7 1979

informatyka

WYDAWNICTWO SIGMA

W NUMERZE

NACZELNA ORGANIZACJA TECHNICZNA

WYDAWNICTWO SIGMA

CZASOPISM I KSIĄŻEK TECHNICZNYCH

ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pokój 331, tel. 27-71-40 lub centrala 26-82-61 w. 285, dyżury redakcji 10.00—13.00

Zakł. Graf. „Tamka". Zam. 246. Papier druk. sat. V kl. 70 g. A1. Obj. 5 ark. druk. Nakład 7350. C-119.

Cena egzemplarza zł 25.— INDEKS 36124 Prenumerata roczna zł 300.—

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 7
MIESIĘCZNIK
1979
ROK XIV
lipiec

ORGAN KOMITETU INFORMATYKI, MINISTERSTWA NAUKI, SZKOLNICTWA WYŻSZEGO I TECHNIKI ORAZ KOMITETU NAUKOWO-TECHNICZNEGO NOT DS. INFORMATYKI

SIEGFRIED GAGSCH
BIFOA
Kolonia (RFN)

Zasady wspomaganej komputerowo syntezy zadań

Określenie „synteza zadań" wyznacza podstawowy zakres tematyczny z teorii organizacji. Dotychczas podstawowe pytanie z tego zakresu dotyczyło kryteriów, według których poszczególne zadania powinny być łączone w większe kompleksy zadań (np. w zadania częściowe) [1]. Natomiast problemy technologiczno-metodyczne, związane z realizacją procesów syntezy zadań, nie były dotąd poruszane. Przede wszystkim nie jest jeszcze zupełnie wyjaśnione, czy i w jaki sposób mogą być przeprowadzane procesy syntezy zadań z zastosowaniem komputera. Wspomagana komputerowo synteza zadań ma praktyczne znaczenie m.in. dla tworzenia komórek funkcjonalnych i modułów programowych we wspomaganym komputerowo systemie informacyjnym [2].

Problemy związane z rozwojem modeli, metod i procedur znajdują się w centrum następujących dwóch zakresów problemowych:

1) problemy teoretyczno-koncepcyjne odnoszące się do podstawowych kryteriów grupowania pojedynczych zadań w większe kompleksy względnie segmenty zadań

2) problemy prakseologiczno-instrumentalne, w skład których wchodzą zagadnienia
— konstrukcji modelu (jakie problemy syntezy zadań mogą być odtwarzane w modelu)
— selekcji metod (jakie metody matematyczne nadają się do optymalizacji względnie suboptymalizacji tych modeli)
— realizacji procedur (jak za pomocą komputera można realizować procesy syntezy zadań).

PODSTAWY TEORETYCZNE SYNTEZY ZADAŃ

Cel syntezy zadań polega na tym, aby łączyć dużą liczbę zadań elementarnych w większe kompleksy względnie segmenty zadań. Istniejące przy tym problemy dają się wyjaśnić przede wszystkim przez formalny opis syntezy zadań [3].

Struktura formalna syntezy zadań

Niech $A = \{a_1, a_2, ..., a_n\}$ będzie zadanym zbiorem zadań elementarnych. Cel syntezy zadań polega na określaniu podziału $A = \{A_1, A_2, ..., A_k\}$ na k segmentów zadań. Każdy podział powinien spełniać następujące warunki:

1) warunek rozłączności — wszystkie segmenty zadań powinny być parami rozłączne:

$$A_1 \cap A_j = \Phi \; (i \neq j; \;\; i, j = 1, 2, ..., k)$$

2) warunki zupełności — suma logiczna wszystkich segmentów powinna dawać zbiór A:

$$\bigcup_{i=1}^{k} A_i = A$$

3) warunek mocy segmentów (zbiorów) — oznaczając moc jako n_i (liczbę zadań elementarnych) segmentów zadań A_i spełnione jest:

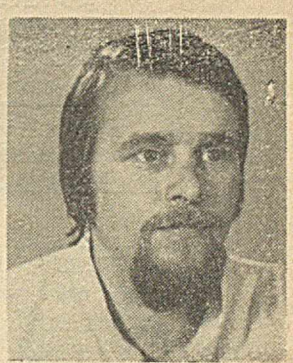
$$2 \leqslant n_i \leqslant n-2 \;\; \text{i} \;\; \sum_{i=1}^{k} n_i = n$$

[1] Por. np. Bleicher K.: Grundlagen der Abteilungsbildung. In: Zeitschrift für Betriebswirtschaft, 29. Jg. 1959, S. 106—118 sowie Kosiol E.: Organisation der Unternehmung. Wiesbaden 1962

[2] Por. Grochla E., Meller F.: Datenverarbeitung. Bd 2: Gestaltung und Anwendung. Reinbek bei Hamburg 1977

[3] Por. Gagsch S.: Budowa podsystemów w systemach informatycznych. Podstawy i metody syntezy zadaniowej opartej na uwzględnianiu współzależności. Międzynarodowa konferencja naukowa „Zastosowania systemów informatycznych do zarządzania w przedsiębiorstwie". Wrocław-Bierutowice, 3—4.X.1977

Siegfried GAGSCH ukończył w 1968 r. Wydział Ekonomiczno-Społeczny Uniwersytetu w Kolonii (RFN). Od tego czasu do chwili obecnej pracuje na tej uczelni w charakterze asystenta na kierunku studiów „Ogólna Ekonomika Przedsiębiorstwa i Nauka o Organizacji" oraz jako pracownik naukowy Instytutu Organizacji i Automatyzacji Przedsiębiorstw (BIFOA) w Kolonii. Specjalizuje się w zagadnieniach teorii organizacji, organizacji zautomatyzowanego przetwarzania danych oraz projektowania i zastosowania systemów informatycznych.



Każdy możliwy podział jest charakteryzowany przez następujące parametry:
— stopień segmentacji k (liczba segmentów)
— struktura podziału $n_1 + n_{2t} ..., n_k$ (moc segmentów zadań)
— forma realizacji (treść segmentów zadań).

W zależności od tego, które parametry podziału są zadane a priori, otrzymuje się różne sytuacje wyjściowe do syntezy.

1) Sytuacje bardzo ustrukturalizowane
Zadana jest struktura podziału. Odpowiedni stopień segmentacji daje się bezpośrednio „odczytać". Ze zbioru dopuszczalnych form realizacji należy wybrać określoną formę.

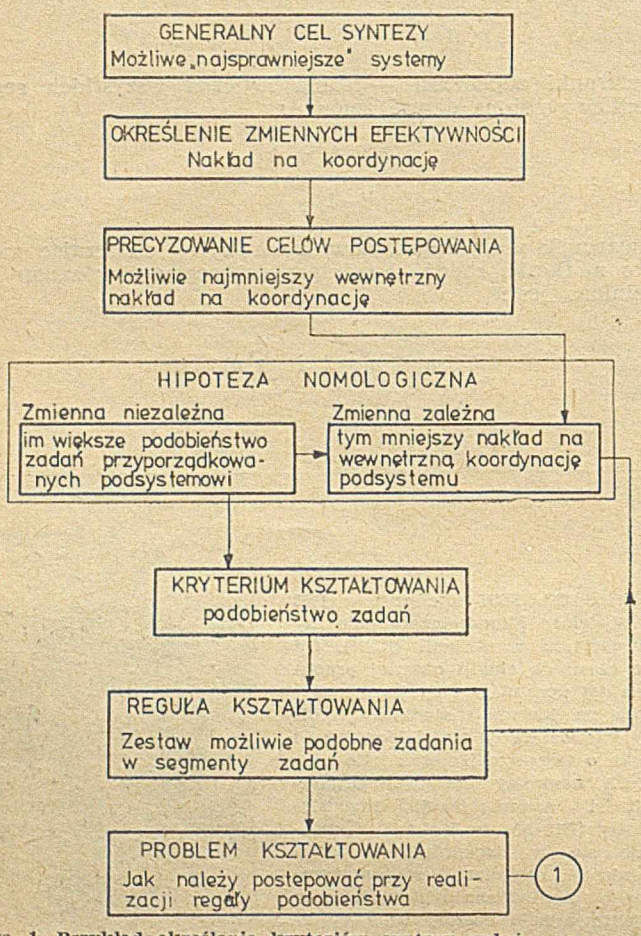
2) Sytuacje częściowo ustrukturalizowane
Stopień segmentacji jest zadany. Odpowiadające mu struktury podziału można względnie łatwo wyprowadzić (określić). Każdej strukturze podziału można przyporządkować różne formy realizacji. Zostaje do rozstrzygnięcia: jakie struktury podziału i formy realizacji należy wybrać.

3) Sytuacje słabo ustrukturalizowane
Nie ma zadanych ani struktur podziału, ani stopni segmentacji. Należy ustalić formę realizacji. Z wyborem tym następuje równocześnie wybór struktury podziału i stopnia segmentacji.

We wszystkich tych sytuacjach główny problem syntezy zadań polega na określeniu zawartości segmentów zadań, tzn. na wyborze określonej formy realizacji ze zbioru dopuszczalnych form realizacji. Nasuwa się przy tym nieuchronnie pytanie, na podstawie jakich kryteriów powinien następować wybór i jakie w tym kontekście należy wybrać formy syntezy zadań.

KRYTERIA I PODSTAWOWE FORMY SYNTEZY ZADAŃ

Segmenty zadań tworzą podstawę do konkretnego kształtowania podsystemów (np. komórek funkcjonalnych czy programów komputerowych). Kryteria syntezy zadań mają istotne znaczenie dla kształtowania podsystemów a więc muszą to być kryteria kształtowania. Kryterium syntezy zadań będzie więc np. „podobieństwo" i w tym aspekcie zasada tworzenia zbiorów będzie brzmieć „Zestawiaj segmenty zadań z zadań możliwie podobnych". Przykład takiego postępowania przedstawiony jest na rysunku 1.



Rys. 1. Przykład określania kryteriów syntezy zadań

Obecnie nie ma nomologicznych hipotez, z których mogą być wyprowadzone — według opisanego sposobu — teoretycznie uzasadnione kryteria i reguły syntezy zadań. Trzeba więc wrócić do mniej lub bardziej zasadnych reguł tworzonych ad hoc, z których wynikają dwie ważne formy syntezy zadań [4]).

1) Synteza zadań zorientowana na cechy
W tym podejściu elementami istotnymi są cechy stosowane do opisu zadań. Należy przy tym wyróżnić syntezę zadań zorientowaną na czynności oraz zorientowaną na obiekty; przy czym w obu przypadkach kryterium syntezy stanowi „podobieństwo" zadań (rys. 1). Opierając się na tym kryterium reguła podobieństwa brzmi: „Łącz te zadania, które odnoszą się do podobnych obiektów będą do podobnych czynności".

2) Synteza zadań zorientowana na relacje
Elementami integrującymi są relacje (zależności, sprzężenia) pomiędzy zadaniami. Kryterium syntezy jest „obfitość relacji" wewnątrz i pomiędzy segmentami zadań. Oparta na tym reguła relacyjna brzmi: „Łącz zadania w segmenty tak, żeby związki pomiędzy segmentami były możliwie małe a wewnątrz segmentów możliwie liczne".

PRAKSEOLOGICZNO-INSTRUMENTALNE PODSTAWY SYNTEZY ZADAŃ

Przy opracowywaniu modeli, metod i procedur dla wspomaganej komputerowo syntezy zadań należy zwrócić uwagę na:

- funkcję celu syntezy zadań, która wynika z zadanej reguły, tzn. reguły podobieństwa bądź reguły relacyjnej
- sytuację wyjściową syntezy zadań, która może być różnie ustrukturalizowana.

Jeśli proces syntezy zadań ma być realizowany z zastosowaniem komputera, to struktura problemu musi być opisana w formie modelu.

KONSTRUKCJA MODELU

Do wspomaganej komputerowo syntezy zadań konieczne są modele matematyczne zróżnicowane w zależności od przyjętego rodzaju syntezy — zorientowanej na cechy bądź zorientowanej relacyjnie.

Modele syntezy zadań zorientowanej na cechy

Do konstrukcji tych modeli są konieczne następujące dane:

— zbiór zadań $A = \{a_1, a_2 ..., a_n\}$, z których będą tworzone segmenty zadań

— zbiór cech $M = \{m_1, m_2, ..., m_p\}$, za pomocą których zadania będą charakteryzowane

— miary podobieństwa, za pomocą których podobieństwo zadań może być określone numerycznie.

Jeśli dysponuje się tymi danymi, konstruuje się macierz danych $X_{nxp} = (x_{ij})$ (tab. 1).

Tabela 1. Macierz danych dla zorientowanej na cechy syntezy zadań

Cechy / Zadania	m_1	m_2	...	m_p
a_1	x_{11}	x_{12}	...	x_{1p}
a_2	x_{21}	x_{22}	...	x_{2p}
.	.	.		.
.	.	.		.
.	.	.		.
.	.	.		.
a_n	x_{n1}	x_{n2}	...	x_{np}

[4]) Por Gagsch S.: Probleme der Partition und Subsystembildung in betrieblichen Informationssystemen. In: Management, Informations-systeme, hrsg. v. E. Grochla u. N. Szyperski, Wiesbaden 1971, S. 623—652, Gagsch S.: Subsystembildung. In: Znadwörterbuch der Organisation, 2. Aufl. hrsg. v. E. Grochla, Stuttgart 1979 (im Druck)

W zależności od tego, czy chodzi o cechy ilościowe, jakościowe czy binarne, możemy tworzyć różne macierze podobieństwa [5]).

Macierz podobieństwa jest najczęściej kwadratowa albo symetryczna $S_{(nxn)} = (s_{ij})$, przy czym elementy s_{ij} określają wielkość podobieństwa pomiędzy zadaniami a_i i a_j $(i, j = 1, 2, ..., n)$.

Taką macierz przedstawia tabela 2.

Tabela 2. Macierz podobieństw dla zorientowanej na cechy syntezy zadań

Zadania / Zadania	a_1	a_2	...	s_n
a_1	s_{11}	s_{12}	...	s_{1n}
a_2	s_{21}	s_{22}	...	s_{2n}
.	.	.		.
.	.	.		.
.	.	.		.
a_n	s_{n1}	s_{n2}	...	s_{nn}

Podstawowy problem w konstrukcji modelu polega na:
— określeniu sposobu liczby i stopnia uszczegółowienia cech uznanych za ważne
— ustaleniu miar podobieństwa, za pomocą których możliwe będzie numeryczne określenie podobieństwa. Po rozwiązaniu tego można odtworzyć adekwatnie i jednoznacznie strukturę formalną zorientowaną na cechy syntezy zadań.

Dla syntezy zadań zorientowanej na relacje konieczne będą specjalne modele w formie macierzy struktury.

● Modele syntezy zadań zorientowanej na relacje

Przy syntezie zadań zorientowanej na relacje obiektami odniesienia (wyróżniającymi) są systemy zadań względnie struktura zadań, tj. zbiory zadań elementarnych łącznie z powiązaniami między nimi [6]).

Do konstrukcji modelu nieodzowne są następujące dane:

- zbiór zadań elementarnych $A = \{a_1, a_2, ..., a_n\}$
- zbiór wszystkich obiektów (np. rodzajów danych przekazywanych pomiędzy zadaniami $D = \{d_1, d_2, ..., d_m\}$
- zbiór obiektów wejściowych dla zadań
- zbiór obiektów wyjściowych dla zadań.

Za pomocą tych danych można opisać strukturę danego systemu zadań i następnie przedstawić w postaci macierzy struktury.

Tok postępowania jest następujący:

1) konstrukcja (nxm) macierzy wejściowej $B = (b_{ij})$ wskazującej, które zadania określają dane obiekty wejścia, tzn.:

$$b_{ij} = \begin{cases} 1, & \text{gdy zadane } a_i \text{ produkuje (wytwarza) obiekt } d_j \\ 0, & \text{w innym przypadku.} \end{cases}$$

2) konstrukcja macierzy wyjściowej (mxn) $C = (c_{jk})$ opisującej, które obiekty będą na wyjściu zadań

$$c_{jk} = \begin{cases} 1, & \text{gdy obiekt } d_j \text{ będzie na wyjściu zadania } a_k \\ 0, & \text{w innym przypadku} \end{cases}$$

3) konstrukcja macierzy struktur $T_{(nxn)} = (t_{ik})$ $(i, k = 1, 2, ..., n)$, w której ujęte są związki pomiędzy zadaniami; otrzymuje się ją drogą mnożenia macierzy B i C, tak że $T_{(nxn)} = B_{(nxm)} C_{(mxn)}$. Macierz taką przedstawia tabela 3.

[5]) Przegląd potrzebnych tu procedur do określenia prawdopodobieństwa dają: Bock H. H.: Automatische Klassifikation, Göttingen 1974; Sodeur W.: Empirische Verfahren zur Klassifikation. Stuttgart 1974; Vogel F.: Probleme und Verfahren der numerischen Klassifikation, Göttingen 1975

[6]) Do odtworzenia kompleksowych struktur zadaniowych używane są w coraz większym stopniu specjalne (graficzno-werbalne) modele opisowe. Prototyp takiego modelu został opracowany w Instytucie Organizacji i Automatyzacji Ekonomiki Przedsiębiorstwa (BIFOA) na Uniwersytecie w Kolonii. Por. szczegółowo Grochla E. und Mitarbeiter: Integrierte Gesamtmodelle der Datenverarbeitung. München—Wien 1974

Tabela 3. Macierz struktury do syntezy zadań zorientowanej na relacje

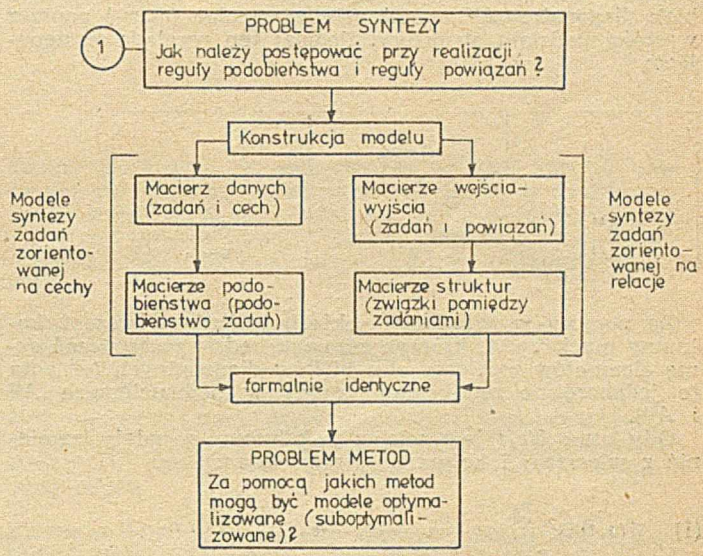
Zadania / Zadanie	a_1	a_2	...	a_n
a_1	t_{11}	t_{12}	...	t_{2n}
a_2	t_{21}	t_{22}	...	t_{2n}
.	.	.		.
.	.	.		.
.	.	.		.
a_n	t_{n1}	t_{n2}	...	t_{nn}

Elementy macierzy struktury (tab. 3) są na ogół liczbami określającymi liczbę obiektów (danych) przekazywanych pomiędzy zadaniami. Często wystarczy zaznaczyć w macierzy istnienie określonych związków (tworzenie macierzy binarnych zew jedynkowych). Decyzja, czy w syntezie zadań zorientowanej na relacje, będą stosowane macierze binarne, czy różnowartościowe, zależy od założonej funkcji celu. Jeśli ilość związków pomiędzy segmentami zadań powinna być minimalizowana (lub maksymalizowana wewnątrz segmentów), stosuje się macierze binarne. Jeśli celem jest minimalizacja mocnych relacji, konstruuje się macierze wielowartościowe.

● Identyfikacja modelu

Formalnie macierz podobieństwa i macierz struktury są identyczne, bowiem pokazują relacje pomiędzy zadaniami. Odtwarzają jednak różne treści: z jednej strony podobieństwo w sensie relacji teoretyczno-koncepcyjnych, z drugiej zaś związki wejścia-wyjścia w sensie relacji faktycznych.

Prakseologiczno-instrumentalne problemy związane z konstrukcją modelu są przedstawione na rys. 2.



Rys. 2. Konstrukcja modeli jako prakseologiczno-instrumentalny problem syntezy zadań

Modele obydwu grup (zorientowane na cechy i związki) są specjalnymi opisami stanów, tj. pokazują określony stan wyjściowy syntezy zadań w postaci sformalizowanej. Aby przeprowadzić dany stan wyjściowy w odpowiadający funkcji celu stan końcowy, nieodzowne jest korzystanie z metod rozwiązywania, przy których dokonuje się specjalnego opisu procesu.

Selekcja metod

Zarówno przy syntezie zadań zorientowanej na relację, jak i na cechy treść problemu stanowi kombinatoryczny problem ekstremalny, który zasadniczo daje się rozwiązać algorytmicznie. Jeśli wychodzi się jednak od modeli realnych, tzn. relatywnie dużych, to rozwiązania algorytmiczne

nie są możliwe lub — z zastosowaniem komputera — nieefektywne. We wszystkich praktycznych przypadkach należy w problemie syntezy zadań wrócić do metod heurystycznych. Umożliwiają one sekwencyjne polepszenie każdej macierzy wyjściowej i dają najczęściej suboptymalną macierz wynikową, tj. lepszą niż macierz wyjściowa, ale bez możliwości stwierdzenia czy jest to rozwiązanie najlepsze ze wszystkich. Tworzenie i wybór metod stanowią specjalne wieloaspektowe pole problemowe, które wchodzi w zakres badań operacyjnych i w tej pracy nie będzie szczegółowo omawiane [7]).

Należy jednak stwierdzić, że obecnie istnieje wiele metod możliwych do zastosowania. Przy wyborze i ocenie metod należy brać pod uwagę (obok funkcji celu) przede wszystkim stopień strukturalizacji, bowiem metody muszą być dostosowane do sytuacji wyjściowej.

Metody dla sytuacji bardzo ustrukturalizowanej

W sytuacjach bardzo ustrukturalizowanych struktura podziału jest zadana. Zarówno liczba jak i moc segmentów zadań (ilość zadań w segmencie) są więc znane i należy określić formę realizacji odpowiadającą funkcji celu. Dla tej sytuacji istnieją metody, które transformują zarówno macierze podobieństw, jak i macierze struktur w odpowiadające funkcji celu macierze wynikowe. Zasada metody jest następująca [8]).

Niech będzie zadana struktura podziału $n_1 + n_2 + n_3 + \ldots + n_k = n$ i $A^{22}{}_{(nxn)} = (a_{ij})$ — macierz podobieństw lub macierz struktury rozłożona na k diagnonalnych bloków $A^{11}{}_{(n_1xn_1)}$, $A^{22}{}_{(x_2xn_2)}$, ..., $A^{kk}{}_{(n_kxn_k)}$ których czynniki są określone poprzez zadane elementy podziału. Funkcja celu ma postać $S \rightarrow max$, przy czym S jest sumą elementów wszystkich bloków diagonalnych. Przy syntezie zadań zorientowanej na cechy oznacza to, że podobieństwo wewnątrz segmentów zadaniowych powinno być możliwie największe. Przy syntezie zadań zorientowanej na cechy powinno się dążyć do możliwie dużego stopnia zależności wewnątrz segmentów zadań. Przede wszystkim należy rozbić daną macierz wyjściową na podmacierze, przy czym pierwszy blok diagonalny A^{11} ma określony wymiar (n_1xn_1) poprzez wartości podziału struktury. Podział ten wygląda następująco:

n_1	$n-n_1$	
A^{11}	A^{12}	n_1
A^{21}	A^{22}	$n-n_1$

Za pomocą wyrażenia (1) określane będą wiersze i kolumny macierzy A, których zamiana będzie podwyższać sumę elementów macierzy obu bloków diagonalnych (oznacza to jednocześnie redukowanie sumy w podmacierzach A^{12} i A^{21}).

Gdy $\max \delta(s,t) > 0$ wiersz i kolumnę s należy zamienić z wierszem i kolumną t, przy czym:

$$(1)\quad \delta(s,t) = \sum_{i=1}^{n_1} (a_{it} + a_{ti} - a_{is} - a_{si}) + \sum_{i=n_1+1}^{n} (a_{is} + a_{si} - a_{it} - a_{ti}) - 2(a_{st} + a_{ts})$$

dla $s = 1,2,\ldots,n_1$, $t = n_1+1, n_1+2, \ldots, n$

Jeśli nie można określić wierszy i kolumn podatnych do wymiany ($\max \delta(s,t) \leqslant 0$), to uważa się, że zmodyfikowany drogą wymiany pierwszy blok diagonalny przechodzi w macierz wynikową. Identycznie zmodyfikowany drugi blok diagonalny jest traktowany jako nowa macierz wyjściowa. Jej pierwszy blok diagonalny przyjmuje w drugim podziale wymiar (n_2xn_2).

Metody dla sytuacji częściowo ustrukturalizowanych

W sytuacji częściowo ustrukturalizowanej jest określony jedynie stopień segmentacji k. Także w tym przypadku istnieją metody możliwe do zastosowania zarówno przy macierzach binarnych, jak i różnowartościowych. Odpowiednie metody wykorzystują z zasady podany związek (1) i umożliwiają stopniowe ustalenie bloków diagonalnych szukanej macierzy wynikowej. Do oceny są stosowane na ogół ilorazy współzależności, w których ilość wszystkich elementów macierzy (ogólnie n^2) zostaje powiązana z sumą faktycznych elementów tego bloku diagonalnego. W zależności od tego, jaki iloraz współzależności zostanie zastosowany, porównanie alternatywnych (konkurencyjnych) bloków diagonalnych wypada różnie, a to z kolei wpływa na wynik końcowy syntezy zadań. Stanowi to podstawowy problem przy selekcji i wyborze metod w sytuacjach częściowo ustrukturalizowanych. Problem ten występuje szczególnie często w sytuacjach słabo ustrukturalizowanych [9]).

Metody dla sytuacji słabo ustrukturalizowanych

W sytuacjach słabo ustrukturalizowanych nie jest zadana ani struktura podziału zadań, ani stopień ich segmentacji. Znane metody możliwe do stosowania nie mają charakteru ani syntetyzującego ani analizującego. W metodach syntetyzujących wychodzi się od pary zadań i tak długo, w kolejnych krokach, dodaje się zadania, aż funkcja celu osiągnie wartość ekstremalną.

Metody analizujące opierają się na zasadzie dekompozycji hierarchicznej. Macierz wyjściowa zostaje podzielona na dwie równe (lub prawie równe) bloki diagonalne, które będą „poprawiane" przez wymianę kolumn i wierszy.

Przedstawione metody opierają się na relatywnie prostych operacjach matematycznych, lecz są one w zależności wymiarów macierzy wyjściowej i stopnia strukturalizacji dość często stosowane.

Procedura realizacji

W przypadkach realnych (macierze wyjściowe z kilkoma setkami wierszy i kolumn) zagadnienia syntezy zadań możliwe są do realizacji jedynie za pomocą komputera [10]). Niezależnie od stopnia strukturalizacji i sytuacji wyjściowej każda metoda i model mogą być sformalizowane w postaci programu komputerowego, na podstawie którego przyjmuje się tylko jedną formę realizacji. Zaakceptowana zostanie ta forma realizacji, która odpowiada oczekiwaniom i nie wymaga zmian założonych parametrów (liczby i zakresu segmentów). W przypadku gdy zawartość segmentów zadań nie jest istotna, to parametry syntezy mogą ulegać zmianom i można wygenerować nową, komputerową formę realizacji.

Nasuwa się tu możliwość realizacji procesu syntezy zadań w formie dialogu [11]), przy którym istnieje możliwość prezentacji wyników każdego kroku segmentacji na monitorze ekranowym lub drukarce oraz takie ukształtowanie programów użytkowych, żeby istniała możliwość modyfika-

---

[7]) Przedstawienie i ocena różnych metod znajduje się u Müller-Merbach H.: OR-Ansätze zur optimalen Abteilungsbildung in Institution. In: Unternehmungsführung und Organisation, hrsg. v. W. Kirsch, Wiesbaden 1973, S. 93—124

[8]) Por. Gagsch S.: Probleme der Partition und Subsystembildung in betrieblichen Informationssystemen, a.a.O., s. 635 ff.; Sommerfeld E.: Heuristische Regeln für das Problem der Schnittstellenminimierung beim Entwurf von Leiterplatten. In: Analyse und Synthese von Problemlösungsprozessen, hrsg. von F. Klix, W. Krause und Sydow, Berlin 1972, s. 160—177; Adler H.-J.: Bestimmung von Teilkomponenten eines Netzwerkes bei gleichzeitiger Minimierung der Schnittstellen: In: Angewandte Informatik, 17 Jg. 1975, s. 52—54

[9]) W obu przypadkach można wrócić do metod heutystycznych, które m.in. znajdują zastosowanie w ramach automatycznej klasyfikacji i klasteringu (clustering-analysis). Por. Bock H. H.: Automatische Klassifikation Göttingen 1974; Vogel F.: Probleme und Verharen der numerischen Klassifikation. Göttingen 1975, Spath H. (Hrsg.): Fallstudien Cluster-Analyse. München—Wien 1977

[10]) Pracochłonność obliczeń i zapotrzebowanie na miejsce w pamięci dają się przy tym znacznie zredukować, gdy dana macierz wyjściowa zostaje przekształcona za pomocą specjalnych programów we (właściwą) górną lub dolną macierz trójkątną

[11]) Przykład na to znaleźć można u Berg C. C.: Quantitative Ansatze der Subsystembildung in Organisationen. In: Quantitative Ansatze in der Betriebswirtschaftslehre. hrsg. von H. Müller-Merbach, München 1978, S. 375—385

cji wyników pośrednich. Wymaga to obszernej pamięci i długich obliczeń, bowiem wszystkie poziomy hierarchiczne należy w pełni „przejrzeć", a wybrane fragmenty (segmenty) wyprowadzić z każdego poziomu na ekran monitora, drukarkę lub plotter. Jeśli się stwierdzi ,że jeden lub więcej segmentów odpowiadają oczekiwaniom użytkownika, to tych segmentów nie trzeba analizować i przerabiać na następnej płaszczyźnie. Cały proces segmentacji zostaje więc w ten sposób skrócony.

PODSUMOWANIE

Aktualną sytuację w zakresie wspomaganej komputerowo syntezy zadań można scharakteryzować następująco:
1) problemy teoretyczne syntezy zadań są rozwiązane tylko częściowo. W dalszym ciągu trudności teoretyczne sprawia zarówno uzasadnienie kryteriów syntezy zadań, jak i pomiar podobieństwa zadań.

2) W przeciwieństwie do tego problemy prakseologiczno-instrumentalne syntezy zadań zostały w znacznej mierze rozwiązane, ponieważ:

— modele matematyczne syntezy mogą być łatwo konstruowane

— istnieją dla różnie ustrukturalizowanych sytuacji wyjściowych matematyczne metody rozwiązania tych modeli

— wspomagane komputerowo procedury realizacji procesu syntezy zadań mogą być bez trudności implementowane na podstawie modeli i metod, które już są do dyspozycji. Należy jednak dać pierwszeństwo procedurom diagonalnym.

WIT DREWNIAK
Zarząd Mechanizacji i Automatyzacji
Opracowań Statystycznych GUS
Warszawa

Niezawodność systemów komputerowych ODRA 1305 i R–32

Szybki rozwój krajowej produkcji komputerów spowodował, że ODRY i RIADY coraz częściej trafiają do najrozmaitszych ośrodków obliczeniowych — m.in. do ośrodków, które poprzednio korzystały ze sprzętu zagranicznego. Po zainstalowaniu polskich komputerów zaczyna się je natychmiast porównywać z maszynami zagranicznymi. Pracownicy obsługi operatorskiej i technicznej stosunkowo szybko uzyskują właściwy obraz niezawodności nowego komputera, natomiast kierownictwo ośrodka obliczeniowego dość długo nie przyjmuje do wiadomości, że istnieją różne klasy niezawodności komputerów.

O różnych klasach niezawodności w ramach rodziny maszyn Jednolitego Systemu traktuje artykuł B. Gliksmana nt. wyników uzyskanych podczas eksploatacji komputerów JS EMC zainstalowanych w ZETO Katowice (INFORMATYKA nr 7—8/78). Na podstawie tabel zamieszczonych w tym artykule można wyliczyć wartości czasów międzyawaryjnych T_m w odniesieniu do wielomiesięcznego okresu eksploatacji [1]). Wartości te dla poszczególnych komputerów eksploatowanych w ZETO Katowice kształtowały się następująco:

R-20 (pierwszy rok eksploatacji) — 16,1 h
R-20 (drugi rok eksploatacji) — 52,8 h
R-22 (pierwszy rok eksploatacji) — 42,8 h
R-32 (11 miesięcy eksploatacji) — 11,1 h
R-50 (9 miesięcy eksploatacji) — 10,5 h

Tak więc w pierwszym roku eksploatacji średnie miesięczne czasy międzyawaryjne wynoszą w ZETO Katowice zaledwie kilkanaście godzin, natomiast w drugim roku — kilkadziesiąt godzin.

[1]) Podane poniżej czasy T_m dla maszyn ZETO Katowice oraz czasy T_λ dla maszyn Ośrodka Elektronicznego GUS w Warszawie wyliczono zgodnie ze wzorem (1)

Należy podkreślić, że wspomniany artykuł stał się poważnym argumentem dla obsługi technicznej w dyskusjach z użytkownikami, którzy chcieliby już w pierwszym roku eksploatacji uzyskać czasy międzyawaryjne rzędu kilkudziesięciu godzin bez względu na klasę niezawodności komputerów.

WSKAŹNIKI NIEZAWODNOŚCI EKSPLOATACJI KOMPUTERÓW W OE GUS WARSZAWA

Z praktyki eksploatacyjnej wiemy, że absolutna doskonałość obiektów technicznych (urządzeń) jest nieosiągalna, a więc obiekty te (szczególnie urządzenia komputerowe ze względu na wyjątkową złożoność konstrukcji) ulegają uszkodzeniom.

Pewną miarą niezawodności obiektu (urządzenia, systemu) jest średni czas pracy między dwoma kolejnymi uszkodzeniami tego obiektu w określonym okresie eksploatacji.

Mgr inż. Wit DREWNIAK od chwili zakończenia studiów na Wydziale Elektronicznym Politechniki Warszawskiej (1961r.) pracuje w pionie informatyki Głównego Urzędu Statystycznego, gdzie obecnie pełni funkcje głównego specjalisty ds. obsługi technicznej w Zarządzie Mechanizacji i Automatyzacji Opracowań Statystycznych. W latach 1962—1977 był wykładowcą w średnich szkołach technicznych o profilu informatycznym w Warszawie.

Czas ten — T_λ [2]) — potocznie nazywany wskaźnikiem niezawodności jest matematycznie określoną wartością średnią zmiennej losowej, oznaczającą czas poprawnej pracy między dwoma kolejnymi uszkodzeniami obiektu. Ujmuje to wzór:

$$T_{\lambda i} = \frac{1}{m_i} \sum_{j=1}^{m_i} t_{ij} \qquad (1)$$

gdzie: m_i — liczba stwierdzonych uszkodzeń i-tego obiektu w rozpatrywanym okresie eksploatacji

t_{ij} — czas pracy i-tego obiektu od momentu ukończenia naprawy obiektu po poprzednim (j-1) uszkodzeniu do następnego (j-tego) uszkodzenia obiektu.

W 1978 r. w Ośrodku Elektronicznym GUS w Warszawie przeprowadzono badania eksploatowanych tam systemów komputerowych pod kątem niezawodności działania. Wyniki badań przedstawiono w niżej podanych tabelach.

Z poniższych danych wynika, że komputery ODRA 1305 charakteryzują się wskaźnikiem niezawodności T_{ii} w od-

[2])W literaturze angielskiej czas T_λ określany jest skrótem MTBF (Mean Time Between Failures)

ODRA 1305, nr fabr. 208, drugi rok eksploatacji

Miesiąc/rok		01/78	02/78	03/78	04/78	05/78	06/78	07/78	08/78	09/78	10/78	11/78	12/78
Łączny czas pracy [h]	t_p	543	504	568	510	516	559	543	575	575	603	538	486
Łączny czas awarii [h]	t_a	16	17	15	25	16	16	9	38	8	13	40	24
Liczba przestojów awaryjnych	m_i	37	31	43	43	58	33	16	27	32	29	32	42
Średni czas między uszkodzeniami [h]	$T_{\lambda i}$	14,2	15,7	12,8	11,2	8,6	16,4	33,3	19,8	17,7	20,3	15,5	11,0

Wartość $T_{\lambda i}$ w odniesieniu do rocznego okresu eksploatacji wyniosła w 1978 r. 14,8 h (w pierwszym roku eksploatacji 15,1 h)

ICL 1903A, nr fabr. 431, dziewiąty rok eksploatacji

Miesiąc/rok		01/78	02/78	03/78	04/78	05/78	06/78	07/78	08/78	09/78	10/78	11/78	12/78
Łączny czas pracy [h]	t_p	552	525	586	520	515	569	549	577	584	603	550	479
Łączny czas awarii [h]	t_a	—	3	2	15	6	5	—	1	—	—	1	5
Liczba przestojów awaryjnych	m_i	2	4	5	5	5	5	2	3	—	3	4	4
Średni czas między uszkodzeniami [h]	$T_{\lambda i}$	276	130,5	116,8	111	101,8	112,8	274,5	192	584	201	183	118,5

Wartość $T_{\lambda i}$ w odniesieniu do rocznego okresu eksploatacji wyniosła w 1978 r. 160,2 h (w poprzednim roku — 55,9 h)

ICL 1905, nr fabr. 201, dwunasty rok eksploatacji

Miesiąc/rok		01/78	02/78	03/78	04/78	05/78	06/78	07/78	08/78	09/78	10/78	11/78	12/78
Łączny czas pracy [h]	t_p	552	514	579	510	508	561	541	575	534	553	528	486
Łączny czas awarii [h]	t_a	4	5	44	11	3	15	5	4	3	2	4	14
Liczba przestojów awaryjnych	m_i	18	12	40	7	6	17	11	10	11	6	9	11
Średni czas między uszkodzeniami [h]	$T_{\lambda i}$	30,4	42,4	13,3	71,2	84,1	32,1	48,7	57,1	48,2	91,8	58,2	42,9

Wartość $T_{\lambda i}$ w odniesieniu do rocznego okresu eksploatacji wyniosła w 1978 r. 40,0 h (w poprzednim roku — 72,5 h)

ODRA 1305, nr fabr. 197, drugi rok eksploatacji

Miesiąc/rok		01/78	02/78	03/78	04/78	05/78	06/78	07/78	08/78	09/78	10/78	11/78	12/78
Łączny czas pracy [h]	t_p	375	335,5	375,5	363	348	393	360,5	398	444,5	552	537,5	486
Łączny czas awarii [h]	t_a	37	57	6,5	10	14	13	14	17	10	10	20	44
Liczba przestojów awaryjnych	m_i	24	23	9	8	10	25	12	15	7	15	15	64
Średni czas między uszkodzeniami [h]	$T_{\lambda i}$	14,0	12,1	41,0	44,1	33,4	15,2	28,8	25,4	62,0	36,1	34,5	6,9

Wartość $T_{\lambda i}$ w odniesieniu do rocznego okresu eksploatacji wyniosła w 1978 r. 20,7 h (w pierwszym roku eksploatacji 23,3 h)

ODRA 1305, nr fabr. 233, drugi rok eksploatacji (w poprzednim roku pracowała tylko 4 miesiące)

Miesiąc/rok		01/78	02/78	03/78	04/78	05/78	06/78	07/78	08/78	09/78	10/78	11/78	12/78
Łączny czas pracy [h]	t_p	375	336	376	363	348	393	361	399,5	454	552	529	484,5
Łączny czas awarii [h]	t_a	4	13	5,5	10,5	9	5	4	5,5	2	3	7,5	7
Liczba przestojów awaryjnych	m_i	15	18	11	16	12	20	13	12	7	8	5	29
Średni czas między uszkodzeniami [h]	$T_{\lambda i}$	24,7	17,9	33,6	22,0	28,2	19,4	27,4	32,8	64,5	68,6	104,3	16,4

Wartość $T_{\lambda i}$ w odniesieniu do rocznego okresu eksploatacji wyniosła w 1978 r. 29,4 h

R-32, nr fabr. 021, drugi rok eksploatacji (w pierwszym roku eksploatacji pracował 8 miesięcy)

Miesiąc/rok		01/78	02/78	03/78	04/78	05/78	06/78	07/78	08/78	09/78	10/78	11/78	12/78
Łączny czas pracy [h]	t_p	180	175,5	191	192,5	183	202	195,5	201	183,5	195	180,5	187
Łączny czas awarii [h]	t_a	33	45	16,5	19,5	58	20,5	53	32,5	75	23,5	30	5,5
Liczba przestojów awaryjnych	m_i	15	13	8	20	20	16	35	22	31	17	18	43
Średni czas między uszkodzeniami [h]	$T_{\lambda i}$	9,8	10,0	21,8	8,6	6,2	11,3	4,0	7,6	3,5	10,0	8,3	4,2

Wartość $T_{\lambda i}$ w odniesieniu do rocznego okresu eksploatacji wyniosła w 1978 r. 7,1 h (w poprzednim roku — 8,8 h)

niesieniu do rocznego okresu eksploatacji w granicach 20—30 h, maszyn R-32 7 + 9 h, maszyn ICL 1905 — około 50 h z tendencją malejącą (12 lat eksploatacji) oraz maszyna ICL 1903A — około 100 h.

Zauważmy, że dla tego samego typu maszyny, a mianowicie R-32, wyniki uzyskane przez ZETO Katowice i OE GUS Warszawa są zbieżne. Inaczej być nie mogło, bowiem niezawodność, jest niewątpliwie wewnętrzną własnością obiektu a nie cechą jakości jego obsługi technicznej. Obsługa techniczna może wprawdzie pogorszyć lub polepszyć pracę systemu komputerowego, ale nie może zmienić wewnętrznych własności tego systemu.

Rozpatrzmy teraz jak otrzymane wyniki w odniesieniu do $T_{\lambda i}$ dla maszyn ODRA 1305, R-32 oraz maszyn zagranicznych, mają się w stosunku do norm obowiązujących w naszym kraju.

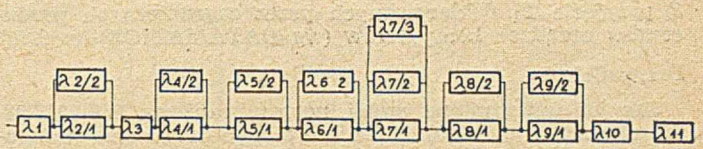
WYMAGANIA NORMALIZACYJNE I SCHEMATY STRUKTURY NIEZAWODNOŚCIOWEJ SYSTEMU KOMPUTEROWEGO

Według normy branżowej BN-78/3108-03 minimalna wartość czasu $T_{\lambda i}$ dla urządzeń komputerowych wynosi 100 godzin.

Przyjmijmy, że typowy system komputerowy składa się z następujących urządzeń: 1 procesor, 3 bloki pamięci operacyjnej (po 32 K słów), 1 konsola operatorska (monitor), 1 jednostka sterująca pamięci dyskowych (wykonanie „podwójne"), 6 pamięci dyskowych, 2 jednostki sterujące pamięci taśmowych, 8 pamięci taśmowych, 2 drukarki wierszowe, 2 czytniki kart, 1 czytnik taśmy papierowej, 1 dziurkarka taśmy papierowej.

W praktyce eksploatacyjnej urządzenia rezerwowe najczęściej pracują, a więc nie są wyłączone i oczekują na awarię (uszkodzenie) urządzenia podstawowego. Naprawa uszkodzonego urządzenia odbywa się dopiero po wyłączeniu tego urządzenia z pracy. Taki system pracy odpowiada równoległej strukturze niezawodnościowej systemu komputerowego z tzw. rezerwą obciążoną bez naprawy.

Jeśli przy przyjęciu takiej struktury niezawodnościowej systemu komputerowego założymy, że wyżej podany typowy system komputerowy zawiera jako urządzenia rezerwowe 1 blok pamięci 32 K słów, 1 jednostkę sterującą pamięci dyskowych, 1 pamięć dyskową, 1 jednostkę sterującą pamięci taśmowych, 2 pamięci taśmowe, drukarkę i 1 czytnik kart oraz że w danej chwili uszkodzeniu ulega tylko jedno urządzenie danego typu, to schemat struktury niezawodnościowej omawianego systemu komputerowego można przedstawić następująco.

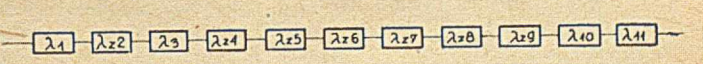


Rys. 1. Schemat struktury niezawodności systemu komputerowego

Symbole λ charakteryzują intensywność uszkodzeń

— procesora (λ_1)
— bloku pamięci operacyjnej 32 K słów ($\lambda_{2/1}$)
— redundancyjnego bloku pamięci 32 K słów ($\lambda_{2/2}$)
— konsoli operatorskiej (λ_3)
— jednostki sterującej pamięci dyskowych ($\lambda_{4/1}$)
— rezerwowej jednostki sterującej pamięci dyskowych ($\lambda_{4/2}$)
— pamięci dyskowej ($\lambda_{5/1}$)
— rezerwowej jednostki pamięci dyskowej ($\lambda_{5/2}$)
— jednostki sterującej pamięci taśmowych ($\lambda_{6/1}$)
— rezerwowej jednostki sterującej pamięci taśmowych ($\lambda_{6/2}$)
— pamięci taśmowej ($\lambda_{7/1}$)
— 2 rezerwowych jednostek pamięci taśmowej ($\lambda_{7/2}$, $\lambda_{7/3}$)
— drukarki wierszowej ($\lambda_{8/1}$)
— rezerwowej drukarki wierszowej ($\lambda_{8/2}$)
— czytnik taśmy papierowej (λ_{10})
— rezerwowego czytnika kart ($\lambda_{9/2}$)
— czytnika taśmy papierowej (λ_{10})
— dziurkarki taśmy papierowej (λ_{11}).

Ostatecznie każdy schemat struktury niezawodnościowej można sprowadzić do postaci szeregowej. W naszym przypadku szeregowy schemat struktury niezawodnościowej systemu komputerowego przedstawia się następująco:



Rys. 2. Szeregowy schemat struktury niezawodności systemu komputerowego

Litera z przy symbolach oznacza intensywność zastępczą uszkodzeń dla danej grupy urządzeń. Wartości zastępcze λ_{zn} obliczamy ze wzoru

$$\lambda_{zn} = \frac{1}{T_{\lambda zn}} \quad [h^{-1}] \tag{2}$$

po wcześniejszym obliczeniu $T_{\lambda zn}$ ze wzoru

$$T_{\lambda zn} = \frac{1}{\lambda} \sum_{i=1}^{k+1} \frac{1}{i} \quad [h] \tag{3}$$

jako odpowiadającego równoległej strukturze niezawodnościowej z rezerwą obciążoną bez naprawy,

gdzie: λ — intensywność uszkodzeń urządzenia (identyczna dla wszystkich urządzeń w danej grupie)
k — liczba urządzeń rezerwowych
n — identyfikator grupy urządzeń.

Intensywność wypadkowa uszkodzeń systemu komputerowego λ_{ws} będzie sumą intensywności poszczególnych członów szeregowego schematu struktury niezawodnościowej wspomnianego systemu, a mianowicie:

$$\lambda_{ws} = \lambda_1 + \lambda_{z2} + \lambda_{z3} + \lambda_{z4} + \lambda_{z5} + \lambda_{z6} + \lambda_{z7} + \lambda_{z8} + \lambda_{z9} + \lambda_{z10} \quad [h^{-1}] \tag{4}$$

Natomiast średnią wartość czasu między kolejnymi dwoma uszkodzeniami systemu komputerowego $T_{\lambda ws}$ obliczamy z zależności

$$T_{\lambda ws} = \frac{1}{\lambda_{ws}} \quad [h] \tag{5}$$

Obliczenia $T_{\lambda ws}$

Rozpatrzmy trzy różne przypadki, a mianowicie:
1) założenie minimalnych T_λ dla poszczególnych urządzeń komputerowych
2) przyjęcie T_λ deklarowanych przez krajowych producentów urządzeń
3) przyjęcie T_λ deklarowanych przez zagranicznych producentów urządzeń komputerów (wg DATAMATION No 9/78).

Przypadek 1

Jak już wspomniano wyżej, norma branżowa BN-78/3208-03 określa $T_{\lambda min} = 100$ h dla dowolnego urządzenia komputerowego. W oparciu o wzory (2), (3), (4) i (5) możemy obliczyć minimalny średni czas między kolejnymi dwoma uszkodzeniami systemu komputerowego $T_{\lambda ws\,min}$ [h]

Przypadek 2

Dostawcy lub producenci krajowych urządzeń komputerowych podają następujące wartości czasów T_λ dla poszczególnych urządzeń:

jednostka centralna ODRA 1305 z pamięcią operacyjną 32 K słów	— 120 h
konsola operatorska (Facit lub DZM 180/05)	— 1500 h
jednostka sterująca pamięci dyskowych (pds 325)	— 1000 h
pamięć dyskowa (produkcji bułgarskiej)	— 1000 h
jednostka sterująca pamięci taśmowych (MTS25-02)	— 450 h
pamięć taśmowa (PT-3)	— 500 h
drukarka wierszowa (DW 325)	— 1000 h
czytnik kart (CK 325)	— 450 h
czytnik taśmy papierowej (CDT325 — część czytnika)	— 500 h
dziurkarka taśmy papierowej (CDT325 — część dziurkarki)	— 200 h

Dla powyższych danych otrzymamy wartość średniego czasu między kolejnymi dwoma uszkodzeniami krajowego systemu komputerowego $T_{\lambda ws} = 37{,}0$ h.

Przypadek 3

Według danych zawartych w DATAMATION nr 9/78 obecnie typowe wartości czasów T_λ dla poszczególnych urządzeń komputerowych w krajach zachodnich są następujące:

procesor	— 1000 h
bloki pamięci wewnętrznej 1 MB	— 4000 h
jednostka sterująca pamięci dyskowych lub taśmowych	— 3000 h
pamięć dyskowa	— 2500 h
pamięć taśmowa	— 1000 h
drukarka wierszowa	— 300 h

Bez popełnienia większego błędu można przyjąć, że T_λ dla czytnika taśmy papierowej wynosi 500 h, czytnika kart — 450 h, dziurkarki taśmy papierowej — 200 h i konsoli operatorskiej — 1500 h.

Dla powyższych danych otrzymamy wartość średniego czasu między dwoma kolejnymi uszkodzeniami systemu komputerowego (zachodniego $T_{\lambda ws} = 72{,}3$ h.

WSTĘPNA ANALIZA WYNIKÓW OBLICZEŃ

Jeśli porównamy rzeczywiste $T_{\lambda 12}$ (patrz tabele) dla komputerów ODRA 1305 (20÷30 h) i R-32 (7÷11 h) z otrzymanym czasem $T_{\lambda ws\,min}$, to ponownie stwierdzamy zgodność obliczeń teoretycznych z praktyką eksploatacyjną.

Komputery ODRA 1305 znajdują się znacznie powyżej minimalnej granicy czasu T_λ (11,7 h) wyliczonego dla typowego systemu komputerowego w oparciu o normę branżową, natomiast komputery R-32 z ZETO Katowice i OE GUS Warszawa są poniżej tej minimalnej granicy.

Porównanie wartości $T_{\lambda ws}$ (37,0 h), obliczonego na podstawie deklarowanych przez krajowych producentów urządzeń komputerowych czasów T_λ, z wartościami rzeczywistymi $T_{\lambda 12}$ (20÷30 h) nawet dla maszyn ODRA 1305 wypada niekorzystnie. Świadczy to albo o błędnych wyliczeniach teoretycznych konstruktorów urządzeń komputerowych, albo o bagatelizowaniu problemu w procesie produkcji. Tym, którzy doszukują się całej winy u użytkownika systemu, a więc w złej pracy personelu obsługi technicznej, można odpowiedzieć, że maszyny zagraniczne obsługiwane przez tę samą kadrę techniczną wytrzymują wszelkie porównania z obliczeniami teoretycznymi, a w praktyce nawet przekraczają — w sensie pozytywnym — założenia teoretyczne.

Warto w tym miejscu dodać, że ustanowienie w normie branżowej $T_{\lambda min} = 100$ h dla urządzeń komputerowych jest za mało mobilizujące dla producentów tych urządzeń i dlatego na etapie tworzenia Polskiej Normy „poprzeczka" wymagań winna być odpowiednio podniesiona.

Na zakończenie warto jeszcze zwrócić uwagę na sprawę oprogramowania systemów komputerowych i ich wyposażenia w części zamienne. Oprogramowanie (systemowe, użytkowe, diagnostyczne) oraz części zamienne są niewątpliwie integralną częścią systemu komputerowego i dlatego muszą być traktowane na równi ze sprzętem. W naszej praktyce eksploatacyjnej niestety nie znajduje to potwierdzenia, bowiem części zamiennych chronicznie brakuje, a oprogramowanie jest tylko częściowo modernizowane i to z dużym opóźnieniem.

HENRYK DĄBROWSKI
OBR-SPIS
Warszawa

Bank Danych Statystycznych ROZWÓJ i kierunki budowy języka użytkownika

Bank Danych Statystycznych ROZWOJ stanowi część składową Centralnego Banku Danych Statystycznych, a tym samym jest jednym z bardzo ważnych elementów Systemu Państwowej Informacji Statystycznej (SPIS). Bank danych ROZWÓJ w swej koncepcji docelowej ma zawierać uporządkowany zbiór podstawowych wskaźników w ujęciu makroekonomicznym, charakteryzujących rozwój społeczno-ekonomiczny kraju w przedziale 20 lat (za punkt wyjścia przyjęto rok 1960). Przyjęto, że bank ten zawiera informacje (wskaźniki) za okresy roczne, bądź rejestrujące stan na określoną datę (rok).

Do banku ROZWÓJ wprowadzane są dane pierwotne nie przetworzone, przy czym użytkownik ma możliwość otrzymania charakterystyk liczbowych zarówno w postaci pierwotnej, jak i przetworzonej.

Użytkownikami Banku są jednostki naczelne administracji państwowej, (ministerstwa, urzędy centralne oraz jednostki równorzędne), instytucje naukowe oraz jednostki organizacyjne Głównego Urzędu Statystycznego.

Istnieją następujące możliwości korzystania z informacji zawartych w Banku:

— w trybie konwersacyjnym na ekranie monitora zainstalowanego w gmachu GUS, z możliwością ich wydrukowawania na drukarce sprzężonej z monitorem
— w formie wydruku na drukarce wierszowej, zamówionych zestawień tabelarycznych w trybie zdalnego dostępu przez upoważnionych użytkowników zewnętrznych.

Obecnie można uzyskać charakterystyki liczbowe w postaci przetworzonej dzięki zastosowaniu następujących algorytmów:

- wskaźniki dynamiki przy stałej podstawie
- wskaźniki dynamiki łańcuchowej
- średnie roczne tempo wzrostu i średnia arytmetyczna
- przyrosty bezwzględne (arytmetyczne)
- wskaźniki struktury
- odejmowanie wybranych wierszy
- dodawanie wybranych wierszy
- dodawanie wybranych rubryk (lat)
- udziały wybranych wierszy w wierszu wybranym.

Prowadzone są prace zmierzające do rozbudowy powyższej listy o dalsze podstawowe procedury statystyczne.

Aktualnie informacje wejściowe przygotowywane są przez jednostki organizacyjne Głównego Urzędu Statystycznego. W przyszłości, w miarę postępu w automatyzacji sprawozdawczości statystycznej, przewiduje się zasilanie Banku bezpośrednio z istniejących maszynowych nośników informacji.

Bank ROZWÓJ obejmuje: wykaz identyfikatorów wraz z odpowiednią usystematyzowaną listą informacji, tworzący zbiór KATALOG oraz wartości liczbowe odpowiadające danemu identyfikatorowi w określonym momencie czasu, tworząc zbiór DANE. Ponadto istnieje trzeci zbiór zawierający podstawowe algorytmy przetwarzania danych stosownie do zasad i wymogów analizy statystyczno-ekonomicznej.

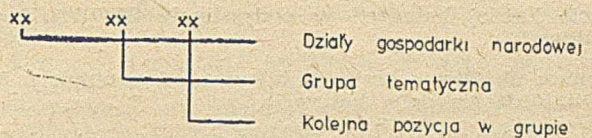
Do banku wprowadzone zostały informacje zgrupowane w następujących blokach tematycznych: Polska, przemysł, budownictwo, rolnictwo, leśnictwo, transport i łączność, handel zagraniczny. Prowadzone są również prace nad uzupełnieniem Banku o następne bloki tematyczne: ochrona zdrowia i opieka społeczna, handel wewnętrzny, praca i płace, nauka, oświata i wychowanie, gospodarka komunalna i mieszkaniowa.

Aktualizacja zawartości Banku jest przeprowadzana dwa razy w roku: 1) wstępna, dotycząca ograniczonego zakresu informacyjnego (do końca marca) oraz 2) ostateczna, dotycząca pełnego zakresu informacji (do końca sierpnia roku następnego po roku sprawozdawczym).

W ramach aktualizacji danych w Banku koryguje się szeregi czasowe z punktu widzenia zachowania porównywalności. Potrzeby takie wynikają ze zmian podziału administracyjnego kraju, ze zmian w klasyfikacji gospodarki narodowej lub zasad jej stosowania oraz innych zmian naruszających porównywlaność danych w czasie.

ROZWÓJ obejmuje około 9500 różnoimiennych informacji o strukturze macierzy lub wektorów danych. Wszechstronne wykorzystanie tak dużych zasobów informacyjnych wymaga, by użytkownik dysponował językiem ułatwiającym identyfikację danych w zbiorze. Powszechnie stosowane katalogi informacji zapewniają z reguły identyfikację w oparciu o kod mieszany: pozycyjno-hierarchiczny. Listę nazw informacji dzieli się na działy, grupy, podgrupy tematyczne. Jest to więc kod klasyfikacyjny, zgodnie z którym każda nazwa ma tylko jeden identyfikator. Taki sposób kodowania informacji może odpowiadać jedynie niektórym użytkownikom w banku ROZWÓJ.

Zastosowano następujący sposób kodowania nazw informacji:



Podział na grupy tematyczne w działach jest różny. Użytkownik interesujący się informacjami dotyczącymi danego zagadnienia w przekroju wielu działów gospodarki narodowej musi więc wielokrotnie sięgać do katalogu i w każdym dziale od nowa poszukiwać potrzebnego identyfikatora. Dlatego w ramach SPIS podjęto prace nad systemem, który między innymi umożliwiłby wieloaspektową identyfikację informacji z możliwością generowania katalogów danych dla różnych użytkowników. Chodzi o to, aby dla każdej klasy użytkowników można było łatwo opracowywać tematyczne katalogi uwzględniające ich specyficzne potrzeby. Służyć temu powinien podsystem instrumentalny SPIS — SŁOWNIK. Jego podstawową funkcją jest generowanie leksyki języka użytkownika w zakresie nazw informacji gromadzonych w bazach danych.

Mgr Henryk DĄBROWSKI ukończył Wydział Ekonomiki Produkcji SGPiS w Warszawie i obecnie jest kierownikiem pracowni Centralnego Banku Danych w Ośrodku Badawczo-Rozwojowym Systemu Państwowej Informacji Statystycznej SPIS. Pracował w Zakładzie Techniki Statystycznej GUS jako organizator maszyn licząco-analitycznych oraz w przemyśle, zajmując się zagadnieniami organizacji przedsiębiorstw i stosowania nowoczesnych technik obliczeniowych, zwłaszcza w dziedzinie zatrudnienia i płac. W latach 1966—1977 był kierownikiem pracowni pracy i płac w Rządowym Centrum systemu CENPLAN.

Inną trudnością, na jaką napotyka się w korzystaniu z dużych baz danych, jest skrótowy sposób formułowania nazw wskaźników statystycznych. W katalogach podaje się zazwyczaj tylko część nazwy charakteryzującej zakres przedmiotowy wskaźnika, oraz pewne atrybuty formalne, jak np. charakter wielkości (stan na dzień, przeciętna itp.), jednostka miary ewentualnie metoda grupowania. Na podstawie takich charakterystyk użytkownik, zwłaszcza bez przygotowania statystycznego, nie ma możliwości dokładnej interpretacji treści liczby, jaką może uzyskać z banku danych. Podchodząc z zaufaniem do danych statystycznych sądzi on, że informacje w bazie są porównywalne zarówno w ujęciu dynamicznym, jak i w strukturze.

W rozwoju języka użytkownika baz danych statystycznych należy więc uwzględnić tworzenie systemu informującego w sposób pełny o treściowej stronie informacji statystycznej. Propozycje rozwiązania tego problemu ujęte zostały również we wspomnianym już podsystemie SŁOWNIK.

Do podstawowych zadań realizowanych przez podsystem SŁOWNIK należą:

- ujednolicenie nazewnictwa kategorii społeczno-ekonomicznych, ustalenie jednolitych jednostek miar, przekrojów, zakresów obiektów opisywanych za pomocą określonej kategorii, okresów, charakteru wielkości i innych cech danej kategorii
- ustalenie trybu wprowadzania nowych kategorii w procesie projektowania badań statystycznych
- formalizacja trybu zatwierdzania formularzy badań statystycznych
- ustalenie jednolitego systemu klasyfikacji kategorii w zbiorach danych statystycznych.

Przyjęto, że kategorią społeczno-ekonomiczną jest nazwa (łącznie z dodatkowymi określeniami, służącymi do uściślenia treści tej nazwy), którą można sformułować na podstawie treści „główki" i „boczka" formularza statystycznego (kategorie statystyczne) względnie formularza planistycznego (kategorie planistyczne).

Ujęcie każdej kategorii w podsystemie SŁOWNIK jest dwojakie:

1) poprzez określenie cech formularza (dokumentu), w którym znajduje się dana kategoria

2) poprzez określenie nazwy kategorii i jej cech, charakteryzujących ją możliwie wszechstronnie, wyczerpująco i jednoznacznie.

Podsystem SŁOWNIK może spełnić funkcje koordynacyjne w skali makroekonomicznej oraz koordynację bieżących prac statystycznych. Funkcje te polegają na utrzymaniu pełnej jednolitości pojęciowej, zakresowej oraz definicyjnej i formalno-językowej wszystkich kategorii społeczno-ekoonmicznych występujących w statystyce i planowaniu. Pozwoliłoby to na koordynację statystyki państwowej i planowania na poziomie kategorii społeczno-ekonomicznej.

Podsystem SŁOWNIK w bieżących pracach statystycznych pozwala na koordynację wielkości ekonomicznych występujących w formularzach różnych badań statystycznych oraz w formularzach planistycznych, a także umożliwia zautomatyzowanie projektowania zasobu informacji czyli (zautomatyzowania projektowania formularzy statystycznych).

W pracach nad podsystemem SŁOWNIK chodzi o wypracowanie wspólnej, jednolitej metodyki doskonalenia języka informacji ekonomicznej, a także wypracowanie takich modyfikacji, które zapewniałyby spójność informacyjną systemów centralnych przy zachowaniu niezbędnej elastyczności. Należy podkreślić, je jednorazowe badanie ewidencyjne nie może raz na zawsze rozwiązać tych problemów. Wszelkie przedsięwzięcia, zmierzające do opracowania zestawień kategorii (wskaźników) mających według przyjętych doraźnie kryteriów odzwierciedlać zapotrzebowanie informacyjne na poziomie języka użytkownika (języka w którym przekazuje się dane na wyjściu systemu informacyjnego) są z góry skazane na niepowodzenie, ponieważ tworzenie baz danych na tej podstawie prowadzi do znacznego usztywnienia systemu. W fazie analizy stanu faktycznego należy dążyć do ustalenia zbioru „kategorii elementarnych zarządzania" badając np. ich odporność na zmiany cenowe, nomenklaturowe itp.

Podsystem SŁOWNIK powinien zapewnić możliwość przeprowadzania tego rodzaju badania w stosunku do całej informacji ekonomicznej zawartej w formularzach z informacją sprawozdawczą, statystyczną, planistyczną i finansową.

Celem zapewnienia spójności centralnych systemów informatycznych konieczne jest zaprojektowanie i wdrożenie podsystemu wspólnego, którego zasadniczą funkcją będzie „sterowanie językiem". SŁOWNIK pełniłby rolę zautomatyzowanego rejestru kategorii wraz z ich pełną charakterystyką. Rejestr kategorii byłby powiązany odpowiednimi procedurami z innymi rejestrami jednostek sprawozdawczych (obiektów informacji) oraz wszystkich powszechnie obowiązujących klasyfikacji i nomenklatur. Wszystkie bazy danych systemów centralnych gromadzące i przechowujące kategorie (wskaźniki) znajdowałyby się pod kontrolą wspomnianego wyżej podsystemu, który spełniłby także funkcję sterowania wymianą danych pomiędzy systemami. Wszelkie modyfikacje w zakresie kategorii elementarnych, klasyfikacji nomenklatur i wykazów jednostek byłyby możliwe jedynie poprzez odpowiednie procedury sterowania językiem.

Takie rozwiązanie zapewniałoby integralność całej sfery informacyjnej makrozarządzania.

Wykorzystując bogatą bazę informacyjną, przy zachowaniu jednolitości kategorii ekonomicznych oraz klasyfikacji i nomenklatur w statystyce, sprawozdawczości i planowaniu, można przystąpić do opracowania specjalnego języka użytkownika pozwalającego na bieżącą eksploatację baz danych.

Wspomniane przedsięwzięcia prowadzą do podwyższenia jakości zasobów informacyjnych i umożliwiają skorzystanie z rozwiniętej aparatury modeli ekonometrycznych.

Znajomość zawartości baz danych, listy dostępnych procedur oraz pokonanie bariery bezpośredniego dialogu pomiędzy (szeroko pojętym) użytkownikiem a komputerem zapewni wszechstronne wykorzystanie istniejących i przyszłych baz danych. Rozważmy dwie metody współpracy użytkownika z systemem. Współpraca użytkownika z systemem metodą pytanie — odpowiedź wymaga katalogu zawierającego słowniki baz danych, nazwy obiektów baz danych, nazwy układów grupowania i innych procedur wyszukiwania i przetwarzania danych oraz znajomości logicznej struktury bazy danych. Zadanie kompletnego pytania wymaga użycia sformalizowanego języka parametrów. Treść pytania zawiera klasyfikacje układów grupowania, nazwy obiektów i nazwy lub identyfikatory danych. Niedogodności takiego języka to konieczność częstego aktualizowania katalogów. Ponadto w warunkach korzystania z baz danych statystycznych język użytkownika składa się z niewielkiej liczby wyrażeń sterujących (słowa kluczowe) oraz dużej liczby zawartych w katalogach nazw rzeczywistych i nazw formalnych.

Metoda konwersacyjna współpracy użytkownika z systemem zakłada opracowanie prostego języka sterującego konwersją i krokowe udostępnianie przez system zasobów informacyjnych baz danych, a zwłaszcza rzeczywistych nazw danych obiektów i układów grupowania, co pozwala użytkownikowi na ostateczne sformułowanie pytania. Warunkiem pracy konwersacyjnej jest odpowiednia budowa słownika baz danych. Elementy wybrane przez użytkownika spośród prezentowanych przez system determinują kolejny niższy poziom hierarchii, zawierający bardziej szczegółowe informacje.

Metoda ta nie wymaga druku katalogów i słowników, a zawartość bazy jest prezentowana użytkownikowi w procesie konwersacji.

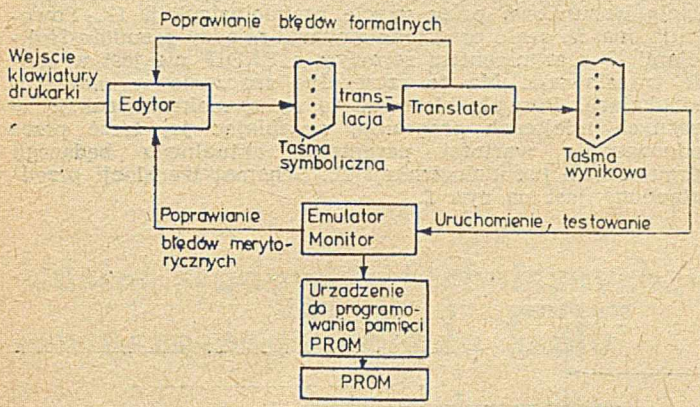
Dotychczasowe doświadczenia wykazały konieczność:

- dalszego rozszerzania banku ROZWÓJ o nowe bloki tematyczne celem zgromadzenia pełnego zbioru podstawowych wskaźników charakteryzujących rozwój społeczno-gospodarczy kraju w skali makroekonomicznej
- rozwoju języka użytkownika baz danych statystycznych, a więc przyspieszenia prac nad projektowanym podsystemem SŁOWNIK
- kontynuowania prac w zakresie przetwarzania konwersacyjnego z zastosowaniem omówionych metod.

TADEUSZ GÓRNICKI
Studium Doktoranckie
MICHAŁ WYRZYKOWSKI
Instytut Podstaw Elektroniki
Politechnika Warszawska

Uruchamianie systemów mikroprocesorowych przy użyciu MERY 303

Stosowanie mikroprocesorów oraz innych układów wielkiej skali integracji LSI do konstruowania nowych urządzeń wymaga odmiennych niż dotychczasowe metod projektowania i testowania. Mikroprocesor wraz z układami pamięci i buforami wejścia/wyjścia tworzą tzw. system mikroprocesorowy, będący częścią sterującą projektowanego urządzenia. Specyfika projektowania takich systemów polega na jednoczesnym tworzeniu konfiguracji sprzętu i oprogramowania, które są bardzo silnie ze sobą związane. Po zaprojektowaniu następuje etap uruchamiania systemu mikroprocesorowego. Wstępną fazą uruchamiania jest zaś testowanie programu sterującego projektowanym urządzeniem. Dopiero w oparciu o tak skontrolowany program można sprawdzać działanie sprzętu. Zagadnieniem uruchamiania i testowania systemów mikroprocesorowych z wykorzystaniem sprzętu krajowego poświęcony jest poniższy artykuł.



Rys. 1. Przygotowanie oprogramowania systemu

Przygotowanie programu dla systemu mikroprocesorowego (rys. 1) wymaga zapisania go w postaci symbolicznej na określonym nośniku informacji (np. na taśmie dziurkowanej). Następnie program jest tłumaczony na język wewnętrzny mikroprocesora i w postaci binarnej wyprowadzany na taśmę dziurkowaną lub bezpośrednio wprowadzany do pamięci. Ewentualne błędy formalne wykryte w fazie translacji są poprawiane za pomocą programu edytora. Uruchamianie i testowanie programu przebiega pod nadzorem tzw. monitora, który pozwala na łatwą detekcję błędów oraz śledzenie wykonania programu. Przygotowanie programu nie może się odbywać bezpośrednio w projektowanym systemie mikroprocesorowym, ponieważ:

— pamięć systemu zawiera tylko specjalizowane programy obsługi projektowanego urządzenia
— docelowy system zazwyczaj nie posiada urządzeń peryferyjnych.

Programy muszą więc być uruchamiane w niezależnym systemie komputerowym zwanym systemem uruchomieniowym.

Przetestowany program nie gwarantuje jeszcze poprawnego działania systemu mikroprocesorowego w konstruowanym urządzeniu. Może to wynikać np. ze złej współpracy programu ze sprzętem. Konieczne jest zatem jednoczesne testowanie całości. Niektóre systemy uruchomieniowe umożliwiają śledzenie wykonania programu w docelowym urządzeniu, poprzez tzw. emulację w czasie rzeczywistym (zastąpienie jednostki centralnej systemu projektowanego przez końcówkę wyprowadzoną z systemu uruchomieniowego). W tym przypadku program sterując urządzeniem znajduje się nadal pod kontrolą wspomnianego wcześniej monitora.

System uruchomieniowy powinien odznaczać się następującymi cechami:

— możliwością emulacji sprzętowej w czasie rzeczywistym
— posiadaniem edytora tekstu
— możliwością programowania w ASEMBLERZE
— możliwością programowania w językach wyższego rzędu (np. BASIC, FORTRAN, PL/M)
— możliwością łączenia programów i ich przemieszczania w pamięci
— posiadaniem manitora ułatwiającego śledzenie wykonania programu i detekcję błędów *(debugger)*
— możliwością programowania pamięci PROM.

Obecnie już kilkanaście firm produkuje systemy uruchomieniowe. W tabeli zestawiono najbardziej znane tego rodzaju systemy [1]. Jako kryterium porównawcze przyjęto istnienie (+) lub brak (—) w tych systemach wyżej wymienionych cech.

Ze względu na zastosowany sprzęt systemy uruchomieniowe można podzielić na dwie grupy:
1) systemy oparte na mikroprocesorach
2) systemy oparte na większych komputerach — systemy skrośne *(cross-systems)*.

Mgr inż. Tadeusz GÓRNICKI ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej w Instytucie Automatyki. Obecnie jest doktorantem w Politechnice Warszawskiej. Zajmuje się systemami cyfrowymi automatyki przemysłowej

Mgr inż. Michał WYRZYKOWSKI ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej w Instytucie Automatyki. Obecnie jest asystentem w Instytucie Podstaw Elektroniki na Wydziale Elektroniki PW. Zajmuje silę techniką cyfrową, a w szczególności zastosowaniem systemów mikroprocesorowych dla celów sterowania i kontroli.

Porównanie systemów uruchomieniowych

Typ systemu	Typ mikroprocesora	Emulacja sprzętu	Detekcja błędów	Edytor	ASSEMBLER	BASIC	FORTRAN	PL/M	Możliwość przemieszczania w pamięci
Data General	Micro NOVA	—	—	+	+	+	+	—	+
DEC	LSI 11	—	—	+	+	+	+	—	+
Fairchild	F8	+	+	+	+	—	—	—	+
Intel Intelec III	8021, 8041 8048, 8049 8080, 8085 3000	+	+	+	+	—	+	+	+
Mostek AID 80F	Z80	+	—	+	+	—	—	—	+
Tektronix 8002	8080, 6806 Z80, 990	+	—	+	+	—	—	—	—
TI 990/10	996	+	+	+	+	+	+	—	+
Motorola EXORCISER	6800	+	—	+	+	+	—	—	+

W systemach pierwszego rodzaju warunki uruchamiania są bardzo zbliżone do rzeczywistych warunków pracy projektowanego systemu. Programy są wykonywane na mikroprocesorach tego samego typu, co zainstalowany w docelowym, projektowanym urządzeniu. W systemach tego typu możliwa jest emulacja sprzętowa. Cena tych systemów jest stosunkowo niska (od 500 do 5000 dolarów) ze względu na niski koszt układów scalonych. Na uwagę zasługuje system TEKTRONIX 8002 [2], nie ogranicza on bowiem projektanta do jednego typu mikroprocesora. W systemie TEKTRONIX 8002 można uruchamiać programy dla mikroprocesorów 8080, 6800, TMS 9900, Z80 — zależnie od zastosowanego modułu (modułem jest płytka z określonym typem mikroprocesora) oraz dysku, na którym znajduje się odpowiednie oprogramowanie. TEKTRONIX 8002 działa w oparciu o trzy mikroprocesory: procesor systemowy, asembler-procesor oraz emulator-procesor, przy czym ten ostatni jest wymienny.

Systemy skrośne tworzone są na dużych komputerach poprzez odpowiednie ich oprogramowanie — nie wymagają zakupu dodatkowego sprzętu (mikroprocesorowego systemu uruchomieniowego). Systemy te — poza wspomnianymi programami EDYTORA, ASEMBLERA i MONITORA — muszą posiadać również programowy EMULATOR danego mikroprocesora (często sprzedawany przez firmy wraz z mikroprocesorem). Główną wadą systemów skośnych jest brak emulatora sprzętu w czasie rzeczywistym; jego wprowadzenie wymagałoby licznych przeróbek drogiego komputera.

Konkretny przykład systemu skrośnego został zrealizowany w Instytucie Automatyki Politechniki Warszawskiej przy współpracy z Instytutem Podstaw Elektroniki PW. System ten powstał w oparciu o minikomputer MERA 303 i pozwala przygotowywać, uruchamiać i testować programy w języku ASEMBLER 8080. Wybór MERY 303 został podyktowany niskim kosztem eksploatacji, łatwością obsługi i dostępu do tej właśnie maszyny, co pozwala na stosowanie systemu do celów dydaktycznych. Jednocześnie skromne możliwości minikomputera MERA 303 wymagały nałożenia pewnych ograniczeń na system, które nie powodują większych zmian w porównaniu z systemami tworzonymi na daleko lepszych maszynach.

Zgodnie z opisanymi już wymaganiami w skład systemu wchodzą następujące programy:

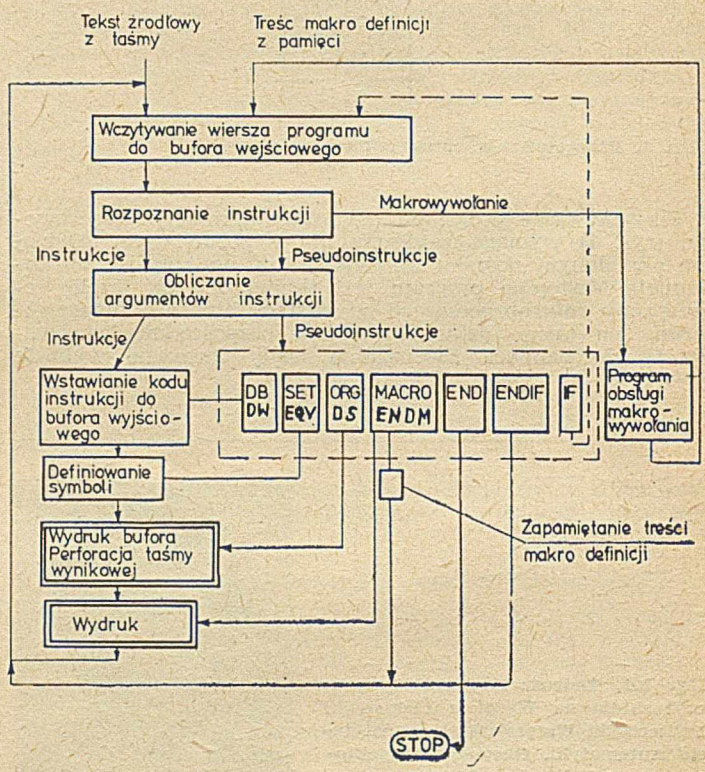
- EDYTOR
- MAKROASEMBLER 8080
- EMULATOR mikroprocesora 8080 wraz z MONITOREM.

EDYTOR służy do przygotowania taśmy papierowej z programem zapisanym w postaci symbolicznej. Tekst programu jest wprowadzany z klawiatury drukarki mozaikowej i wyprowadzany na taśmę papierową. Proces poprawiania jest analogiczny jak w edytorze XKYA powszechnie stosowanym przez ICL. Taśma symboliczna przygotowana za pomocą EDYTORA jest formą zapisu programu, akceptowaną przez MAKROASEMBLER 8080 i może być poddana translacji.

MAKROASEMBLER akceptuje język ASEMBLER 8080 opisany w „Intel 8080 Assembly Language Programming Manual" i jest implementacją na minikomputerze MERA 303 firmowego ASEMBLERA. Zgodnie ze standardem program pozwala na definiowanie symboli będących ciągiem znaków alfanumerycznych oraz adresowanie symboliczne. Oprócz instrukcji mikroprocesora 8080 ASEMBLER akceptuje również szereg pseudoinstrukcji pomocniczych, takich jak:

SET, EQU — nadawanie wartości symbolom
ORG — nadawanie wartości licznikowi adresów
DB, DW — deklaracja stałych
DS — deklaracja tablic.

Możliwe jest również deklarowanie makroinstrukcji z maksymalnie dziesięcioma parametrami. Dopuszcza się translacją warunkową przez użycie pseudoinstrukcji IF. Powoduje ona, że część programu zawarta między pseudoinstrukcjami IF z argumentem zerowym a ENDIF nie jest w ogóle translowana. Ma to szczególne znaczenie w makroinstrukcjach. Pozwala stosować tę samą makroinstrukcję w różnych miejscach programu, zmieniając jej treść przez odpowiednie wartości parametru aktualnego będącego argumentem pseudoinstrukcji IF. Schemat translacji przedstawiony jest na rys. 2.



Rys. 2. Schemat translacji

Wynikiem działania MAKROASEMBLERA jest:
— lista symboli drukowana zawsze w pierwszym przebiegu translacji
— wydruk programu wraz z dokładną mapą pamięci oraz wykaz błędów
— taśma wynikowa w formie znakowym.

Dokładny obraz pamięci, tzn. wydruk zawartości pamięci wraz z adresami, przydaje się przy późniejszym uruchamianiu i testowaniu programów. Zarówno wydruk, jak i taśma są tworzone opcjonalnie w zależności od położenia kluczy na pulpicie technicznym MERY.

MAKROASEMBLER może sygnalizować pojawienie się różnego rodzaju błędów. W razie wystąpienia błędu w nazwie instrukcji ASEMBLER wstawia w miejsce niewłaściwej instrukcji trzy instrukcje NOP *(no operation)*, co pozwala na ewentualne poprawianie programu już wczytanego do pamięci w fazie uruchamiania. Taśma wynikowa zawierająca przetłumaczony program i będąca produktem wyjściowym ASEMBLERA jest jednocześnie taśmą wejściową EMULATORA.

Kolejną fazą obróbki programu jest uruchomienie i testowanie. Do tego celu służy **EMULATOR 8080** wraz z MONITOREM. EMULATOR umożliwia testowanie programów o maksymalnej długości 4,5 K słów. Oprócz interpretowania instrukcji mikroprocesora 8080 EMULATOR realizuje również odwzorowanie 65 K słów pamięci mikroprocesora w 4,5 K słów pamięci MERY. Pamięć mikroprocesora została umownie podzielona na 256 bloków, z których 18 ma swój fizyczny odpowiednik w pamięci MERY. Użytkownik ma możliwość zadeklarowania obszaru pamięci potrzebnego dla testowanego programu, zmieniając numery odwzorowanych bloków, co pozwala testować programy niespójne umieszczone w dowolnych obszarach pamięci. Cały program EMULATORA wraz z MONITOREM jest objęty protekcją pamięci, tzn. nie może być zniszczony wskutek błędu użytkownika. Każde odwołanie się testowanego programu do nieodwzorowanego obszaru pamięci jest sygnalizowane jako błąd, przy czym podawany jest adres nielegalnego odwołania. EMULATOR może pracować w różnych trybach — tzn. krokowo lub ciągle, z wydrukami lub bez. Wydruk obejmuje zawartość wszystkich rejestrów mikroprocesora, wskaźnika stosu oraz licznika rozkazów.

Użytkownik ma możliwość zgłoszenia przerwania przez wciśnięcie odpowiedniego klucza na pulpicie technicznym MERY. Po zgłoszeniu przerwania do rejestru rozkazów zostaje wprowadzona instrukcja z pulpitu drukarki. Urządzenia we/wy są symulowane przez drukarkę mozaikową, na której podawany jest numer aktualnie symulowanego urządzenia. Tylko czytnik i perforator mogą być używane jako rzeczywiste urządzenia zewnętrzne emulowanego mikroprocesora.

Programem nadzorującym pracę EMULATORA jest **MONITOR.** Umożliwia on konwersację użytkownika z EMULATOREM, zawiera mechanizmy ułatwiające detekcję błędów, daje możliwość bezpośredniego dostępu do pamięci i rejestrów emulowanego mikroprocesora oraz pozwala wprowadzać i wyprowadzać taśmę dziurkowaną. Częścią MONITORA jest również DEASEMBLER umożliwiający wydruk zawartości pamięci w postaci symbolicznej. Użytkownik ma możliwość wstrzymania realizacji programu w dowolnym miejscu przez wprowadzenie tzw. pułapek. Natrafiwszy na adres, przy którym zastawiono pułapkę, EMULATOR wydrukuje zawartość rejestrów oraz przekaże sterowanie do MONITORA, dając operatorowi możliwość wydania nowych poleceń.

W przyszłości cały system urchomieniowy zostanie uzupełniony urządzeniem do programowania pamięci PROM, pozwalającym przenieść uruchomiony i przetestowany program z pamięci MERY wprost do pamięci stałej systemu mikroprocesorowego.

LITERATURA

[1] Santoni A.: Designing microprocessor software in modules' speeds development, ups reliabity. „Electronic Design" 11 z 24 maja 1978
[2] Lowry D., Stofer B.: The 8002 — a new design tool for microprocessor users. „Tekscope" vol. 9, no. 2, 1977
[3] „Intel 8080 Assembly Language Programming Manual". Intel Technical Library, Part no 98—301
[4] „MCS-80 User's Manual". Intel Technical Library, Part no. 98—153
[5] „MDC ICE-80 Reference Manual". Intel Technical Library, Part no. 98—167

Udział środowisk technicznych w kształtowaniu nowoczesnego oblicza kraju

W dniu 7 lipca br. I sekretarz KC PZPR Edward Gierek przyjął delegację Naczelnej Organizacji Technicznej. Delegacji przewodniczył prezes NOT, minister Przemysłu Maszynowego, dr inż. Aleksander Kopeć.

W toku spotkania podkreślono, że społeczne działanie środowisk technicznych jest ważnym czynnikiem rozwoju różnorodnych form pracy NOT i stowarzyszeń naukowo-technicznych, które swą działalnością przyczyniają się w istotny sposób do realizacji zadań społeczno-gospodarczego rozwoju kraju.

W wyniku decyzji podjętych przed rokiem — po I Krajowej Naradzie Przedstawicieli Samorządu Robotniczego — wzrosła rola stowarzyszeń naukowo-technicznych w pracach KSR. Ułatwiło to NOT realizację jej zadań zgodnie z potrzebami gospodarki narodowej.

Stwierdzono, iż federacja NOT dostrzega rezerwy tkwiące w wykorzystaniu obecnego potencjału kadr technicznych oraz możliwości stworzenia warunków zwiększających aktywność zawodową tej grupy — przez podniesienie rangi zawodu, poprawę wyposażenia warsztatu pracy, opracowanie i uruchomienie ogólnopaństwowego systemu doskonalenia wiedzy.

Uczestnicy spotkania poinformowali Edwarda Gierka, że z myślą o zbliżającym się VIII Zjeździe partii przygotowują inżynierskie prognozy rozwoju poszczególnych dziedzin i problemów techniki. Przekazali też meldunki o realizacji przez środowiska techniczne czynów dla uczczenia 35-lecia PRL.

W rozmowie wskazywano, że zbliżający się VIII Kongres Techników Polskich powinien określić zadania środowisk technicznych, wynikające z planów społeczno-gospodarczych przyszłej 5-latki, ustalić kierunki działania dla kształtowania nowoczesnego oblicza kraju oraz sformułować zadania organizacji technicznych w podnoszeniu kwalifikacji i etyki zawodowej inżynierów i techników.

Zwracając się do uczestników spotkania Edward Gierek wskazał, że zbiega się ono z przystąpieniem do dyskusji nad projektem tez do VIII Zjazdu partii.

Jeszcze 4 lata temu, gdy przystępowaliśmy do dyskusji nad tezami na VII Zjazd partii — powiedział I sekretarz KC PZPR — *sytuacja w świecie wyglądała zupełnie inaczej niż obecnie. Już wtedy przewidywaliśmy wzrost trudności, ich zakres przerósł jednak nasze oczekiwania. Dlatego teraz musimy tak sterować naszą gospodarką, by rozwijała się nadal i pomyślnie rozwiązywać problemy społeczne, z korzyścią dla wszystkich obywateli.*

I sekretarz KC PZPR przekazał zebranym, a za ich pośrednictwem całemu środowisku ludzi techniki życzenia pomyślnej i owocnej działalności dla dobra kraju.

PIOTR STRZAŁKOWSKI
Instytut Badań Jądrowych (CYFRONET)
Świerk

Modularyzacja. Część 2

Języki modularne

Języki modularne powstały na przecięciu dwu nurtów: starszego, związanego z potrzebą opracowania narzędzia ułatwiającego tworzenie dużych systemów oprogramowania a także prostszego operowania ich częściami (zajmowaliśmy się tym w części pierwszej artykułu) oraz nowszego rozwijającego się przede wszystkim w latach siedemdziesiątych, nawiązującego do próby całościowego ujęcia „cyklu życia" oprogramowania, stworzenia odpowiednich metodologii i zoptymalizowania kosztów wykonania i eksploatacji oprogramowania.

Modularyzację można traktować jako pewną technikę. W sposób naturalny została ona zaimplementowana w wielu nowych językach oprogramowania.

Scharakteryzujemy niektóre z tych języków — będziemy nazywać je modularnymi — powstałych w ostatnich latach. Wybrane ich cechy będziemy opisywać poprzez zestawienia związane z:

- współbieżnością
- rozdziałem idei od jej realizacji oraz „zasłanianiem" informacji
- badaniem poprawności programu.

WSPÓŁBIEŻNOŚĆ

Pojawienie się maszyn wieloprocesorowych umożliwiło jednoczesne (ansychroniczne) wykonywanie programów i zadań (zagadnienie równoległości wykonywania występuje także w przypadku maszyn wieloprogramowych).

Konieczne stało się oprogramowanie wykorzystujące tę nową możliwość. M.in. dla celów tworzenia i opisu systemów operacyjnych zaistniała potrzeba posiadania języków programowania, w których równoległe wykonanie można by było opisywać i nim sterować. Najbardziej znanymi językami tego typu są PASCAL Współbieżny (C-PASCAL, CONCURRENT PASCAL) Brinch Hansena [2, 3, 4] i MODULA Wirtha [5, 6, 7].

Rozważmy następujący przykład współbieżności (podajemy go za [14]).

W maszynie istnieje bufor wykorzystywany jednocześnie przez proces obliczania i proces drukowania. Pierwszy wysyła do bufora porcje przeznaczone do druku, drugi pobiera z niego porcje w celu zredagowania tekstu i drukowania. Bufor jest zasobem wspólnie dzielonym przez oba procesy. Zasobem takim może być również np. tablica. Kiedy kilka jednocześnie przebiegających procesów dąży w tej samej chwili do skorzystania z tego samego zasobu, może dojść do konfliktu i niekontrolowanego dostępu do zasobu. Konieczne jest więc opracowanie reguł dotyczących użytkowania dzielonych zasobów. Zwykle sprowadzają się one do tego, że w danej chwili z dzielonego zasobu może korzystać tylko jeden proces; jeśli inne także żądają dostępu do tego zasobu, to zostają one „zawieszone" do czasu, gdy zasób będzie „wolny" [2]).

Taką koncepcję wykorzystano przy opracowywaniu następujących nowych struktur programowych:
— tzw. „procesów" wykonywanych równolegle i składających się ze struktury lokalnych danych oraz z programu sekwencyjnego, a więc będących czymś w rodzaju wyróżnionych procedur
— tzw. „monitorów", skupiających w sobie dzieloną strukturę danych, wszystkie możliwe operacje na tej strukturze oraz część inicjalizującą strukturę danych; monitory synchronizują procesy współbieżne (za pomocą operacji ich zawieszania i ponownego uaktywniania), a także umożliwiają przepływ danych między tymi procesami.

Jak nie trudno zauważyć, struktura monitorów odpowiada całkowicie strukturze modułu „funkcjonalnego" opisanego w pierwszej części artykułu, natomiast proces odpowiada modułowi „klasycznemu".

Ideę monitorów podał Hoare [8], a języki MODULA i PASCAL Współbieżny są przykładami, różniącymi się nieco podejściem do realizacji tej idei.

W MODULI monitor reprezentuje tzw. moduł wiążący (ang. *interface module*), natomiast w PASCALU Współbieżnym istnieją po prostu „monitory". W obu językach procesy są wyodrębnionymi jednostkami programowymi.

Przykład 1

```
var p: process                  var v:monitor: (pl: tl; .. pk:tk);
var vl : T1; ... vm: Tm;        var yl: Tyl; ... yl: Tyl;
begin                           procedure entry Pl (...); begin...end;
sl;                             ..
...                             procedure entry Pn (...); begin...end;
sn                              begin
end                             so
                                end
```

W powyższym przykładzie deklaracje procesu i monitora w PASCALU Współbieżnym v1, ... vm, y1 ... yl są zmiennymi typów odpowiednio t1, ... Tm, Ty1 ... Tyl; p1, ... pk są parametrami typów t1, ... tk; s1, ... sn są instrukcjami procesu; so jest częścią inicjalizującą monitora: P1, ... Pn procedurami — operacjami na dzielonej strukturze danych. Jak widać, istnieje możliwość deklarowania zmiennych typu „proces" lub „monitor" (p i v). Procedury P1, ... Pn można wykorzystywać w ciele (ang. *body*) procesu po inicjalizacji monitora v za pomocą instrukcji init v w sposób następujący: v.P1 (...).

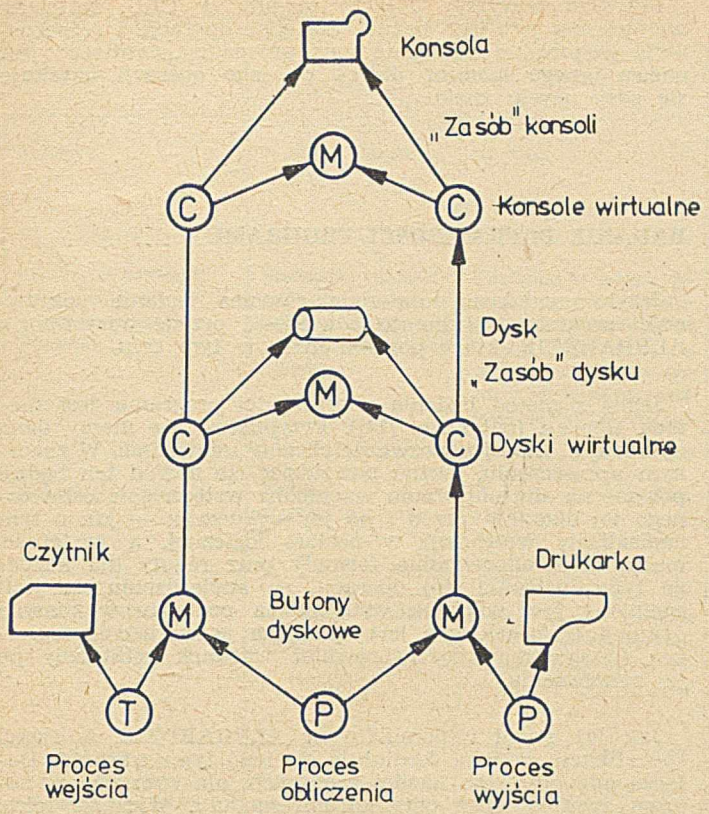
W PASCALU Współbieżnym zawieszenie procesu następuje przez wprowadzenie go do kolejki procesów oczekujących na zwolnienie zasobu za pomocą instrukcji „delay"; po zwolnieniu tego zasobu można czekający proces ponownie uaktywnić za pomocą instrukcji „continue". Kolejki idenyfikują zasoby.

W MODULI identyfikacji dokonuje się wg wyróżnionych zmiennych zwanych „sygnałami", a zawieszenie i wznowienie procesu sterowane jest za pomocą instrukcji „wait" i „send", wspomnianego monitora wiążącego („interface module"), które operują na sygnałach.

W obu językach oprócz monitorów i procesów wyróżnia się inne jeszcze jednostki programowe: w przypadku PASCALA Współbieżnego istotne jest pojęcie klasy, zewnętrznie podobnej do monitora z tą jednak różnicą, że dostęp do jej danych i operacji możliwy jest tylko poprzez jedną składową programu. W przypadku MODULI wyróżnione zostały moduły (program w MODULI przypomina nieco blokową strukturę ALGOLU 60 i może składać się z zagnieżdżonych „modułów") oraz procesy opisujące sterowanie urządzeniami fizycznymi („device module" i „device process").

[1]) Przez proces należy rozumieć ogólne ciągi czynności zmierzających do osiągnięcia zaplanowanego celu (obliczenia, drukowania)

[2]) Dla zilustrowania tej sytuacji podaje się czasem przykład dwóch robotników korzystających ze wspólnego narzędzia np. śrubokręta. Z kolei współbieżność dla maszyny jednoprocesorowej można zobrazować podając jako przykład kucharza przygotowującego jednocześnie kilka dań



Rysunek ilustruje przykładowe powiązanie modułów w PASCALU Współbieżnym: P oznacza proces, M — monitor, C — klasę.

Prawa dostępu do zmiennych i procedur modułu zostaną scharakteryzowane w dalszej części artykułu.

ROZDZIAŁ IDEI OD JEJ REALIZACJI ORAZ „ZASŁANIANIE" INFORMACJI

W pierwszej części artykułu mówiliśmy o oddzieleniu informacji o tym, co program robi (ma robić), od informacji o tym, jak program to realizuje (ma realizować), a także o znaczeniu tego oddzielenia dla procesu projektowania. Mając na względzie znaczenie tego faktu dla użytkownika projektowanego lub gotowego już systemu, można powiedzieć o oddzieleniu idei (abstrakcji) od jej realizacji (implementacji). W MODULI i C-PASCALU osiągnięte to zostało przez sam podział na moduły (za moduły można tu uważać procesy, monitory i klasy w C-PASCALU oraz różnego rodzaju „moduły" w MODULI) i informacje o tym, co one zawierają, oraz prawa dostępu do odpowiednich struktur danych czy operacji. W MODULI są one uzupełnione przez dodatkowe „zasłanianie". W module część zmiennych można zadeklarować jako „importowane" (deklaracja use), tzn. zadeklarowane na zewnątrz modułu, a dostępne wewnątrz modułu, część zaś jako „eksportowane" (deklaracja define), tzn. zdeklarowane wewnątrz i dostępne na zewnątrz modułu (patrz przykład 2).

Przykład 2

```
.. <(dekl. a >...
module M 1;
define b, c;
use a;
< dekl. d >
   module M 2;
   define c, e;
   use d;
   < dekl. c, e, f >
   ...dostępne c, d, e, f
   end M2;
   procedure b;
   < dekl. f >
   ...dostępne a, b, c, d, e, f
   end b;
...dostępne a, b, c, d, e
end M1 ;
```

Przykład powyższy ilustruje „zasłanianie" zmiennych w MODULI. Jak widać, moduły mogą być zagnieżdżone tekstowo, podobnie jak bloki w ALGOLU 60. Prawa dostępu są wyznaczone przez deklaracje define i use.

Podobna idea znalazła zastosowanie w języku DDN systemu DREAM [9], gdzie dostępem do informacji sterują fragmenty programu nazywane przez autorów DREAMu „portami" wejściowym (typu in) i wyjściowymi (typu out), zależnie od tego, czy informacje są „importowane czy „eksportowane".

Rozdział abstrakcji od implementacji bardziej jawnie został zastosowany w innych językach: MESIE [10], ALPHARDZIE [2, 3] i CLU [11]. W MESIE abstrakcję reprezentuje tzw. „definicja". Znajduje się w niej opis typów i procedur bez podania, jak się je realizuje. Implementacja znajduje się natomiast w „programie" dotyczącym tej definicji. Tak więc jedną abstrakcją można implementować na kilka sposobów.

Przykład 3

```
Abstraction: DEFINITION =
BEGIN
...
it: TYPE = ...; rt: TYPE = ...;
p: PROCEDURE;
p1: PROCEDURE [INTEGER];
p: PROCEDURE [it] RETURNS [rt];
..
END
Implementer: PROGRAM IMPLEMENTING Abstraction =
BEGIN
  OPEN Abstraction;
  x: INTEGER;
  ..
  p PUBLIC PROCEDURE = < kod dla p >;
  p1: PUBLIC PROCEDURE [1: INTEGER] =
  < kod dla p1 >
  ...
  pi: PUBLIC PROCEDURE [x:it] RETURNS [y:rt] =
  < kod dla pi>
  ...
  END
```

Przykład powyższy ilustruje abstrakcję i implementację w języku MESA. Abstrakcja o nazwie „Abstraction" zawiera opis typów oraz listę dostępnych procedur, „implementer", program implementujący tę abstrakcję, kod każdej z procedur. Słowo „PUBLIC" określa status procedury (stopień jej dostępności), x jest zmienną lokalną. Dostęp do procedur jest regulowany w MESIE przez słowa kluczowe przed słowem „PROCEDURE" w implementacji: „PUBLIC", „PRIVATE" lub „READ ONLY".

Zauważmy (patrz przykład 3), że koncepcja modułu w MESIE jest taka sama jak ta, którą przedstawiono w pierwszej części artykułu.

Opis tego, co program robi, można przedstawić w postaci zestawienia warunków, które muszą być spełnione przed jego wykonaniem, z warunkami po jego wykonaniu. Tego typu warunki zapisuje się zwykle w postaci wyrażeń logicznych.

W ALPHARDZIE rozdział na abstrakcję i implementację dokonany jest wewnątrz każdego modułu, zwanego „formą". Forma składa się z trzech części: specyfikacji, reprezentacji i implementacji. W specyfikacjach opisuje się ogólnie „abstrakcyjną" strukturę danych i operacje, podobnie jak w definicjach MESY, lecz z dodatkowymi założeniami dotyczącymi inicjalizacji oraz wyrażeniami logicznymi związanymi z funkcjami (operacjami) formy (warunki przed i po wykonaniu każdej operacji). Jest to jedyne źródło informacji o module przeznaczone dla użytkownika. W części reprezentacji mieści się opis „konkretnej" struktury danych, reprezentującej abstrakcję, w części implementacji znajdzie się zestaw instrukcji dla funkcji (operacji) wymienionych w specyfikacjach. Przykład 4 ilustruje formę opisu stosu.

Przykład 4

```
form stos (n: integer)
beginform
  specifications
  ...
  function
  ..
  usuńelementzestosu (s: stos) pre 0 < length (s) ⩽ n
                               post s = leader (s'),
  ...
  representation
  ilel: integer init ilel ← 0;
  ...
  states
  mit when ilel = 0,
  normalny when 0 < ilel < n,
  pełny when sp = n,
  błąd otherwise
  implementation
  ...
  body usuńelementzestosu out (s.ilel. = s.ilel — 1) =
      normalny, pełny: s.ilel. ← s.ilel — 1;
      otherwise: BŁĄD;
  ...
  endform;
```

Jest to uproszczony przykład modułu opisującego stos i operacje na stosie w języku ALPHARD. Moduł ten podzielono na trzy części poprzedzone słowami kluczowymi „specifications", „representation" i „implementation". Informacje istotne dla użytkownika znajdują się w nagłówku i części pierwszej, gdzie są m.in. wymienione operacje na stosie i wyrażenia logiczne opisujące efekty ich wykonania — *n* wskazuje ile elementów może zmieścić się na stosie, length (stos) jest funkcją przekazującą informacje o bieżącym zapełnieniu stosu, *(s')* oznacza stos s przed operacją, leader *(s)* jest funkcją symbolizującą stos bez ostatniego elementu. Część druga zawiera opis reprezentacji stosu i jego stanów, ilel jest zmienną lokalną pokazującą liczbę elementów stosu, początkowa jej wartość (inicjalizacja!) jest równa zeru. W części trzeciej znajdują się ciała procedur wymienionych w specyfikacjach. Sposób ich wykonania zależy od stanu stosu. Formy w ALPHARDZIE można rozszerzać, dodając do nich nowe operacje, nie „niszcząc" jednocześnie form starych (podobnie jak to się dzieje w SIMULI 67 przy prefiksowaniu klas [1]).

W CLU moduły programu składają się z zestawu procedur, dostępnych z zewnątrz, operujących na wewnętrznej, niedostępnej dla użytkownika, strukturze danych. Moduły te nazywają się „wiązkami" (ang. *clusters*). Moduły tworzą bibliotekę CLU, a każda z abstrakcji posiada w niej swój opis. Można je wykorzystywać, a także powiększać bibliotekę.

Przykład 5

```
wordbag = cluster is
  create,
  insert,
  print ;
rep = record [contents: wordtree, total: int];
create = proc () returns (cvt);
  return (rep $ {contents: wordtree $ create (), total: 0});
  end create;
insert = proc (x: cvt, v: string);
  x.contents: x wordtree $ insert (x.contents, v);
  x. total: = x. total + 1;
  end insert;
print = proc (x: cvt, o: outstream);
  wordtree $ print (x.contents, x. total, o);
  end print;
end wordbag;
```

Przykład powyższy dotyczy modułu w CLU. Najpierw wymieniona jest jego nazwa, dostępne operacje i reprezentacja (rekord), a następnie opis procedur „wordtree" jest nazwą innego modułu, dostęp do jego operacji uzyskuje się przy użyciu znaku „$".

BADANIE POPRAWNOŚCI PROGRAMU

Języki modularne ułatwiają również badanie poprawności programu. Można to powiedzieć przede wszystkim o ALPHARDZIE, który powstał m.in. w tym celu.

Jedną z metod badania poprawności programu jest metoda Hoare'a (patrz np. [14]). Przypuśćmy, że mamy daną operację P, której poprawność chcemy udowodnić. W znacznym uproszczeniu, można powiedzieć, że dowód ten będzie polegać na sformułowaniu warunków wstępnego i ostatecznego tej operacji (przed i po jej wykonaniu — jak o tym napisaliśmy wcześniej) w postaci logicznej, a następnie, mając dane odpowiednie formuły oraz reguły dla każdego typu instrukcji tej operacji, na stwierdzeniu, że jeśli spełniony jest warunek wstępny na początku wykonania operacji P (prawdziwe jest opisujące je wyrażenie logiczne), to spełniony będzie również warunek ostateczny po jej zakończeniu.

Jak już został wspomniany w ALPHARDZIE, w części specyfikacji modułu, formułuje się tego typu warunki. Dotyczą one nie tylko każdej z operacji, ale również np. założeń związanych z parametrami modułu. Warunki także formułowane są w innych częściach modułu. Pozwala to na dokonanie rachunku logicznego i sprawdzenie, czy implementacja specyfikacji jest zgodna z oczekiwaniami i w związku z tym, czy moduł jako całość napisany jest poprawnie.

Poprawność formy (modułu ALPHARDU) bada się w czterech krokach: najpierw dla każdej formy, poprawność reprezentacji (1) oraz inicjacji (2) — a następnie dla każdej funkcji (operacji) z osobna — sprawdzenie jej poprawności — (3) oraz związku między konkretnymi i abstrakcyjnymi specyfikacjami logicznymi (4).

Chociaż mechanizmy tego typu wbudowane są przede wszystkim w ALPHARDZIE, dobrze przeprowadzona modularyzacja, niewątpliwie ułatwia badanie poprawności i to niekoniecznie zaprezentowaną wyżej metodą. Poza tym w niektórych systemach oprogramowania, jeśli moduły wyprowadzane są i programowane według sformalizowanych reguł, często już samo to rozwiązanie powoduje zmniejszenie liczby możliwych błędów. Błędy w programach są jednak złożonym zagadnieniem i nie zawsze łatwo jest w sposób jednoznaczny podać receptę jak ich uniknąć.

Przy okazji można zasygnalizować problem rozłożenia odpowiedzialności za błędy w modułach. W złożonych systemach o wielu modułach nie zawsze jest rzeczą prostą zdecydować, w których modułach umieścić odpowiednie testy oraz jak zsynchronizować reakcje na błąd.

Dodatkowo chodzi o to, by testy nie zajmowały zbyt dużo czasu. W związku z tym często dzieli się je na takie, które muszą być wykonywane zarówno w trakcie tworzenia jak i eksploatacji oprogramowania i takie, które są niezbędne jedynie w trakcie tworzenia systemu, a następnie zostają usunięte.

Implementacja powiązań modułów jest często realizowana za pomocą dodatkowych tablic. W MESIE istnieje tzw. „mechanizm wiążący" (ang. *binder*), ustalający związki między modułami na podstawie „ścieżek wiązania" (patrz część pierwsza artykułu), przedstawiających kolejność wykonywania modułów. W CLU istnieje tzw. „lista stowarzyszenia (ang. *„association list"*), przyporządkowująca (podczas kompilacji) nazwom „zewnętrznym" występującym w module jednostki opisowe i dołączona do biblioteki jako część tego modułu.

Porównanie własności opisanych języków modularnych zawiera tabela.

Porównanie niektórych cech wybranych języków modularnych

	PASCAL współbieżny	MODULA	MESA	ALPHARD	CLU
Autorzy	P. Brinch Hansen	N. Wirth	C. M. Geschke, J. H. Morris Jr., E. H. Satterthwaite	M. Shaw, W. A. Wulf, R. L. London	B. Liskov, A. Snyder, R. Atkinson, C. Schoffert
Główny cel powstania (poza samym wprowadzeniem modularyzacji)	Realizacja współbieżności dla opisu systemów operacyjnych zwłaszcza małych komputerów	Połączenie w jednym języku programowania sekwencyjnego, równoległego i czasu rzeczywistego dla strukturalizacji opisu systemów operacyjnych i działania sprzętu informatycznego	Oddzielenie abstrakcji od implementacji	Ułatwienie weryfikacji programów, oddzielenie abstrakcji od implementacji	Oddzielenie abstrakcji od implementacji
Wykorzystane języki	PASCAL, SIMULA 67	PASCAL współbieżny, SIMULA 67, PASCAL	PASCAL, ALGOL 68, SIMULA 67	SIMULA 67 PASCAL, IPL-V, LISP	SIMULA 67, LISP, PASCAL
Postać modułu	Procesy, monitory, klasy	Moduły proste, interface module, device module; można też używać procesów i zwykłych procedur	Definicje, programy	Formy dzielące się na części specyfikacji, reprezentacji i implementacji	Abstrakcje danych (*cluster*), procedur (*procedure*), sterowania (*iterator*), cluster może zawierać procedury i iteratory
„Zasłanianie" informacji (oprócz reguł dostępu do określonych typów)	Tylko przez reguły dostępu do określonych typów modułu	Zmienne i procedury eksportowane (define) i importowane (use), reguły struktury blokowej	Słowa kluczowe definiujące sposób korzystania z operacji: PUBLIC PRIVATE, READ-ONLY	Zmienne „wewnętrzne" — w części niedostępnej „z zewnątrz", możliwość rozszerzania form i dodawania nowych operacji i zmiennych	Tylko przez reguły dostępu
Inicjalizacja modułu (podstawienie wartości początkowych i wykonanie pierwszych operacji)	W ciele monitora i klasy (instrukcje modułu)	Instrukcje modułu	Wewnątrz implementacji (programu)	Wewnątrz modułu specjalne wskaźniki inicjalizacji	Podstawienie przy deklarowaniu; poza tym operacje poprzez procedury w wiązkach
Powiązanie modułów	Sterowanie w „programie głównym"; ograniczona struktura blokowa	Moduł „globalny" zawiera pozostałe; struktura blokowa powiązana z „zasłanianiem" informacji	Wewnętrzna reprezentacja poprzez mechanizm wiążący i ścieżki powiązań	Moduły można traktować jako typy i deklarować zmienne tych typów	„Lista stowarzyszenia", tworzona w czasie kompilacji
Możliwość tworzenia egzemplarzy modułów	Ewentualnie poprzez typy	W zasadzie brak, ewentualnie (nieco sztucznie, poprzez typy	Podobnie jak w SIMULI 67	Poprzez zmienne typów „modułowych"	Nazwy zmienych traktowane są jak referencje
Możliwość programowania współbieżności	Tak: procesy, operacje „delay" i "continue" na kolejkach	Tak: procesy, operacje "send" i "wait" na sygnałach	Nie	Nie	Nie
Iteratory (uogólnienie pętli)	Cycle-end, ciąg instrukcji wykonywanych cyklicznie	1. typu while 2. typu repeat 3. typu loop: loop S1 when B1 do X1 exit S2 when B2 do X2 exit ... Sn when Bn do Xn exit S end	1. typu REPEAT 2. typu FOR: FOR (zm. kontrolna) IN (zbiór) DO... ENDLOOP	1. for x: gen (y) while B (x) do ST (x, y, z) 2. first x:gen (y) suchthat β (x) then S_1 (x, y, z) else S_2 (y, z) gen (y) jest generatorem wartości zmiennej kontrolnej i definiowany jest jako forma	for (zm. kontr.) in (wywołanie generatora) do ... end generator deklaruje się osobno
Weryfikacja (oprócz ułatwień wynikających z samego podziału programu i rozdziału abstrakcji od implementacji)	Brak specjalnych mechanizmów	Brak specjalnych mechanizmów, możliwe użycie metody Hoare'a	Dokładne sprawdzanie poprawności typów	Za pomocą specjalnie umieszczonych w odpowiednich miejscach każdego z modułów predykatów; weryfikacja przeprowadzana jest w czterech krokach, przy czym dwa dotyczą każdego modułu, a dwa każdej z operacji (funkcji) modułu	Brak specjalnych mechanizmów

Trudno obecnie określić, jak w inżynierii oprogramowania będzie rozwijało się dalej podejście modułowe. Istnienie dużych systemów wykorzystujących to podejście jest już faktem. Wspomnieliśmy np. o DREAMie, który umożliwia modelowanie i symulację systemu w trakcie jego projektowania, istnieje też np. dość sformalizowany system HOS [15], używany z powodzeniem przy realizacji amerykańskich lotów kosmicznych i wiele innych. Przypomnijmy główne zalety podejścia modułowego:

- pozwala lepiej projektować systemy oprogramowania (łatwiejszy podział pracy i bardziej skuteczne zarządzanie realizacją)
- pozwala lepiej zrozumieć ideę systemu oraz łatwiej sporządzać dokumentację.
- pozwala łatwiej modyfikować oprogramowanie (ograniczenie zmiany często do jednego tylko modułu, łatwość wymiany i dołączania nowych modułów)
- dzięki oddzieleniu abstrakcji od implementacji pozwala użytkownikowi systemu skoncentrować się na idei problemu ignorując podczas realizacji jego wewnętrzną „maszynową" złożoność
  — umożliwia lepszą strukturalizację systemu (eliminację jawnych skoków, uogólnienie i teratorów)
  — zwiększa możliwość badania poprawności programu i stopień niezawodności systemu
  — umożliwia opisywanie współbieżności.

Na tle powyższych zalet bledną podnoszone czasem zarzuty, dotyczące nie zawsze optymalnych czasów wykonania programów czy pewnych problemów „technicznych", jak np. wspomniane zagadnienie rozłożenia odpowiedzialności za błędy.

Zauważmy że zainteresowanie modularyzacją wiąże się z wykorzystaniem języków „uniwersyteckich", mających w miarę dobry opis formalny, takich jak SIMULA 67, LISP czy PASCAL, a z odejście od języków typu FORTRAN. Być może oznacza to wyższy poziom programowania i jego metodologii.

LITERATURA

[1] Dahl O-I, Myhrhang B., Nygaard K.: The SIMULA 67 — common base language. Oslo, Norvegian Computing Centre, 1967

[2] Brinch Hansen P.: Concurrent Pascal: A programming language for operating system design. Cornegie Mellon Unievrsity, IST Rep. No 10, April 1974

[3] Brinch Hansen P.: The programing language Concurrent Pascal. IEEE Trans. on Soft. Eng., June 1975

[4] Brinch Hansen P.: Experience with modular Concurrent Pascal. IEEE Trans. on Soft. Eng., March 1977

[5] Wirth N.: Modula: A language for modular multiprogramming. Institut für Informatik ETH, Rep. Nr 18, 1976

[6] Wirth N.: The use of Modula. Design and implementation of Modula. Institut für Informatik ETH, Rep. Nr 19, 1976

[7] Wirth N.: Toward a discipline of real-time programming. CACM, Aug. 1977

[8] Hoare C. A. R.: Monitors: An operating system structuring concept. CACM, Aug. 1977

[9] Riddle W. E. et al.: Behavior modeling during software design. IEEE Trans. on Soft. Eng., July 1978

[10] Geschke C. M. et al.: Early experiece with MESA. CACM. Aug. 1977

[11] Liskov B. et al.: Abstraction mechanisms in CLU. CACM, Aug. 1977

[12] Wulf W. A. et al.: An introduction to the construction and verification of Alphard programs. IEEE Trans. on Soft. Eng., Dec. 1977

[13] Shaw M. et al.: Abstraction and verification in Alphard: Defining and specifying iteration and generators. CACM, Aug. 1977

[14] Turski W. M.: Metodologia programowania. WNT 1978

[15] Hamilton M., Zeldin S.: Higher Order Software — A methodology for defining software. IEEE Trans. on Soft. Eng., March 1976

Brytyjskie centrum badań symulacyjnych

Symulacja komputerowa jest dogodną techniką rozwiązywania wielu złożonych problemów, jakie pojawiają się podczas projektowania rzeczywistych, zwłaszcza dużych systemów. Projektant konstruuje model matematyczny systemu i testuje go na maszynie matematycznej celem zbadania jego własności. Aby pomóc projektantom przy tworzeniu modeli, w Lancaster University (Wielka Brytania) uruchomiono centrum symulacyjne — centralny ośrodek szkolenia i badań symulacyjnych.

Centrum zostało założone w 1977 roku w Department of Operational Research na Uniwersytecie w Lancaster, znanym z praktycznego podejścia do badań operacyjnych. Dyrektorem Centrum jest profesor M. G. Simpson, jednocześnie przewodniczący Operational Research Society; zastępca dyrektora — pionier badań symulacyjnych, profesor K. D. Tocher [1]) jest doradcą w sprawach prac badawczych.

Badania mają być prowadzone w ścisłej współpracy z użytkownikami metod symulacyjnych w przemyśle i zarządzaniu. Symulacja bowiem jest tylko techniką służącą rozwiązywaniu konkretnych problemów, które według naukowców najlepiej może określić sam użytkownik. Nowe rozwiązania techniczne często pochodzą właśnie od pracowników przemysłu. Należy spodziewać się, że pracownicy ci zostaną członkami-korespondentami centrum i będą przyczyniać się do jego rozwoju, a także brać udział w jego działalności szkoleniowej i badawczej.

Zadania centrum

Planowane są dwa kierunki działalności centrum: 1) rozwój i wdrażanie niezbędnego oprogramowania oraz 2) poszukiwanie nowych rozwiązań metodologicznych. Oczywiście kierunki te są wzajemnie powiązane i osiągnięcia w dziedzinie oprogramowania będą wykorzystywane w pracy nad metodologią.

Najnowszym osiągnięciem centrum jest język LORBS (**L**anguage for **O**perational **R**esearch in **B**ritish **S**teel) ułatwiający symulację lub tworzenie specjalistycznego oprogramowania. Jest to bardzo wygodne narzędzie tworzenia systemów, usprawniające proces projektowania i wykorzystujące koncepcje zawarte w różnych językach programowania.

[1]) Prof. K. D. Tocher, wraz z D. G. Owenem, opublikował w 1958 roku pierwszy artykuł z dziedziny symulacji. Obecnie jest również koordynatorem w dziale organizacji zarządzania British Steel Corporation

Badania metodologiczne

Pewien zakres badań związany jest z wykorzystaniem idei symulacji komórkowej. W metodzie tej dzieli się system na części, z których każda może być traktowana jako podsystem dający się symulować oddzielnie. Podział ten może być dokonany tak, że komunikacja między poszczególnymi częściami będzie realizowana przez synchroniczne meldunki, a symulację będzie można przeprowadzać w prosty, standardowy sposób. Tak więc każda komórka systemu może być rozpatrywana jako program symulacyjny zbudowany na swoich własnych zasadach. Oczywiście programy symulacyjne podsystemów mogą być łączone w model pełnego systemu. Przy współpracy z przemysłem, realizację tej idei można przetestować w praktyce.

Inną korzyścią z zastosowania komórkowej struktury modelu, jest zmniejszenie czasu wykonywania programów symulacyjnych. Nie wszystkie części programu symulacyjnego muszą być ponownie wykonane po zmianie jednego parametru. Wystarczy wznowić symulację od miejsca pierwszego uaktywnienia zmodyfikowanej komórki, oczywiście pod warunkiem, że zostały zapamiętane informacje z poprzedniego przebiegu.

Efektywność symulacji zależy w znacznym stopniu od procedur wyszukiwania. Znane są różne metody realizacji takiego wyszukiwania, jednak nigdy nie były one systematycznie przebadane. Centrum w Lancaster zamierza dokonać szczegółowej analizy i porównać zalety tych metod.

Podczas tworzenia wielkich modeli służących do rozwiązywania wielu różnych problemów nie ma konieczności zachowania tego samego poziomu szczegółowości we wszystkich modułach. Pożądane jest konstruowanie zestawów alternatywnych modułów o różnym poziomie szczegółowości i dobieranie ich w zależności od aktualnie badanego zagadnienia. Centrum będzie próbowało znaleźć metody i określić zasady konstruowania modułów szczególnie dogodnych do przetwarzania na maszynie.

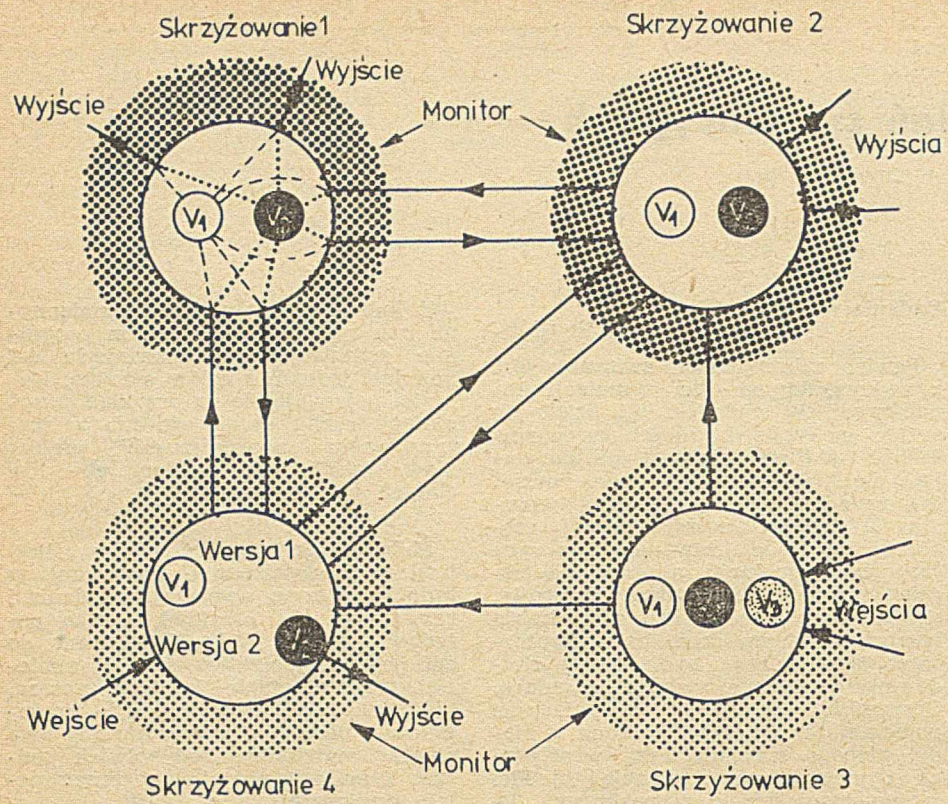
Badania w dziedzinie oprogramowania

W symulacji pewne zmienne mogą być traktowane jako zmienne zmieniające swą wartość (reprezentującą stany rzeczywistych zjawisk) tylko w specyficznych momentach, takich jak zakończenie procesu. Inne lepiej oddają rzeczywistość — jeśli można przyjąć, że zmieniają swą wartość w sposób ciągły. Powszechnie stosowane są inne metody i języki dla symulacji tych dwu typów zmiennych. Możliwe jest jednak posługiwanie się zmiennymi obu typów w tym samym programie symulacyjnym. Rozwinięcie tej idei jest jednym z celów ośrodka. Korzyści pojawiają się na przykład w symulacji ruchu drogowego lub w rozwiązywaniu „wewnętrznych" problemów, takich jak problem obiektów stałych (materialnych).

Metody weryfikacji wyników symulacji są niedostatecznie opanowane, w związku z czym wymagają zbadania i rozwinięcia. Na przykład, bardzo pożądana byłaby zdolność systemu do definiowania obiektów stałych (materialnych) i do takiej kontroli procesu, która zapewni wyprowadzenie komunikatu o błędzie w przypadku gdy dwa obiekty materialne kiedykolwiek podczas symulacji przeniknęły przez siebie wzajemnie.

W symulacji dotyczącej przemieszczania się ciężkich pojazdów, takich jak pociągi czy dźwigi samojezdne, może zaistnieć sytuacja, w której po sprawdzeniu, że trasa jest wolna, zostanie zainicjowany ruch dwu pojazdów i zrealizowane całkowite ich przemieszczenie. W efekcie otrzymamy pozornie dobre wyniki, chociaż w rzeczywistości pojazdy te musiałyby przeniknąć się wzajemnie. Procedury zapewniające unikanie takich błędów mogą być zaprogramowane oraz na stałe umiejscowione w systemie. Zwolniłoby to przyszłych użytkowników od uciążliwej pracy.

Dalszy rozwój symulacji komórkowej pozwoliłby na konstruowanie modelu systemu poprzez składanie go z modeli elementów systemu. Model każdego elementu symulowanego systemu byłby wybierany w zależności od okoliczności, spośród alternatywnych modeli tego elementu. Wybieranie to mogłoby być dynamiczne, umożliwiając wymianę elementów w ramach jednego przebiegu symulacji. Do realizacji takiej metody byłby potrzebny specjalny program nadzorczy, zawierający procedury określające momenty w których należy wymieniać modele. Modele elementów byłyby przechowywane w bibliotece i pobierane z niej przez program nadzorczy ilekroć zaszłaby konieczność wymiany. Stosując metodę symulacji komórkowej u wyżej opisanych rozwiązań, możliwe jest stworzenie systemu przeznaczonego do stymulacji ruchu ulicznego, w którym wielka sieć byłaby konstruowana z elementów biblioteki modeli skrzyżowań. Można by konstruować alternatywne formy modeli dla konkretnego skrzyżowania, od modeli szczegółowych, zbudowanych na zasadzie metody kolejnych zdarzeń i metody stałego kroku, do prostych modeli przepływowych, które są względnie oszczędne jeśli chodzi o czas przetwarzania.



Przetwarzanie modelu bardziej szczegółowego niż jest to potrzebne dla przeprowadzenia badań, może spowodować ogromne skomplikowanie programu symulacyjnego, zwłaszcza gdy system symulowany jest duży. Przedstawiony na rysunku prosty model zaprojektowany został w celu zbadania sieci ruchu drogowego. Alternatywne podmodele skrzyżowań (V_1, V_2, V_3) zbudowane są na różnych stopniach szczegółowości. Linie ze strzałkami reprezentują przepływy informacji o przemieszczaniu się pojazdów. Na podstawie tych informacji następuje wprowadzanie i wyprowadzanie danych. Oddzielny monitor danych dla każdego skrzyżowania określa, który podmodel powinien być użyty w celu przyjęcia i przetworzenia danych aktualnie dopływających do danego skrzyżowania. Pozwala to uniknąć przetwarzania bardziej szczegółowych modeli niż jest to w danej chwili konieczne.

Prostsze modele byłyby wykorzystywane jedynie w niewielu momentach i często wymagałyby wymiany w ramach przebiegu programu. Jest to jedno z zastosowań koncepcji biblioteki i programu nadzorczego.

Często warto rozwijać istniejące systemy specjalizowane w celu rozwiązywania problemów typowych. Na przykład pracownicy National Coal Board (Zarząd Kopalń Państwowych) uzyskali doskonałe wyniki w symulacji systemów transportu węgla podziemnym przenośnikiem taśmowym.

Znacznie częściej jednak mamy do czynienia z systemami o strukturze hierarchicznej. Wszystkie poziomy systemu hierarchicznego mogą być modelowane oddzielnie i łączone w pełny model. Jako przykład może służyć system do symulowania awarii urządzeń w zakładach British Steel.

Współpraca z pracownikami przemysłu jest w tych przypadkach niezbędna.

Dalsze plany badawcze

Wyniki pierwszych lat działalności centrum wpłyną oczywiście na przyszłe kierunki badań. Część prac znajdujących się obecnie w stadium projektów będzie kontynuowana.

Gdy powstanie sieć ośrodków komputerowych współpracujących z centrum, symulacja ruchu ulicznego, wykorzystująca bibliotekę modeli oraz program nadzorczy, powinna przynieść nowatorskie rozwiązania. Zakłada się, że centrum będzie dostarczać narzędzi dla ośrodków wielu instytucji umożliwiając im korzystanie ze wspólnych osiągnięć a także wnoszenie swojego wkładu w jego rozwój. Nie odnosi się to, naturalnie, jedynie do systemów ruchu ulicznego.

Struktura symulacji komórkowej narzuca naturalny sposób wykorzystania komputerów wieloprocesorowych.

W symulacji na takich komputerach każda „komórka" mogłaby mieć własny zegar, który nie musiałby być przez cały czas synchroniczny z zegarami innych komórek.

Bardzo interesującym zagadnieniem jest automatyczne pisanie programów w oparciu o kwestionariusz albo konwersacyjny tryb pracy. Dr. A. Chementson z Birmingham University stworzył imponujący system wspomagający programowanie w języku ECSL. Jak dotychczas nie wszystkie rozwiązania, przydatne w symulacji systemów, mogą być generowane za pomocą tej metody, ale jest to obiecujący kierunek badań eksperymentalnych i teoretycznych.

Jedną z cech szczególnych symulacji jest istniejąca obecnie wielka różnorodność języków, dająca początkującemu programiście większy wybór niż jest to rzeczywiście potrzebne, ponieważ wiele z pozornych różnic dotyczy bardziej formy niż treści. Zaletą wspomnianego już systemu LORBS jest jego uniwersalność osiągnięta dzięki uwzględnieniu struktur charakterystycznych dla różnych języków.

Należy stwierdzić, że symulacja jest metodą szeroko stosowaną nie tylko przez teoretyków badań operacyjnych, ale i w takich dziedzinach, jak technika, biologia, fizyka jądrowa, statystyka, wojskowość. Osiągnięcia w tych dziedzinach mogą być oczywiście wykorzystane we wspomnianych badaniach operacyjnych.

Opracował Krzysztof KAMIŃSKI na podstawie „A centre for simulation", SPECTRUM No 7/78

Informatyka w procesach zarządzania produkcją

Pod egidą Oddziału Wojewódzkiego NOT w Wałbrzychu i Zakładu Elektronicznej Techniki Obliczeniowej w Świdnicy, w dniach 4 i 5 kwietnia br. na Zamku w Książu, odbyło się sympozjum naukowe pod hasłem „Zastosowanie minikomputerów i teletransmisji w procesie zarządzania produkcją przemysłową". Obrady, w których udział wzięło blisko dwustu uczestników z całej Polski, otworzył w imieniu organizatorów wiceprezes Oddziału NOT w Wałbrzychu, mgr Jerzy Kubica.

Omawianie walorów imprezy należy rozpocząć od podkreślenia trafnego doboru tematów, a co ważniejsze dojrzałej metodyki całego sympozjum. Wygłoszono 16 referatów. Inauguracyjny wykład dr inż. H. Pietrowskiego z warszawskiego IOPM-u stanowił osnowę, na którą nakładały się tematy o wysokim stopniu szczegółowości. Referat wprowadzający traktował o „Zintegrowanym systemie informatycznym sterowania procesami i zarządzania przedsiębiorstwami przemysłu maszynowego" i jedną z podniesionych w nim kwestii była pozytywna ocena przetwarzania rozproszonego w przedsiębiorstwach o wielopoziomowej sieci hierarchicznej. Kwestię tę rozwijał referat następny: „Systemy rozproszone — nowy model zastosowania minikomputerów" — autorstwa mgr Teresy Lubińskiej i mgr. Jerzego Marcinkiewicza. Uzupełnieniem jego był referat mgr. Ryszarda Plichty „Kierunki i formy zastosowań minikomputerów".

Drugim walorem jest coraz bardziej upowszechniająca się na konferencjach i sympozjach koncentracja na zagadnieniach związanych ze stosowaniem popularnego sprzętu, a więc urządzeń seryjnych dostępnych na krajowym rynku, najczęściej produkowanych w ramach Jednolitego Systemu — ale nie tylko.

Trzecim — jest zaproszenie do wygłoszenia referatów przedstawicieli producentów sprzętu. Tak np. o systemie monitorów ekranowych dla komputerów R-32 mówiła przedstawicielka MERA-ELZAB z Zabrza mgr Anna Piłko.

Natomiast zastrzeżenie budzi bierna rola wyznaczona przez organizatorów uczestnikom sympozjum — na dyskusję poświęcono zaledwie 30 minut. Wprawdzie wielokrotnie sceptycznie ocenialiśmy przebieg różnych czasochłonnych dyskusji, które często tracą z pola widzenia główny temat sympozjów, schodząc na manowce utyskiwań, ale można było się spodziewać, że tak dobrze uzupełniający się blok referatów spowoduje jednak jakąś konstruktywną dyskusję. Inna rzecz, że w komnatach zamkowych tego dnia było szczególnie zimno...

Poprzez oceny i opinie referentów sympozjum dowiodło, że sprzęt, którym obecnie dysponujemy, nie jest ani dobry ani zły. Na tyle dobry, aby można było serio i powszechnie zabrać się do wprowadzania informatyki w procesy zarządzania produkcją przemysłową, na tyle słaby (lub jeszcze trudno dostępny), aby to wprowadzanie przebiegało w tempie odpowiadającym aktualnym potrzebom. Tu również sporo zależy przede wszystkim od kadry.

(K.B.)

II Fabryczny Rocznik Organizacji i Informatyki FSM

W 1978 r. nastąpił dalszy intensywny, rozwój produkcji i efektywności gospodarowania w FSM POLMO w Bielsku Białej oraz w zakładach w Tychach i Skoczowie. M. in. Fabryka osiągnęła przed zaplanowanym terminem docelową zdolność produkcyjną, znacznie zwiększyła wielkość eksportu samochodów i dostawy na rynek krajowy. W roku tym miały miejsce dwa wielkie wydarzenia: wyprodukowanie 500 000 małych fiatów i 1 000 000 zespołów napędowych.

W osiągnięciach produkcyjnych i ekonomicznych FSM sporo zasług mają również informatycy i organizatorzy zatrudnieni w trzech zakładowych ośrodkach komputerowych, w przyzakładowych komórkach organizacji i informatyki oraz w Centralnym (Fabrycznym) Ośrodku Organizacji i Informatyki FSM „POLMO" w Bielsku Białej.

Przede wszystkim należy odnotować dalsze rozszerzenie obszaru zastosowań systemów informatycznych na nowe dziedziny zarządzania oraz opanowanie trudnej techniki związanej z monitorowym systemem zarządzania w zakładach montażowych w Bielsku-Białej i Tychach oraz komputeryzację stanowisk badawczych w Bielsku (Ośrodku Badawczo-Rozwojowym FSM) przy pracach konstrukcyjnych nowych wersji i modeli samochodów oraz badaniach jakościowych trwałości i niezawodności poszczególnych podzespołów, części, detali.

W roku 1978 w FSM-owskich ośrodkach organizacji i informatyki (fabrycznym i zakładowych):

— opracowano i uruchomiono 248 nowych programów dla 13 różnych agend i podsystemów informatycznych

— zaprojektowano i wdrożono 137 nowych tabulogramów

— opracowano i wdrożono kilka nowych podsystemów informatycznych, np. OSIRIS (analiza badań społecznych i socjologicznych), GONAR (gospodarka narzędziowa), IMPUR (analiza importu i nowych uruchomień), MOSTER (system obsługi teletransmisji i programów zastosowań — rozwiązanie własne FSM, pierwsze tego typu w Polsce), INFORMAT (system szybkiej informacji z wykorzystaniem monitorów ekranowych o materiałach występujących w magazynach FSM), UNIREK (uniwersalny system wyszukiwania informacji w dowolnym zbiorze), PROGAT (system wspomagania programowania z wykorzystaniem monitorów ekranowych), KARDAN (system ewidencji i analizy kart drogowych), SIKOM (system informacyjno-komunikacyjny z wykorzystaniem monitorów ekranowych), EKSPRESS (system poszukiwania zbiorów i wiele innych

— opracowano szczegółowy program kompleksowej automatyzacji zarządzania, sterowania produkcją i obliczeń naukowo-technicznych

— roszerzono też zakres i obszar zastosowań wdrożonych w latach ubiegłych podsystemów

— nastąpił dalszy rozwój bazy danych.

Wszystkie te interesujące informacje można znaleźć w „Fabrycznym Roczniku Organizacji i Informatyki FSM POLMO 1978/79". W publikacji tej scharakteryzowano również najważniejsze prace ośmiu zakładowych służb organizacji i informatyki dotyczące realizacji zadań w zakresie organizacji produkcji i organizacji zarządzania (procedury i instrukcje organizacyjne) oraz przedstawiono syntetyczny program ważniejszych przedsięwzięć tematycznych w dziedzinie organizacji i informatyki na rok 1979.

Liczne tabele, wykresy i diagramy zamieszczone w „Roczniku", mają nie tylko walory informacyjne, ale także wartość poglądową i porównawczą.

Autorami „Rocznika" są: dr Czesław Paczuła, mgr Marian Migdalski, mgr inż. Andrzej Wernerowski, mgr inż. Cecylia Wolamin, mgr Stanisław Rudowicz, mgr inż. Marian Wyczyszczewski, mgr Krystyna Janosz, mgr Konrad Janusz, mgr inż. Jacek Bargiełowski i mgr inż. Józef Chromik.

P.S. Skoro omawiana publikacja traktuje o bogatej i wielokierunkowej działalności Fabrycznego Ośrodka Organizacji i Informatyki FSM POLMO w Bielsku-Białej, trudno nie wspomnieć o najnowszym sukcesie Fabryki. Otóż na I Krajowych Targach Systemów Oprogramowania Maszyn Cyfrowych (Katowice, 26—31 marca br.) Ośrodkowi FSM przyznano 2 złote medale za systemy ASOM (gospodarka materiałowa) i KADRA (gospodarka kadrami) oraz medal złoty dla Kierownictwa Ośrodka Organizacji i Informatyki FSM za całokształt osiągnięć i wyróżniającą się ekspozycję na Targach. Takie wyróżnienie otrzymała FSM jako jedyne przedsiębiorstwo przemysłowe w kraju. Gratulujemy i obiecujemy przedstawić oba nagrodzone systemy w jednym z pobliższych numerów INFORMATYKI.

(E.K.)

ZETO

WCT/184/K/79

Socjalistyczne współzawodnictwo

Jak już informowaliśmy skrótowo w numerze majowym, w dniu 30 kwietnia br. w ZETO Łódź odbyła się uroczystość ogłoszenia wyników współzawodnictwa o tytuł najlepszego przedsiębiorstwa Zjednoczenia Informatyki w roku 1978. Laureatem pierwszego miejsca był właśnie ośrodek łódzki, którego kierownictwo postarało się nadać temu wydarzeniu w powiązaniu z tradycyjną akademią 1-Majową szczególnie uroczysty charakter.

W tym miejscu należą się Czytelnikowi najbardziej zwięzłe informacje na temat genezy oraz zasad tej niewątpliwie pierwszej w środowisku informatycznym akcji socjalistycznego współzawodnictwa pracy.

Jego inicjatorem był Zarząd Główny Związku Zawodowego Energetyków, którego propozycja oparta na Uchwale nr 49 Rady Ministrów z dnia 14 lutego 1974 roku w sprawie socjalistycznego współzawodnictwa pracy, spotkała się z pełnym poparciem kierownictwa ZI oraz Ministerstwa Nauki, Szkolnictwa Wyższego i Techniki. Warto również przypomnieć, że podstawowym celem współzawodnictwa jest pobudzenie aktywności społeczno-produkcyjnej kolektywów pracowniczych, wyrażającej się m.in. w świadomym i dobrowolnym podejmowaniu i konsekwentnej realizacji dodatkowych zadań przyspieszających nasz rozwój gospodarczy.

Działania organizacyjne w tym zakresie rozpoczęło w 1977 r. rozesłanie przez ZI wytycznych regulaminu współzawodnictwa wewnątrzzakładowego pracy do wszystkich ZETO. Bezpośrednio po tym przygotowany został regulamin współzawodnictwa międzyzakładowego, który rozesłano do tych samych adresatów na początku 1978 r. W roku tym wiele przedsiębiorstw miało już zorganizowane współzawodnictwo wewnątrzzakładowe, spełniając tym samym podstawowy warunek możliwości przystąpienia do współzawodnictwa międzyzakładowego. Warto podkreślić, że zorganizowanie współzawodnictwa wewnątrzzakładowego było zadaniem nielatwym z uwagi na specyfikę naszej branży, która utrudniała znalezienie właściwego wzorca do naśladowania i adaptacji, Oficjalne przystępowanie do współzawodnictwa odbywało się na Konferencjach Ekonomicznych poszczególnych przedsiębiorstw, a następnie na Konferencjach Samorządu Robotniczego z chwilą oficjalnego ich powołania w ZI.

Ostatecznie współzawodnictwo pracy, do którego przystąpienie jest aktem całkowicie dobrowolnym, zostało zorganizowane w 9 na łączną liczbę 14 przedsiębiorstw ZI.

Dla współzawodnictwa wewnątrzzakładowego przyjęto następujące podstawowe kryteria oceny:

- wydajność pracy
- jakości i terminowość wykonanych zleceń
- dyscyplina pracy
- podnoszenie przez pracowników kwalifikacji zawodowych
- postawa społeczna pracowników.

We współzawodnictwie międzyzakładowym dla oceny wyników działalności przedsiębiorstwa przyjęto natomiast następujące wskaźniki:

- wydajność pracy ogółem
- produktywność maszyn i urządzeń
- wykorzystanie czasu pracy komputerów
- wypadkowość
- liczba uzyskanych tytułów Brygady Socjalistycznej
- liczba aktywistów BHP
- poziom warunków socjalnych załogi
- liczba zobowiązań społecznych
- wydajność pracy projektantów i programistów
- liczba uznanych reklamacji.

Zgodnie ze wspomnianą Uchwałą nr 49, Dyrektor Naczelny ZI powołał przy Zjednoczeniu Komisję Współzawodnictwa Pracy, której podstawowymi zadaniami są inicjowanie i koordynacja współzawodnictwa w ramach całej sieci ZETO, a także ocena jego wyników oraz przedstawianie jednostkom nadrzędnym propozycji rozdziału nagród pomiędzy zwycięskie załogi.

Podsumowanie wyników pierwszego roku współzawodnictwa wyłoniło jako zdecydowanego zwycięzcę załogę ZETO Łódź, która tym samym zdobyła

Akt wręczenia sztandaru przez min. W. Kujawskiego

WCT/184/K/70

w przedsiębiorstwach ZETO

sztandar przechodni Ministra Nauki. Szkolnictwa Wyższego i Techniki oraz Zarządu Głównego Związku Zawodowego Energetyków, dyplom uznania oraz nagrodę pieniężną. Dyplomy uznania oraz nagrody pieniężne otrzymali leureaci drugiego (ZETO Poznań) i trzeciego miejsca (ZETO Kielce), natomiast laureaci czwartego (ZETO Jelenia Góra) oraz piątego miejsca (ZETO Katowice) musili zadowolić się jedynie dyplomami uznania.

Uroczystość ogłoszenia wyników współzawodnictwa miała wyjątkowo uroczysty charakter, akcentując nowy etap integracji załóg oraz wzrostu sprawności i efektywności usług świadczonych przez sieć ZETO. W uroczystości tej wzięli udział: wiceminister MNSzWiT doc. dr Walery Kujawski, dyrektor naczelny Zjednoczenia Informatyki mgr Zbigniew Substyk, Sekretarz Komitetu Łódzkiego PZPR tow. Jerzy Grabowski, Sekretarz Komitetu Dzielnicowego PZPR Łódź-Polesie tzw. Witold Kotras oraz Sekretarz Zarządu Głównego Związku Zawodowego Energetyków tow. Józef Żmijewski. Za stołem prezydialnym zasiedli również przedstawiciele gospodarzy uroczystości w osobach dyrektora naczelnego dr inż. Zygmunta Łuczaka, I Sekretarza POP PZPR tow. Grzegorza Marynowicza i zastępcy przewodniczącego Rady Zakładowej tow. Mariusza Młynarskiego. Niewielka sala konferencyjna ośrodka Łódź, w której dominował na ścianie aktualny cytat „Wszyscy jesteśmy odpowiedzialni za dalszy rozwój kraju", z trudem pomieściła licznych zaproszonych gości, reprezentujących instytucje zaprzyjaźnione i współpracujące z ośrodkiem gospodarzy.

Uroczystość zainaugurowało przemówienie dyr. Łuczaka, który w ramach okolicznościowego referatu 1-majowego w prostych bezpośrednich sformułowaniach scharakteryzował dotychczasowe osiągnięcia i najbliższe zamiary ośrodka. W kolejnym przemówieniu dyr. Substyk przedstawił szczegółowo oficjalne wyniki współzawodnictwa, po którym nastąpiło uroczyste wręczenie laureatom sztandaru i dyplomów przez min. W. Kujawskiego.

Laureaci współzawodnictwa w komplecie

Minister w zwięzłych słowach podkreślił fakt szczególnych trudności oceny wyników współzawodnictwa informatyków oraz zachęcił do kontynuowania i rozszerzenia idei współzawodnictwa.

Z programu części oficjalnej uroczystości warto zanotować jeszcze przemówienia tow. Grabowskiego (Komitet Łódzki PZPR), akcentującego znamienny fakt zwycięstwa we współzawodnictwie w działalności intelektualnej właśnie na terenie Łodzi, w której pierwszą wyższą uczelnię powołano dopiero 34 lata temu, a także przemówienie tow. Żmijewskiego (ZG ZZE), eksponującego pierwsze w kraju przyznanie sztandaru w dziedzinie działalności informatycznej. Oficjalną część uroczystości zakończyło przyznanie wyróżnionym pracownikom ośrodka łódzkiego nagród i dyplomów, które wraz z serdecznymi gratulacjami osobiście wręczył dyr. Substyk.

Trzeba stwierdzić, że pomimo opisanej koncentracji tak wielu uroczystych aktów, uczestnicy akademii nie mieli powodu uskarżać się na znużenie częścią oficjalną, która przebiegała w sposób bardzo sprawny oraz w bardzo miłej i bezpośredniej atmosferze. Tym przyjemniejszym był odbiór części artystycznej uroczystości, którą wypełnili bardzo wszechstronnym programem wokalistycznym wybitni soliści Opery Łódzkiej oraz interesujący amatorski zespół muzyczno-wokalny, w którym bierze udział jedna z młodych pracownic ZETO Łódź. Przyłączając się do licznych gratulacji, jakie otrzymała ambitna załoga łódzka, należy życzyć, aby zdobyty sztandar pozostał w jej rękach również w roku przyszłym.

Władysław KLEPACZ

PICTURE SYSTEM

Jednym z punktów programu naszej wizyty w ETH w Zurichu była prezentacja systemu zwanego PICTURE SYSTEM, opracowanego przez firmę Evans & Sutherland.

Wprowadzono nas do przyciemnionego pomieszczenia. Stał tam pokaźnych rozmiarów monitor ekranowy, pod którym, na środku stołu operatorskiego (około 2 metry długości i 1 metr szerokości), umieszczona była płyta (ok. 0,5 × 0,5 m), sprzężona z piórem świetlnym. Ruch piórem po płycie powodował wyświetlenie zakreślonej drogi na ekranie.

System demonstrował nam student wydziału architektury. Zaczął od obrysu jakiejś dowolnej figury [1]). Pod wpływem każdego ruchu piórem linia wydłużała się, a jednocześnie na dole ekranu dwie liczby wskazywały współrzędne narysowanego właśnie punktu. Dyskretyzacja płaszczyzny była dla oka niewidoczna.

Pióro zostało następnie przełączone na rysowanie odcinków. Odcinek wyznaczały dwa kolejno zaznaczone punkty. Powstał w ten sposób trójkąt równoramienny i w chwilę potem wewnętrzny mały trójkącik. Dokładne umiejscowienie odpowiednich punktów umożliwiły podawane przez system współrzędne, migające przy szybkim ruchu pióra, ale stabilizujące się przy precyzowaniu punktu. Jeden ruch gałką sterowniczą wystarczył, by trójkąt zaczął się obracać, ruch inną powodował, że oś OX wydłużała się, deformując trójkąt. Oba te przekształcenia nakładały się bez trudu.

Mamy teraz ochotę obejrzeć trójkąt w płaszczyźnie prostopadłej (YOZ). Odcinek na ekranie przekształca się w „dziurawy graniastosłup" (otwór zrobił mały wewnętrzny trójkącik). Oglądamy go ze wszystkich stron: graniastosłup powoli obraca się, deformuje. Wrażenie trójwymiarowości — zupełne. „Wjeżdżamy" w dziurę. Przez chwilę ekran jest czysty, po czym, „przejechawszy" bezkolizyjnie przez ten maleńki otworek, jesteśmy już po drugiej stronie. Została również włączona translacja.

Teraz następuje pokaz architektoniczny. Po włożeniu elastycznego dysku do PDP na ekranie ukazuje się fragment krajobrazu: wzgórza, drzewa, szosa i parę budynków. Jeden z nich jest właśnie projektowany przez studenta. Wymodelowanie takiego krajobrazu technikami tradycyjnymi trwa ok. 3 miesięcy. Tu trwało ok. pół godziny.

„Najeżdżamy" na projektowany budynek. Oglądamy go z bliska. Istotnie znakomicie harmonizuje z otoczeniem, ale może lepiej będzie go trochę podwyższyć? Próbujemy. Lekkie drgnienie konstrukcji. Za chwilę oglądamy projekt od dołu — prawdziwe curiosum. Prosimy o wyłączenie mało widocznych linii: dla niewprawnego oka obraz będzie teraz łatwiejszy do oglądania.

Budynek podzielony jest na piętra. Aby obejrzeć jedno z nich, górne kondygnacje łagodnie unoszą się do góry. Oto plan pomieszczeń: ściany, drzwi, okna. Zmiana punktu odniesienia powiększa wrażenie trójwymiarowości obrazu.

Z powrotem „sklejamy" budynek. Ręcznie osadzamy usuniętą warstwę. Wydaje się, że robimy to bardzo precyzyjnie. Niestety powiększony obraz ujawnia naszą niedokładność. Poprawiamy się i znów wywołujemy obraz. Po kilkukrotnym przeprowadzeniu takiego zabiegu nareszcie wszystkie elementy idealnie do siebie pasują.

Po pokazie w inny już sposób patrzymy na wspaniałe, nowoczesne bryły budynków uczelni.

Piotr CHRZĄSTOWSKI

[1]) Na ekranie można również wymodelować okrąg i wszystkie pozostałe krzywe stożkowe

Czas podsumować

Celem wyjazdu Koła Naukowego Informatyków UW było poznanie możliwości sprzętowych i organizacji ośrodków obliczeniowych firm turystycznych i uniwersytetów w krajach Europy Zachodniej. Do programu naukowego wchodziły również prace związane z testowaniem kompilatora Pascala (Pascal Stone Brook Compiler) na uniwersytecie w Brnie.

Jako użytkownicy krajowych uczelnianych ośrodków obliczeniowych chcielibyśmy przede wszystkim podzielić się, naszymi wrażeniami z podobnych ośrodków za granicą.

W uczelnianych ośrodkach obliczeniowych wykorzystuje się w pierwszym rzędzie prace o charakterze dydaktycznym i badawczym. Temu celowi służy na przykład system priorytetów (Tybinga) — najwyższy priorytet posiadają studenci, chociaż ich prace ośrodek wykonuje bezpłatnie. Często wykonuje się również prace związane z administracją uczelni (Tybinga) lub dla celów wyspecjalizowanych instytucji uczelni (Zurich — dla biblioteki ETH). Dopiero w ostatniej kolejności — jeżeli moc obliczeniowa ośrodka nie jest jeszcze w pełni wykorzystana — wykonywane są prace dla użytkowników z zewnątrz.

Prace dydaktyczne i badawcze mogą być wykonywane efektywnie jedynie wówczas, gdy moc obliczeniowa ośrodków jest duża. Tak więc zainstalowane we wszystkich ośrodkach uczelnianych komputery podstawowe są dużymi maszynami (CDC, IBM 360 i 370, TR 440). Rzadko też zdarza się, żeby ośrodek posiadał tylko jeden komputer. Najczęściej są dwa, trzy komputery różnej klasy, połączone siecią terminali lub przez wspólną pamięć masową.

Pracę komputerów podstawowych wspomagają minikomputery (najczęściej PDP 11/45), na których prowadzone są interakcyjnie prace konstrukcyjne architektów (Zurich), prace badawcze fizyków (Tybinga) bądź inne mniejsze zadania.

Moc obliczeniową zainstalowaną w tych ośrodkach trudno porównywać z możliwościami ośrodka CYFRONET w Świerku, z którego korzysta m.in. Uniwersytet Warszawski. Tak np. nasz uniwersytet jest jednym z wielu użytkowników komputera CDC 6400, który w Zurichu pełni rolę małej maszyny w konfiguracji trzech komputerów będących niemal do wyłącznej dyspozycji studentów i pracowników ETH.

Ośrodek w Świerku ma zdecydowanie niewystarczającą pojemność pamięci masowej. W centrum obliczeniowym Bawarskiej Akademii Nauk powiększenie pamięci odbyło się przez wprowadzenie nowego jej typu — „cartidge storage", podobnego nieco do karuzeli produkowanych kiedyś przez REGNE CENTRALEN. Jedna stacja tej pamięci składa się z 2000 rolek szerokiej taśmy magnetycznej (60 × 9 ścieżek). Pojemność stacji wynosi około 20 GB, a czas dostępu do informacji — 2—3 sekundy. Stosunkowo niski koszt urządzenia (ok. miliona marek) pozwolił na zlikwidowanie sytuacji, w której brakowałoby miejsca na pliki stałe. Pliki robocze znajdują się na dyskach.

Uczestnicy wyjazdu organizowanego przez Koło Naukowe Informatyków UW w ub. r. Siedzą od lewej: K. Miączyńska-Koteras, J. Siemień, A. Okseniuk, T. Tarlecka, M. Lao, L. Szcześniak, J. Mincer, A. Tarlecki, M. Biderman, M. Bańkowski. Stoją: M. Cichy, P. Chrząstowski, K. Koteras, J. Deminet, J. Franecki, B. Cichocki i J. Kania *Fot. J. Szczęsny*

Normalnym trybem pracy we wszystkich ośrodkach jest praca interakcyjna. Sprzyja temu bardzo duża liczba terminali. W Lozannie zainstalowano 150-terminalową sieć, która rozciąga się nawet poza francuskojęzyczną część Szwajcarii. Dobrze rozwinięta jest sieć przesyłu danych (linie telefoniczne, łącza mikrofalowe). Umożliwia to wygodne korzystanie z komputera bez względu na miejsce, w którym znajduje się użytkownik.

Wprowadzono też wiele pomysłowych rozwiązań organizacyjnych. Niektóre można zastosować również w naszych ośrodkach. Czytnik obsługiwany przez użytkowników odciąża pracę operatorów i przyspiesza wprowadzenie zadania do systemu. Specjalne stojaki, na których wydrukowane wyniki zadań układane są alfabetycznie, umożliwiają szybkie ich odnalezienie. Monitory ustawione na terenie całego ośrodka, wyświetlające cyklicznie wszystkie kolejki systemowe, pozwalają na śledzenie przebiegu zadania w systemie. Nie są wtedy konieczne pytania użytkowników kierowane do operatorów: „Gdzie jest mój program?".

Wizyty w ośrodkach obliczeniowych uniwersytetów i firm turystycznych umożliwiły nam poznanie nie tylko nowoczesnego sprzętu o olbrzymiej mocy obliczeniowej. Przy testowaniu kompilatora Pascala w Bernie wielu z nas po raz pierwszy miało możliwość pracy interakcyjnej, która przecież w niedalekiej przyszłości będzie podstawową formą pracy informatyka. I tak — chociaż niewiele mieliśmy czasu na zwiedzanie poszczególnych ośrodków — udało nam się zorientować, w jaki sposób można rozwiązać niektóre ważne problemy w projektowaniu systemów.

Marek J. LAO

Wszystkim naszym Czytelnikom życzymy równie ciekawych wrażeń. Czekamy na relacje.

Redakcja

Co słychać w informatyce francuskiej ?

Park komputerowy

W dniu 1 stycznia 1978 roku zarejestrowano we Francji 23 646 komputerów wszystkich kategorii — poza komputerami biurowymi i minikomputerami o wartości poniżej 50 tys. franków. Rok wcześniej liczba ta wynosiła 19 142, co oznacza istotny wzrost (23,5 %). Warto jednak zauważyć, że tak jak w latach poprzednich największy przyrost dotyczył przede wszystkim bardzo małych systemów, o wartości poniżej 250 tys. franków (35 %), a stanowią one zaledwie 3,7 % całkowitej wartości parku (1085 mln franków wobec ponad 29 mld franków). Znaczny wzrost zanotowano również w ogólnej wartości tego sprzętu: 17,5 % w 1977 roku, w porównaniu z 12,2 i 12 % w latach 1976 i 1975, oraz 14,6 % — średni wzrost przewidziany na lata 1978—1983.

Skok ten, nieco zaskakujący dla francuskich prognostyków, wynika z:

- podjęcia prac związanych z nowymi zastosowaniami; od roku 1975 były one opóźnione w stosunku do planów, bądź zarzucone; o ile jednak w pewnych dziedzinach można sobie było na to pozwolić, o tyle np. w przypadku problemów związanych z bazami danych czy sieciami (które charakteryzują się dużą chłonnością dodatkowego sprzętu), nie można było odwlekać prac w nieskończoność
- przekroczenia pewnego poziomu wykorzystania zainstalowanych komputerów; w związku z ograniczeniem zamówień w latach 1975—1976 zwiększył się stopień wykorzystania istniejącego już parku; wreszcie osiągnięto poziom, przy którym nie można było już dalej zwiększać obciążenia maszyn i dlatego od drugiej połowy 1976 roku pojawiły się nowe zamówienia
- zmiany stanowiska, spowodowanej przez producentów; wielu użytkowników odwlekało decyzję zakupów sprzętu, ponieważ oczekiwało na nowe, „rewolucyjne" komputery, anonsowane przez niektóre firmy, szczególnie IBM.

Ok. 5 lat temu zauważono, że rozwój francuskiego parku komputerowego wykazuje tendencję do większego wzrostu ilościowego minikomputerów i dużych maszyn kosztem małych i przede wszystkim średnich komputerów. Obecna analiza wykazuje pewną stabilizację na poziomie systemów o wartości przekraczającej 250 tys. franków. Przewiduje się do roku 1983 średni ilościowy przyrost roczny w grupie komputerów małych o 16 %, średnich — o 12 %, dużych i bardzo dużych — o 13 %, a także dalszy szczególnie prędki wzrost ilościowy systemów o wartości poniżej 250 tys. franków (do roku 1983 średnio o 36,2 %).

Opublikowane informacje nie podają, jak kształtuje się udział na rynku francuskim wyrobów poszczególnych producentów komputerów.

Zatrudnienie

Mimo pogłębiania się bezrobocia we Francji w latach 1977 i 1978, informatyka nie została dotknięta kryzysem. W roku 1975 pracowało we Francji 170 tys. informatyków, pod koniec 1977 — ok. 200 tys. (wzrost ok. 18 %). Dla porównania, w najlepszych latach rozwoju informatyki francuskiej notowano wzrost zapotrzebowania rzędu 20 %.

Analiza wykazuje znaczne wahania w zatrudnieniu, największe zanotowano między styczniem 1977 a styczniem 1978 roku (+155 %).

Pracodawców można podzielić na trzy grupy: producentów sprzętu, przedsiębiorstwa usługowe i użytkowników. Stosunek potrzeb poszczególnych grup w roku 1976 wynosił 15 : 18 : 67, w roku 1977 — 10 : 18 : 72 i w pierwszych trzech miesiącach roku 1978 — 14 : 22 : 64.

Zapotrzebowanie na pracowników w poszczególnych grupach pracodawców (w procentach)

	Producenci			Usługi			Użytkownicy		
	1976	1977	1978 (I kw.)	1976	1977	1978 (I kw.)	1976	1977	1978 (I kw.)
Kadra kierownicza	3	4	—	9	9	6	88	87	94
Sprzedawcy	65	59	73	35	41	37	—	—	—
Eksploatacja	16	21	23	10	19	19	73	60	48
Obsługa systemów	14	4	8,5	15	13	25	71	83	66,5
Analitycy i programiści	9	5	6	22	15	21	69	80	73
Główni projektanci	21	15	18	24	21	24	55	64	58

W roku 1978 nastąpił wyraźny wzrost liczby ofert. W 1976 roku było ich 2770, w 1977 — 3962 (wzrost o 43 %), natomiast w ciągu trzech pierwszych miesięcy 1978 — 1635. Pojawiły się nawet — szczególnie w pewnych regionach — trudności w skompletowaniu kadry informatycznej. Sytuację tę spowodowały zarówno szybki rozwój miniinformatyki i decentralizacja systemów informatycznych, jak też niewystarczająca liczba wykształconych specjalistów w tej dziedzinie. Problemy takie występują np. w regionie paryskim, mimo że jest to największy ośrodek uczelniany. Okazuje się, że absolwenci z prowincji odmawiają podejmowania pracy w metropolii.

Rok 1978 wydaje się być szczególnie dobry jeśli chodzi o zapotrzebowanie rynku pracy w dziedzinie informatyki francuskiej. Trudno jednak teraz o perspektywiczną ocenę tego zjawiska. **(P.S.)**

Na podstakwie ZERO-UN-INFORMATIQUE, kwiecień br.

Centrum banków informacji na Riwierze

W grudniu ubiegłego roku rząd Francji przeznaczył ok. 30 mln dolarów na dofinansowanie rozwoju dziedzinowych banków informacji. Plan rozwoju przewiduje stworzenie ogólnokrajowego centrum dziedzinowych banków informacji w Valbonne na Riwierze, obejmującego banki informacji z zakresu chemii, medycyny, rolnictwa oraz polityki międzynarodowej. Banki te realizowane będą na sprzęcie francuskim i powstaną drogą przekazania lub powielania obecnie funkcjonujących systemów informacyjno-wyszukiwawczych, takich jak PASCAL czy CHEMICAL ABSTRACTS. Ponadto przewidywane jest usprawnienie krajowej sieci usług w zakresie planowania i informacji ekonomicznej Gapset. Relizatorem projektu jest rządowe centrum badań naukowych oraz firma Telesystemes.

Posunięcia te mają na celu uniezależnienie się od zagranicznych banków informacji oraz stanowią próbę przygotowania Francji do eksportu usług w tym zakresie. **(I. S.)**

Na podstawie DATAMATION, marzec br.

Nowy ośrodek w Zielonej Górze

Mgr Leon Judka — jako dobry gospodarz z zamiłowania i Wielkopolanin z pochodzenia — w ciągu blisko piętnastoletniego dyrektorowania poznańskiemu ETOB-owi — borykał się przeważnie z kontrastowym niejako problemem: dużo kilometrów kwadratowych do obsługi informatycznej i mało metrów kwadratowych w etobowskich pomieszczeniach. Nie szukał logicznego na pozór kompromisu, traktując jako naturalny obowiązek obsługi budownictwa na terenie objętym obecnie dziewięcioma województwami. Mógłby pod tym względem konkurować jedynie z ETOB-em Warszawa. Szukał głównie rozwiązania trudności lokalowych, bo to warunkowało możność instalowania tzw. mocy obliczeniowych, czyli komputerów i kadry informatycznej.

Pierwszy wynik pozytywny, to ukończenie w 1974 r. budynku ETOB w Poznaniu. Dzięki temu można było dość szybko zainstalować dwie ODRY 1305 i prostować skrzydła na rzecz regionalnych usług informatycznych. Pozostał jednak problem dużego oddalenia użytkowników od ośrodka obliczeniowego. Trudno jest bowiem działać operatywnie na obszarze od Szczecina do Jeleniej Góry i od Zielonej Góry do Konina.

Z tego względu już w 1972 r. pomyślano o utworzeniu skromnej filii w Zielonej Górze, kierując tam wycofywane stopniowo z Poznania maszyny licząco-analityczne, wraz z urządzeniami do przygotowania danych. Więcej niż skromne warunki lokalowe pozwoliły na zainstalowanie trzech zaledwie zestawów MLA. Był to jednak pierwszy krok, który niewiele wprawdzie odciążając ETOB-Poznań, dość skutecznie rozbudził zainteresowanie informatyką w zielonogórskim środowisku budowlanym. A kiedy powstały możliwości wybudowania nowoczesnego budynku dla zespołu przedsiębiorstw budowlanych, zadbano o to, aby jego efektowny fragment przeznaczyć dla ośrodka ETOB

W ten sposób zmaterializowało się obopólne zainteresowanie Zielonogórskiego Zjednoczenia Budownictwa i poznańskiego ETOB-u. Zaangażowania tego ostatniego i dyr. L. Judki osobiście eksponować w tym miejscu nie wypada. Wypada natomiast i trzeba podkreślić konstruktywną atmosferę, kreowaną przez miejscowe władze polityczne i administracyjne, w czym wielka zasługa mgr Józefa Grzelaka — sekretarza ekonomicznego KW PZPR, uznanego na Ziemi Lubuskiej orędownika nowoczesności w ogóle, a informatyki w szczególności. Tak się bowiem składa, że decyzje przyzwalające na realizację przedsięwzięcia zwykły nabierać finalnego poloru dopiero wtedy, gdy się je wspiera żywym i skutecznym duchem zainteresowania. A rzadko wystarcza dać ludziom nawet atrakcyjne miejsce pracy, nie oglądając się np. na to gdzie im mieszkać wypadnie.

Zielonogórski ośrodek ETOB zwiedzają (od lewej): Przewodniczący Rady Zakładowej w ETOB Poznań — J. Wąchalska, Prezydent Zielonej Góry — mgr S. Ostręga, Wiceprezydent — mgr L. Szczygieł, dyrektor ETOB-u Poznań — mgr L. Judka, główny elektronik Ośrodka — mgr inż. W. Kaczmarek, I Sekretarz KM PZPR — tow. J. Trzeszcz, kierownik budowy — W. Grzebieniuk.

Pomieszczenie ośrodka obliczeniowego w Zielonej Górze liczy 1200 m², w czym dwie duże sale — po 120 m² każda — jedna dla komputera, druga dla przygotowania danych. Kosztowało to — łącznie z wyposażeniem — zaledwie 52 mln. zł (o 20 mln. zł mniej niż przewidywały założenia). Na przełomie lat 1978/79 rozpoczęto instalowanie Odry 1305 w tzw. konfiguracji podstawowej, niestety (PAO-64k, 6 × PT-3, DZM-180, jedna drukarka wierszowa i jeden czytnik kart) oraz 20 stanowisk do przygotowania danych na kartach dziurkowanych (SOEMTRON).

Oficjalne otwarcie ośrodka w nowym jego pomieszczeniu (centrum miasta, ul. Westerplatte) odbyło się 27 kwietnia 1970 r. Uroczystość była nader skromna, traktowana jako zwyczajne, robocze nieomal wydarzenie, zauważone jednak odpowiednio przez władze Ziemi Lubuskiej. Świadczyło o tym dokonanie aktu otwarcia przez I Sekretarza KM PZPR tow. Jana Trzeszcza, wespół z Prezydentem Miasta mgr Stanisławem Ostręgą. Uczestniczyli przy tym: Wiceprezydent Zielonej Góry — L. Szczygieł, Z-ca Kierownika Wydziału Ekonomicznego KW PZPR — H. Piechocki, Naczelny Dyrektor Zielonogórskiego Zjednoczenia Budownictwa — A. Macniak, Prorektor WSI — L. Faryniak, Z-ca Przewodniczącego Wojewódzkiej Komisji Planowania — M. Kopij, Z-ca Naczelnego Dyrektora Centrum ETOB — T. Niemunis.

Wobec pilnych obowiązków służbowych nie mógł być obecny mgr J. Grzelak. Natomiast sentyment do sprawy spowodował gremialną niemal obecność wykonawców obiektu, z Naczelnym Dyrektorem Kombinatu w Zielonej Górze — H. Augustynowiczem, jego zastępcą E. Piwockim i kierownikiem budowy W. Grzebieniukiem,, którzy — występując w asyście mistrzów bezpośrednio odpowiedzial-

nych za jakość prac budowlanych — chcieli zapewne udowodnić, że nie obawiają się spojrzeć w oczy przyszłemu użytkownikowi obiektu. Należy to z szacunkiem uznać za ewenement godny naśladowania w skali krajowej. A wiedzieli co czynią, bowiem nie tylko zarzutów nie było, ale było coś w rodzaju zdziwienia, że krajowe przepisy budowlane mogą nie przeszkadzać rzetelnemu, a nawet eleganckiemu wykończeniu obiektu. Oby był to przykład zaraźliwy.

Powinności gospodarzy pełnili — z należytym poświęceniem — wspomniany już dyr. L. Judka oraz kierownik ośrodka — mgr inż. Sylwester Szmigiel.

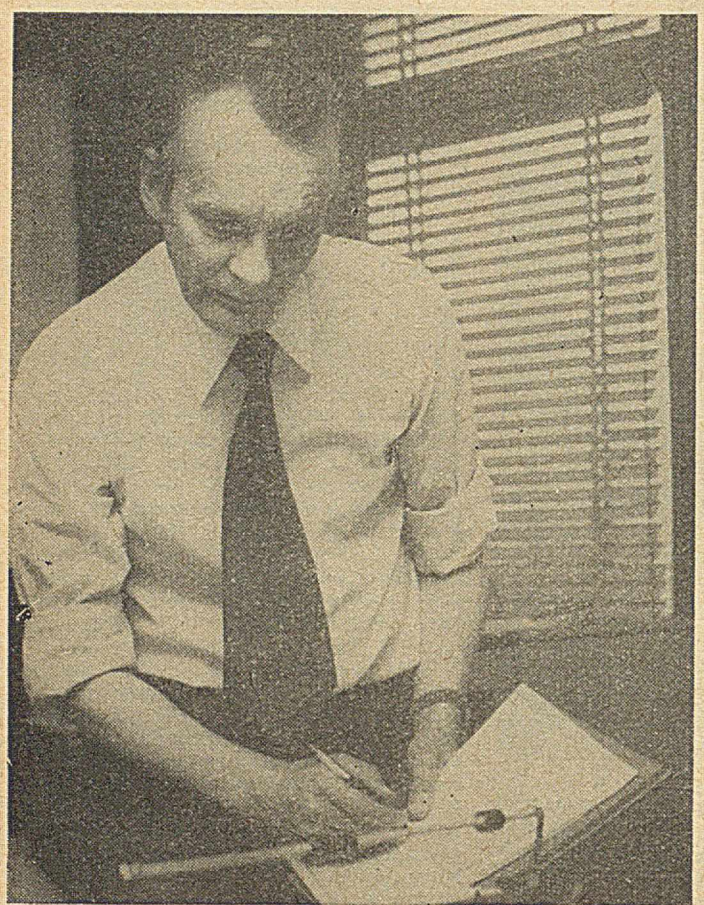
Uroczystości to mają do siebie, że szybko przechodzą do kroniki wydarzeń. Zostaje codzienna rzeczywistość. Z tej zaś wynika dość jasno, że zielonogórski ośrodek powstał po to, aby odciążyć Poznań (a także inne ETOB-y) w obsłudze informatycznej województw zielonogórskiego, legnickiego i jeleniogórskiego, które już wcześniej zostały mu — niejako na wyrost — przypisane. Macierzysty poznański ETOB chciałby widzieć więcej — obsługę Zjednoczenia Budownictwa Przemysłowego „Zachód" — a więc od Jeleniej Góry po Szczecin dla przedsiębiorstw tego zjednoczenia. Zjednoczenie zielonogórskie — według oczekiwań wypowiadanych przez dyr. H. Augustynowicza, jako najbardziej w sprawie doświadczonego — liczy na rozszerzenie usług konwencjonalnych i ich pogłębienie, zwłaszcza na rzecz fabryk domów i budownictwa mieszkaniowego. Są to dość konkretne obowiązki, wymagające spełnienia.

Życzymy młodemu ośrodkowi sukcesów w wypełnianiu tych niełatwych zadań.

Krystyn BERNATOWICZ

Portrety zawodowe

Juliusz Nalewajski



Zamysł tej rubryki polega na „zatrzymywaniu w kadrze" — póki jeszcze nie za późno — doznań i karier zawodowych, charakterystycznych dla pierwszego w Polsce ćwierćwiecza dziedziny, którą od lat dziesięciu informatyką przyjęło się nazywać. W każdym więc przypadku kluczem do wyboru bohatera szkicu, a następnie sposobu jego prezentacji jest znalezienie formuły przewodniej, syntetyzującej niejako dany portret zawodowy.

Szukanie takiej formuły dla Juliusza Nalewajskiego z Krakowa, przywiodło do odgrzebania w lamusie językowym celnego słowa „kontrapunkt". Wypada wyjaśnić, iż jest to termin o muzycznym rodowodzie, oznaczający sztukę prowadzenia odrębnych melodii tak, że tworzą one harmonijną całość. Co ma jednak J. Nalewajski do kontrapunktu? Za chwilę się okaże, a tymczasem warto zachęcać do refleksji nad tym życiorysem, bo o przyczynek do ćwierćwiecza losów naszej informatyki użytkowej tu chodzi, a nasz bohater wydaje się nader pouczającą tych losów egzemplifikacją. Niełatwo w końcu znaleźć człowieka na tak miarodajnym punkcie obserwacyjnym, jakim jest ponad dwudziestoletnie pełnienie funkcji szefa „produkcji" w profesjonalnym ośrodku obliczeniowym.

Kontrapunktem J. Nalewajskiego była i jest sztuka harmonizowania aż czterech równoległych wątków, które raczej do wzajemnych dysonansów niż do owej harmonii skłonności wykazują. Pierwszy — to trudności lokalowe, drugi — notoryczna niejednolitość parku maszynowego, trzeci — tempo koniecznych zmian kwalifikacji kadry, czwarty — osobiste nadążanie za tym co się dzieje i zmienia.

J. Nalewajski nie jawi się jako oryginał, lubiący pracować w nienormalnych warunkach. Jego podopieczny personel — tym bardziej. A zaczęło się w 1953 roku od przysposobienia dla Biura Rozliczeń Budownictwa trzech mieszkań przy ul. Manifestu Lipcowego. Wystarczyło na krótko. Szczęśliwie, w 1958 r. władze miejskie zamknęły knajpę „Lotos" w Nowej Hucie, z całkiem nie informatycznych powodów. Zaadaptowano ją dla potrzeb zautomatyzowanej techniki obliczeniowej, co pozwoliło ulokować ARITMY i objąć obsługą wszystkie związane z Hutą przedsiębiorstwa bu-

dowlane. Pojawiła się wkrótce nowa komplikacja: komputery Mińsk 32, których nijak w byłym „Lotosie" nie dało się ulokować, nie tylko z powodu tradycji pomieszczenia. Pozyskano więc lokal w budowlanym biurowcu przy ul. Wadowickiej, gdzie do dziś trwają kolejne jego adaptacje i trzeba mieć sporo wyobraźni, aby widzieć w nim ośrodek obliczeniowy z prawdziwego zdarzenia.

Z maszynami było nieco podobnie: najpierw SAMY 45 i 80, potem ARITMY i wreszcie komputery — dwa Miński 32, Odra 1305, a obecnie także R 32. Nie ma już wprawdzie MLA, ale cztery komputery pracują równocześnie i nie ma się co czarować, że zgodnie to robią. J. Nalewajski z sentymentem wspomina czasy, gdy mógł sie nie przejmować awarią nawet paru zestawów MLA. Miał ich bowiem kilkanaście. Teraz lepiej nie myśleć o awarii na Odrze. A jeszcze niewiele lat temu wierzył, że komputery przyniosą stabilizację warunków pracy i spokojniejszy tryb obsługi klientów. Czy była to więc wiara naiwna? Nie — odpowiada — bo kto mógł przewidzieć niemożność dokupienia do Odry paru urządzeń peryferyjnych i pamięci operacyjnej, aby zestaw ten pracował jak prawdziwy komputer, a nie tylko udawał szybszą wersję MLA. Prowokowany perspektywami riadowskimi, reaguje nie bez pewnej irytacji: — trudno ufać dostawcy, który jeden program produkcji rozbabrał i zarzuca, a zamiast tego epatuje w podobny sposób kolejną perspektywą. Byłem za należytym wykorzystaniem najpierw tego co mamy kupione i mieliśmy obiecane, a przeciew pośpiesznym reorientacjom. I natychmiast się sumituje: — w swoim zakresie dołożę starań, aby Riad w Krakowie był jak najlepiej wykorzystany. Ale przecież jeszcze przez parę lat moim podstawowym warsztatem pracy będą dwa stare Miński i Odra 1305 w konfiguracji, którą nie ja pierwszy nazywam kadłubową!

Sprawy kadrowe są nie mniej trudne. Nie wszyscy wytrzymują pracę trzyzmianową, a zwłaszcza szybkość przestawiania swych kwalifikacji na nowy sprzęt. Całe szczęście, że choć systemy użytkowe są dość stabilne, bo dominują w nich ewidencja i rozliczenia. J. Nalewajski czynnie uczestniczył w tworzeniu filialnych ośrodków w Rzeszowie (już samodzielne przedsiębiorstwo ETOB) i w Kielcach (jeszcze podlega Krakowowi). Trochę im nawet zadrości, bo od razu wchodziły na ETO, nie mając obciążeń rutyną MLA, która w Krakowie ciągle jeszcze daje się we znaki.

Wątkiem w „Kontrapunkcie Nalewajskiego" najdelikatniejszym, a i najciekawszym na tym miejscu zarazem, jest jego osobiste odnajdywanie się w tym konglomeracie spraw przeżywanym od ponad 25 lat.

W 1951 roku ukończył blisko 800-letnią płocką „Małachowiankę". Trzeba znać z bliska atmosferę tego liceum, aby zrozumieć dlaczego marzeniem wielu jej absolwentów było podjęcie studiów na Uniwersytecie Jagiellońskim. J. Nalewajski okazał się jeszcze większym konformistą, bo po nieudanym starcie na UJ (czym się w końcu nie przejął, zapewne na skutek typowej dla tego wieku szybkości reorientacji) — zwerbował do Krakowa młodszą koleżankę licealną. Nie chce się przyznać, czy już wtedy przeczuwał, że owa pani Julita stanie się wkrótce domowym egzaminatorem tego, co pan Juliusz będzie robił w BRB i w ETOBie.

Zainteresowania naszego bohatera skierowały się ku dwuletniemu Zakładowi Wiedzy Handlowej, któremu patronował prof. dr Kazimierz Sowa, „ojciec duchowy", założyciel i pierwszy dyrektor BRB w Warszawie, w Krakowie i w ogóle. On to zaproponował J. Nalewajskiemu i paru innym osobom pracę w BRB. I tak w dniu 15 września 1953 r. na etacie BRB pojawia się jeden z pierwszych operatorów tabulatora. Potem następne szczeble wtajemniczenia: kontrola wyjścia z MLA, rozpowszechnianie systemów, a od 1958 do dziś — szef produkcji, choć w ciągu 20 lat różnie się to stanowisko nazywało.

Nie odbyło się bez zwątpienia w samego siebie, gdy trzeba było przechodzić z MLA na komputery. Po prostu zląkł się ogromu nowej wiedzy, którą trzeba było szybko opanować. Kończy odpowiednie kursy, jeździ do Mińska, uczy się sam. I teraz — po latach — mówi, że polubił te urządzenia. — Natura ludzka ciągnie do rutyny, a one na to nie pozwalają — dodaje. A doświadczenia operatora tabulatorów są także przydatne, bo pozostał nawyk wnikania w szczegóły pracy personelu. Dobrze też rozumie się ze swym bezpośrednim przełożonym — dyr. Henrykiem Rajchlem, który także rozpoczynał w BRB jako operator MLA.

Juliusz Nalewajski twierdzi, że lubi pracować z mądrzejszymi od siebie, ale stara się nadmiernie od nich nie „odstawać". M.in. dlatego w 1966 roku ukończył dwuletnie studium socjologii pracy przy krakowskim SNS. — Chciałem sprawdzić teoretycznie, czy moje postępowanie w pracy ma pokrycie w zaleceniach naukowych. Zwłaszcza, że uważano mnie za autokratę — wyjaśnia.

W swą pracę jest autentycznie zaangażowany. Sam czuje się doceniany i potrzebny, a współpracownicy i przełożeni mówią bez ogródek, że nie wyobrażają sobie bez Nalewajskiego tak sprawnego funkcjonowania ośrodka w tak trudnych warunkach. Był wielokrotnie wyróżniany: Złoty i Srebrny Krzyż Zasługi, liczne dyplomy uznania.

Pani Julita Nalewajska pracuje w krakowskim „INSTALU" też od 25 lat — obecnie jako zastępca głównego księgowego — i przez cały ten czas jest odbiorczynią usług ETOBu. Oboje twierdzą, że w sposobie propagowania zastosowań informatyki tkwi jakiś istotny błąd. Dyrekcje przedsiębiorstw nie rozumieją bowiem na ogół potrzeb i problemów własnych służb w tej dziedzinie, sądząc, że współpraca z ETOBem — to wyłącznie sprawa np. księgowości. Warto dążyć do zmiany tej sytuacji.

Poproszony o finalną refleksję na temat swej pracy, Juliusz Nalewajski odpowiedział: — gdybym miał swe życie „repetować", zrobiłbym to w tym samym zawodzie.

(wlad)

Prosty system ewidencji studentów

Centralną komórką administracyjną każdego wydziału wyższej uczelni jest dziekanat. Zakres prac np. dziekanatu Wydziału Automatyki i Informatyki Politechniki Śląskiej w Gliwicach ustala „Regulamin organizacyjny Politechniki Śląskiej" [1]. Do najważniejszych prac przewidzianych tym regulaminem należą:

— wszystkie prace organizacyjne związane z dydaktyką
— prowadzenie dokumentacji i sprawozdawczość
— prace związane z rekrutacją na studia
— opracowywanie planu zajęć dydaktycznych
— całokształt zagadnień związanych ze świadczeniami stypendialnymi i sprawami bytowymi studentów
— prowadzenie archiwum wydziału.

Celem usprawnienia tych praco- i czasochłonnych zadań na Wydziale Automatyki i Informatyki podjęto prace projektowe i wdrożono systemy informatyczne rekrutacji kandydatów na I rok studiów [2] oraz układania rozkładu zajęć [3]. Kolejnym krokiem na drodze ułatwiania pracy dziekanatu było zaprojektowanie w latach 1977—78 i przygotowanie do wdrożenia opisanego poniżej systemu ewidencji studentów KOSEWIST [4]. Prace nad tym systemem były prowadzone pod kierunkiem doc. dr hab. Antoniego Niederlińskiego oraz przy konsultacjach mgr Ireny Lipowej z Centralnego Ośrodka Informatyki Górnictwa.

ZADANIA SYSTEMU

Podstawowym zadaniem systemu KOSEWIST jest stworzenie zbioru informacji o wszystkich studentach wydziału (około 1500 osób). W związku z tym system:

— ujmuje obiek dokumentów źródłowych w obrębie dziekanatu
— precyzuje sposób przenoszenia danych z dokumentów źródłowych na maszynowe nośniki danych
— zapewnia programowe środki w celu założenia, aktualizacji i kontroli zbioru informacji o studentach (ZBIOR GŁÓWNY) oraz ZBIORU ARCHIWALNEGO, zawierającego dane o studentach opuszczających wydział — korzysta z podstawowych danych o strukturze wydziału (kierunki — specjalizacje), a przede wszystkim zawiera kompletny i aktualny plan studiów realizowanych na wydziale (zbiór PLAN STUDIOW).

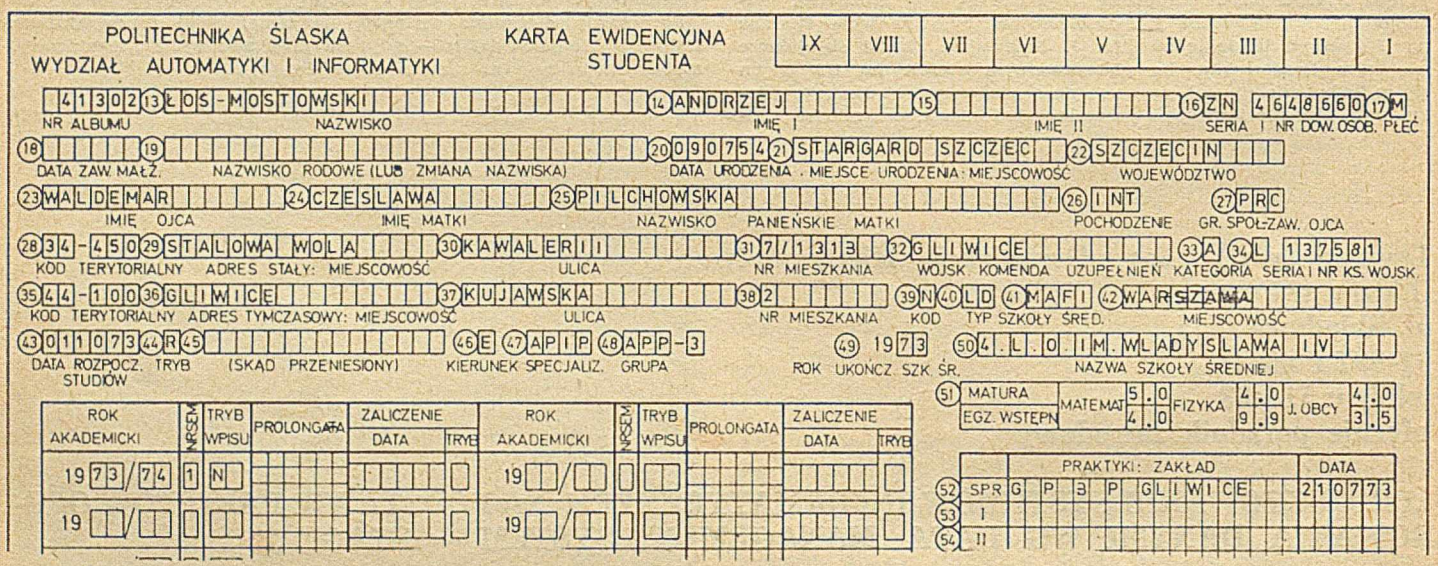
Reasumując — KOSEWIST gwarantuje pełny, stale aktualny i rzetelny zbiór informacji o studentach, który jest wykorzystywany przez programy użytkowe [1]) wspomagające prace dziekanatu.

Liczba programów użytkowych będzie się zwiększała podczas wdrażania systemu. Powinny znaleźć się wśród nich programy sporządzające różnego rodzaju listy, charakterystyki studentów i protokoły egzaminacyjne, a także różne sprawozdania (dla GUS, Rektoratu itp.). System powinien przejąć większość prostych lecz masowych prac ewidencyjnych i sprawozdawczych dziekanatu.

PROJEKT DOKUMENTÓW

Dane o każdym studencie znajdują się w około 20 różnych dokumentach, których obecność w dziekanacie wynika z przepisów resortowych i uczelnianych lub z potrzeby ułatwienia prac ewidencyjnych. Prowadzi to do dużej redundancji i powoduje, że praca personelu dziekanatu, polegająca na aktualizowaniu tych dokumentów, ich prze-

[1]) Przez program użytkowy rozumie się w tym artykule program korzystający z określonej kombinacji zbiorów: ZBIÓR GŁÓWNY, ZBIÓR ARCHIWALNY, PLAN STUDIÓW. Programami użytkowymi nie są w tym rozumieniu programy tworzące, aktualizujące i kontrolujące wymienione zbiory.



Rys. 1. Przykładowo wypełniona karta ewidencyjna systemu KOSEWIST

glądaniu i przepisywaniu, jest bardzo nużąca i prowadzi do licznych błędów. W związku z tym zaprojektowano kartę ewidencyjną (rys. 1), zawierającą wszystkie istotne dla dziekanatu informacje o studencie. Grupuje ona dane personalne, dane o zaliczeniach poszczególnych semestrów, dokumentację akcji socjalnej, praktyk, nagród, spraw dyscyplinarnych oraz informacje o wyjazdach za granicę.

Sposób wypełniania karty ewidencyjnej został dokładnie opisany w specjalnej instrukcji dla dziekanatu [4].

WYKORZYSTYWANY SPRZĘT KOMPUTEROWY

Projekt systemu KOSEWIST podporządkowany jest wymaganiom systemu komputerowego ODRA 1300. W opisanej realizacji korzystano z maszyny cyfrowej ODRA 1325, zainstalowanej w Ośrodku Elektronicznej Techniki Obliczeniowej Politechniki Śląskiej. Maszyna ta pracuje pod nadzorem programu sterującego (EXECUTIVE) w konfiguracji standardowej:
— 32 K słów pamięci operacyjnej
— 6 jednostek pamięci taśmowej
— czytnik kart dziurkowanych
— czytnik i dziurkarka taśmy papierowej.

Zbiory w systemie KOSEWIST zapisane są na taśmach magnetycznych, natomiast dane z formularzy źródłowych przenoszone są na karty dziurkowane. KOSEWIST jest więc typowym tradycyjnym systemem z przetwarzaniem danych w trybie wsadowym.

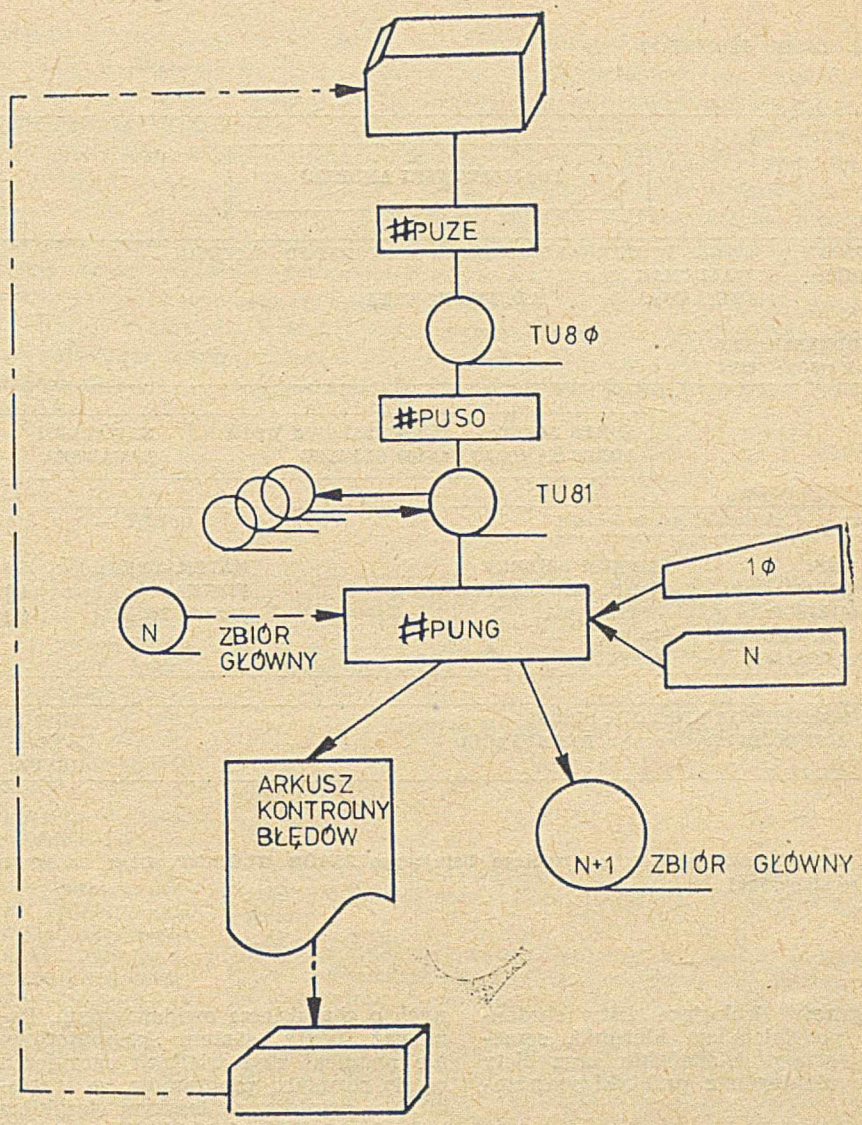
ZBIORY SYSTEMU

System korzysta z następujących trzech zbiorów: ZBIORU GŁÓWNEGO, ZBIORU ARCHIWALNEGO oraz PLANU STUDIÓW.

Każdy element ZBIORU GŁÓWNEGO (informacje o jednym studencie) rozpoznawany jest przez numer albumu studenta, zaś poszczególne rekordy przez ich typ. W ZBIORZE GŁÓWNYM znajduje się 16 typów różnej długości rekordów [2]), w których zapisuje się informacje z karty ewidencyjnej oraz wyniki nauczania. Dane o jednym studencie nie przekraczają 1,5 K słów.

Z chwilą opuszczania wydziału niektóre informacje o danym studencie ze ZBIORU GŁÓWNEGO, uzupełnione dodatkowymi informacjami o trybie i dacie opuszczenia wydziału oraz o dyplomie, zapisywane są w ZBIORZE ARCHIWALNYM w postaci pojedynczego rekordu. Zbiór ten jest pomocny przy realizacji przepisowego 50-letniego okresu przechowywania dokumentacji dotyczącej absolwentów szkół wyższych.

[2]) Klucz, według którego rozpoznaje się rekord, to numer albumu i typ rekordu (8 pozycji znakowych)



Rys. 2. Schemat przetwarzania modułu nagrywającego ZBIÓR GŁÓWNY

W zbiorze PLAN STUDIÓW informacje zapisane są w czterech typach rekordów: o przedmiotach nauczanych na wydziale, o prowadzących zajęcia dydaktyczne, o strukturze wydziału oraz o praktykach studenckich. Zbiór ten umożliwia aktualizowanie danych o wynikach nauczania.

PROGRAMY SYSTEMU

Ze względu na pełnione funkcje programy systemu podzielono na trzy grupy:

1) programy organizacyjne — zakładają i aktualizują zbiory danych systemu, dla przykładu przedstawiono (rys. 2) schemat przetwarzania modułu nagrywającego ZBIÓR GŁÓWNY; podobną konstrukcję mają moduły zakładające ZBIÓR ARCHIWALNY i PLAN STUDIÓW; zbiory te mają znacznie prostszą budowę niż ZBIÓR GŁÓWNY, w związku z czym moduły zakładające służą tu równocześnie do aktualizacji; ZBIÓR GŁÓWNY jest aktualizowany dwoma modułami:

— dla informacji pochodzących z karty ewidencyjnej

— dla danych o wynikach nauczania pochodzących z kart zaliczeniowych

2) programy pomocnicze — służą do kontroli zbiorów danych systemu; znajduje się tu moduł kontrolujący kompletność i poprawność ZBIORU GŁÓWNEGO oraz programy, które w zredagowany sposób wypisują poszczególne zbiory w całości lub selektywnie; powyższe programy, a szczególnie bardzo rozbudowany moduł kontroli ZBIORU GŁÓWNEGO, są nieodzowne dla poprawnej pracy systemu

3) programy użytkowe; obejmują one:

— programy wydawnicze drukujące informacje o jednej lub kilku (do 10) osobach ze ZBIORU GŁÓWNEGO (rys. 3) lub ZBIORU ARCHIWALNEGO bądź określone informacje z PLANU STUDIÓW

— program sporządzający specjalne sprawozdanie dla GUS-u

EDYCJA ZBIORU GLOWNEGO ***KOSEWIST ***
DATA: 07/02/78
GODZINA: 15/56/35

LOS-MOSTOWSKI ANDRZEJ	NUMER ALBUMU 41302

URODZONY: 09/07/54 W STARGARD SZCZEC. WOJEWODZTWO: SZCZECIN	DOW. OSOBISTY: ZN 4648660
RODZICE: WALDEMAR	KS. WOJSKOWA: L 0137581
CZESLAWA Z.D. PILCHOWSKA	WKU: GLIWICE
POCHODZENIE: INT	KATEGORIA: A
GR. S-Z OJCA: PRC	

ADRES STALY:	34-450 STALOWA WOLA	KAWALERII	7/131 B
ADRES TYMCZAS.	44-100 GLIWICE	KUJAWSKA	2

Informacje o ukończonej szkole średniej		OCENY — MATURA		OCENY — EZGAMIN WSTEPNY	
TYP:	LICEUM DZIENNE	MATEMATYKA:	5.0	MATEMATYKA:	4.0
NAZWA:	4 L.O. IM. WLADYSLAWA IV	FIZYKA:	4.0	FIZYKA:	ZWOLN.
MIEJSCE:	WARSZAWA	JEZYK OBCY:	4.0	JEZYK OBCY:	3.5 LN.
ROK UKONCZ:	73				
SPECJALIZ.	MAFI				

DATA ROZPOCZECIA STUDIOW:	01/10/73	KIERUNEK:	E
TRYB ROZPOCZECIA:	REKRUTACJA	SPECJALIZACJA:	APIP
		GRUPA:	APP3

Rys. 3. Przykładowy wydruk z przebiegu listującego ZBIÓR GŁÓWNY. Dane na wydruku odpowiadają danym przedstawionym na karcie ewidencyjnej

-- programy drukujące listy studentów z podziałem na kierunki, semestry i grupy dziekańskie oraz listy osób zwolnionych z praktyki warsztatowej.

Wszystkie programy systemu (ok. 30) umieszczone są w bibliotece. Niewielka, jak dotąd, liczba programów użytkowych wynika stąd, iż główny nacisk położono na właściwe opracowanie grupy programów organizacyjnych. Rzetelne i pełne zbiory danych umożliwiają pisanie programów użytkowych. W tym też kierunku będzie rozwijany system KOSEWIST.

JĘZYK PROGRAMOWANIA

W większości przypadków zdecydowano się na opracowanie programów w języku COBOL. Język ten ma oczywiste zalety przy przetwarzaniu danych o charakterze ewidencyjnym. Ponieważ użyty system komputerowy nie narzucał zbyt wielkich ograniczeń co do rozmiaru programów, ze względu na prostotę najlepszym językiem był niewątpliwie COBOL. W systemie wykorzystano także do celów pomocniczych programy standardowe.

Przewiduje się, że w przyszłości liczba programów użytkowych zostanie ograniczona do podstawowych okresowych funkcji systemu, zaś wszelkie nowe, pojawiające się sporadycznie zadania związane z wyszukiwaniem informacji będzie można rozwiązywać za pomocą pakietu standardowego FIND. Dotychczas wykonane prace stwarzają podstawę do pełnego wdrożenia opisanego systemu.

Lubomir KIESZCZYŃSKI
COIG, Katowice

LITERATURA

[1] Regulamin organizacyjny Politechniki Śląskiej. Gliwice 1977

[2] System informatyczny dla rekrutacji kandydatów na I rok studiów na Wydziale Automatyki i Informatyki (nie publikowana praca Instytutu Konstrukcji i Technologii Urządzeń Automatyki i Informatyki Politechniki Śląskiej). Gwlice 1975

[3] Hanuszkiewicz Z., Lisiak K.: Opracowanie algorytmu i programu dla układania rozkładu zajęć Wydziału Automatyki i Informatyki przy użyciu komputera (praca magisterka). Gliwice 1977

[4] Kieszczyński L., Ogonowski Z.: Opracowanie pakietu programów dla celów komputeryzacji prac ewidencyjnych Dziekanatu Wydziału Automatyki i Informatyki (praca magisterska), Gliwice 1978

Zbigniew OGONOWSKI
Politechnika Śląska, Gliwice

Marnotrawstwo papieru w ośrodkach obliczeniowych bulwersuje coraz szerszy krąg osób, praktycznie bezsilnych wobec niektórych przyczyn tego zjawiska.

Chodzi tu głównie o rozrzutność w drukowaniu informacji pomocniczych przez oprogramowanie podstawowe, a także o uwarunkowany przez to oprogramowanie bardzo niski stopień zagęszczenia wydruków. Jakie to są ilości papieru możemy coraz częściej przekonać się w wielu naszych sklepach, gdzie wydruki te wykorzystywane są na opakowanie... Likwidacja takiego marnotrawstwa uzależniona jest jednak — jak to wskazują autorzy poniższych listów — od odpowiedniej modyfikacji oprogramowania podstawowego, co na pewno nie jest sprawą łatwą i prostą. Problem wymaga szybkiego rozwiązania — zwłaszcza wobec zmniejszonych ostatnio możliwości produkcyjnych naszego przemysłu papierniczego. Autorów i konserwatorów oprogramowania podstawowego zachęcamy do nadsyłania wypowiedzi na temat uzyskania również na tej drodze oszczędności papieru tabulogramowego.

Redakcja

Z uwagi na odczuwalne braki papieru (przedostatnie miejsce w druku książek w krajach socjalistycznych na jednego mieszkańca) artykuł Zbigniewa Ładosia drukowany w Informatyce 4/79 jest jak najbardziej na czasie. Prócz wymienionych przez Autora sposobów oszczędności zużycia papieru tabulogramowego chciałbym zasygnalizować jeszcze jeden, a mianowicie uzależniony od pewnych modyfikacji oprogramowania systemowego. Dotyczy to zarówno systemu DOS dla maszyn JS jak i systemu Komputer Biurowy dla minikomputerów serii MERA 300.

Tak więc nie wydaje się specjalnie uzasadnionym z punktu widzenia czytelności wydruku, czy rzeczywiście po niektórych kartach programów sterujących, a także po niektórych powtarzających się informacjach systemu, należy pozostawiać resztę strony niezadrukowaną. To samo odnosi się do przypadku wykonywania pewnych standardowych czynności, takich jak np. katalogowanie ciągu podprogramów. Również przy kompilowaniu programów, zwłaszcza liczących kilka instrukcji, mamy do czynienia ze szczególnie drastycznym marnotrawstwem papieru, np. 3 instrukcje programu w języku FORTRAN powodują — 4—9 wierszy informacji dostarczonych przez kompilator — rozproszonych aż na 4 stronach wydruku!

Oprogramowanie firmowe minikomputerów serii MERA-300 przyczynia się również do nadmiernego zwiększenia zużycia papieru, co tylko częściowo można usprawiedliwić małą pojemnością pamięci operacyjnej tych maszyn. Między innymi tekst programu realizowanego w systemie Komputer Biurowy uzyskujemy zawsze w postaci: 1 instrukcja — 1 wiersz, co powoduje nieuzasadnione zużycie, b. dużych ilości papieru, zwłaszcza przy poprawianiu i testowaniu programów. Te same uwagi odnoszą się do bezdyskowej wersji języka MINI-MERA-BASIC. Konkretnym przykładem możliwych oszczędności jest tzw. wersja „gdańska" tego języka, gdzie istnieje możliwość otrzywywania wyników w kilku kolumnach.

Na tle omówionych wyżej przykładów marnotrawstwa papieru, stosunkowo korzystnie prezentuje się system SOM-3 dla minikomputera MERA-400. Przewidziano w nim możliwości pomijania wydruku pewnych informacji, co nie tylko oszczędza papier, ale również zwiększa efektywność wykorzystania tego minikomputera.

Na zakończenie pragnąłbym poruszyć jeszcze inną sprawę. Do maszyn serii MERA-300 istnieje możliwość dołączenia pamięci taśmowej kasetowej PK-1, co pozwoliłoby również oszczędzić wiele taśmy papierowej i papieru tabulogramowego. Niestyty jak dotąd nie zostało to uwzględnione w systemie operacyjnym, uniemożliwiając praktycznie wykorzystanie tego urządzenia.

mgr Piotr SŁUGOCKI
Zakład Automatyki Przemysłu Cukrowniczego
Toruń

Zachęcony inicjatywą Pana Zbigniewa Ładosia z NBP chciałbym przedstawić swoje uwagi na temat oszczędności papieru.

Wielokrotnie usiłowałem wpłynąć na moich kolegów, a także zwierzchników, aby zainteresowali się możliwościami oszczędności papieru, lecz temat ten jest bardzo niepopularny.

Drukarki wierszowe DW produkcji MERA-BŁONIE, a także dostarczane przez firmę IBM mają przełącznik umożliwiający drukowanie tabulogramów z gęstością 6 lub 8 wierszy na 1 cal. Niestety nie udało mi się zauważyć, aby w jakimkolwiek ośrodku obliczeniowym drukowano tabulogramy ze zwiększoną gęstością druku. Argumenty na stosowanie gęstości 6 wierszy/cal, są m.in. takie, że stosowane kompilatory i parametry systemu operacyjnego przewidują drukowanie 55 wierszy na stronie. Modyfikacja parametrów określających liczbę wierszy na stronie jest zwykle bardzo prosta, zaś oszczędność może wynieść 30% papieru.

Często także drukuje się wielokrotnie te same informacje w różnych układach. Radykalnym rozwiązaniem byłoby zapisywanie na taśmie magnetycznej rekordów drukarskich i przeglądanie zawartości taśmy na monitorze ekranowym. Dołączona równolegle drukarka umożliwia wydrukowanie tylko potrzebnych w danym momencie stron.

Jak nieprzemyślane i lekceważone są sprawy oszczędności papieru świadczą choćby nowe wzory rachunków telefonicznych.

Jerzy DRYJAŃSKI
Ośrodek Informacji Technicznej i Przetwarzania Danych
Okręgowego Laboratorium Poczty i Telekomunikacji
Warszawa

Trudna sztuka programowania

Rola programów użytkowych, aczkolwiek nie mniej ważna od tzw. oprogramowania podstawowego i od samego sprzętu, jest w pewnym sensie mało doceniana. Brakuje też poważniejszych prac z zakresu metodyki tworzenia programów użytkowych. Co więcej — w zasadzie nie prowadzi się w kraju systematycznych badań z tej dziedziny. Pojęcie „metodologii programowania" jest mało znane i jeszcze mniej uprawiane jako odrębna gałąź dociekań poznawczych (jeżeli pominąć wydaną niedawno pracę W. S. Turskiego pod tymże tytułem).

Równocześnie jednak właśnie programowanie użytkowe jest być może najszerzej uprawianą dziedziną informatyki. Programują profesjonaliści w dużych usługowych ośrodkach obliczeniowych typu ZETO i w ośrodkach przedsiębiorstw lub instytucji. Programują „półzawodowcy" z małych pracowni informatycznych w biurach projektowych i konstrukcyjnych, w ośrodkach badawczych i w placówkach naukowych. Programuje również stale zwiększająca się rzesza użytkowników coraz łatwiej dostępnego sprzętu minikomputerowego.

W większości przypadków ludzie ci uczą się dość pobieżnie jakiegoś języka programowania, a następnie zaczynają eksperymentować. Każdy na własną rękę, według własnych predyspozycji psychicznych i zgodnie z własnym przygotowaniem zawodowym wypracowuje sobie swój warsztat programisty, jakąś swoją metodykę postępowania. Poszczególne ośrodki próbują na swój użytek ujednolicać pewne zasady postępowania. Jest to na ogół działalność mniej lub bardziej amatorska, a co gorsza wynikająca jedynie z lokalnych doświadczeń. Zbyt wiele spraw jest w tej sytuacji rozwiązywanych administracyjnie, poprzez wydanie odpowiednich instrukcji i zarządzeń, a nie w oparciu o pogłębione analizy metodyczne.

Dlatego dobrze, że redakcja WNT-owskiej serii „Biblioteka inżynierii oprogramowania" oddała do rąk polskiego czytelnika doskonale napisaną pracę **D. Van Tassela** [1]), która zdaniem autora „...jest przeznaczona dla tych, którzy potrafią już programować i chcą osiągnąć biegłość w programowaniu...". **Praktyka programowania** może być też doskonałym poradnikiem metodycznym dla wykładowców z zakresu programowania, aczkolwiek tak traktowany przedmiot jest w kraju słabo rozwinięty. Zwykle wykłada się taki czy inny język, zamiast mówić o programowaniu rozumianym szerzej niż umiejętność zgrabnego posługiwania się konkretną wersją danego języka.

Na podkreślenie zasługuje fakt, że pierwsze wydanie oryginału ukazało się w USA w roku 1974. Upłynęło więc wiele lat powszechnego stosowania komputerów i społecznego gromadzenia doświadczeń o procesie programowania użytkowego, zanim zaczęły ukazywać się prace metodyczne na ten temat.

Książka składa się z 6 rozdziałów i 3 dodatków, z których szczególnie ciekawy jest Dodatek II — przedruk artykułu D. H. H. Ingallsa na temat „Oceniania czasu wykonania (programu) w FORTRANIE".

Zasadniczą treść książki stanowi 5 pierwszych rozdziałów, w których kolejno omówiono podstawowe — zdaniem autora — aspekty procesu tworzenia programów użytkowych.

Nie jest to jednak pełna analiza tego procesu, gdyż autor omawia explicite tylko trzy jego etapy, czy też fragmenty: projektowanie programu (rozdział II), uruchamianie (rozdział IV) i testowanie (rozdział V). Dwa pozostałe rozdziały (I i III) zawierają liczne wskazówki, jak powinien program wyklądać, i rozważania na temat czego programista powinien unikać. Nie jest to jednak analiza etapu „pisanie programu". Niemniej właśnie te rozdziały zawierają najciekawszy, najwięcej dający do myślenia materiał metodyczny.

Zapewne wiele zasad podawanych przez D. Van Tassela może robić w pierwszej chwili wrażenie trywialnych, ale w rzeczywistości wartość tych zasad ocenią w pełni programiści, którzy już jakieś programy pisali, już popełnili liczne błędy w sztuce i boleśnie odczuli niedostatki własnego warsztatu.

Z licznych zaleceń podawanych w książce warto przykładowo zwrócić uwagę na dążenie do prostoty, trzymanie się raz ustalonych standardów, stosowanie licznych komentarzy, precyzyjne akapitowanie.

Prawie połowę książki stanowią rozdziały IV („Uruchamianie") i V („Testowanie"), zawierające liczne praktyczne wskazówki co do zasad postępowania i szereg ogólniejszych rozważań metodycznych. Autor bardzo silnie podkreśla różnice między tymi czynnościami, aczkolwiek nie przeczy, że na ogół przeplatają się one z sobą. W rozdziałach tych można spotkać pewne powtórzenia. Są to jednak zwykle konsekwentne rozwinięcia sygnalizowanych uprzednio problemów.

Reasumując, trzeba stwierdzić, że oprócz ogromnej liczby konkretnych wskazówek dla programisty udało się autorowi wykazać, że pisanie, a raczej tworzenie programów użytkowych jest problemem samym w sobie. Wbrew tytułowi „praktyka", książkę można by raczej nazwać „sztuka programowania". Porządkuje ona liczne problemy procesu tworzenia oprogramowania i stanowi niewątpliwie źródło inspiracji metodycznych.

Układ książki jest czytelny i jasny, dobrze dobrane liczne przykłady oraz ćwiczenia i zadania ułatwiają zrozumienie omawianych zagadnień. Na podkreślenie zasługuje również proponowana polska terminologia — co jest już zasługą tłumacza. Nazwy i określenia są na ogół celne, zgodne z duchem języka polskiego i wyrastają z powszechnie już używanej gwary informatycznej.

Książka roi się od lekkich, dowcipnych a celnych powiedzionek, które znacznie lepiej oddają myśl niż długie i przyciężkie wywody. A oto próbka takich „złotych myśli", mogących stać się dewizą pracy każdego programisty. Aby uświadomić czytelnikowi konieczność pisania prostych i czytelnych programów — co ułatwia ich rozumienie i późniejsze modyfikacje — Van Tassel twierdzi:

— *Program jest przeznaczony do czytania bardziej przez ludzi niż przez maszyny*
— *Czytelny program stwarza wrażenie, że autor wiedział co robi*
— *Rzadko jest za dużo komentarzy.*

Zdaniem Van Tassela zawsze warto zastanowić się:

— *Czy zaprogramowano właśnie to zadanie, które sformułowano*
— *Dlaczego nie mamy nigdy czasu, by to (program) dobrze zrobić, natomiast zawsze dosyć czasu, by to przerobić.*

Jak również warto sobie uświadomić, że:

— *Bez względu na ilość dostępnej pamięci, jest jej zawsze za mało*
— *Bezwzględny program jest abstrakcyjnym pojęciem teoretycznym*
— *Mało jest zajęć, które stwarzają tyle okazji do popełniania błędów w programowaniu*
— *Wszyscy programiści popełniają błędy*
— *Najlepszym sposobem usuwania błędów jest ich unikanie*
— *Błędny program jest wart mniej niż żaden*
— *Programista uczy się programowania, rzadko natomiast uruchamiania*
— *Produkowanie bezbłędnych programów jest zajęciem czasochłonnym.*

Na zakończenie można by tylko dodać uwagę pod adresem całej serii. Czy nie byłoby dobrze, żeby poszczególne pozycje zaopatrywane były w krótkie notki bibliograficzne o autorze, o innych jego pracach, zasadniczej dziedzinie naukowej, jaką uprawia. Dla czytelnika polskiego informacje takie mogą być szczególnie ciekawe.

Stanisława BONKOWICZ-SITTAUER

[1]) Dennie Van Tassel: Praktyka programowania. WNT, „Biblioteka Inżynierii Oprogramowania". Warszawa 1978, oryginał 1974 r. Z angielskiego przełożył Stanisław Szpakowicz. 6 rozdziałów, 3 dodatki, str. 230, rys. 14, tab. 3, poz. lit. 79, nakład 6250 egz., cena 50 zł

Piąte wydanie na temat FORTRANU

Ostatnio na półkach księgarskich ukazała się książka pt. „Programowanie w języku FORTRAN", której autorami są Jacek Bańkowski, Konrad Fiałkowski i Zbigniew Odrowąż-Sypniewski [1]).

FORTRAN jest dziś powszechnie stosowanym językiem automatycznego programowania maszyn cyfrowych, a według niektórych statystyk najpopularniejszym językiem w naszym kraju. Nic więc dziwnego, że ogromne jest zapotrzebowanie na publikacje o FORTRANIE — stąd piąte wydanie omawianej pozycji.

Książka wywołała duże zainteresowanie nie tylko wśród informatyków, ale również wśród wszystkich, którzy pragną w swojej pracy korzystać z możliwości współczesnych komputerów.

W stosunku do wydań poprzednich autorzy wprowadzili do obecnego wydania istotne rozszerzenia oraz pewne modyfikacje. Rozszerzenie materiału polega głównie na wprowadzeniu do pracy opisu tych cech języka FORTRAN, które są charakterystyczne dla realizacji w komputerach ODRA serii 1300, ICL serii 1900 i CDC CYBER 70 (modele 72, 73, 74) oraz CDC serii 6000. Ponadto autorzy rozbudowali i dostosowali do nowych realizacji zbiór przykładów.

Na dużą uwagę zasługuje zarówno sposób prezentowania pełnej wersji języka, jak i jego cech, które są charakterystyczne wyłącznie w realizacji FORTRANU dla określonych komputerów, polegające na użyciu druku w różnym kolorze. Sprawia to, że czytelnik może bardzo szybko zapoznać się z tymi elementami tego języka, które interesują go ze względu na konkretną realizację komputerową.

W kolejnych rozdziałach autorzy omawiają poszczególne elementy języka, począwszy od ogólnej struktury programów pisanych w tym języku poprzez liczby i stałe logiczne, zmienne, funkcje standardowe, wyrażenia logiczne i arytmetyczne, do instrukcji i realizacji cyklu. Dla wszystkich wyróżnionych elementów języka podane są proste przykłady z uwzględnieniem poszczególnych wersji dla określonych maszyn cyfrowych. Nieco szerzej została w książce omówiona realizacja cyklu, którą zakończono przykładem programu. W rozdziale tym autorzy zwracają ponadto uwagę na pewne nieprawidłowości mogące powstać przy konstruowaniu w FORTRANIE pętli programowej.

Najszerzej potrakowany jest rozdział dotyczący problemu instrukcji wejścia i wyjścia. Nie jest to chyba przypadkowe, gdyż większość programów pisanych w tym języku przez początkujących programistów oraz studentów zatrzymuje się właśnie na błędnych specyfikacjach deklaracji FORMAT.

Po omówieniu diagnozy błędów, oddzielnie dla wersji ICL 1900, ODRA 1300 i CDC CYBER 70 w rozdziale ostatnim umieszczono przykłady programów. Przykłady te przedstawiono kolejno według wzrastającego stopnia trudności, z uwzględnieniem cech istotnych w realizacji dla poszczególnych komputerów. Dotyczą one obliczeń naukowo-technicznych, takich jak np. tablicowanie amplitudy prądu zmiennego w szeregowym obwodzie RCL, czy też klasyfikacja 100 egzemplarzy określonego wyrobu.

Dodatkową cenną zaletą zbioru przykładów jest dołączenie do niektórych programów precyzyjnie określonej formy przygotowania danych.

Książka została zaopatrzona w indeks zwrotów, zestawienie funkcji standardowych oraz wykaz błędów „kompilacji" i „wykonanie" dla komputerów serii ODRA 1300 i ICL 1900. Zarówno sposób przedstawiania problemów, jak i umiejętne stopniowanie trudności w doborze treści świadczy o dużym dydaktycznym doświadczeniu autorów.

Sądzę, że dzięki opisanym rozszerzeniom treści zaspokoi ona nowe większe wymagania szybko rosnącego kręgu użytkowników tego podręcznika.

[1]) Bańkowski Jacek, Fiałkowski Konrad, Odrowąż-Sypniewski Zbigniew: Programowanie w języku FORTRAN. Wydanie piąte. Warszawa 1978, PWN.

Tadeusz SZUSZKIEWICZ

Bibliografia wydawnictw polskich z dziedziny informatyki

● **Projektowanie układów elektronicznych za pomocą maszyny cyfrowej — CALAHAN D. A. Tłum. wyd. ang. z 1972 r. WNT, Warszawa 1978, s. 302, cena 65 zł**

Projektowanie układów elektronicznych za pomocą maszyny cyfrowej. Analiza układów liniowych. Analiza stałoprądowa układów nieliniowych. Analiza stanów przejściowych układów dynamicznych. Obliczanie wrażliwości. Projektowanie automatyczne. Analiza tolerancji. Wprowadzenie do nowoczesnych metod układania równań układu elektronicznego. Nowoczesne metody numeryczne w analizie stanów przejściowych. Macierz rzadka — metody obliczeniowe oraz towarzyszące im zagadnienia. Metody optymalizacji układów elektronicznych. Obliczanie wrażliwości w dziedzinie czasu. Dodatki. Dowód twierdzenia Tellegena. Format danych wejściowych do programów analizy obwodów elektrycznych (RCAP, DCAP, TCAP). Format danych wejściowych do programów DSAP, DCOP. Programy.

Książka przeznaczona jest dla inżynierów elektroników — projektantów układów elektronicznych.

● **Informatyczne rozwiązywanie zadań matematycznych — NIEVERGELT J., FARRAR J. C., REINGOLD E. M. Tłum. wyd. ang. z 1974 r. WNT Warszawa, 1978, s. 239, cena 58 zł**

Co to jest wyrażenie arytmetyczne. Kombinatoryka a techniki komputerowe. Gry i podejmowanie decyzji. Procesy losowe w deterministycznych komputerach. Przetwarzanie liczb rzeczywistych. Co maszyny mogą, a czego nie mogą robić. Tematem książki są związki między informatyką i matematyką, pomocne w formułowaniu oraz rozwiązywaniu problemów.

Książka przeznaczona jest dla informatyków i inżynierów różnych specjalności. Z książki mogą też korzystać średnio zaawansowani czytelnicy, znający podstawy programowania.

● **Projektowanie urządzeń cyfrowych — WAGNER F. WNT, cykl „Automatyka", Warszawa 1978, s. 304, cena 55 zł**

Kodowanie liczb w systemie dwójkowym. Operacje arytmetyczne. Podstawowe elementy układów cyfrowych — bramki i przerzutniki. Komutatory. Dekodery. Liczniki. Rejestry przesuwające. Pamięci półprzewodnikowe. Pamięci ferrytowe. Pamięci magnetyczne kinetyczne. Realizacja części sterującej urządzenia. Układy arytmetyczne. Konwertery kodów. Realizacja zależności czasowych. Wyświetlanie informacji cyfrowej. Przetwarzanie cyfrowo-analogowe i analogowo-cyfrowe. Zadawanie sygnałów wejściowych. Współpraca elektronicznych elementów przełączających różnego rodzaju. Kanały wejście-wyjście komputera. Systemy zbierania informacji CAMAC. Przesyłanie sygnałów cyfrowych. Zagadnienia technologiczne konstrukcji urządzeń cyfrowych. Testowanie urządzeń cyfrowych.

Książka zawiera podstawowe wiadomości z zakresu projektowania urządzeń cyfrowych i jest przeznaczona dla inżynierów projektujących te urządzenia.

● **Minikomputery biurowe. Zastosowania w systemach informatycznych — NOWAKOWSKI A., OLEJNICZAK W. PWE, cykl „Informatyka w praktyce", Warszawa 1978, s. 167, cena 25 zł**

Zastosowania średniej techniki obliczeniowej. Charakterystyka minikomputerów biurowych. Konstrukcja modelu systemu informatycznego zarządzania z zastosowaniem minikomputerów biurowych. Wybrane możliwości zastosowań minikomputerów biurowych. Dodatek: przegląd maszyn średniej techniki obilczeniowej.

Materiały przeznaczone są dla projektantów systemów informatycznych zarządzania, kadry kierowniczej, pracowników służb ekonomicznych przedsiębiorstw i instytucji oraz studentów wyższych szkół ekonomicznych.

● **Tendencje w pozyskiwaniu informacji: 1980, 1985, 1990. Tłum. wyd. ang. z 1977 r. Wyd. Ośrodka Badawczo-Rozwojowego Informatyki, Warszawa 1978, s. 46, cena 92 zł. „Europejski Program Badawczy Diebolda". Zeszyt 94 (E 150)**

Wstęp i streszczenie na potrzeby personelu kierowniczego. Wprowadzenie klawiaturowe. Czytniki. Bezpośrednie ręczne wprowadzanie danych. Wprowadzanie danych głosem. Specjalne systemy wejścia: punkt sprzedaży detalicznej (POS), zbieranie danych produkcyjnych.

Oprac. A.K.

ANDRZEJ BRANDT
Centrum Projektowania i Zastosowań Informatyki
Warszawa

Administrator zastosowań

Rozwój zastosowań systemów informatycznych opartych na wspólnej bazie danych prowadzi do wyodrębnienia nowej jakościowo grupy problemów, których rozwiązywanie prowadzi do zmiany charakteru funkcji, jakie spełniają uczestnicy procesu projektowania i realizacji zastosowań bazy danych. Tak na przykład o powodzeniu w realizacji zastosowań bazy danych (także w warunkach polskich) decyduje m.in. ścisłe oddzielenie funkcji Administratora Bazy Danych od funkcji użytkownika rozumianych w sposób konwencjonalny [2], [5]. W takim ujęciu podstawowe obszary działania Administratora Bazy Danych obejmują:

- opis i organizację bazy danych: analiza wymagań różnych typów zastosowań z punktu widzenia struktur danych, koordynacja prac nad modelem danych, opis danych w formie schematu, koordynacja prac nad projektem zastosowań w części dotyczącej opisów cząstkowych bazy danych w formie podschematów, utrzymywanie aktualnego katalogu bazy danych
- kierowanie eksploatacją i rozwojem bazy danych: zapewnienie odpowiedniej jakości danych w bazie danych, bieżąca ocena efektywności eksploatacyjnej bazy danych, weryfikacja opisu struktury pamięci, reorganizacja struktury pamięci i rewizja struktury logicznej bazy danych
- dokumentację bazy danych: katalog bazy danych, standardy wykorzystania, procedury operacyjne eksploatacji, wyniki pomiarów eksploatacyjnych, opis procedur awaryjnych i systemu zabezpieczeń, identyfikacja użytkowników — hasła i klucze dostępu.

Szczegółowe rozważania na temat zakresu funkcji Administratora Bazy Danych można znaleźć w licznych publikacjach, przykładowo: [3, 4, 5, 6]. Przede wszystkim jednak rola Administratora Bazy Danych wymaga od niego idealnej znajomości i właściwego zrozumienia informacyjnych potrzeb użytkowników, które — dysponując odpowiednimi możliwościami — powinien zaspokoić [8].

Aby dobrze wykonywać podstawowe funkcje, tzn. koordynować i kontrolować organizację oraz sposób wykorzystania bazy danych, Administrator Bazy Danych 1° musi mieć odpowiednio wysokie kwalifikacje i 2° musi być wyposażony w odpowiednio rozległe kompetencje (z punktu widzenia struktury organizacyjnej objętej bazą danych).

KWALIFIKACJE ADMINISTRATORA BAZY DANYCH

Od kandydatów na Administratorów Bazy Danych wymaga się:

— biegłej znajomości potrzeb informacyjnych użytkowników wszystkich typów objętych strukturą organizacyjną (strukturami organizacyjnymi), przewidywaną do objęcia wspólną bazą danych
— biegłej znajomości wszystkich elementów struktury bazy danych
— biegłej znajomości mechanizmów systemu zarządzania bazą danych pośredniczących między Administratorem Bazy Danych a bazą danych
— praktyki w projektowaniu systemów informatycznych
— praktyki w programowaniu użytkowym (zastosowań)
— praktyki w programowaniu systemowym, tj. wykorzystywaniu możliwości systemu operacyjnego komputera
— praktyki w organizacji eksploatacji komputera (w tym specjalistycznego przygotowania technicznego w zakresie pomiarów eksploatacyjnych i metod optymalizacji cyklu przetwarzania).

W miarę rozwoju technologii bazy danych lista wymagań w odniesieniu do Admiinstratora Bazy Danych rośnie szybko i nieustannie. Już w chwili obecnej poziom kwalifikacji wymaganych od Administratora Bazy Danych jest tak wysoki, że istnieją poważne trudności ze znalezieniem osoby, która mogłaby sprostać swoim zadaniom — zwłaszcza w warunkach polskich, ale także w krajach, gdzie zastosowania bazy danych liczą się w tysiącach.

Różne są metody rozwiązania tego problemu:
— permanentne kształcenie i samokształcenie Administratora Bazy Danych
— wieloosobowa obsada stanowiska Administratora Bazy Danych
— przejęcie części funkcji Administratora Bazy Danych przez innych uczestników procesu projektowania i realizacji zastosowań bazy danych. Poniżej zajmiemy się bliżej możliwością przejęcia części funkcji Admiinstratora Bazy Danych przez Administratora Zastosowań.

ADMINISTRATOR ZASTOSOWAŃ W ARCHITEKTURZE SZBD GRUPY ANSI/SPARC

Pogłębienie rozdziału funkcji administracyjnych od funkcji użytkowych w systemie zarządzania bazą danych oraz wewnętrzny podział funkcji administracyjnych pojawiają się po raz pierwszy w architekturze SZBD, opracowanej przez Grupę ds. Baz Danych ANSI/SPARC w roku 1975. Opis architektury SZBD ANSI-SPARC zawierają m.in. publikacje: [1, 7].

W architekturze tej wprowadzono trzy poziomy opisu bazy danych:
1) schemat pojęciowy *(conceptual schema)* — modelujący rzeczywistość gospodarczą struktury organizacyjnej objętej wspólną bazą danych w dłuższym okresie czasu i bez wewnętrznych ograniczeń modelu (poza ograniczeniami dostępu do danych)
2) schemat wewnętrzny *(internal schema)* — odzwierciedlający fizyczną strukturę danych oraz strukturę pamięci
3) schemat(y) zewnętrzny(e) *(external schema)* — odzwierciedlające wycinki schematu wewnętrznego obejmujące zakresem poszczególne zastosowania. W dalszej części artykułu będziemy go nazywali podschematem.

W konsekwencji zdefiniowane zostały trzy grupy funkcji administracyjnych w systemie zarządzania bazą danych, reprezentowane przez trzy następujące bloki funkcjonalne (osoby wyposażone w odpowiednie mechanizmy pośredniczące):

1) administrator organizacji gospodarczej *(enterprise administrator)*, odpowiedzialny za zdefiniowanie schematu pojęciowego (definicja obiektów i ich atrybutów, relacji między obiektami oraz ograniczeń dostępu do danych)
2) Administrator Bazy Danych *(data base administrator)*, odpowiedzialny w tym przypadku za zdefiniowanie schematu wewnętrznego oraz organizację bazy danych i odpowiednią jakość danych w bazie danych (utrzymywanie bazy danych w stanie integralności)
3) administrator zastosowań *(application administrator)* — jedna lub większa liczba osób, odpowiedzialna za zdefiniowanie podschematu(ów) dla poszczególnego zastosowania (zastosowań): podschemat jest jedynym „mechanizmem pośredniczącym" między bazą danych a programistą zastosowań, a więc i programem użytkowym.

W związku z tym, że podschemat odzwierciedla wycinek schematu wewnętrznego, wykorzystywany w konkretnym zastosowaniu, a schemat wewnętrzny jest pełnym od-

zwierciedleniem schematu pojęciowego, podschemat nie może odwoływać się do tych obiektów i ich atrybutów (ani związanych z takimi obiektami danych), które nie zostały zdefiniowane przez administratora organizacji gospodarczej w schemacie pojęciowym, jak również nie zostały odwzorowane w schemacie wewnętrznym przez Administratora Bazy Danych.

W architekturze ANSI-SPARC prawidłowość opisu na poziomie podschematu jest więc zależna również od prawidłowości opisu dokonanego na obu wyższych poziomach opisu. Mechanizmem zapewniającym utrzymanie zgodności wewnętrznej opisu na wszystkich trzech poziomach jest katalog bazy danych *(data dictionary/directory facility)*.

Zdefiniowanie nowego schematu wewnętrznego bazy danych przez Administratora Bazy Danych powoduje konieczność redefinicji wyłącznie schematu pojęciowego bez konieczności redefinicji podschematów, a więc również bez konieczności dokonywania zmian w programach użytkowych. Aktualizacja katalogu bazy danych w przypadku zmiany schematu wewnętrznego odbywa się w architekturze ANSI/SPARC automatycznie.

Administrator Zastosowań w architekturze ANSI/SPARC przejmuje od Administratora Bazy Danych praktycznie wszystkie funkcje związane z realizacją zastosowań. Osiągnięcie pełnej niezależności programów od danych umożliwia niezależne działanie tych bloków funkcjonalnych.

ADMINISTRATOR ZASTOSOWAŃ W OBECNIE DZIAŁAJĄCYCH SZBD

Zrealizowanie systemu zarządzania bazą danych w oparciu o architekturę ANSI/SPARC przewiduje się na rok 1990. Aktualnie działające SZBD charakteryzują się jednym lub dwoma poziomami opisu bazy danych. W systemach zbudowanych według specyfikacji komitetu CODASYL, na przykład w polskim systemie RODAN — występują dwa poziomy opisu: poziom schematu i poziom podschematu. Systemy te, jak zresztą wszystkie aktualnie dostępne SZBD, zorientowane są przede wszystkim na programistę, dostarczając środków językowych dostosowanych do jego sposobu widzenia struktur danych.

Konieczność redefinicji podschematów w przypadku zmiany schematu bazy danych oraz brak możliwości automatycznej aktualizacji katalogu bazy danych (nawet jeśli prowadzony jest on całkowicie przy użyciu komputera) powodują, że wszystkie funkcje administracyjne skupione są w jednym ręku — Administratora Bazy Danych.

Ze względu na trudności praktyczne dotychczasowego rozwiązania wydaje się jednak konieczne przejęcie części funkcji administracyjnych przez Administratora Zastosowań już na obecnym etapie. Rola Administratora Zastosowań będzie z konieczności uboższa od przedstawionej w architekturze ANSI/SPARC, co nie oznacza jednak, że ma się ograniczać wyłącznie do odciążania Administratora Bazy Danych.

Administrator Zastosowań winien działać w obszarze jednego zastosowania bazy danych i spełniać następujące funkcje:
— określanie potrzeb informacyjnych użytkowników zastosowania
— współpraca przy opracowaniu modelu danych bazy danych
— dokonywanie opisów cząstkowych bazy danych w formie podschematów
— określanie prawa dostępu do danych dla użytkowników zastosowania
— prowadzenie pomiarów efektywności eksploatacyjnej wycinka bazy danych objętego zastosowaniem
— weryfikacja poprawności działania programów użytkowych zastosowania
— zgłaszanie Administratowi Bazy Danych postulatów dotyczących:

  nowych zapotrzebowań na dane dla zastosowania
  weryfikacji opisu struktury pamięci
  reorganizacji struktury pamięci
  rewizji struktury logicznej wycinka bazy danych objętego zastosowaniem
  potrzeby aktualizacji katalogu bazy danych

— prowadzenie dokumentacji zastosowania (jako składowej dokumentacji bazy danych).

Administrator Zastosowania powinien więc spełniać rolę nadrzędną w stosunku do wszystkich użytkowników zastosowania, a jednocześnie rolę łącznika między użytkownikami zastosowania i Administratorem Bazy Danych.

Kwalifikacje Administratora Zastosowania powinny być również wysokie, chociaż nie tak wygórowane jak Administratora Bazy Danych: może nim zostać dobry programista zastosowań, mający praktykę w projektowaniu systemów informatycznych i biegłą znajomość:
— struktury organizacji i obiegu informacji dotyczących zastosowania
— systemu zarządzania bazą danych
— elementów struktury danych wycinka bazy danych objętego (lub przewidzianego do objęcia) zastosowaniem.

W szczególnych przypadkach Administrator Zastosowań może działać w obszarze więcej niż jednego zastosowania. Niecelowe jest jednak przydzielanie kilku administratorów do jednego zastosowania.

LITERATURA

[1] The ANSI/SPARC DBMS Model. Proceedings of the Second SHARE Working Conference on DBMS Montreal (kwiecień 1976). North-Holland 1977

[2] Bogucki W., Staniszkis W.: Rola administratora bazy danych. INFORMATYKA nr 1/1975

[3] BCS/CODASYL DDLC-Data Base Administration Working Group. Report, czerwiec 1975

[4] Lyon J. K.: The Data Base Administrator. Willey Interscience Series, New York 1976

[5] Peck R. L.: The Data Base Administrator. Eurocomp Conference Procedings 1974

[6] SHARE Data Base Administration Comittee Report. SHARE secretary Distribution 246, czerwiec 1974

[7] Steel T. B. Jr. (chirman): ANSI/X3/SPARC Study Group On Data Base Management Systems Prec. IKEE vol. 63 no. 9, wrzesień 1975

[8] Wiederhold Gio: Database Design. Mc Graw-Hill Book Company. New York 1977

WITOLD REKUĆ
Instytut Organizacji i Zarządzania
Politechnika Wrocławska

Obiektowa struktura bazy danych

Szczególne zainteresowanie problematyką baz danych (dalej oznaczanych w skrócie BD) wyraża się — między innymi — powstawaniem wielu ich modeli. Modele te ujmują różne aspekty BD.

W aspekcie lingwinistycznym [4] BD jest rozpatrywana jako ciąg zdań pewnego języka. W ujęciu strukturalnym modele baz danych opisują zależności występujące w ich strukturze logicznej (np. model zaprezentowany w pracy [2]). Inne podejście przedstawiono w pracy [1], w której autorzy opisują mechanizm funkcjanowania banku danych za pomocą modelu teorii automatów.

Wydaje się, że równie istotne znaczenie dla analizy zasad budowy i wykorzystania BD ma aspekt semantyczny. Baza danych jest odwzorowaniem pewnego fragmentu świata rzeczywistego. Semantyka danych może być określana drogą porównania relacji między danymi z relacjami występującymi w odwzorowanym systemie. Taki pogląd jest podstawą budowy modelu semantycznego relacyjnych baz danych, przedstawionego w pracy [3]. Elementy tego modelu (obiekty niezależne, obiekty charakteryzujące, relacje charakteryzujące i asocjacje) są ściśle związane z elementami logicznej struktury bazy danych.

W niniejszym artykule przedstawiono model BD ujmujący semantykę danych w bazie niezależnie od sztucznego, jak się wydaje, podziału BD na logiczne jednostki danych, a także wprowadzono pojęcie obiektowej struktury bazy danych [1]).

BAZA DANYCH JAKO ODWZOROWANIE OBIEKTÓW ŚWIATA RZECZYWISTEGO

Pojęcie struktury obiektu jest pojęciem pierwotnym. Strukturę obiektu opisujemy prezentując pewne elementy (składowe) tego obiektu i relacje między nimi. Dlatego nie będziemy formułowali definicji obiektowej struktury bazy danych, lecz opiszemy ją za pomocą niżej wprowadzonych pojęć obiektu, klasy obiektów, ich atrybutów i relacji między obiektami.

Baza danych jest ciągiem zdań pewnego języka, czyli danych. Dane te są w określony sposób organizowane, to znaczy:

- bazę danych dzieli się na pewne skupienia danych [2])
- między tymi skupieniami (lub między niektórymi z nich) definiuje się powiązania.

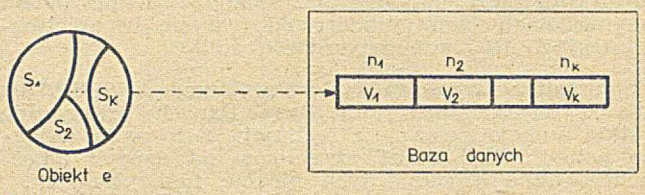
W taki sposób bazie danych nadaje się strukturę logiczną.

Najmniejszym skupieniem danych w BD jest dana elementarna.

Odwozrowywany w bazie danych fragment świata rzeczywistego może być traktowany jako system wzajemnie powiązanych obiektów. Obiekty tego systemu pogrupowane są w klasy jednorodnych obiektów. Każdy obiekt posiada skończony zbiór atrybutów prostych, które go charakteryzują. Przy tym obiekty tej samej klasy mają identyczną kolekcję atrybutów. W systemie występują określone relacje między obiektami, zdefiniowane ogólnie jako relacje na klasach obiektów.

Każdy atrybut S skojarzony z klasą E posiada nazwę i zbiór wartości. Konkretnemu obiektowi należącemu do klasy E przyporządkowana jest jedna wartość atrybutu.

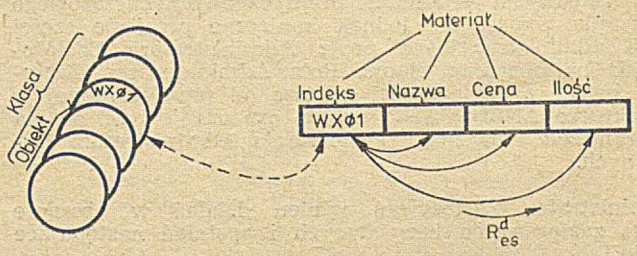
Odwzorowanie obiektu w bazie danych polega na zapamiętaniu w BD wartości jego atrybutu w jednym lub kilku [3]) elementach danych (rys. 1).



Rys. 1. Odwzorowanie obiektu w bazie danych

W ogólnym przypadku obiekty z klasy E są charakteryzowane zbiorem atrybutów. W rozpatrywanym systemie obiektów istnieje przyporządkowanie tego zbioru do obiektów klasy E, zwane tu relacją „obiekt-atrybut" i w określony sposób realizowane w bazie danych. Odwzorowanie to można pokazać następująco.

Niech atrybut s_1 będzie jednoznacznym identyfikatorem obiektu w klasie wybranym arbitralnie ze zbioru jego atrybutów. Wartości atrybutów obiektu (wśród nich również s_1) są zapamiętane w zbiorze elementów danych. Elementy te mogą być rozproszone w bazie danych, ale projektant BD musi zapewnić ich właściwe powiązanie. Oznacza to, że relacji „obiekt-atrybut" w systemie obiektów odpowiada w bazie relacja na wspomnianym wyżej zbiorze elementów danych, która realizowana jest drogą ustalenia powiązań (zazwyczaj programowych) między nimi. Brak tych powiązań uniemożliwia identyfikację opisu obiektu w bazie danych.



Rys. 2. Struktura modelu informacyjnego obiektu i jego związek z obiektem

Między obiektem świata rzeczywistego i odpowiadającymi mu elementami w BD istnieje więc związek, który można scharakteryzować następująco (rys. 2):

- element zawierający identyfikator obiektu, który jest bezpośrednio skojarzony z obiektem — przy założeniu, że znana jest klasa, do której obiekt należy
- pozostałe elementy — kojarzone są z obiektem pośrednio, dzięki logicznemu ich powiązaniu z identyfikatorem obiektu i identyfikatora z samym obiektem.

[1]) Obiektowa struktura bazy danych jest adekwatna do struktury systemu realnych obiektów, odwzorowanego w bazie danych

[2]) Termin „skupienie danych" oznacza tu ogólne pojęcie jednostki danych, będącej elementem logicznej struktury bazy danych (element, grupa danych, rekord, zbiór itp.)

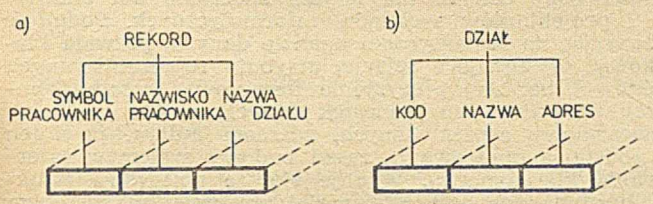
[3]) W tym drugim przypadku mamy do czynienia z kopiami danych, które z punktu widzenia prezentowanego modelu należy traktować jako jeden (logiczny) element

Analogicznie, klasy jako zbiory obiektów są odwzorowywane w BD w odpowiednich kolekcjach zbiorów danych elementarnych, na których (zbiorach) utworzone są relacje typu „obiekt-atrybut".

Relacja „obiekt-klasa" zwykle jest reprezentowana w BD poprzez powiązania nazw atrybutów tego obiektu z nazwą klasy, a mianowicie:

- nazwy atrybutów obiektów tworzy się zestawiając rzeczywistą nazwę atrybutu i nazwę klasy, np. NAZWA-DZIAŁU (rys. 3a) lub
- poprzez nadanie nazwy grupie elementów zawierających wartości atrybutów tego obiektu (rys. 3b).

wencji — relacje między klasami obiektów. Odwzorowanie



Rys. 3. Sposoby reprezentacji w bazie danych relacji przynależności obiektu do klasy drogą powiązania odwzorowania obiektu z nazwą klasy

Jak widać szczególną rolę spełnia tu schemat (opis bazy danych).

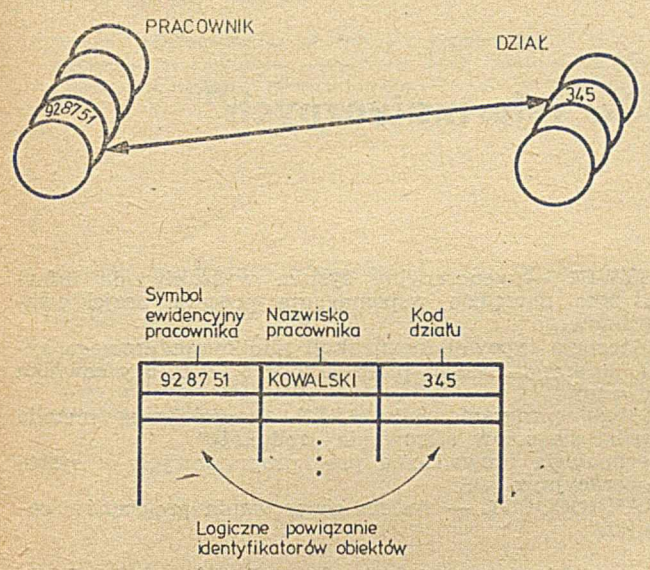
Jak już wspomniano w systemie realnych obiektów występują określone relacje między obiektami i, w konsekwencji, relacji w BD uzupełnia jego obraz informacyjny. Ogólnie relacje te można sklasyfikować w dwóch podstawowych grupach:

1) relacje asocjacyjne
2) relacje klasyfikacji.

Relacje klasyfikacji stanowią o podziale klas obiektów na podklasy, natomiast relacje asocjacyjne wynikają z określonych asocjacji między obiektami (np.: relacje „część-całość", „przyczyna-skutek" itp.).

RELACJE ASOCJACYJNE (R_A)

Relacje asocjacyjne między klasami E_i i E_j wynikają z istnienia relacji między obiektami z tych klas. Przynależność obiektów z klasy PRACOWNIK do obiektów z klasy DZIAŁ jest przykładem relacji tego typu. W szczególnym przypadku relacja asocjacyjna może być określona na jednej klasie obiektów, np. następstwo w czasie obiektów typu OPERACJA.



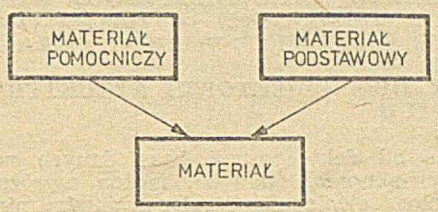
Rys. 4. Reprezentacja w bazie danych asocjacyjnej relacji między obiektami (przykład)

Relacje asocjacyjne między klasami obiektów są reprezentowane w BD przez logiczne powiązania identyfikatorów obiektów (rys. 4), tzn. między zbiorami danych elementarnych reprezentujących te obiekty. Relacje odwrotne są też relacjami asocjacyjnymi i ich odwzorowanie może odbywać się podobnie [4]).

RELACJE KLASYFIKACJI (R_K)

Między dwoma klasami obiektów E_i i E_j istnieje relacja klasyfikacji, jeżeli dowolny obiekt z klasy E_i jest elementem klasy E_j.

Relacje klasyfikacji zwykle nie są jawnie deklarowane w opisie bazy danych, tzn. nie stanowią elementu logicznej struktury BD. Często wynika to z dążenia do maksymalnej prostoty struktur bazy i związane jest z ograniczeniami języków opisu danych. Rozpatrzmy następujący przykład. Pewien zbiór rekordów opisuje klasę obiektów typu MATERIAŁ. Podział tej klasy na dwie podklasy: MATERIAŁ-POMOCNICZY i MATERIAŁ-PODSTAWOWY wymaga zdefiniowania dwóch podzbiorów rekordów opisujących wyznaczone podklasy. Taki podział może być dokonany na przykład w sieciowych bazach danych (rys. 5).



Rys. 5. Podział zbioru rekordów MATERIAL na dwa podzbiory poprzez utworzenie dwóch setów rekordów

Niezależnie od definicji relacji klasyfikacji w bazie danych, istnieją w BD takie relacje między danymi, które je reprezentują. Odpowiednikiem relacji klasyfikacji realnych obiektów są w bazie danych relacje zawierania się zbiorów danych elementarnych reprezentujących klasy tych obiektów. Analogiczne rozważania można przeprowadzić dla relacji klasyfikacji odwrotnych i dopełnieniowych.

KLASYFIKACJA ATRYBUTÓW OBIEKTÓW ODWZOROWYWANYCH W BAZIE DANYCH

Pokazany sposób interpretacji związków między danymi w BD pozwala wyróżnić pewne rodzaje atrybutów odwzorowywanych obiektów. Atrybuty obiektów należących do klasy E_j sklasyfikujemy ze względu na zależność ich wartości w stosunku do relacji danego obiektu systemu z innymi obiektami. Wyróżnimy tu:

- atrybuty własne (typu α) — atrybuty, których wartość dla dowolnego obiektu z klasy E_j nie zależy od relacji tego obiektu z pozostałymi obiektami systemu
- atrybuty zewnętrzne (typu β) — atrybuty, których wartość dla danego obiektu z klasy E_j uwarunkowana jest jego relacjami z innymi obiektami systemu.

Zanim przejdziemy do wyjaśnienia proponowanej klasyfikacji w grupie atrybutów typu α wydzielimy dwie podgrupy:

- identyfikatory (typ γ) — atrybuty własne, które jednoznacznie identyfikują obiekt w jego klasie
- atrybuty opisowe (typ δ) — wszystkie pozostałe atrybuty własne, które nie są identyfikatorami obiektu.

Zauważmy, że zakwalifikowanie atrybutów obiektów do poszczególnych grup nie może odbywać się na podstawie zawartości, którą dysponuje baza danych w krótkim okresie. Okres ten musi być dostatecznie długi, aby klasyfikacja atrybutów dla konkretnej BD (i systemu obiektów) nie zależała od składu klas i relacji w niej odwzorowywanych. W większości przypadków podstawą do klasyfikacji atrybutów może być doświadczenie projektantów i analityków systemów, wynikające z ich dostatecznej wiedzy o danym systemie.

[4]) Zwykle odwzorowanie relacji wprost implikuje odwzorowanie relacji odwrotnej. W przypadku relacji dopełnieniowych są one zazwyczaj ustalane proceduralnie

Przypuśćmy, że w BD są zdefiniowane rekordy o strukturze, którą określimy posługując się elementami opisu danych języka COBOL:

```
01 REKORD-1
   02 INDEKS-MATERIALU                (γ)
   02 NAZWA-MATERIALU                 (δ)
   02 CENA-MATERIALU                  (δ)
   02 JEDNOSTKA-MIARY-MATERIALU       (δ)
01 REKORD-2
   02 NUMER-MAGAZYNU                  (γ)
   02 INDEKS-MATERIALU                (γ)
   02 POZIOM-MATERIALU                (β)
```

W przykładzie tym w nawiasie określono typ atrybutu reprezentowanego przez każdy element w rekordach. Atrybut POZIOM-MATERIALU jest atrybutem zewnętrznym, ponieważ w ogólnym przypadku wartość tego atrybutu dla konkretnego obiektu z klasy MATERIAL zależy od relacji tego obiektu z obiektami klasy MAGAZYN.

Każdy atrybut zewnętrzny obiektu związany jest z określoną relacją klasy obiektów, od której zależy wartość tego atrybutu. W przyotocznym wyżej przykładzie jest to relacja < MAGAZYN, MATERIAL >. Artybur OCENA obiektu z klasy STUDENT zależy od związku czterech klas < SEMESTER, PRZEDMIOT, STUDENT, EGZAMIN >.

FORMALNY OPIS OBIEKTOWEJ STRUKTURY BAZY DANYCH

Niezależnie od dekompozycji bazy danych na logiczne (programowe) skupienia danych, można wyróżnić w BD określone zbiory elementów i relacje między nimi, będące odwzorowaniem obiektów i związków między obiektami świata rzeczywistego. Na tej podstawie można wnioskować o istnieniu szczególnej struktury BD, adekwatnej do struktury systemu obiektów, nazwanej tu strukturą obiektową. Formalnie, korzystając z form notacji przyjętych w teorii zbiorów i relacji, obiektową strukturę BD można opisać za pomocą pary:

$$< D^T, R^d >$$

gdzie:

$$R^d = R^d_{es} \cup R^d_A \cup R^d_K$$

D^T — zbiory danych elementarnych reprezentujące obiekty systemu

R^d_{es} — relacje typu „obiekt-atrybut" występujące na zbiorach elementów danych reprezentujących obiekty

R^d_A — relacje na zbiorach danych elementarnych reprezentujące relacje asocjacyjne między klasami obiektów

R^d_K — relacje reprezentujące w BD relacje klasyfikacji między klasami obiektów.

Obiektowa struktura zajmuje najwyższe miejsce w hierarchii struktur BD. Stanowi ona niezmiennik bazy danych, tzn. jest zachowywana przy reorganizacji jej struktury logicznej, gdy struktura systemu realnych obiektów odwzorowywanego w BD pozostaje stała.

* * *

Wydaje się, że przedstawiony tu sposób interpretacji zawartości bazy danych ma istotne znaczenie dla analityków i projektantów systemów informatycznych. Podejście to, bazujące na semantycznej analizie danych, pozwala klasyfikować występujące relacje, atrybuty reprezentowanych obiektów i bardziej precyzyjnie formułować zasady kompozycji podstawowych jednostek danych w BD.

Znaczeniowy aspekt danych, zdaniem autora, powinien być podstawą opracowania zasad budowy systemów informowania kierownictwa. Wydaje się, że obiektowa struktura bazy danych jest najbardziej zrozumiałym sposobem prezentacji modelu informacyjnego zawartego w bazie danych dla użytkowinka nie programującego. Wynika to z dobrej orientacji użytkownika w strukturze odwzorowywanego w BD systemu obiektów. Natomiast trudniej jest mu przyswoić sobie zasady odwzorowania obiektu drogą nadania BD struktury logicznej. Oczywiście budowa systemu informowania kierownictwa wymaga stworzenia opisu obiektowej struktury bazy danych (tezaurusa), efektywnego języka zapytań oraz jego interpretatora. Ich szczegółowe rozpatrzenie wykracza poza ramy niniejszego artykułu.

LITERATURA

[1] Balcerzak W., Subieta K.: Automat skończony jako model zawartości bazy danych. Prace IPI PAN, Nr 291, Warszawa 1977

[2] Durcholtz R. i inni: Das Datenmodel der „Feature Analysis of Generalized Date Base Management Systems". Agnew Inform. 1972, vol. 14, No 12

[3] Schmid H. A., Swenson J. R.: One the Sementics of the Relational Data Model. ACM-SIGMOND Conf. May 1975, ACM New York 1975

[4] Subieta K.: Lingwinistyczny model bazy danych. Cz. I i II. Prace CO PAN, Nr 257, Warszawa 1976

„Techniki i metody tworzenia systemów informatycznych"

Zjednoczenie Informatyki, Ośrodek Doskonalenia Kadr i Postępu Technicznego NOT w Lublinie oraz Zakład Elektronicznej Techniki Obliczeniowej w Lublinie organizują w listopadzie br. dwudniową konferencję nt. „Techniki i metody tworzenia systemów informatycznych".

Tegoroczna konferencja będzie już trzecim spotkaniem informatyków poświęconym tej tematyce. Będzie ona okazją do ogólnokrajowej wymiany doświadczeń i rozpropagowania nowych osiągnięć, a także inspiracją do prowadzenia dalszych prac nad doskonaleniem systemów informatycznych, a więc przyczyni się do wzrostu znaczenia komputera w życiu społeczno-gospodarczym naszego kraju.

Tematyka zaprezentowanych prac obejmuje bardzo interesujący wachlarz zagadnień. Organizatorzy przedstawiają uczestnikom konferencji m.in. następujące referaty:

- „SYWIN — generacyjny system wyszukiwania informacji jako narzędzie wspomagania procesu projektowo-programowego"
- „Koncepcja typowej konstrukcji minikomputerowego systemu informatycznego przeznaczonego dla użytkownika końcowego"
- „System oprogramowania STEP — efektywna metoda tworzenia systemów sterowania produkcją"
- „Technologia prowadzenia prac programowych w systemie operacyjnym OS"
- „MAKROCOBOL — system generowania programów cobolowych"

W sprawach dotyczących konferencji informacji udziela Ośrodek Doskonalenia Kadr i Postępu Technicznego NOT, ul. M. Curie-Skłodowskiej 3, 20-919 Lublin, tel. 277-56.

Łamy INFORMATYKI otwarte dla wszystkich!

Zanim jednak nasi Autorzy sięgną po pióro, prosimy, by zechcieli zapoznać się z poniższymi informacjami.

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

W artykułach można omawiać, prezentować lub proponować wszystko, co dotyczy współczesnej informatyki, oraz wszystko, co wiąże się z jej kierunkami rozwoju — zarówno z pozycji informatyka, jak i użytkownika informatyki.

Materiał, oprócz tekstu zasadniczego, powinien zawierać — na oddzielnych stronach — kartę tytułową (strona 1), krótki życiorys zawodowy autora (strona 2) i jego zdjęcie, wykaz literatury, tabele, rysunki, podpisy pod rysunki, zdjęcia.

Na stronie 1 należy podać tytuł naukowy, imię i nazwisko, nazwę zakładu pracy, adres prywatny i telefon, tytuł artykułu oraz informację, jaką drogą przesłać honorarium po opublikowaniu artykułu: kasa Wydawnictwa, poczta, bank (w takim przypadku prosimy podać numer konta PKO).

Konstrukcja artykułu powinno być zwarta i przejrzysta; wstęp musi wprowadzić czytelnika w zagadnienie, w podsumowaniu należy sformułować wnioski; podział na rozdziały, podrozdziały i akapity powinien być logiczny i konsekwentny. Należy zwrócić szczególną uwagę na poprawność stylistyczną i terminologiczną, unikać skrótów, rzadko stosowanych wyrażeń obcych i żargonu fachowego; starannie definiować nowe terminy. Należy również wystrzegać się nieczytelnych i zbyt rozbudowanych wzorów.

Tekst powinien być napisany na maszynie, jednostronnie, na papierze nieprzebitkowym formatu A-4, z marginesem 5 cm (30 wierszy na 1 stronie, 60 znaków w 1 wierszu).

Wykaz literatury powinien zawierać: kolejny numer pozycji (w nawiasie kwadratowym), nazwisko i imię autora, tytuł publikacji (książki lub artykułu), ewentualnie tytuł i numer czasopisma (w przypadku artykułu), miejsce i rok wydania.

Tabele — każda na oddzielnej stronie — powinny być numerowane i opatrzone tytułem oraz ściśle związane z tekstem (odniesienie na marginesie).

Rysunki — każdy oddzielnie (uwaga: nie wklejać rysunków w tekst!) — powinny być czytelne i również ściśle związane z tekstem (odniesienie na marginesie). Format rysunku nie może być mniejszy niż 10 × 10 cm.

Podpisy pod rysunkami, napisane również na oddzielnej stronie, oprócz kolejnego numeru powinny zawierać tytuł rysunku i ewentualnie legendę dotyczącą poszczególnych elementów.

Łączna objętość materiału nie powinna przekraczać w przypadku

- artykułu problemowego — 12 stron
- reportażu — 8 stron
- recenzji, relacji z imprezy — 6 stron
- informacji — 4 stron maszynopisu

Tak przygotowany materiał prosimy dostarczyć w dwóch egzemplarzach pod adresem: redakcja INFORMATYKI, ul. Jasna 14/16, 00-041 Warszawa. Wszelkich dodatkowych informacji udzielamy pod telefonem 27-71-40.

Autor opublikowanego w INFORMATYCE artykułu otrzymuje bezpłatnie egzemplarz okazowy.

Materiałów nie zakwalifikowanych do druku redakcja nie zwraca.

Isotimpex

Elektroniczne rejestrujące urządzenie kasowe

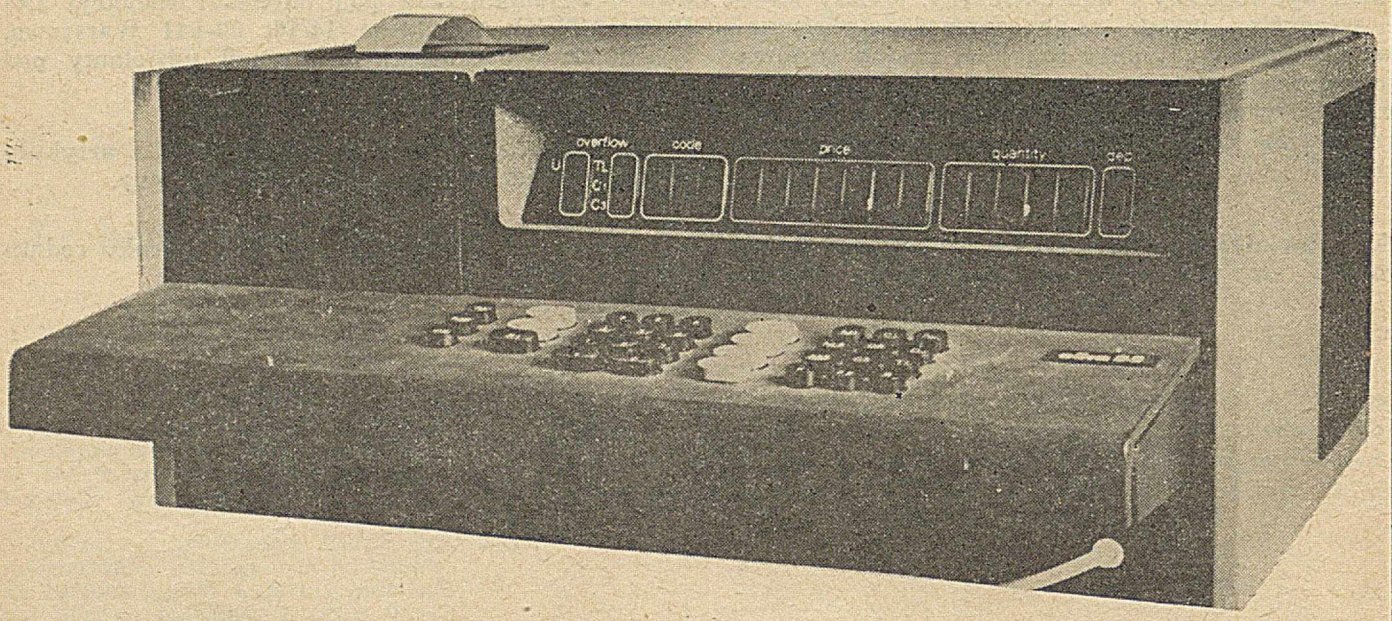
„ELKA 88"

Dokładna, szybka, niezawodna obsługa i rozliczenie w zakładach zbiorowego żywienia.

Nowoczesna technologia produkcji oparta na wykorzystaniu mikroprocesorów i ferrytowej pamięci operacyjnej 4 k × 4 bity — oto podstawa dużych możliwości operacyjnych rejestracji kasowej za pomocą urządzenia „ELKA 88".

DANE TECHNICZNE:

— 200 rejestrów pamięci ceny jednostkowej
— 200 rejestrów pozycji rachunku
— przeznaczony dla 7 grup kelnerów, dla każdej po 4 rejestry pamięci — kuchnia, bufet, anulowanie, liczba obsłużonych klientów
— magnetycznie zakodowane klucze dla każdego kelnera
— anulowanie błędnego zamówienia o równowartość zwracanych pozycji
— rozrachunek według brygad, asortymentu i całej kasy
— drukuje 3 czeki — dla klienta, dla kuchni i dla bufetu
— automatyczna blokada przy przekroczeniu pojemności rejestrów
— drukarka SEIKO — 2,5 wiersza/sekundę
— wyświetlanie — 17-pozycyjne, o kolorze zielonym; wyświetlacze segmentowe
— wymiary — 230 × 530 × 560 mm
— masa — 31 kg



Eksporter:

Przedsiębiorstwo Handlu Zagranicznego ISOTIMPEX
Bułgaria, Sofia, ul. Czapajewa 51. Telefon: 73-61. Teleks: 022731, 022732

WCT/732/K/79