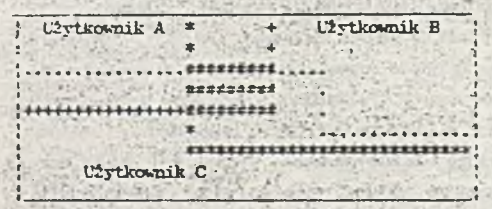
1) wyodrębnienie niezbędnych danych,

2) klasyfikacje danych z punktu widzenia ich roli w systemie,

3) opracowanie standardowych struktur danych.

Podstawową przesłanką determinującą zakres merytoryczny i formalny struktur danych jest struktura funkcjonalna agendy systemu. Zakres realizowanych procedur wraz ze szczegółowymi algorytmami przetwarzania dość jednoznacznie wyznacza rodzaj i postać danych niezbędnych w systemie powielanym. Etap polegający na wyodrębnieniu omawianych danych wynika z przeprowadzanej analizy systemu informacyjnego i tradycyjnie sprowadza się do odwzorowanie zasobów informacyjnych występujących w ramach danej agendy wraz z ich charakterystyką funkcjonalną. Znajduje to wyraz w opracowywanych metodykach budowy modeli konceptualnych.

Graficzny obraz takich wyodrębnionych struktur standardowych wyrażono na rys. 1.



Struktury :
+ - użytkownika A
* - użytkownika B
. - użytkownika C
- standardowe

Rys. 1. Standardowe struktury danych

Istotnego znaczenia w procesie standaryzacji nabiera realizacja drugiego z etapów. Polegać on ma na zakwalifikowaniu określonych danych do grupy tzw. danych systemowych, bądź metasystemowych. Rozgraniczenie to ma decydujący wpływ na wykonanie ostatniego z etapów. Dane oznaczone jako metasystemowe będą stanowiły nadbudowę

informacyjną, która pozwoli dostosowywać system powielarny do innych warunków przetwarzania. Zasadniczo zatem, zaliczyć tutaj można określony zespół kodów (np. komórek organizacyjnych, pewnych nazw własnych) oraz relacji, które są specyficzne dla danego użytkownika. Pozostałe dane określone mianem systemowych powinny zachować swoją postać dla dowolnych użytkowników.

Ostatni z wyodrębnionych etapów sprowadzać się ma do opracowania odpowiednich struktur danych (logicznych, fizycznych) zgodnie ze środowiskiem programowym, w jakim będą one użytkowane. Zachowuje oczywiście swoje znaczenie dokonana w ramach poprzedniego etapu klasyfikacja danych, która jest tu wzmocniona przez utworzenie odpowiednich relacji między danymi metasystemowymi oraz systemowymi.

W procesie standaryzacji należy uwzględnić możliwość rozbudowy struktur zarówno meta- jak i systemowych (odpowiada to formułowanym poprzednio właściwościom wynikającym ze stopnia sformalizowania i poziomu zmienności struktur danych).

4. Adaptacja struktur danych

Jednym z zasadniczych problemów związanych z procesem wdrażania systemów powielarnych jest operacja dostosowania struktur danych do warunków funkcjonowania tego systemu u użytkownika.

Jak wykazano poprzednio, proces ten wyraźnie się upraszcza jeżeli w fazie konstruowania systemu zadbamy o to, żeby przeważająca część struktur danych zrealizować w konwencji standardowych struktur danych. Oznacza to, że w systemie powielarnym istnieje wyodrębniony fragment struktur danych, które bez jakichkolwiek modyfikacji mogą być zastosowane u nabywcy systemu. Część pozostała wymaga natomiast "dostrojenia" do potrzeb użytkownika (por. rys 1). Proces dostosowywania całego systemu do wymagań użytkownika (w tym "dopasowywanie" struktur danych) określamy zazwyczaj mianem adaptacji. W niniejszej części referatu zostanie

przedstawiona ogólna charakterystyka tego procesu w odniesieniu do informacyjnej części systemu.

O przebiegu adaptacji decydują zastosowane w systemie rozwiązania szczegółowe związane z odwzorowywaniem struktur danych. Dla uproszczenia rozważań przyjęto, że adaptacja struktur danych będzie realizowana wg dwu zasadniczych wzorców jako:

a) pasywna - oznaczająca konieczność niewielkich zmian struktur danych,

b) aktywna - polegająca na istotnym przeobrażeniu strony informacyjnej systemu.

Oczywiście w praktyce trudno jest wyznaczyć granice obydwu wzorców postępowania i można nawet przyjąć, że dominować będą rozwiązania pośrednie, które jednak nie zmieniają istoty powyższej klasyfikacji.

Adaptacja struktur danych (niezależnie od przyjętego wzorca) jest procesem, który można zdekomponować na następujące etapy:

1) rozpoznanie systemu informacyjnego użytkownika,

2) ocena struktur danych systemu powielarnego z punktu widzenia obiektu wdrażającego,

3) modyfikacja struktur danych.

W ramach pierwszego z wyspecyfikowanych etapów zostaje przygotowana charakterystyka struktur systemu informacyjnego (funkcjonalnej, informacyjnej, technicznej i przestrzennej). Etap ten powinien być realizowany w sposób konstruktywny, czyli należałoby w nim sformułować wnioski oceniające rozpoznane zasoby informacyjne. Ocena obejmuje sugestie zachowania określonych struktur danych (wraz z wzajemnymi powiązaniami) a także wnioski związane z ich modyfikacją. W ten sposób wyrażona zostanie docelowa postać systemu, co oczywiście w określony sposób koresponduje z rozwiązaniami zawartymi w systemie powielarnym. Rozpoznanie powinno być dokonywane zgodnie z metodyką wypracowaną przez zespół autorski systemu powielarnego (por. np. [SKWA89]). W przypadku kolejnych adaptacji systemu informacyjnego metodyka ta powinna być sukcesywnie doskonalona i przybierać postać

procedury algorytmicznej, a nawet może być wspomagana komputerowo.

Drugi z etapów ma na celu określenie stopnia koniecznych zmian struktury danych w systemie powielarnym. Następuje w nim skonfrontowanie docelowych struktur wynikających z realizacji pierwszego etapu ze strukturami zdefiniowanymi w systemie powielarnym. Ocena powinna się sprowadzić do porównania struktur danych obydwu modeli m.in. w zakresie:

a) kompletności danych z punktu widzenia realizowanych funkcji w systemie,

b) sposobu wiązania danych,

c) formy prezentowania danych,

d) zasad używania zbiorów danych,

e) parametrów efektywnościowych (wielkość zbiorów, czasu dostępu itp.).

Na podstawie dokonanej analizy można określić wzorzec adaptacji konieczny do zastosowania w obiekcie wdrażającym. Podobnie jak poprzednio realizacja drugiego z etapów może przybrać postać sformalizowanej procedury ze wspomaganiem komputerowym włącznie.

Ostatni z wymienionych etapów będzie realizowany odmiennie, zależnie od zdefiniowanego wzorca adaptacji. Procedura modyfikacji struktur danych jest zależna od wyników osiągniętych w poprzednich etapach a także od sygnalizowanych uprzednio rodzaju struktur danych przyjętych w systemie powielarnym.

Zgodnie z przedstawioną tabelą największe możliwości realizacji adaptacji aktywnej istnieją w przypadku zastosowania w systemie powielarnym struktur swobodnych, zmiennych i przy skonstruowaniu systemu z wyodrębnionymi danymi metasystemowymi, obsługiwanych przez oprogramowanie narzędziowe. Adaptacja pasywna może mieć miejsce w przypadku dobrze zdefiniowanego systemu o niewielkiej skali zmienności co odpowiada strukturom sformatowanym, stałym, systemowym obsługiwanym przez oprogramowanie narzędziowe.

Tablica 1

Zależności pomiędzy typami struktur danych a wzorcami adaptacji

Adaptacja ⟶	pasywna	aktywna
Rodzaj struktury		
- sformatowana	+	-
- swobodna	-	+
- stała	+	-
- zmienna	-	+
- systemowa	+	-
- metasystemowa	+	+
wyrażona w oprogramowaniu:		
- podstawowa	+	-
- narzędziowa	+	+

Objaśnienia:

\+ - zgodność struktury ze wzorcem adaptacji

\- - istotne ograniczenie struktury dla wzorca

. - niewykorzystana struktura dla wzorca

5. Zakończenie

Na podstawie określonych uprzednio warunków osiągnięcia standardowych struktur danych oraz ogólnej charakterystyki procesu adaptacji tych struktur można sformułować następujące konkluzje:

a) proces opracowania systemu powielarnego powinien się rozpoczynać od jawnie wyrażonych parametrów jego uniwersalności (klasa użytkowników, zakres funkcjonalny itp.),

b) w ramach definiowanego systemu powielarnego należy zdefiniować w sposób modelowy struktury danych: mają one w odniesieniu do danego systemu postać standardu,

c) proces adaptacji systemu powielarnego w obszarze "dopasowywania" struktur danych musi być realizowany zgodnie ze sformułowaną procedurą.

Bibliografia

[BANA87] Banachowski L., Kreczmar A., Rytter W.: Analiza algorytmów i struktur danych. WNT Warszawa 1987.

[KOWA86] Kowalczyk S., Owoc M.: Konceptualizacja metasystemu dla hierarchicznych systemów baz danych. Mat. z konferencji "Problemy tworzenia i eksploatacji systemów baz danych". Prace Naukowe AE Wrocław Nr 362 Wrocław 1986.

[MAKO89] Makowska-Łukasik B.: Standaryzacja systemów powielarnych (artykuł w niniejszych materiałach).

[NICH80] Nicholls J.E.: Struktury języków programowania WNT Warszawa 1980.

[SKWA89] Skwarnik M.: Standaryzacja warunków eksploatacji systemów powielarnych (artykuł w niniejszych materiałach).

[TURS71] Turski W.M.: Struktury danych NT Warszawa 1971.

[WIRT80] Wirth N.: Algorytmy + struktury danych = programy. WNT Warszawa 1980.

dr M. Skwarnik
Akademia Ekonomiczna
we Wrocławiu

ANALIZA STRUKTUR OBIEKTÓW W TRAKCIE TWORZENIA SYSTEMÓW POWIELALNYCH

W ostatnich latach w sposób istotny wzrosło zainteresowanie zastosowaniem komputerów w zarządzaniu. Ma to ścisły związek z pojawieniem się na rynku mikrokomputerowych systemów informatycznych obsługujących tą sferę działalności obiektów gospodarczych. Większość producentów systemów reklamuje swoje produkty jako systemy powielarne, standardowe lub powtarzalne.

W dalszej części opracowania powielarność będzie traktowana jako cecha użytkowa systemu, określająca możliwości jego krotnego wdrożenia w obiektach gospodarczych. Powielarność jest cechą względną - ocenić ją można dopiero wtedy, gdy system odniesiemy do konkretnego obiektu lub grupy obiektów. Dążenie do powielarności systemu współgra z próbami opracowania systemu stanowiącego standard dla pewnej klasy/grupy obiektów gospodarczych - jeśli system faktycznie jest standardowy to szanse jego powielarności istotnie rosną, jeśli nie to maleją.

W chwili obecnej dominują na rynku systemy, które dają szansę dopasowania co najmniej części rozwiązań systemu do potrzeb obiektu poprzez selekcję rozwiązań i parametryzację systemu w ramach instalacji/inicjacji lub reinstalacji/reinicjacji systemu. W opracowaniu zajmiemy się analizą metodyki tworzenia tej grupy systemów. Nazwiemy je tu umownie systemami powielarnymi elastycznymi. Ich charakterystykę zawarto w [ŁUKASOa].

Można mówić o dwóch podstawowych strategiach tworzenia systemów tej klasy:
1/ diagnostycznej (ewolucyjnej),
2/ prognostycznej.

W przypadku pierwszej z nich punktem wyjścia jest system indywidualny opracowany dla pojedynczego obiektu, wdrażany następnie po zmianach i przeróbkach w kolejnych obiektach. Po pewnym okresie eksploatacji i po wykonaniu pewnej liczby wdrożeń twórcy systemu dochodzą do wniosku, że

nieopłacalne staje się generowanie wielu odmiennych wersji systemu i równoległa ich konserwacja. Starają się więc uogólnić nabyte doświadczenia modyfikując istniejący system w taki sposób, aby stał się on funkcjonalnie i realizacyjnie równoważny wielu wersjom systemu dotychczas eksploatowanym. Mechanizmami adaptacyjnymi obejmuje się w tej sytuacji przede wszystkim te rozwiązania, które realizowano odmiennie w poszczególnych wersjach systemu wcześniejszego.

Strategia druga polega na tworzeniu od podstaw systemu, który zgodnie z przyjętymi założeniami ma być powielarny dla pewnej grupy obiektów.

Obydwie strategie tworzenia systemów mają swoje wady i zalety. Do głównych zalet pierwszej z nich należą:

1/ doświadczenie zespołu projektowo-programistycznego w rozwiązywaniu zagadnień merytoryczno-technologicznych związanych z danym typem systemu,

2/ doświadczenia uzyskane w efekcie wdrożenia danego typu systemu, pozwalające zidentyfikować i ominąć usterki i mankamenty wcześniejszych wersji systemu,

3/ możliwość bezpośredniego przeniesienia do nowego systemu tej części rozwiązań systemowych, która nie wymaga zmian i uzupełnień,

4/ wynikające z poprzednich zalet oszczędności w zakresie czasu i kosztów realizacji systemu powielarnego oraz znaczne prawdopodobieństwo realizacji postawionego celu.

Za podstawowe wady tej strategii można uznać:

1/ samoistne ograniczenie zakresu elementów zmiennych systemu powielarnego i ich wariantowania (ewidentne szczególnie wtedy, gdy wcześniejsze wersje systemu były realizowane w obiektach w miarę jednorodnych z punktu widzenia danej aplikacji),

2/ większość modyfikacji dotyczy rozwiązań "na dzisiaj"; w niewielkim zakresie uwzględniając rozwiązania prorozwojowe lub standaryzację w ramach systemu lub międzysystemową - zazwyczaj bardzo trudno (z przyczyn subiektywnych i obiektywnych) zmieniać generalnie coś, co funkcjonuje aktualnie poprawnie.

Do zalet drugiej spośród wskazanych strategii należy zaliczyć:

1/ możliwość opracowania systemu o znacznie wyższych walorach użytkowych i eksploatacyjnych,

2/ możliwość opracowania systemu o znacznie zwiększonym zasięgu powtarzalności,

3/ możliwość tworzenia systemów o rozwiązaniach proroswojowych, znacznie łatwiej poddających się następnym modyfikacjom i rozbudowom.

Główne wady strategii prognostycznej to:

1/ znacznie zwiększone koszty i znacznie dłuższy czas opracowania systemu,

2/ problemy metodyczne wynikające w trakcie prac analityczno-projektowych,

3/ ograniczone prawdopodobieństwo pełnego osiągnięcia zamierzonego celu.

4/ znacznie zwiększona dyscyplina prac nad systemem.

Analiza zalet i wad poszczególnych strategii tworzenia systemów powtarzalnych elastycznych wykazuje praktyczną wyższość pierwszej z wskazanych strategii nad drugą. Istnieją jednak sytuacje, w których zastosowanie strategii diagnostycznej jest utrudnione:

1/ nie dotyczy tych firm, które aktualnie wchodzą na rynek z nadzieją uzyskania znaczących wyników w zakresie dystrybucji systemów użytkowych,

2/ dotyczy wyłącznie ewidencyjnych systemów dziedzinowych o najszerszej skali zastosowań (systemy finansowo-księgowe, systemy ewidencji materiałowej, systemy ewidencji środków trwałych, systemy ewidencji kadrowej, systemy naliczania i ewidencji płac, systemy fakturowania sprzedaży, systemy kosztorysowania prac budowlanych), których rynek ulega powoli nasyceniu,

3/ nie dotyczy systemów wielodziedzinowych i zintegrowanych,

4/ nie dotyczy systemów przekraczających poziom ewidencji gospodarczej (systemy planistyczne, systemy informowania kierownictwa, systemy wspomagania decyzji, itp.).

W pracy [ŁUKA89a] wskazano cztery podstawowe kierunki uelastyczniania systemu do obiektu:

1/ realizacja systemu na sprzęcie o zróżnicowanej klasie i konfiguracji,

2/ uwzględnienie specyficznych rozwiązań obiektu,

3/ zróżnicowanie warunków eksploatacji systemu,

4/ wprowadzenie rozwiązań proroswojowych.

Aktualnie dostępne systemy tworzone za pomocą strategii diagnostycznej

mają zazwyczaj wbudowane mechanizmy adaptacyjne związane z pierwszym i trzecim kierunkiem uelastyczniaania systemu. Kierunki drugi i czwarty są w nich potraktowane marginesowo - do systemów wbudowuje zięrozwiązania cząstkowe znacznie ograniczające swobodę użytkowania i zmian w trakcie eksploatacji. Niekiedy dochodzi do sytuacji karykaturalnych - miejsce mechanizmów adaptacyjnych zajmują rozwiązania zwane żargonowo wodotryskami(rozwiązania o marginesowej użyteczności przemawiające do nabywców szatą graficzną i niestandardowymi pomysłami). Nie zawsze jest to skutkiem zamierzonym - strategia diagnostyczna dopuszcza przeniesienie do systemu niedoważonych bądź nie do końca przemyślanych rozwiązan sugerowanych przez użytkowników.

W opisanej wyżej sytuacji Autor opowiada się za strategią drugą lub mieszaną (z przewagą prognozy w trakcie definiowania zakresu funkcjonalnego systemu i zbioru mechanizmów adaptacyjnych), uzależniając słuszność takiego podejścia od spełnienia podstawowego warunku - opracowania odpowiedniej metodyki tworzenia systemów powielarnych elastycznych. Zwróćmy bowiem uwagę na fakt, że zalety drugiej spośród omawianych strategii mają charakter warunkowy - dają pewne możliwości ale nie gwarantują ich osiągnięcia.

Jako punkt odniesienia dla metodyki działania w trakcie tworzenia systemów powielarnych elastycznych proponuje się klasyczną metodykę diagnostyczną tworzenia systemów indywidualnych (szeroko znaną i stosowaną). W takim przypadku dwa zagadnienia wysuwają się na pierwszy plan:

1/ znacznie większa złożoność zadania projektowego,

2/ zwiększenie liczby, znaczenia i rangi rozwiązań o charakterze modelowym (uogólniających rozwiązania ściśle dostosowane do wymagań poszczególnych obiektów gospodarczych).

Podzielmy proces opracowania systemu na pięć etapów;

1/ identyfikacja i analiza systemu informacyjnego,

2/ koncepcja systemu,

3/ projektowanie systemu,

4/ oprogramowanie systemu,

5/ wdrażanie systemu w obiekcie gospodarczym.

Większość wyróżnionych etapów ma w trakcie tworzenia systemu

powielarnego zadania dodatkowe. Podstawowa różnica polega na konieczności identyfikacji różnorodności rozwiązań w systemach informacyjnych obiektów, które powinny być objęte tworzonym systemem i na opracowaniu mechanizmów adaptacyjnych neutralizujących te rozbieżności.

Uzupełniające zadania etapu identyfikacji i analizy są następujące:

- ustalenie podobieństw i rozbieżności systemów informacyjnych badanych obiektów,
- analiza sprawności rozwiązań stosowanych w ramach poszczególnych obiektów,
- opracowanie modelu systemu powielarnego.

Do podstawowych zadań etapu koncepcji (obok tych klasycznych) należy zaliczyć:

1/ ostateczne określenie zakresu i zasad powielarności opracowywanego systemu,

2/ określenie poziomu standaryzacji systemu i ogólnych reguł jej realizacji,

3/ opracowanie modelu instalacji/inicjacji systemu,

4/ ustalenie rozwiązań konstrukcyjnych przyjętych w systemie.

Uzupełniające zadania etapu projektowania to:

- szczegółowy projekt procesu instalacji/inicjacji systemu,
- szczegółowy projekt mechanizmów uelastyczniających,
- standaryzacja rozwiązań w skali systemu i (ewentualnie) międzysystemowej.

Z powyższego opisu wynika, że główne różnice o charakterze metodycznym dotyczą fazy analityczno-projektowej procesu tworzenia systemu. Ogólnie przyjęta reguła - im wcześniej (w trakcie prac nad systemem) popełniono błąd, tym większy obszar jego oddziaływania i tym trudniejsza jego neutralizacja - sugeruje szczególnie precyzyjne i ostrożne prowadzenie prac identyfikacyjno-analitycznych.

Podstawowym problemem analitycznym jest prowadzenie prac równolegle w kilku obiektach gospodarczych. Chodzi o wykrycie i zbadanie istniejących rozbieżności w celu późniejszego ich zneutralizowania przy pomocy mechanizmów adaptacyjnych. Właściwe wykonanie tego zadania wymaga następujących działań:

1/ wytypowania reprezentatywnego podzbioru obiektów poddanych analizie,
2/ organizacji pracy zespołu analityków,
3/ standaryzacji sposobu i zakresu opisu obiektów.

Przeprowadzenie analizy poprzedza najczęściej przyjęcie mniej lub bardziej sformalizowanych założeń odnośnie zasięgu powielarności tworzonego systemu. Założenia te dotyczą zazwyczaj typów obiektów (przedsiębiorstwa, jednostki budżetowe, spółki, spółdzielnie, warsztaty rzemieślnicze, itp), skali obiektów (duże, średnie, małe), rodzajów obiektów (przemysłowe, handlowe, rolnicze, budownictwo, itp). Często bierze się również pod uwagę ograniczenia o charakterze merytorycznym, ściśle związane z wstępnie określonym zakresem funkcjonalnym systemu. Założenia takie wyróżniają grupę obiektów stanowiących obszar analizy. Ze względów czasowych i kosztowych obszar ten powinien być następnie ograniczony do kilku (ca 3-5) obiektów o możliwie zróżnicowanych warunkach i zasadach działalności. Nie jest wykluczone, że wybór obiektów powinien być poprzedzony pracami studialnymi określającymi ich reprezentatywność.

Pozostałe dwa zadania wstępne - organizacja prac i standaryzacja opisu - powinny być realizowane łącznie. Ich celem jest zapewnienie porównywalności wyników uzyskanych w różnych obiektach. Zazwyczaj pracochłonność prac identyfikacyjnych i krótki termin ich realizacji wyklucza pracę indywidualną - należy powołać zespół analityków. Podział prac między poszczególnych analityków powinien mieć charakter przedmiotowy (1 analityk - 1 obszar merytoryczny w ramach wszystkich badanych obiektów). Zapewni to jednorodność opisu ogółu obiektów. Przed przystąpieniem do prac należy ustalić wyczerpujące zasady prowadzenia identyfikacji (przede wszystkim ujednolicić metody i techniki analityczne oraz techniki prezentacji wyników badań).

Właściwa identyfikacja i analiza systemów informacyjnych wybranych obiektów powinna być realizowana według sformalizowanych i praktycznie zweryfikowanych zasad. Proponuje się zastosowanie do tego celu metod zstępującej (top-down) analizy strukturalnej. Jako punkt wyjścia wzięto pod uwagę metodykę prac analitycznych rozpracowaną szczegółowo i zweryfikowaną praktycznie przez zespół z Akademii Ekonomicznej we Wrocławiu [DOMI84]. Zakłada ona badanie obiektu poprzez równoległą

analizę czterech podstawowych struktur:
1/ funkcjonalnej,
2/ informacyjnej,
3/ przestrzennej i
4/ techniczno-technologicznej.

W ramach analizy struktury funkcjonalnej badaniu podlegają funkcje i zadania wyróżnione merytorycznie, przedmiotowo i czasowo (aspekt statyczny) oraz procesy, procedury i operacje związane z ich realizacją (aspekt dynamiczny). Przedmiotem badań w ramach struktury informacyjnej są kompleksy danych (zbiory danych, grupy danych, pojedyncze dane) przetwarzane w ramach obiektu. Struktura przestrzenna lokalizuje uczestników procesu przetwarzania, opisując odległości między nimi i metody komunikowania się. Ostatnia spośród wyodrębnionych struktur charakteryzuje infrastrukturę techniczną i warunki przetwarzania danych w obiekcie.

Należy zwrócić uwagę na fakt, że prowadzona analiza musi mieć charakter systemowy. Z jednej strony chodzi o rozpatrywanie uwzględniające powiązania przedmiotu badań z otoczeniem, z drugiej o wykrycie wzajemnych powiązań elementów poszczególnych struktur.

Badanie dwóch pierwszych struktur (podstawowych z punktu widzenia rozpoznania systemu informacyjnego) - jest wielostopniowe. Pozwoli to - w przypadkach stwierdzenia zasadniczych rozbieżności między obiektami, wykluczającymi objęcie ich jednym systemem powielarnym - ograniczyć zakres szczegółowych prac identyfikacyjnych.

Postuluje się następującą szczegółową procedurę prac analitycznych.

1/ wstępna identyfikacja struktury funkcjonalnej w wybranych obiektach (badanie funkcji i podstawowych zadań),

2/ analiza porównawcza, pozwalająca:
- określić istniejące rozbieżności,
- wstępnie oszacować poziom złożoności systemu powielarnego, który realizowałby wszystkie zidentyfikowane funkcje i zadania,
- wyeliminować te zadania (i ewentualnie funkcje), które występują sporadycznie i mają ograniczone uzasadnienie merytoryczne,
- sprecyzować standard dalszego, bardziej szczegółowego opisu obiektów (terminologia, techniki precyzyjnie oddające złożoność strukturalną);

3/ bazowa identyfikacja struktury funkcjonalnej (ogół zadań, procesy je realizujące, uwarunkowania wykonania poszczególnych zadań i wzajemne ich powiązania)i wstępna identyfikacja struktury informacyjnej (zbiory danych - kartoteki, raporty - i ich powiązania z realizowanymi procesami);

4/ analiza porównawcza, umożliwiająca:

- określenie spójności funkcjonalno - informacyjnej obszaru zakreślonego dla systemu powielarnego,
- wypracowanie standardu struktury funkcjonalnej systemu powielarnego poprzez dalszą eliminację zadań unikalnych i merytorycznie wątpliwych,
- identyfikację zadań, realizowanych odmiennymi metodami lub technikami i przy unikalnych ograniczeniach;
- wyznaczenie podstawowych struktur danych systemu powielarnego;
- wskazanie powiązań międzysystemowych (dla systemów wielodziedzinowych i zintegrowanych);

5/ bazowa identyfikacja struktury informacyjnej obiektów (dokumenty, struktury zbiorów, podstawowe zasady i warunki ich przetwarzania), identyfikacja struktury przestrzennej i techniczno-technologicznej (w tym ograniczenia czasowe i realizacyjne związane z przetwarzaniem danych);

6/ analiza porównawcza, pozwalająca:

- wariantować warunki eksploatacji (cykle obliczeniowe, ograniczenia czasowe, itp.) oraz generalne rozwiązania technologiczne systemu powielarnego (klasa sprzętu, sieci komputerowe, układy wielodostępne),
- sprecyzować struktury danych systemu powielarnego,
- zbudować w miarę wyczerpujący i wieloaspektowy ogólny model systemu, poddany konsultacji i weryfikacji w poszczególnych obiektach;

7/ szczegółowa identyfikacja struktury funkcjonalnej i informacyjnej wybranych obiektów (badanie typu wejście-algorytm-wyjście ze wskazaniem szczegółowych ograniczeń i wymagań, dotyczące ogółu zadań i operacji włączonych do systemu powielarnego);

8/ analiza porównawcza, zapewniająca:

- wykrycie algorytmicznych różnic w sposobie realizacji poszczególnych zadań i podjęcie decyzji o standaryzacji bądź wielostyczniemiu rozwiązań systemu powielarnego.

- szczegółową ocenę spójności i kompletności rozwiązań stosowanych aktualnie w badanych obiektach,
- uszczegółowienie modelu systemu powielarnego, który należy ponownie poddać weryfikacji.

Przedstawiona wyżej procedura może mieć charakter iteracyjny. Potencjalnymi źródłami takiego zagrożenia są przede wszystkim:

1/ niski poziom merytorycznego i metodycznego przygotowania analityków,
2/ nieprawidłowa organizacja prac zespołu analitycznego,
3/ stosowanie odmiennych metod i technik analizy przez poszczególnych analityków (szczególnie odradza się technikę wywiadów prowadzonych ad hoc),
4/ niski poziom standaryzacji opisu poszczególnych struktur systemu informacyjnego obiektów,
5/ pobieżne i rutynowe prowadzenie analiz porównawczych,
6/ budowanie modeli systemu powielarnego zubożonych semantycznie.

Jeśli do tego dodać konieczność bieżącego i selektywnego operowania dużą liczbą niesformatowanych, niesformalizowanych informacji identyfikacyjnych, to przeprowadzenie prac analitycznych w sposób skuteczny staje się problematyczne. Brakuje bowiem narzędzi usprawniających i dyscyplinujących taką działalność. Rozwiązaniem, co najmniej częściowo neutralizującym ten problem powinien stać się komputerowy Słownik/SkorowidzDanych (SSD). Koncepcje zastosowania narzędzia tej klasy są powszechnie znane i akceptowane [ZACH82][SAKA82]. Tym niemniej w dniu dzisiejszym na rynku krajowym brak implementacji mikrokomputerowych, a implementacje na duże komputery obsługę zadań identyfikacyjno-analitycznych traktują w sposób uproszczony i szablonowy [AMBR84][SAKA82]. Ideę i ogólną strukturę funkcjonalną narzędzia, które w znacznie większym stopniu wspomaga omawiane prace przedstawiono w [ŁUKA88a].

Należy zastrzec się, że dokonana powyżej analiza metodyki prac nad systemem powielarnym zmierza do wypracowania szczegółowego modelu tego procesu. W trakcie opisu strategii tworzenia systemu i rozważań dotyczących analizy struktur obiektów pominięto rozwiązania mieszane (np. wyjście od systemu indywidualnego uzupełnione wycinkową identyfikacją opisywanego typu). Możliwości klasyfikacji takich

rozwiązań są ograniczone ze względu na dużą liczbę uwarunkowań praktycznych metodyk tworzenia systemów poziomych elastycznych (poziom merytoryczny personelu, czas, koszty, przyzwyczajenia, sytuacje rynkowe, faktycznie zgłaszane zapotrzebowanie na systemy wirtualne, itp.). Należy jedynie wyrazić nadzieję, że im bardziej zwinne i elastyczne będą tworzone systemy, tym bardziej racjonalne, spójne i skuteczne będą metody ich tworzenia.

LITERATURA

[AMBR84] Ambrosetti R., Ciriani T. A., Pennacchi R., An application analyzer, IBM Systems Journal 1984 nr 4

[DOMI84] Domiński W., Dyczkowski M., Małachowski A., Nowicki A., Zastosowanie analizy strukturalnej w modernizacji i rozwoju systemu informacyjnego przedsiębiorstwa, Materiały konferencyjne "Informatyka w zarządzaniu przedsiębiorstwem", INFOGRYF'84, TNOiK i Politechnika Szczecińska 1984

[ŁUKA88a] B.Łukasik-Makowska, M.Skwarnik, Metodologiczne aspekty tworzenia powielarnych informatycznych systemów zarządzania, INFOGRYF'88 TNOiK i Uniwersytet Szczeciński Kołobrzeg 1988

[ŁUKA89a] B.Łukasik-Makowska, Standaryzacja systemów powielarnych (w tym tomie)

[SAKA82] Sakamoto J. G., Ball F.W., Supporting Business System Planning studies with the DB/DC Dictionary, IBM Systems Journal 1982 nr 1

[ZACH82] Zachman J. A., Business Systems Planning and Business Information Control Study : A Comparison, IBM Systems Journal 1982 nr 1

Ireneusz Szydłowski

Instytut Cybernetyki Ekonomicznej i Informatyki

Uniwersytetu Szczecińskiego

METODA WSPOMAGANEGO PROJEKTOWANIA SYSTEMÓW INFORMATYCZNYCH

1. Potrzeba wspomagania procesu projektowania

W miarę rozwoju zastosowań informatyki obserwuje się wzrost wielkości i złożoności zadań projektowych, które wymagają wspomagania. Wśród szczegółowych przesłanek komputerowego wspomagania projektowania wyróżnić można:

- dążenie do skrócenia czasu projektowania,
- osiąganie lepszych rozwiązań, poszukiwanie i wybieranie rozwiązań suboptymalnych,
- usprawnienie ilościowe i jakościowe procesu projektowania,
- poprawność i spójność projektowanych elementów systemu informatycznego,
- pomoc w formułowaniu problemów projektowych,
- możliwość efektywnego zarządzania realizacją prac projektowych,

- możliwość bezkonfliktowego współdziałania wielu osób w procesie projektowania,
- możliwość analizy już poniesionych i przewidywanych kosztów projektowania,
- automatyzację procedur kontrolnych procesu projektowania,
- automatyczną emisję dokumentacji projektowej [por. CHOJ'84 s.19],
- standaryzację procesu projektowania.

Duża różnorodność i liczba podejść metodycznych do budowy systemów informatycznych, które charakteryzują się wysokim stopniem formalizacji, wpływają na opracowywanie wielu propozycji użycia komputera do wspomagania. Ponieważ nie ma metodyki uniwersalnej, dającej się zastosować do wszystkich typów tworzonych systemów informatycznych (każdy typ systemu informatycznego wymaga specyficznego podejścia - każda metodyka ma swoje wady i zalety), w efekcie daje się zauważyć tworzenie coraz to nowych podejść metodycznych, które implikują powstawanie kolejnych systemów wspomagających.

Aby taki system wspomagający powstał niezbędne jest posiadanie:
- sformalizowanej metody projektownia systemu,
- języka opisu służącego do przedstawiania projektowanego systemu,
- modelu konceptualnego systemu - jako podstawy do szczegółowego projektowania systemu [por. DLEJ'85 s. 92].

Większość dostępnych obecnie narzędzi wspomaga tylko pewien fragment pełnego cyklu budowy systemu informatycznego, np. tworzenie schematów przetwarzania czy też generowanie oprogramowania. Pewna grupa najnowszych narzędzi wspomagających wykorzystuje koncepcje Information Engineering (inżynierii informacyjnej) i Computer Aided Systems Engineering (CASE - komputerowo wspomagana inżynieria systemów). Bazuje się na idei, że systemy informatyczne mogą być budowane w sposób mniej lub bardziej

standardowy, podobnie jak projekty domów z wykorzystaniem systemów CAD [por. BOST'88]. Mimo że projektowanie z natury swej jest działalnością chaotyczną, to wydaje się, że możliwe jest zbudowanie systemu komputerowego wspomagającego pełen cykl budowy systemu informatycznego. Wymaga to jednak zastosowania odpowiedniej metodologii i właściwe jej odwzorowanie.

2. Metoda wspomaganego projektowania systemu informatycznego

Wprawdzie system komputerowy nie jest w stanie całkowicie wyeliminować projektanta, to może jednak uwolnić go od wielu uciążliwych i rutynowych czynności. W pierwszym rzędzie komputerowy system wspomagający pełni funkcję ewidencyjną. Duża liczba różnorodnych danych występujących podczas tworzenia systemu informatycznego wymagających przede wszystkim uchwycenia (również zmian w czasie), a także uporządkowania, jest już wystarczającym powodem zastosowania komputera w projektowaniu. Jest to zaledwie ułamek możliwości komputerowo wspomaganego projektowania. Dostarczenie odpowiedniego zestawu danych początkowych umożliwia poprzez przetworzenie tych danych uzyskać szybko produkty dające się bezpośrednio wykorzystać przy budowie oprogramowania - różnego rodzaju schematy będące podstawą 'odręcznie' tworzonych programów lub dane dla generatorów programów. Niezależnie od tego przebieg procesu projektowania systemu informatycznego jest na bieżąco dokumentowany, co umożliwia uzyskiwanie różnego rodzaju dokumentacji - zawsze aktualnej.

W prezentowanej metodzie zastosowano podejście zorientowane obiektowo, przyjmując, że rzeczywistość zbudowana jest z obiektów posiadających powiązania między sobą. Obiekty i powiązania (relacje) są charakteryzowane przez właściwości (cechy) i są

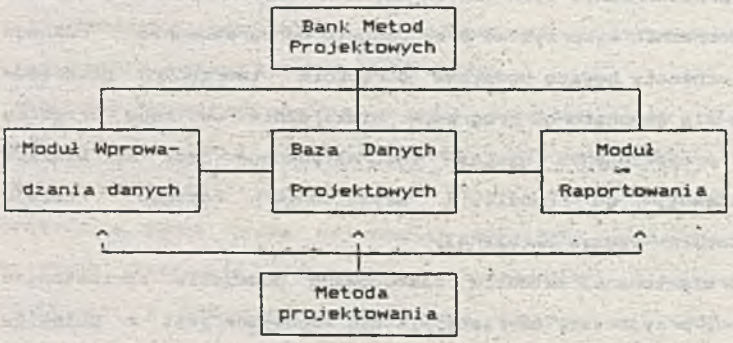
zgrupowane w klasy oraz typy. Zachowanie się (dynamika) systemu (rzeczywistości) opisywane jest przez 3 kategorie zjawisk : obiekt, operacja i zdarzenie mających 3 kategorie powiązań : modyfikuje, stwierdza, wyzwala. Bliższe określenie tych pojęć dla naszych celów można znaleźć w pracy [OLEJ'89]. Przedstawione podejście jest podstawą budowy systemu wspomagającego ukierunkowanego na odwzorowanie przedstawionego modelu danych, z własnymi mechanizmami opisu i zaszytymi procedurami postępowania oraz kontroli.

3. System wspomagający projektowanie

3.1. Struktura systemu wspomagającego projektowanie

Prezentowany system wspomagający projektowanie (rys. 1.) składa się z następujących 4 głównych modułów :

- baza danych projektowych,
- moduł wprowadzania danych,
- moduł raportowania,
- bank metod projektowych.



Rys. 1. Struktura systemu wspomagającego projektowanie

Przyjęta metoda projektowania ma zasadniczy wpływ na strukturę poszczególnych modułów systemu wspomagającego projektowanie. Przy opisie systemu rzeczywistego oraz projektowanego korzysta się przede wszystkim z pojęcia obiektu, relacji technologicznych oraz mechanizmów zachowania, hierarchizacji, przypisywania. Dla ostatniego poziomu hierarchii obiektów (dane elementarne) stosuje się ponadto mechanizmy oznaczania i wyróżniania. Pojęcie operacji i pojęcie relacji funkcjonalnej są podstawowymi narzędziami opisu działania systemu. Dodatkowe pojęcia zdarzenia, predykatu, warunku i czynnika umożliwiają wpełni odzwierciedlić dynamikę systemu.

3.2. Baza danych projektowych

Baza danych projektowych stanowi źródło danych dla modułu raportowania. Jej aktualizacja odbywa się z wykorzystaniem modułu wprowadzania danych wspieranym przez bank metod projektowych. Baza danych projektowych zawiera system zbiorów pozwalających zapamiętać dane o systemie rzeczywistym i projektowanym wyrażone za pomocą pojęć przedstawionych powyżej. W szczególności są to następujące zbiory:

- katalog systemów,
- katalog kategorii obiektów:
  - standardowych,
  - kreowanych przez projektanta,
- katalog obiektów,
- katalog cech,
- katalog relacji technologicznych,
- katalog powiązań hierarchicznych,
- katalog cech obiektów,
- katalog przypisań,

- katalog rekordów,
- katalog danych elementarnych,
- katalog relecji funkcjonalnych.
- katalog operacji,
- katalog zdarzeń,
- katalog predykatów,
- katalog warunków,
- katalog standardów.

Odpowiednie rozbudowanie katalogu standardów może uczynić z niego bazę metodyczną i normatywną projektowania.

Ponadto w systemie wspomagającym występuje wiele zbiorów pomocniczych, które ułatwiają przetwarzanie i przyspieszają wyszukiwanie informacji.

3.3. Moduł wprowadzania danych

Moduł wprowadzania danych jest modułem aktualizacji bazy danych projektowych. Dane mogą być wprowadzane w dwojaki sposób:

- poszczególne katalogi są aktualizowane indywidualnie,
- dane wprowadzane są z wykorzystaniem standardowych formatów wejściowych obsługujących jednocześnie wiele katalogów.

Istotą jednego jak i drugiego sposobu jest wprowadzanie tylko poprawnych danych (wewnętrznie spójnych) tzn. mechanizmy kontrolne zapewniają unikalność różnego rodzaju identyfikatorów, prawidłową hierarchizację obiektów. Obecnie projektant ma do dyspozycji kilka standardowych formatów. Założeniem twórców systemu jest dążenie do umożliwienia definiowania formatów wejściowych przez projektanta (użytkownika). Rys. 2. prezentuje przykładowy, standardowy format wejściowy.

```
      SYSTEM: SD        NAZWA: System demonstracyjny

    WYJŚCIE: LISTAPR   NAZWA: Lista pracowników

        OPIS: Lista pracowników wyprowadzana jest w zależności
              od potrzeb na ekran lub drukarkę. Linia dla każ-
              dego pracownika ma stałą postać.

SKLADA SIĘ Z: NAGL I PR NAZWA: Nagłówek listy pracowników
              LINIALP  NAZWA: Linia listy pracowników
              STPLIPR  NAZWA: Stopka listy pracowników

UZYWA ZBIORU: ZBPR     NAZWA: Zbiór pracowników

REALIZUJE
     PROCES: WYDRLIPR NAZWA: Wydruk listy pracowników
```

Rys. 2. Standardowy format wejściowy.

Na etapie wprowadzania danych projektant ma możliwość kreowania własnych kategorii obiektów, stosowania dla nich dostępnych mechanizmów opisu. Pozwala to na pełniejsze odwzorowanie rzeczywistości w sytuacjach, gdy standardowe kategorie obiektów są niewystarczające.

3.4. Bank metod projektowych

Bank metod projektowych jest modułem wspomagającym opisywanie systemu rzeczywistego i projektowanego (moduł wprowadzania danych), a także stanowi wsparcie dla modułu raportowania, pozwalając na pełniejszą prezentację systemu projektowanego. M.in. bank metod projektowych zawiera procedury automatycznej klasyfikacji obiektów wspomagające tworzenie (opis) struktur funkcjonalnej i organizacyjnej systemu, procedury wspomagające tworzenie schematów logicznych baz danych i schematów fizycznej lokalizacji - algorytmy rozmieszczania.

3.5. Moduł raportowania

Wydaje się, że jakość modułu raportowania w dużej mierze określa jakość całego systemu wspomagającego. Wyniki tego modułu to wyniki zastosowania całego systemu wspomagającego projektowanie. Moduł ten pozwala prezentować wprowadzone dane w różnych układach, generować fragmenty dokumentacji, wyszukiwać dane o projektowanym systemie. Przede wszystkim moduł raportowania pozwala prezentować dane zgodnie z formatami wejściowymi — standardowymi i określanymi przez projektanta. Ponadto na poziomu tego modułu możliwe jest uruchamianie procedur z banku metod projektowych, które umożliwiają uzyskanie obrazu systemu innego niż wynikałoby to bezpośrednio z wprowadzonych danych. Moduł ten jest ciągle rozwijany, a pośrednio można do niego zaliczyć również inne programy, nie będące częścią prezentowanego systemu wspomagającego, które wykorzystują przedstawioną wcześniej bazę danych projektowych.

3.6. Mechanizmy kontrolne realizacji projektu

Prezentowany system wspomagający umożliwia dwojakiego rodzaju kontrolę realizacji projektu. Pierwszy sposób polega na wykorzystaniu kategorii obiektów projektant i mechanizmu przypisywania. Obiekty kategorii projektant są odpowiedzialne za realizację opisu np. wejść, wyjść lub zbiorów → taką formułę zapewnia mechanizm przypisywania. Pozwala nam to stwierdzić, co zostało zrobione oraz co jest do zrobienia.

Mechanizm kontrolny drugiego rodzaju jest wbudowany - wykorzystywany w module wprowadzania danych. Sposób postępowania - kolejność wprowadzania danych jest dowolna — dany obiekt może być

wprowadzony 'po' pewnych obiektach, które jeszcze nie są wprowadzone - jednak pewne czynności logicznie poprzedzają inne. Akceptowane są 4 zasadnicze podejścia: z góry do dołu, z dołu do góry, od środka oraz do środka. Projektant ma możliwość zdefiniowania standardowego - własnego toku postępowania (sieci działań), którego realizacja jest kontrolowana przez system, tzn. pamiętane jest wykonanie poszczególnych kroków i ustalana droga dalszego postępowania z danego punktu sieci działań. Mechanizm ten jest swoistym dedykowanym pomocnikiem projektanta.

3.7. Wykorzystanie systemu wspomagającego projektowanie

Założeniem twórców jest, aby z prezentowanego systemu mógł korzystać każdy, również użytkownik nieprzygotowany (projektowanie partycypacyjne). Wymagane jest tylko zapoznanie się z instrukcją obsługi oraz podstawami metody, które zresztą dostępne są z poziomu programu. Przystępując do realizacji projektu, projektant musi podać jego unikalny identyfikator (katalog systemów), nazwę oraz ew. opis. Wprowadzanie dalszych danych jest wspomagane przez standardowe formaty wejścia oraz mechanizmy kontrolne systemu. W miarę rozpoznawania systemu wspomagającego możliwe jest korzystanie z jego kolejnych elementów, wraz z kreacją własnych kategorii obiektów, formatów wejścia i kontrolowanej sieci działań. Projektant na bieżąco ma do dostęp do instrukcji dotyczącej aktualnie wykonywanych czynności. Wprowadzenie odpowiedniego zestawu danych wejściowych warunkuje wykorzystanie pełnych możliwości modułu raportowania.

4. Zakończenie

Problem wspomaganego projektowania jest obecnie bardzo popularny, szczególnie ze względu na dynamiczny rozwój narzędzi i pakietów możliwych do wykorzystania na sprzęcie mikrokomputerowym. Wśród narzędzi i pakietów wspomagających zauważyć można również takie, które wprawdzie konsekwentnie realizują założenia metodologiczne, lecz w praktyce są zupełnie nieprzydatne. Prezentowana metoda projektowania wraz z systemem wspomagającym ukierunkowana jest na praktyczne zastosowanie w całym procesie tworzenia systemu informatycznego z myślą wykorzystania jej również przez niedoświadczonych projektantów.

LITERATURA

CHOJ'84 R. Chojnowska, Z. Drążek, A. Nowakowski: Projektowanie i dokumentowanie projektu systemu informatycznego, TNOiK Oddział Szczecin, Szczecin 1984

OLEJ'85 W. Olejniczak: Projektowanie systemów informacji ekonomicznej, Wydawnictwo Uczelniane Politechniki Szczecińskiej, Szczecin 1985

OLEJ'89 W. Olejniczak, J. Szydłowski: Globalna metoda projektowania systemów informatycznych, Druga Wiosenna Szkoła PTI, Świnoujście 1989

BOST'88 Information System Development Supporting Methodologies With Computeriezed Tools w pracy Current Trends In Information Systems Development Methodologies, Seminarium Polsko - Skandynawskie, Paraszyno, 27-30 czerwca 1988

Zdzisław Szyjewski
Uniwersytet Szczeciński

Metoda Jacksona planowania programów

1. Wprowadzenie

Proces tworzenia programu rozpoczyna się od specyfikacji wymagań stawianych przed programem. Specyfikacja ta dotyczy realizowanych przez program funkcji oraz danych używanych do rozwiązania problemu. Specyfikacja najczęściej wykonana jest w notacji i terminologii użytkownika. Zadaniem programisty jest zamiana tej specyfikacji na wynikowy kod programu realizowanego przez komputer.

Jak to należy zrobić ?

Jednym za sposobów, który zdobył sobie popularność jest podejście strukturalne. Podejście strukturalne do procesu programowania powoduje przeniesienie akcentów z fazy pisania kodu źródłowego programu na fazę projektowania oprogramowania. Bardzo ważnym elementem tej fazy prac jest planowanie programu. Istotą prac planowania jest odpowiednie zorganizowanie procesu programowania tak, aby mógł być opracowany szybko, sprawnie i niezawodnie dając w wyniku program poprawny, czytelny i łatwy do modyfikacji. W toku prac planowania następuje szczegółowe wyspecyfikowanie struktury rozwiązywanego problemu, zapisanie go w odpowiednio sformalizowanej postaci, która umożliwi łatwą weryfikację zastosowanych rozwiązań oraz przygotuje do kodowania.

Można zaobserwować kilka różnych sposobów podejscia do planowania programów. Duże walory praktyczne w przetwarzaniu danych posiada metoda Jacksona, która za podstawę planowania przyjmuje strukturę danych, które program ma przetwarzać.

2. Kroki metody Jacksona

2.1. Założenia ogólne metody

M. Jackson proponuje metodę planowania programu, która umożliwia pisanie programów "dobrych" w rozumieniu wyżej określonych kryterium. Wychodząc z założenia, że o strukturze

programu decyduje struktura problemu do rozwiązania. Jackson metodą kolejnych uściśleń dochodzi do ostatecznego rozwiązania. Dzieląc problem na podproblemy, na podstawie analizy struktury danych, dochodzi do składników elementarnych odpowiadających modułom programowym. Na wszystkich poziomach dekompozycji należy dążyć do stosowania trzech podstawowych struktur: sekwencji, wyboru i powtarzania.

Struktura danych jest formułowana na podstawie danych problemowych i poprzez poszukiwanie związków pomiędzy danymi a realizowanym procesem, definiowana jest struktura programu. Po zdefiniowaniu listy operacji koniecznych do realizacji programu następuje allokacja tych operacji na strukturze programu. W rezultacie zamiany struktury programu na schemat logiczny opisujący w pseudokodzie problem, otrzymujemy postać łatwą do zamiany na kod programu w języku programowania.

Metoda Jacksona realizowana jest w pięciu następujących krokach :

1. Zdefiniowanie struktury danych wejściowych i wyjściowych.
2. Budowa diagramu strukturalnego programu na podstawie analizy związków między strukturą danych a realizowanym procesem.
3. Specyfikacja listy operacji do wykonania.
4. Allokacja operacji na diagramie strukturalnym programu.
5. Zamiana diagramu strukturalnego na schemat logiczny.

W wyniku ostatniego kroku metody otrzymujemy program w pseudokodzie, co pozwala na łatwą zamianę na dowolny język programowania.

2.2. Definicja struktury danych

Wszystkie dane, przetwarzane przez program będą reprezentować hierarchiczną strukturę danych rzeczywistych. W związku z tym, że struktura programu będzie budowana z wykorzystaniem podstawowych struktur programowych, struktury danych muszą być tworzone tylko z wykorzystaniem zależności typu:

- sekwencja,
- wybór,
- powtarzanie.

Przeanalizujemy notację i zasady każdej z wymienionych struktur.

Sekwencja

Sekwencja opisuje dane rzeczywiste, które występują jedna po drugiej w określonym porządku. Należy przedstawić wszystkie składniki sekwencji oraz ich ewentualną strukturę dalszego podziału.

Sekwencja jest reprezentowana przez następujący diagram:

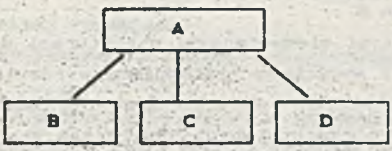
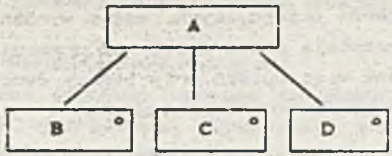


Diagram ten przedstawia dane rzeczywiste A, które zawierają dane B, C i D w takiej kolejności występowania.

Wybór

Wybór opisuje dane rzeczywiste, w których jedna część jest wybierana z zestawu określonych możliwości. Jedna i tylko jedna część musi być wybrana ze zdefiniowanych niezależnych składników, na podstawie opisanego warunku.

Na diagramie wybór jest reprezentowany przez kółko w prawym górnym rogu każdej niezależnej części składającej się na strukturę wyboru.

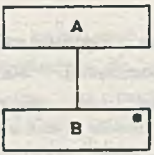


Dane rzeczywiste A zawierają dane B lub C lub D, jedną spodanych możliwości wystąpienia w każdym konkretnym przypadku.

Powtarzanie

Powtarzanie opisuje dane rzeczywiste zawierające tylko jeden składnik(jednorodny), powielając całkowitą ilość razy (włączając w to zero).

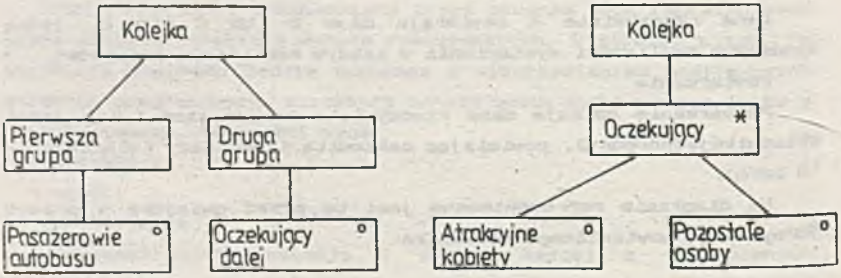
Na diagramie reprezentowane jest to przez gwiazdkę w prawym górnym rogu powtarzanego składnika.



Dane rzeczywiste A składają się z zera lub większej liczby wystąpień danych B.

Dysponując tymi elementarnymi strukturami danych należy opisać dane wejściowe i wyjściowe występujące w rozwiązywanym problemie. Struktura opisywanych danych musi przedstawiać związki i zależności danych rzeczywistych wynikające z wymagań użytkownika, to znaczy ma to być spojrzenie na dane z punktu widzenia rozwiązywanego problemu. Związki między danymi mogą być pokazane z różnych punktów widzenia, ale diagram strukturalny opisujący dane dla konkretnego programu musi eksponować te związki, które są istotne z punktu widzenia specyfikacji wymagań rozwiązywanego problemu.

Dla przykładu rozważmy kolejkę pasażerów oczekujących na autobus. Z punktu widzenia konduktora autobusu kolejka ta składa się z dwóch części. Jedna to grupa pasażerów, którzy będą mieć miejsce w tym autobusie i druga grupa, tych którzy będą musieli poczekać na następny autobus. Z punktu widzenia kierowcy autobusu, w kolejce pasażerów oczekujących, można wyróżnić atrakcyjne młode kobiety oraz pozostała grupę pasażerów oczekujących. Te same dane rzeczywiste można więc opisać przy pomocy dwóch różnych diagramów w zależności od punktu widzenia problemu.



Opisana struktura danych musi ilustrować naturę wymagań stawianych w danym problemie. Przykładowo, jeżeli porządek występowania jest najważniejszy w przypadku danych rzeczywistych, to reprezentującą te dane strukturą będzie sekwencja.

Przedstawiając strukturę danych w postaci diagramu należy mieć na uwadze:

- kompletność, co należy rozumieć że:
  . wszystkie poziomy zastosowań danych rzeczywistych są uwzględnione,
  . pierwsze/ostatnie wystąpienie danych w strukturze powtarzania jest przedstawione, jeśli ma ono istotne znaczenie,
  . wszystkie grupy danych są zawarte w diagramie.
- zgodność, co należy rozumieć że:
  . każdy składnik sekwencji zawiera tylko sekwencyjne części,
  . każdy składnik wyboru zawiera tylko możliwe do wybrania części,
  . każdy składnik powtarzania zawiera pojedyncze powtarzane części.
- prawidłowe użycie struktury wyboru, to znaczy,
  . części będące składnikami wyboru są rzeczywiście niezależne i nie mają związków z wyższymi poziomami hierarchi,
  . każdy wybór jest wykonany na odpowiednim poziomie hierarchi.

2.3. Budowa diagramu strukturalnego programu

Prawie każdy program użytkowy operuje na dwóch kategoriach zbiorów: wejściowych i wyjściowych. Każdy zbiór posiada własną strukturę danych i program musi posiadać strukturę, która będzie odzwierciedlała obydwie struktury danych oddzielnie. Zadanie przedstawienia struktury programu jest proste jeżeli poszczególne moduły programu mogą przetwarzać odpowiednie części danych niezależnie. W drugim kroku metody zajmujemy się budową struktury programu szukając związków i zależności pomiędzy strukturami przetwarzanych danych wejściowych w wyjściowe.

Poszukiwane zależności pomiędzy składowymi danych muszą spełniać następujące warunki:

- muszą posiadać taką samą ilość powtórzeń jak dane rzeczywiste,
- musi być możliwe przetwarzanie powtarzanych części w tym samym

czasie.

Jeśli zadanie polega na skopiowaniu zbioru wejściowego na zbiór wyjściowy, to mamy przykład korespondujących danych wejściowych i wyjściowych. Z jednego rekordu wejścia powstaje jeden rekord wyjścia i ponadto przetwarzanie ich może być realizowane w tym samym czasie.

W przypadku zadania sortowania zbioru wejściowego na zbiór wyjściowy, warunek przetwarzania w tym samym czasie nie jest spełniony. Natomiast warunek takiej samej ilości powtórzeń dla zbioru wejściowego i wyjściowego jest zachowany.

Przypadek niekorespondujących danych obrazuje zadanie polegające na wydruku, który w jednym wierszu zawiera zbiorcze dane obliczane na podstawie danych z dwóch różnych typów rekordów zbioru wejściowego, przy czym ilość rekordów każdego typu może być różna. W takim przypadku korespondencji danych będziemy szukać na wyższym poziomie - zbiorów. Z jednego zbioru wejściowego wyprowadzany jest jeden raport, przetwarzanie zbioru wejściowego i wyprowadzanie raportu odbywa się w tym samym czasie. Na niższym poziomie znajdujemy korespondujące dane w postaci jednej linii raportu i grupy rekordów potrzebnych do jej obliczenia.

Brak możliwości znalezienia korespondujących danych będzie się charakteryzował jednym z dwóch typów:

- nie ma korespondencji między danymi oraz brak wymagań co do przetwarzania danych,
- niezgodność struktur, określone są wymagania co do wspólnego przetwarzania danych ale brak korespondencji między ich strukturami uniemożliwia to.

W przypadku braku możliwości prostego przekształcenia struktury danych w strukturę programu należy w sposób indywidualny budować strukturę programu. We wszystkich innych przypadkach, struktura programu przetwarzającego dane musi zawierać elementy korespondujące. Jest to konieczne, ponieważ wykonywane operacje są powiązane z elementami danych i musimy zapewnić żeby wykonywane operacje były allokowane w odpowiednich punktach struktury programu.

Dla programów, które przetwarzają więcej niż jedną strukturę danych, struktura programu jest budowana na podstawie kombinacji

poszczególnych struktur danych. W procesie tym należy wykorzystywać związki i relacje jakie zachodzą pomiędzy elementami danych wejściowych przetwarzanych w dane wyjściowe.

Zasady dla przekształcania struktur danych w jedną strukturę programu są następujące:

a) Tworzone są moduły programowe dla każdych korespondujących struktur danych, utrzymując hierarchiczne relacje między komponentami występującymi w oryginalnych strukturach danych, wykorzystując zasady strukturalnej notacji.

b) Dodaje się jeden moduł do struktury programu, dla każdych nie korespondujących danych rzeczywistych dla wszystkich wejściowych struktur danych. Moduł taki nie jest konieczny jeśli niekorespondujące dane nie wymagają żadnego przetwarzania przez program. Relacje pomiędzy elementami danych muszą być przeniesione do struktury programu.

c) Powtórzyć powyższy krok dla każdej struktury danych wejściowych, zapewniając żeby przetwarzanie w tym samym czasie było wykonalne, czyli dostępne będą niezbędne dane dla otrzymania wyników. W przypadku gdy dostępność danych nie jest gwarantowana, to struktura programu wymaga reorganizacji.

2.4. Specyfikacja listy operacji do wykonania

Mając określoną strukturę programu w następnym kroku metody definiujemy listę operacji, jakie należy wykonać aby otrzymać wynik programu zgodnie ze specyfikacją. Wymienione operacje będą w następnym kroku allokowane na strukturę programu i w końcowej fazie przekształcone w schemat logiczny. Generowanie kodu źródłowego z schematu logicznego, jest czynnością bardzo prostą i może być wykonane automatycznie z wykorzystaniem preprocesora.

Specyfikacja pełnej listy operacji koniecznych do wykonania dla realizacji algorytmu programu jest istotną czynnością. Porządek na liście specyfikowanych operacji nie jest istotny, a jedynie kompletność. W praktyce najlepszą metodą zidentyfikowania wszystkich operacji jest praca od wyjścia programu do danych wejściowych, rozważając w kolejnych fazach następujące sytuacje:

a) operacje terminalne, czyli operacje startu i zatrzymania programu, a w przypadku wykorzystywania podprogramów.

analogiczne operacje dla podprogramów.

b) operacje otwarcia i zamknięcia zbiorów, czyli operacje rozpoczęcia i zakończenia wprowadzania danych do/z programu i otoczenia.

c) operacje wykonania danych wyjściowych, czyli operacje konieczne dla wyprowadzenia wyników z programu do zbiorów wynikowych.

d) operacje obliczania wyników, czyli operacje wykonywane w programie, konieczne dla otrzymania wyników przed ich zapisaniem do zbiorów.

e) operacje wprowadzania, czyli niezbędne operacje wczytania wszystkich zbiorów wejściowych.

f) operacje wewnętrznego operowania zmiennymi, czyli operacje zapamiętywania, odtwarzania zmiennych, inicjowanie wartości początkowych, zwiększanie, zmniejszanie liczników, sum itp.

Warunki testowane w operacjach wyboru i kończących struktury powtarzania, nie są umieszczane na liście. Będą one dodane w trakcie tworzenia schematu logicznego w ostatnim kroku metody.

2.5. Allokacja operacji na diagramie strukturalnym programu

Każda operacja z listy operacji do wykonania, jest umieszczona w jednym lub w kilku elementach programu. Dla każdej operacji należy odpowiedzieć na dwa pytania:

- dla których elementów programu dana operacja musi być wykonana raz i tylko raz ?
- gdzie w sekwencji elementów składających się na program, należy umiejscowić daną operacje ?

Numer danej operacji z listy piszemy w kółku i kreską łączymy to kółko z miejscem, gdzie na strukturze programu znajduje się odpowiedni element programu , w ramach którego dana operacja ma być wykonana. Jeśli diagram strukturalny programu przypomina drzewko, choinkę, to allokacja operacji z listy, przypomina zawieszanie ozdób, w postaci numerków operacji, na tej choince.

Umieszczając operacje na diagramie strukturalnym programu należy zwracać uwagę na oznaczenia sekwencji, powtarzania i wyboru, aby operacji wykonywanych jednorazowo nie allokować w nieodpowiednim miejscu.

Szczególną uwagę należy zwracać przy alokowaniu operacji czytania. Chcąc ustrzec się błędnego przetwarzania zbiorów należy pstrzestrzegać następujących zasad:

- umieścić operacje czytania po operacji otwarcia zbioru,
- umieścić operacje czytania po przetworzeniu każdego rekordu.

W większości języków programowania, warunek końca zbioru jest rozpoznawany podczas wykonywania operacji czytania. Często stosowanym rozwiązaniem jest bezwarunkowe przekazywanie sterowania w określone miejsce programu. Rozwiązanie takie burzy strukturalną konstrukcję programu i nie jest zalecane.

Proponowane są dwie drogi rozwiązania tego problemu:

- przyjęcie wartości maksymalnej klucza rekordu, w przypadku napotkania końca zbioru.

Np. READ ZB AT END MOVE HIGH-VALUE TO KEY.

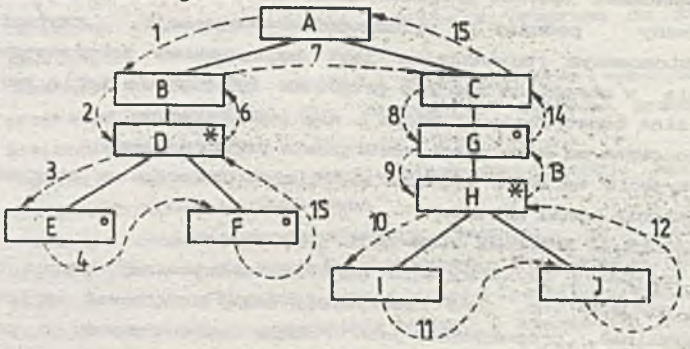
- w przypadku gdy przyjęcie wartości maksymalnej klucza nie jest możliwe (np. dane rzeczywiste mogą przyjmować wartości maksymalne), proponowane jest rozszerzenie rekordu o pole wskaźnikowe, które będzie miało nadane wartość zero dla wszystkich rekordów a wartość jeden dla rekordu końca.

2.6. Zamiana diagramu strukturalnego na schemat logiczny

W wyniku wykonanych dotychczas kroków metody, mamy diagram strukturalny programu i umieszczone na nim w odpowiednim miejscu, operacje, które należy wykonać dla realizacji algorytmu programu. Ostatnim krokiem metody, przed zapisaniem programu w języku programowania, jest zamiana opisanego diagramu na schemat logiczny.

Zasada zamiany diagramu strukturalnego na schemat logiczny jest zaczerpnięta ze sposobu przechodzenia przez wierzchołki(węzły) struktur dendrytowych tzn. top-to-bottom i left-to-right. Diagram strukturalny traktujemy jak strukturę dendrytową, gdzie prostokąty spełniają rolę węzłów. Przy przetwarzaniu danych zapisanych w postaci struktury dendrytowej istotny jest porządek(kolejność) przechodzenia przez węzły takiej struktury. Kolejność przetwarzania węzłów jest dlatego sprawą istotną, ponieważ sama struktura jest nośnikiem informacji, oprócz informacji szczegółowej zapisanej w każdym z węzłów.

Schemat logiczny otrzymujemy z diagramu strukturalnego rozpoczynając analizę diagramu od wierzchołka i zapisując od strony lewej do prawej kolejno napotkane symbole. Graficznie sposób analizy przedstawiają ponumerowane kolejno przerywane linie zaznaczone na diagramie.



Wykorzystując notację przyjętą przez Jacksona i analizując diagram, jak pokazują strzałki, otrzymujemy schemat logiczny odpowiadający przedstawionemu diagramowi strukturalnemu:

```
A seq
   B iter while war-D
     D select war-E
       E
     D or
       F
     D end
   B end
     C select war-G
       G iter while war-H
         H seq
           I
           J
         H end
       G end
     C end
A end
```

Schemat logiczny jest programem napisanym w pseudokodzie z uwzględnieniem trzech podtawowych struktur. Zamiana tak zapisanego programu na wybrany język programowania wyższego rzędu jest już czynnością stosunkowo prostą.

LITERATURA

Jackson M., Principles of Program Design, Academic Press. New York 1975

Szyjewski Z., Programowanie strukturalne w przetwarzaniu danych. PWE. Warszawa 1984

Stanisław Wrycza
Uniwersytet Gdański

NARZĘDZIA KOMPUTEROWO WSPOMAGANEGO TWORZENIA SYSTEMÓW INFORMATYCZNYCH W ŚWIETLE CASExpo EUROPE

1. Wprowadzenie

Do zestawu licznych akronimów, skrótów - kluczy słownika współczesnego informatyka, dynamicznie wchodzi w ostatnich 2-3 latach określenie CASE (computer-aided software engineering) lub (computer - aided systems engineering). Nie wchodząc w szczegółową dyskusję nad możliwymi opcjami tłumaczenia, przyjmujemy, iż bardziej z i s t o t y a mniej z brzmienia poszczególnych słów, oznacza ono komputerowe wspomaganie tworzenia systemów informatycznych (KWTSI).

Sens koncepcji CASE zawiera się we wspomaganiu odpowiednimi narzędziami programowymi zespołu twórców systemu informatycznego (analityków, projektantów, programistów, użytkowników) w całym p r o c e s i e jego tworzenia od analizy wstępnej do wdrożenia. Zatem u podstaw zainteresowania rozwojem tych narzędzi leży potrzeba skrócenia czasu realizacji cyklu życia systemu przy jednoczesnym podniesieniu jakości produktu finalnego - systemu użytkowego.

Były dwa podstawowe ź r ó d ł a , zresztą wzajemnie się stymulujące, inspiracji w rozwoju pakietów CASE. Pierwszym z nich były doświadczenia w użytkowaniu współczesnych m e t o d y k tworzenia systemów informatycznych. Są one w znacznym stopniu sformalizowane, co sprzyja opracowaniu procedur automatyzujących czynności projektotwórcze. Drugim obszarem oddziaływania były pakiety CAD/CAM. Odnoszą się one wprawdzie do projektów w dziedzinie techniki, lecz ogólny stosowany tam sposób wspomagania i stymulowania twórcy systemu może i powinien być wykorzystywany w TSI zarządzania. Jednak w dziedzinie zastosowań gospodarczych informatyki mamy na ogół do czynienia z koniecznością rozwiązywania zadań źle wyspecyfikowanych (TURS 85), niestrukturalizowanych, toteż trudno

mówić o p e ł n e j automatyzacji całego cyklu. Wiedza , doświadczenie i umiejętności projektantów mogą być właśnie jedynie komputerowo wspomagane, choć szereg zespołów pracuje nad znacznym ograniczeniem czynności wykonywanych metodami ręcznymi na rzecz technik zautomatyzowanych. Spośród czynników oddziałujących na rozwój narzędzi CASE niewątpliwie należy podkreślić pionierską rolę, jaką od początku lat siedemdziesiątych odgrywał pakiet PSL/PSA (również w Polsce), którego olejne wersje są nadal doskonalone.

2. Składniki i klasyfikacja narzędzi CASE

Koncepcja a następnie konkretne rozwiązania narzędzi CASE wyszły na ogół z małych zespołów badawczo-rozwojowych, wywodzących się ze środowiska akademickiego. Ocenia się (BOST 88) iż aktualnie dostępnych jest h a n d l o w o na rynku oprogramowania ponad 100 pakietów wspomagających proces TSI. Jest to więc zjawisko t r w a ł e i poważne, które będzie miało istotny wpływ na strategię komputeryzacji organizacji gospodarczych i administracyjnych w najbliższej przyszłości. Wprawdzie sprzedaż narzędzi CASE na rynku w 1987r. była niewielka i wynosiła 140 mln $ lecz w następnym roku zanotowano wzrost o 100% (COOKB88).

W minionym okresie niewiele i n w e s t o w a ł o się w tą dziedzinę na świecie (JONES88). Informatycy dążą do usprawnienia, automatyzacji pracy niemal wszystkich dziedzin działalności gospodarczej. Sami jednak są na ogół bardzo konserwatywni w automatyzacji własnej pracy - analizy i projektowania oraz programowania. W związku z tym powstaje p a r a d o k s - podczas gdy praca wielu działów przedsiębiorstwa jest niemal w pełni skomputeryzowana, analitycy i projektanci wykonują swoją pracę ręcznie. Pojawia się zatem potrzeba automatyzacji pracy "automatyzatorów". Ocenia się, iż nawet w krajach zachodnich pakiety CASE nie będą jeszcze w powszechnym użyciu w ciągu najbliższych kilku lat. Wymaga to p r z e s z k o l e n i a kadry w dziedzinie TSI zwłaszcza w kategoriach "filozofii" projektowania oraz technologii pracy, co jest porównywane z przejściem z produkcji jednostkowej na taśmową.

Stosunkowo obszerne rozważania na tematy istoty i s k ł a d n i k ó w metodyki TSI, zależności pomiędzy metodami, technikami i narzędziami, krótkiej charakterystyki wybranych metodyk i wspo-

magających je narzędzi CASE dokonano w opracowaniach (WRYC89a) i (WRYC88b). Przedstawiono tam m.in. koncepcję warsztatu i stanowiska pracy projektanta TSI z wykorzystaniem pakietów komputerowo wspomagających proces TSI . W tym miejscu warto więc przeprowadzić choćby ogólną klasyfikację narzędzi, ocenić kwestię kosztów i efektów oraz stosowanego sprzętu komputerowego.

Ze względu na powiązanie pakietów CASE z cyklem życia systemu można je podzielić na :

- zintegrowane (I - CASE) oraz
- cząstkowe.

Pierwszy rodzaj narzędzi w pełni wspomaga c a ł y cykl życia systemu, od analizy po wdrażanie, ocenę i adaptację. Wyniki projektowania na jednym etapie są transformowane do drugiego. Projektanci współpracujący w danym zespole mogą wykorzystywać w różnych fazach rozwoju systemu różnorodne składniki. Zakres takiego pakietu jest zindywidualizowany, jednak w jego centrum jest b a z a d a n y c h o tworzonym systemie, służąca generowaniu zestawień czy diagramów, tworzeniu systemu. Inne s k ł a d n i k i to: edytory diagramów, generatory raportów, edytory formatek, systemy dialogowe, słowniki (zkorowidze danych, edytory tekstu). Każde z tych narzędzi taktowane oddzielnie to narzędzie cząstkowe. Najczęściej są to różnorodne edytory diagramów.

Interesującą systematyzację narzędzi komputerowego wspomagania TSI podał Bostrom (BOST88). Wyróżnia on, z uwzględnieniem kryterium metodycznej f u n k c j o n a l n o ś c i , następujące rodzaje narzędzi :

- pakiety wykonujące jedynie funkcje graficzne,dostosowane do przyjętej notacji i pomijające inne reguły metodyczne,
- z wbudowanymi regułami metodycznymi, umożliwiającymi kontrolę poprawności i spójności w ramach jednego segmentu,
- poszerzone w stosunku do wyżej wymienionych o możliwość sprawdzenia w ramach wszystkich segmentów,
- narzędzia ogólne, gdzie można wprowadzać i sprawdzać nowe reguły oraz generować grafy i modele danych na podstawie tekstu.

C e n y produktów CASE różnią się od niskich (w warunkach krajów zachodnich) — 5.000$, poprzez średnie - do 15.000$, do wysokich, przekraczających niekiedy 100.000$. Mimo to, zainteresowanie narzędziami CASE rośnie. Ocenia się (COOK88), iż projekt średniej wielkości na ogół kosztuje 2-krotnie więcej aniżeli wynosi

początkowo zaplanowany budżet, przy czym są one zakończone rok po terminie. Czwarta część projektów nigdy nie jest zakończona a pozostałe trzy czwarte zawierają tak wiele różnorodnych błędów, że zespoły spędzają więcej czasu na korektach użytkowanych systemów aniżeli na rozwoju nowych systemów. A zatem istnieje potrzeba narzędzi utrzymujących automatycznie spójność, kompletność i poprawność a w dziedzinie przetwarzania - automatycznych generatorów kodów.

Użytkownicy narzędzi CASE podkreślają, że ich zastosowanie pociąga za sobą wzrost efektywności projektowania poprzez istotną redukcję czasu i kosztów tworzenia SI, podniesienie j a k o ś c i systemu użytkowego. Jednocześnie akcentuje się brak szerokiej akceptacji tych narzędzi, ich pewien uniwersalizm - ograniczenie elastyczności produktów CASE w dostosowaniu do poszczególnych zespołów projektowych i ich doświadczeń a także specyfiki zastosowań. Pozostaje kwestią twórców narzędzi jak osiągnąć te cechy narzędzi przy zachowaniu systematyczności metodyki i sposobu użytkowania.

Większość pakietów CASE może być użytkowana na mikrokomputerach IBM PC, choć przy narzędziach zintegrowanych sięga się po stacje robocze SUN,APOLLO czy komputery (mainframe) głównego szeregu.

3. Konferencja i wystawa CASExpo EUROPE

Wiele spośród narzędzi CASE osiągnęło w ostatnich kilku latach poziom dojrzałości handlowej i od dłuższego czasu były one reklamowane w czasopismach fachowych. Pierwsza konferencja i wystawa odbyła się w USA, których osiągnięcia w tej dziedzinie są najbardziej znaczące. CASExpo EUROPE była potwierdzeniem z jednej strony wejścia wielu firm amerykańskich na rynek europejski a z drugiej również, własnych oryginalnych osiągnięć zachodnioeuropejskich. Zarówno konferencja jak i wystawa trwały 4 dni. Uzupełniały się one wzajemnie.

Oto niektóre tematy wykładów i paneli, które odbyły się w trakcie konferencji :

- Metodyki ISI - podstawa sterowania systemem ,
- Wprowadzenie do CASE ,
- Porównanie i przegląd narzędzi CASE ,
- Zastosowanie narzędzi czwartej generacji (4 GL) i systemów zarządzania bazami danych z pakietami CASE ,
- Metodyczne zastosowanie CASE ,

- CASE - doskonalenie efektywności oprogramowania w ostatniej dekadzie dwudziestego wieku ,
- Rozwój standardów dla systemów CASE ,
- Doświadczenia w realizacji projektów z wykorzystaniem narzędzi CASE ,
- Wpływ języków programowania na CASE ,
- Przygotowanie organizacji do skutecznego wdrożenia CASE.

Na wystawie a właściwie targach CASExpo zaprezentowało się kilkudziesięciu producentów oprogramowania CASE. Dominowały pakiety zintegrowane, w zasadzie brak było narzędzi drobnych, cząstkowych. Oto n a z w y niektórych, znaczących pakietów (w nawiasie ich producenci) :

- Information Engineering Facility - IEF (James Martin Associates)
- Excelerator , PC Prism (Excelerator Software Products)
- MAESTRO (Softlab)
- Software Engineering Toolkit SET (PA Consulting Group)
- Auto - G (Advanced Systems Architectures)
- Prokit Workbench (Mc Donnel Douglas Information Systems)
- Toolkit (Yourdon International)
- CORVISION (CORTEX)
- BIS CI PSE BIS Applied Systems)
- Design Aid (NASTEC Corporation)
- JSP - Tool (Michael Jackson)
- Information Engineering Workbench IEW (Arthur Young)

Listę tą można kontynuować. Wyżej wymienione pakiety odpowiadają podanym uprzednio kryteriom zintegrowanego CASE. Zazwyczaj związane są z którąś ze znaczących metodyk TSI a zwłaszcza :

- metodyką opartą o model Obiekt - Atrybut - Związek
- metodyką analizy i projektowania strukturalnego w wydaniu zarówno Yourdona (De Marco jak i Gane'a) Sarsona .

4. Uwagi końcowe.

Dokładna analiza porównawcza istniejących narzędzi CASE wymaga oddzielnego obszernego opracowania. W niniejszym referacie, w związku z jego skróconym zakresem i ograniczonym czasem przygotowania, zwrócono uwagę na pewne kierunki i cechy narzędzi CASE zarysowane na wystawie CASExpo. Szereg zagadnień szczegółowych, w tym obszerny materiał graficzny zostanie przedstawione w trakcie

wykładu.

Bibliografia

[BUDE88] Budenberg D.,The Strategy of Integrating CASE, Software Management, November 1988

[BOST88] Bostrom B.T.,Information Systems Development Supporting Methodologies with Computerized Tools, w:[CURR88]

[CURR88] Current Trends in Information Systems Development Methodologies, Polish-Scandinavian Seminar,Paraszyno,June 1988

[COOK88] Cookson C.,Program to Solve the "Software Crisis", Financial Times, September 20,1988

[JONE88] Jones R.,The Case for Automating the Automation Business, DEC User, March 1988

[Pres87] Pressman R.S.,Software Engineering. A Practitioner's Approach, Mc Graw Hill 1987

[TURS85] Turski W.M.,O informatycznym rozwiązywaniu problemów nie tylko matematycznych, Sprawozdania Instytutu Informatyki UW, 148/1985

[WRYC88a] Wrycza S., Kierunki i uwarunkowania automatyzacji procesu tworzenia systemów informatycznych.Pierwsza Wiosenna Szkoła PTI,Świnoujście , maj 1988

[WRYC88b] Wrycza S.,Współczesne metodyki tworzenia systemów informatycznych zarządzania , Orgservice, Gdańsk 1988

SSP "Bratniak" ZUP 67 /69 420 +6