10 1980

informatyka

WYDAWNICTWO NOT SIGMA

Proponujemy współpracę

Piszcie!

Środowisko informatyków, do którego przede wszystkim adresowana jest INFORMATYKA, mimo że formalnie dość jednolite, sprawia wrażenie zdezintegrowanego. Postępujący coraz szybciej podział na wąskie specjalizacje i sfery zainteresowań może w rezultacie spowodować zanik kontaktów pomiędzy poszczególnymi grupami. Informatyka jako całość stałaby się wtedy dziedziną abstrakcyjną, w ramach której praktyka ograniczona by została wyłącznie do wycinkowych prac konstruktorów sprzętu i oprogramowania, projektantów lub analityków systemów.

Powolny przepływ informacji i wspomniane trudności wymiany poglądów pomiędzy informatykami mogą doprowadzić nie tylko do znacznych strat gospodarczych i opóźnień technologicznych, ale także do następstw typu psychicznego — obojętności i zniechęcenia. Drogą do dającego satysfakcję współdziałania jest przecież przede wszystkim porozumienie.

INFORMATYKA, jako pismo fachowe, jest szansą dla środowiska. Jej łamy, otwarte dla wszystkich zainteresowanych, powinny zostać w większym niż dotychczas stopniu wykorzystane do szerokiej wymiany doświadczeń i poglądów.

Piszcie do nas o wszystkim, co w ramach informatyki wydaje Wam się ważne. Spróbujmy razem opisać rzeczywisty stan tej dziedziny w Polsce. Ułatwi to niewątpliwie ujawnienie istniejących możliwości i ograniczeń, a także spowoduje, że pojęcie „informatyka" stanie się czymś realnym, a w afekcie — w pełni zrozumiałym dla całego społeczeństwa. Ze swojej strony gwarantujemy każdą, możliwą w warunkach Redakcji pomoc.

W rubryce POGLĄDY będziemy zamieszczać rzeczowe, wnikliwe wypowiedzi, dotyczące najważniejszych problemów informatyki, nowych zastosowań, interesujących pomysłów i koncepcji. Znajdą tu miejsce dyskusje, wywiady, a także omówienia ważkich problemów i osiągnięć zagranicznych.

Jest ponadto rubryka LISTY, którą wykorzystać może każdy dla przekazania swoich racji — niekoniecznie z pozycji wybitnego eksperta. Spodziewamy się tutaj między innymi głosów krytycznych, informujących o wszelkich nieprawidłowościach, z jakimi niestety często stykają się użytkownicy informatyki, a także uwag oraz postulatów w stosunku do treści naszego pisma.

Zatem — czekamy.

REDAKCJA

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pokój 326, tel. 27-71-40, dyżury redakcji 10.00—13.00

Zakł. Graf. „Tamka". Zam. 361. Papier druk. sat. V kl. 70 g. A1. Obj. 5 ark. druk. Nakład 7000 egz. O-66.

Cena egzemplarza zł 30.— INDEKS 36124 Prenumerata roczna zł 360.—

Brady P. R., Rudak B., Rudakowa B.: Połączenie lokalne dwu maszyn cyfrowych ODRA 1305
INFORMATYKA 1980, nr 10, s. 4

Szczegółowa charakterystyka rozwiązań sprzętowych i programowych lokalnego połączenia dwu komputerów ODRA 1305, zrealizowanego w Centrum Obliczeniowym Politechniki Wrocławskiej. Omówiono wyniki eksploatacji próbnej systemu oraz uzyskane efekty.

Брады П. Р., Рудак Б., Рудакова Б.: Местные связи двух цифровых машин Одра 1305
ИНФОРМАТИКА 1980, № 10, стр. 4

Подробная характеристика решений технологического оборудования и программных решений местной связи двух вычислительных машин Одра 1305, созданной в Вычислительном Центре Вроцлавского Политехнического Института. Обсуждаются результаты опытной эксплуатации системы и полученные эффекты.

Drabent W.: Definiowanie języków programowania — Druga Metoda Wiedeńska (składnia)
INFORMATYKA 1980, nr 10 s. 7

Prezentacja sposobu precyzyjnego definiowania języków programowania w oparciu o Drugą Metodę Wiedeńską (Vienna Development Method). Jest to pierwszy z dwóch artykułów na ten temat i dotyczy definiowania składni.

Драбэнт В.: Определение языков программирования — Второй венский метод (синтаксис)
ИНФОРМАТИКА 1980, № 10, стр. 7

Представление способа точного определения языков программирования на основе Второго венского метода (Vienna Development Method). Это первая из двух статей на эту тему, касается она определения синтаксиса.

Roczniak J.: Projektowanie programów strukturalnych metodą Jacksona
INFORMATYKA 1980, nr 10, s. 10

Charakterystyka podstaw projektowania programów strukturalnych metodą Jacksona. Omówiono stosowaną symbolikę oraz podano przykład ilustrujący proces konstruowania programów.

Рочняк Й.: Проектирование структуральных программ методом Jacksona
ИНФОРМАТИКА 1980, № 10, стр. 10

Характеристика основ проектирования структуральных программ методом Jacksona. Обсуждается применяемая символика и дается пример иллюстрирующий процесс конструирования программ.

Gospodarowicz A.: Możliwości automatyzacji układania harmonogramu zajęć

INFORMATYKA 1980, nr 10, s. 12
Omówienie i matematyczne uzasadnienie istoty trudności komputerowego układania harmonogramu zajęć szkolnych. Wskazano na przydatność stosowania do tego celu metody heurystycznej oraz podano sposoby formułowania zadań.

Господарович А.: Возможности автоматизации составления графиков занятий
ИНФОРМАТИКА 1980, № 10, стр. 12

Обсуждение и математическое обоснование сущности затруднений автоматического составления графиков школьных занятий. Указывается на пригодность применения для этой цели евристического метода и даются способы формулировки задач.

Bracha J., Fortecki J., Kubale M.: ROZA — system układania rozkładów zajęć
INFORMATYKA 1980, nr 10, s. 14

Szczegółowa charakterystyka rozwiązań systemu układania rozkładów zajęć szkolnych ROZA, zrealizowanego w Instytucie Informatyki Politechniki Gdańskiej do stosowania na komputerach serii ODRA 1300. Omówiono warunki i dotychczasowe wyniki eksploatacji systemu, a także perspektywy jego dalszego rozwoju.

Браха Й., Фортэцки Й., Кубале М.: ROZA — система составления расписаний занятий
ИНФОРМАТИКА 1980, № 10, стр. 14

Подробная характеристика решений системы составления расписаний школьных занятий ROZA, созданной в Институте вычислительной техники Гданского политехнического института для применения на вычислительных машинах серии Одра 1300. Обсуждаются условия и до сих пор полученные результаты эксплуатации системы, а также перспективы ее дальнейшего развития.

Roszkowski J.: Określenie i realizacja spójności językowej i informacyjnej systemów informatycznych
INFORMATYKA 1980, nr 10, s. 16

Charakterystyka metody określania spójności językowej i informacyjnej systemów informatycznych przy wykorzystaniu aparatu lingwistyki matematycznej. Podano sposób technicznej realizacji tej metody w Resortowym Ośrodku Informatyki Przemysłu Chemicznego ETOCHEM.

Рошковски Й.: Определение и реализация языковой и информационной связности вычислительных систем
ИНФОРМАТИКА 1980, № 10, стр. 16

Характеристика метода определения языковой и информационной связности вычислительных систем при использовании аппарата математической лингвистики. Дается способ технической реализации этого метода в Отраслевом Центре вычислительной техники химической промышленности ЭТОХЭМ.

Korpal A., Kubiak A.: Architektura systemów wielomikroprocesorowych
INFORMATYKA 1980, nr 10, s. 19

Charakterystyka systemów wielomikroprocesorowych, decydujących o postępie w konstrukcji współczesnych systemów komputerowych. Omówiono podstawy rozwiązań sprzętowych, systemów operacyjnych, komunikacji wewnętrznej oraz sterowania zasobami systemu.

Корпаль А., Кубяк А.: Архитектура многомикропроцессорных систем
ИНФОРМАТИКА 1980, № 10, стр. 19

Характеристика многомикропроцессорных систем решающих о прогрессе в конструкции современных вычислительных систем. Обсуждаются основы решений технического оборудования оперативных систем внутреннего сообщения и управления ресурсами системы.

Bylicki A.: Szybkość efektywna transmisji synchronicznej
INFORMATYKA 1980, nr 10, s. 24

Omówienie źródeł ograniczenia przepustowości informacyjnej, jakie stanowią łącza transmisji danych (linie telefoniczne i modemy). Podano praktyczny sposób określania oraz uzyskane wyniki pomiarów efektywnej szybkości transmisji synchronicznej.

Былицки А.: Эффективная скорость синхронной передачи
ИНФОРМАТИКА 1980, № 10, стр. 24

Обсуждение источников ограничения пропускной способности информации, какой является сеть передачи данных т(елефонные линии и модемы). Указывается практический способ определения и полученные результаты измерений эффективной скорости синхронной передачи.

Brady P. R., Rudak B., Rudakowa B.: Local connection of two ODRA 1305 computers
INFORMATYKA 1980, No 10, p. 4

Detailed characteristics of hardware and software solutions of the local connection of two ODRA 1305 computers, realized in the Computing Center of the Wrocław Technical University. Discussed the results of the system's experimental operation and achieved effects.

Brady P. R., Rudak B., Rudakowa B.: Lokale Verbindung von zwei ODRA 1305 Rechnern
INFORMATYKA 1980, Nr. 10, S. 4

Ausführliche Charakteristik der Hardware- und Softwarelösungen der lokalen Verbindung von zwei ODRA 1305 Rechnern, die im Rechenzentrum der Wroclawer Technischen Universität realisiert wurde. Es wurden die Ergebnisse des Systemsprobebtriebes und die erzielten Effekte besprochen.

Drabent W.: Defining of programming languages — the Vienna Development Method (Syntax)
INFORMATYKA 1980, No 10, p. 7

Presentation of the way for precise programming languages defining used by the Vienna Development Method. It is the first of the two articles concerning this problem and is devoted to syntax definition.

Drabent W.: Definieren von Programmierungssprachen — die Zweite Wiener Methode (die Syntax)
INFORMATYKA 1980, Nr. 10, S. 7

Die Vorstellung eines Verfahrens für präzise Definieren der Programmierungssprachen, gestützt auf der Zweiten Wiener Methode (the Vienna Development Method). Es ist der erste von zwei Artikeln zu diesem Thema und betrifft der Syntaxdefinierung.

Roczniak J.: Structural programs designms using Jackson's method
INFORMATYKA 1980, No 10, p. 10

Characteristics of structural programs designing principles using Jackson's method. Discussed applied symbolic representation and presented an example illustrating the process of programs construction.

Roczniak J.: Projektierung der strukturellen Programme mit Verwendung der Jackson's Methode
INFORMATYKA 1980, Nr. 10, S. 10

Die Charakteristik der Grundsätze für die Strukturalprogrammenprojektierung mit Verwendung der Jackson's Methode. Es wurde die verwendete Symbolik besprochen und ein Beispiel für die Veranschaulichung des Programmgestaltungsprozesses angegeben.

Gospodarowicz A.: Automation possibilities of the school schedule arrangement
INFORMATYKA 1980, No 10, p. 12

Presentation and mathematic justification of the school schedule arrangement difficulties. Pointed out usefulness of heuristic method application for this purpose, as well presented methods of task formulation.

Gospodarowicz A.: Automatisierungsmöglichkeiten der Stundenplanerstellung
INFORMATYKA 1980, Nr 10, S. 12

Besprechung und die mathematische Begründung der Schwierigkeiten, die bei der Studenplanerstellung vorkommen. Es wurde die Brauchbarkeit der heuristischen Methode gezeigt, sowie die Weise der Aufgabeformulierung angegeben.

Bracha J., Fortecki J., Kubale M.: ROZA — the system for school schedule arrangement
INFORMATYKA 1980, No 10, p. 14

Detailed characteristics of the school schedule arrangement system ROZA, realized in the Data Processing Institute of Gdańsk Technical University for ODRA 1300 computer series. Discussed conditions and hitherto results of the system operation, as well prospects for future development.

Bracha J., Fortecki J., Kubale M.: ROZA — ein System für die Stundenplanerstellung
INFORMATYKA 1980, Nr 10, S. 14

Die ausführliche Charakteristik des Stundenplanerstellungsystems ROZA, das im Datenverarbeitungsinstitut der Technischen Universität in Gdańsk für die Verwendung auf ODRA 1300 Rechnerserie realisiert wurde. Es wurden die Bedingungen und die bisherigen Ergebnisse der Systemausnutzung, sowie die Aussichten für ihre weitere Entwicklung, besprochen.

Roszkowski J.: Designation and realization of the data procesing systems linguistic and informative compatibility
INFORMATYKA 1980, No 10, p. 16

Characteristics of the designation method for data processing systems linguistic and informative compatibility using methematical linguistic apparatus. Presented the way for technologic realization of the method in the Ministerial Data Processing Center of Chemical Industry ETOCHEM.

Roszkowski J.: Bezeichnung und Realisierung der Sprach- und Informationskompatibilität von den EDV-Systemen
INFORMATYKA 1980, Nr 10, S. 16

Die Charakteristik der Bezeichnungsmethode für die EDV--Systeme Sprach- und Informationskompatibilität mit Verwendung des mathematischen Linguistikapparats. Es wurde die Arbeitsweise der technischen Realisation dieser Methode im EDV-Resortsrechenzentrum für chemische Industrie ETOCHEM angegeben.

Korpal A., Kubiak A.: Multiprocessor systems architecture.
INFORMATYKA 1980, No 10, p. 19

Characteristics of the multiprocessor systems, which determine the progress of contemporary computer systems construction. Discussed solution principles of the hardware, operating systems, internal communication and systems resources control.

Korpal A., Kubiak A.: Die Architektur der Multiprocessorsysteme
INFORMATYKA 1980, Nr. 10, S. 19

Die Charakteristik der Multiprozessorsysteme, die für den Konstruktionsfortschritt der gegenwärtigen Rechnersysteme entscheidend sind. Es wurden die Lösungsgrundsätze für die Hardware, Betriebssysteme, innere Übertragung und Systemsbetriebsmittelsteuerung besprochen.

Bylicki A.: Effective speed of synchronic data transmission
INFORMATYKA 1980, No 10, p. 24

Discussion of information capacity limitation sources, which makes data links (communication lines and modems). Presented the practical method for designation of the synchronic data transmission effective speed and achieved measurement results.

Bylicki A.: Effektive Geschwindigkeit der synchronen Datenübertragung
INFORMATYKA 1980, Nr. 10, S. 24

Besprechung der Begrenzungsquellen, die über die Informationskapazität der Datenübermittlungsabschnitte (Fernmeldelinien und Modems) entscheiden. Es wurde die praktische Arbeitsweise für die Bezeichnung der effektiven Geschwindigkeit von Synchrondatenübertragung, sowie die erzielten Bemessungenergebnisse angegeben.

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 10
MIESIĘCZNIK
1980
ROK XV
Październik

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

PHILIP R. BRADY
Computer Centre
University College of Swansea

BRONISŁAW RUDAK
BARBARA RUDAKOWA
Centrum Obliczeniowe
Politechniki Wrocławskiej
Wrocław

Połączenie lokalne dwu maszyn cyfrowych ODRA 1305

W Centrum Obliczeniowym Politechniki Wrocławskiej prowadzone są m.in. prace nad metodami lokalnej i zdalnej komunikacji między systemami cyfrowymi typu ODRA. W publikacji [1] przedstawiono koncepcję budowy systemu połączeń lokalnych i zdalnych między maszynami cyfrowymi typu ODRA serii 1300. Poniższe opracowanie zawiera opis programowego wykorzystania lokalnego połączenia dwu systemów ODRA 1305 przez adapter międzymaszynowy dla celów przeniesienia pewnych funkcji (programów) systemu operacyjnego GEORGE 3 na drugą maszynę cyfrową, pracującą pod kontrolą programu sterującego (egzekutora). Zakłada się, że do celów komunikacji międzymaszynowej służy adapter międzymaszynowy typu ADM 305/1 (ICL 7210), którego charakterystykę techniczną zawiera opracowanie [3]. Technicznie jest to połączenie bliskie przez standardowy interfejs.

W programowaniu łącza międzymaszynowego stosuje się zasady obwiązujące w systemach ICL dla łącz międzyprogramowych, które są obsługiwane przez programy sterujące (egzekutory) jako pseudourządzenia pracujące w trybie bezpośredniej odpowiedzi (ang. *direct response mode*). Komunikacja między programami w różnych maszynach cyfrowych jest rozważana jako rozszerzenie komunikacji między programami znajdującymi się w pamięci jednej maszyny cyfrowej.

W celu przeniesienia na drugi procesor części funkcji obsługi transmisji z urządzeniami terminalowymi zastosowano pakiet obsługi urządzeń teletransmisji firmy ILC (Communication Manager) i opracowano moduły komunikacji przez łącze międzymaszynowe. Dzięki takiemu rozwiązaniu zwiększono liczbę urządzeń zdalnych, obsługiwanych przez system GEORGE 3 oraz uzyskano przyspieszenie obsługi urządzeń, angażując dla tych celów około 5% czasu procesora drugiej maszyny i około 9 K słów jej pamięci operacyjnej.

ROZDZIAŁ FUNKCJI OBSŁUGI URZĄDZEŃ ZDALNYCH W SYSTEMIE OPERACYJNYM GEORGE 3 POMIĘDZY DWIE MASZYNY ODRA 1305

W systemie operacyjnym GEORGE 3 urządzenia zdalne mogą być obsługiwane dwoma sposobami:
- bezpośrednio przez rozdziały systemu GEORGE 3
- przez programy użytkowe komunikujące się z systemem przez specjalny interfejs (tzw. wydawca komend, czyli symulacja urządzeń MOP).

Pierwszy z powyższych sposobów jest realizowany w przypadku urządzeń MOP (Multiple on-line Programming) oraz RJE (Remote Job Entry). Urządzenia lub konfiguracje sprzętowe, których nie uwzględnia bezpośrednio system GEORGE 3 (tzn. takie, których nie obsługuje MOP ani RJE) mogą być również włączone do systemu cyfrowego, ale wymagają odpowiedniego oprogramowania. Można dla tych celów wykorzystać wyżej wymieniony pakiet Communications Manager, z którego generuje się programy obsługi różnego typu urządzeń teleprzetwarzania lub opracować program (programy) własne. W przypadku użycia pakietu komunikacyjnego należy opracować dodatkowo program pośredniczący między nim i systemem GEORGE 3. Takie rozwiązanie przyjął producent np. dla monitorów ekranowych zdalnych przyłączonych przez skaner (bez procesora komunikacyjnego), dostarczając gotowy program o nazwie #XKL8 dla zapewnienia interfejsu z systemem GEORGE 3.

Wobec dużego zapotrzebowania na terminale dialogowe rozpatrzono możliwość rozbudowania systemu cyfrowego, tak aby mogły w nich być użyte równocześnie: drukarki mozaikowe DZM 180 przyłączone przez multipleksor (MPX), dalekopisy typu ICL-7071 przyłączone przez drugi MPX oraz monitory ekranowe przyłączone przez skaner.

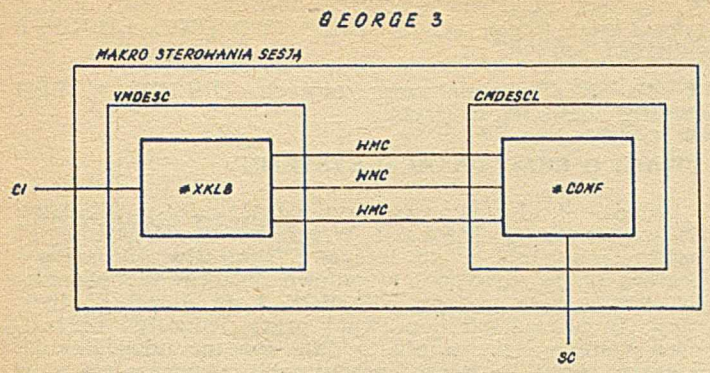
Chcąc zapewnić wszystkim terminalom pracę w systemie GEORGE 3, a dysponując dwiema jednostkami centralnymi ODRA 1305, połączonymi przez adapter międzymaszynowy, zdecydowano rozdzielić funkcje obsługi urządzeń zdalnych systemu GEORGE 3 pomiędzy te dwie maszyny. Rozpatrzono warianty: przeniesienie obsługi multipleksera na drugą maszynę, przeniesienie obsługi skanera na drugą maszynę oraz przeniesienie obsługi wszystkich urządzeń teletransmisji na drugą maszynę. Przy wyborze wzięto pod uwagę obciążenie jakie stanowią dla systemu programy obsługi danych urządzeń, jakość obsługi urządzeń terminalowych (czas reakcji) oraz możliwości programu sterującego — egzekutora maszyny ODRA 1305.

Kierując się tymi kryteriami wybrano przeniesienie obsługi skanera na drugą maszynę. Całkowite odciążenie systemu GEORGE 3 od funkcji kompletacji i rozdzielania komunikatów od i do terminali tzn. zainstalowanie uniwersalnego procesora komunikacyjnego, który przejmuje te funkcje, zwiększa przepustowość zadań w systemie GEORGE 3 o około 30% (odpowiednio mniej w przypadku przeniesienia części tych funkcji na drugą maszynę).

W założonej konfiguracji druga ODRA 1305 realizuje poza tym normalne przetwarzanie pod kontrolą egzekutora. Dzięki takiej obsłudze transmisji uzyskuje się również przyspieszenie czasu reakcji systemu na monitorach ekranowych.

STRUKTURA OPROGRAMOWANIA

Na rysunku 1 przedstawiono strukturę oprogramowania dla monitorów ekranowych zdalnych w systemie GEORGE 3: program #COMF obsługi urządzeń teletransmisji, pełniący funkcję podobne jak program sterujący procesora komunikacyjnego ICL 7903 i wygenerowany z pakietu Communication Manager oraz program o nazwie #XKL8, zapewniający interfejs z systemem operacyjnym GEORGE 3 dzięki mechanizmowi tzw. programowego wydawcy komend (ang. *Command Issuer — CI*). Program #XKL8 jest uruchamiany i sterowany przez makro VMDESC, a program #COMF przez makro o nazwie CMDESCL, które znajdują się w kartotece makroinstrukcji systemu. Te dwa zadania pozostają w pełni uruchomione przez cały czas trwania sesji dla monitorów ekranowych.

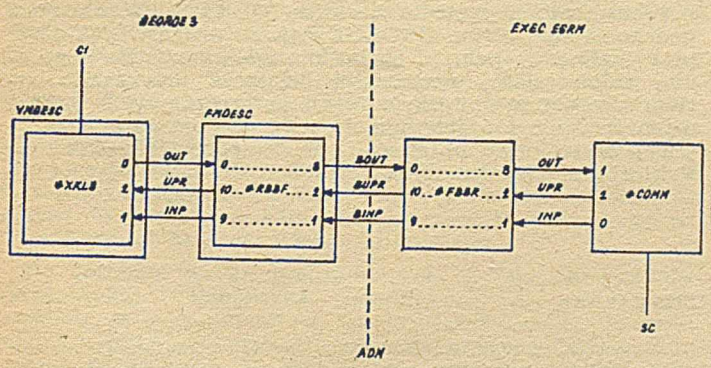


Rys. 1. Struktura oprogramowania dla monitorów ekranowych zdalnych w systemie GEORGE 3

WMC	— łącze międzyprogramowe wewnątrzmaszynowe
# XKL8	— program interfejs do systemu operacyjnego
# COMF	— program obsługi urządzeń wygenerowany z pakietu Communications Manager
VMDESC	— makro obsługujące przerwania programu # XKL8
CMDESCL	— makro obsługujące przerwania programu # COMF
CI	— wydawca komend (Command Issuer)
SC	— skaner

Przystępując do projektu oprogramowania przeprowadzono eksperyment z włączeniem ogniwa symulującego stację transportową, ale pośredniczącą jedynie w transmisjach wewnątrzmaszynowych. Miał on na celu zbadanie procesów transmisji, łącz oraz struktury bloków informacji przekazywanych pomiędzy programami # XKL8 i # COMF. Ponadto zbadano sprawność takiego rozbudowanego układu programowego (czas odpowiedzi w zdaniach interakcyjnych). Po stwierdzeniu, że czas odpowiedzi jest zadowalający i uzyskaniu niezbędnych danych dla rozdzielenia programów # XKL8 i # COMF przystąpiono do realizacji zadania.

Na rysunku 2 widoczna jest struktura oprogramowania dla zdalnych monitorów ekranowych, obsługiwanych przez system połączonych maszyn cyfrowych ODRA 1305. Program # COMM pakietu Communication Manager został wprowadzony na drugą maszynę pod kontrolę egzekutora E6RM. Komunikacja między # XKL8 i # COMM przebiega przez dwie stacje transportowe i adapter międzymaszynowy. Stacje transportowe nieznacznie tylko absorbują procesor (pod egzekutorem E6RM stacja # FBBR wykorzystuje mniej niż 1% czasu procesora).



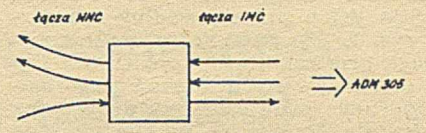
Rys. 2. Struktura oprogramowania dla zdalnych monitorów ekranowych obsługiwanych przez system połączonych maszyn ODRA 1305

Program # XKL8 pracuje pod kontrolą systemu GEORGE 3, tak jak stacja # RBBF. Do komunikacji przez adapter międzymaszynowy służą w tych stacjach łącza międzyprogramowe międzymaszynowe. Stacje transportowe pośredniczą między programami znajdującymi się pod kontrolą systemu GEORGE 3 i egzekutora E6RM.

Jak pokazano na rys. 2 do celów sterowania systemem służy makro uruchamiane przez operatora z konsoli operatorskiej. Makro to powoduje rozpoczęcie sesji MOP oraz pracy łącza międzymaszynowego uruchamiając dwa zadania: VMØ i FMØ.

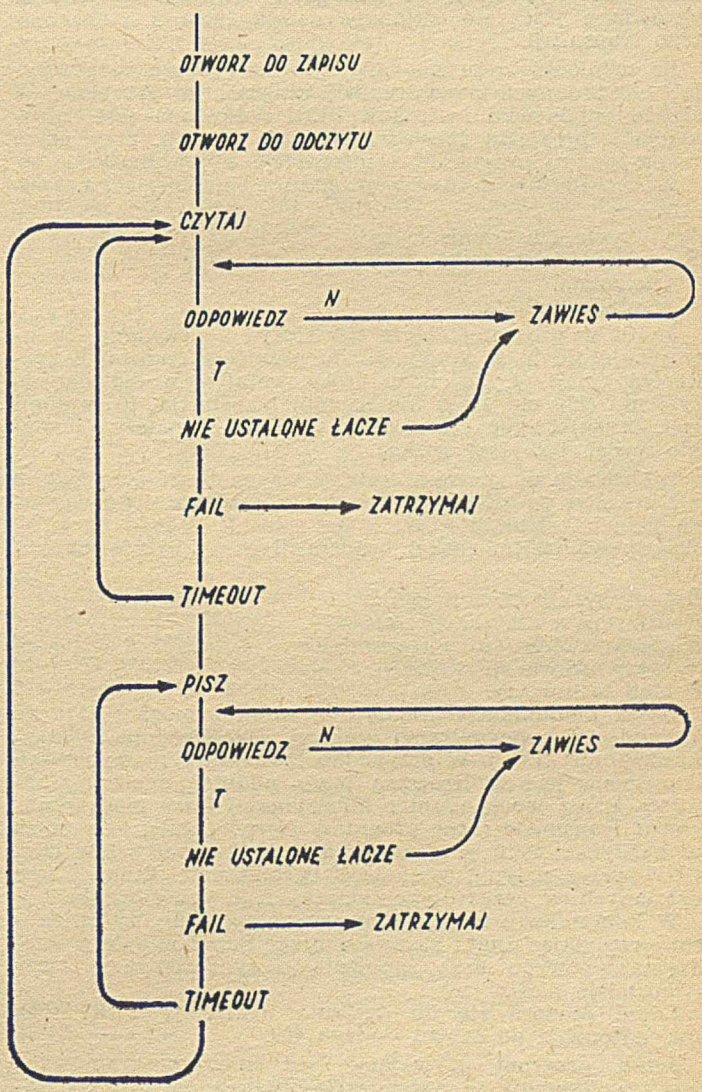
KONCEPCJA PROSTEJ STACJI TRANSPORTOWEJ

Przez stację transportową rozumiemy program realizujący równolegle kilka procesów, z których każdy polega na odbieraniu bloków informacji przez łącze międzyprogramowe i przekazywaniu go na inne łącze komunikacji międzyprogramowej (rys. 3).



Rys. 3. Przykład łącz międzyprogramowych w prostej stacji transportowej

Część tych łącz stanowią łącza międzymaszynowe, przez które stacja transportowa przerzuca bloki informacji do bliźniaczej stacji transportowej w drugiej maszynie. Procesy realizowane w stacji transportowej są uaktywniane cyklicznie. Na rys. 4 przedstawiono schemat przebiegu takiego procesu (po ustaleniu łącz) dla jednej pary łącz: jednego — do czytania, drugiego — do zapisu.



Rys. 4. Schemat przebiegu procesu transmisji

Stacja transportowa może przyjmować bloki informacji od różnych programów, które tworzą łącza z odpowiednimi, ustalonymi dla stacji transportowej, nazwami i przekazywać je do drugiej stacji transportowej oraz odbierać od niej bloki informacji i przekazywać je programom. Procesy realizujące transmisję w obu kierunkach są równouprawnione.

Dla programu rozpatrywanego w tym opracowaniu przyjęto założenie, że łącza wejściowe i wyjściowe w stacji transportowej są sprzężone parami na stałe, ponieważ stacja ta pośredniczy tylko w procesie obsługi urządzeń teletransmisji.

Zgodnie z tą koncepcją są w niej otwierane trzy łącza do odczytu i trzy do zapisu. Jednemu łączu komunikacji międzymaszynowej odpowiada jedno łącze komunikacji wewnątrzmaszynowej tworząc drogę transportu. Każdą parę łącz (drogę) obsługuje jeden proces, według schematu przedstawionego na rys. 2. W akumulatorze X1 znajduje się numer aktywnego procesu. Każdy proces ma bufor dla wprowadzenia przekazywanych bloków informacji. Poprzez stacje transportowe prowadzą trzy drogi, przy czym dwie z nich przekazują transmisje w kierunku do systemu GEORGE 3, a jedna w kierunku przeciwnym (rys. 4).

ZASADY KOMUNIKACJI MIĘDZYPROGRAMOWEJ

Komunikacja między programami odbywa się przez łącza międzyprogramowe, które są identyfikowane przez nazwy. Transmisja w łączu może się odbywać tylko w jednym kierunku, tzn. jeden program — pisze, drugi — czyta. Można użyć wielu łącz pomiędzy dwoma programami i program może mieć łącza z kilkoma innymi programami równocześnie.

Jeżeli łącze przydzielono uprzednio komendą ONLINE i jest ono następnie otwierane w programie, to jest aktualizowane pole sterujące instrukcji otwarcia PERI oraz wstawiana jest tam aktualna nazwa łącza i efektywny tryb transmisji.

Po ustaleniu łącza międzyprogramowego można przesyłać bloki danych dopuszczalnej długości do 511 słów. W słowie odpowiedzi instrukcji PERI ustawiana jest informacja o przyjęciu lub odrzuceniu transmisji, przy czym w przypadku przyjęcia — informacje o jej przebiegu.

Proces obsługi łącza w programie przebiega następująco:

- otwarcie i ustalenie łącza
- synchronizacja łącza
- transmisje
- zamknięcie łącza.

Jeżeli w programie łącze ma służyć do czytania, to utrzymuje się je w gotowości do przyjęcia transmisji rozkazem CZYTAJ. Jeżeli w czasie ustalonym dla danego systemu (np. ok. 10 s) nie nastąpi transmisja, to wystąpi tzw. przekroczenie czasu (ang. *time-out*) i należy ponownie wydać powyższy rozkaz.

Informację do przesłania dzieli się na bloki nie większe niż 511 słów 24-bitowych. Przesłanie każdego bloku wymaga jednej instrukcji PISZ.

W odpowiedzi na rozkaz transmisji mogą być sygnalizowane:

- brak synchronizacji łącza
- transmisja nie zakończona
- przekroczenie czasu transmisji
- błędy transmisji
- błąd techniczny.

Łącza komunikacji międzyprogramowej mogą służyć do komunikacji wewnątrzmaszynowej (WMC) lub międzymaszynowej (IMC). W przypadku łącz międzymaszynowych transmisja jest realizowana przez adapter międzymaszynowy. Przez jeden adapter międzymaszynowy mogą realizować transmisje różne programy równocześnie, przy czym w każdym z tych programów można otwierać wiele łącz. Obowiązuje jedynie generalne ograniczenie liczby łącz ustalanych w egzekutorze podczas jego generowania.

W komunikacji międzyprogramowej pośredniczy egzekutor przesyłając znaki sterujące przez kanał adaptera międzymaszynowego oraz zapewniając synchronizację łącza (transmisje nadzorcze).

Łącze otwiera się do czytania instrukcją PERI tryb 200, natomiast do zapisu — tryb 400.

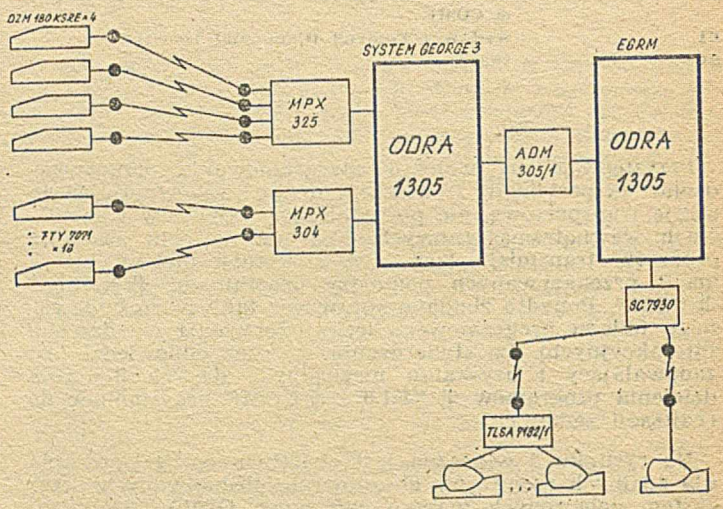
Jeżeli program używający łącz międzyprogramowych jest uruchamiany w systemie GEORGE 3, to w opisie zadania można przydzielić programowi odpowiednie łącza komendą ONLINE w postaci:

- dla łącz międzymaszynowych ONLINE * PBN (IMC), nazwa łącza (tryb), np. ONLINE * PBØ (IMC), LINKOUT (WRITE), lub
- dla łącz wewnątrzmaszynowych ONLINE * PBN (WMC), nazwa łącza (tryb).

UWAGI O EKSPLOATACJI SYSTEMU

Próbna eksploatacja systemu odbyła się w Centrum Obliczeniowym Politechniki Wrocławskiej w okresie od października 1978 r. do lutego 1979 r. Stwierdzono poprawną pracę systemu i stopniowo dostosowano parametry instalacyjne systemu GEORGE 3 do nowej konfiguracji. W warunkach poprawnej pracy jednostek centralnych systemu rozwiązanie to daje znaczne udogodnienia: większą liczbę terminali pracujących równocześnie oraz przyspieszenie czasu obsługi transmisji.

Powstała również możliwość rozbudowy instalacji poprzez dołączania do skanera innych typów urządzeń terminalowych (dalekopisy, drukarki mozaikowe) poprzez rozszerzanie opracowanych programów o moduły realizujące zamianę protokołu danego urządzenia.



Rys. 5. Schemat konfiguracji sprzętowej, eksploatowanej obecnie w Centrum Obliczeniowym Politechniki Wrocławskiej

W okresie eksploatacji próbnej komunikaty błędów typu FAIL Ø, FAIL 1, FAIL 2 od adaptera międzymaszynowego występowały z częstością 1—2 dziennie. Po konsultacjach z ELWRO-SERVICE wprowadzono poprawki do nakładek egzekutora E6RM. Ponadto ulepszono programy obsługi łącz międzymaszynowych. W trakcie próbnej eksploatacji systemu zarówno ODRA 1305 pracująca pod kontrolą systemu GEORGE 3, jak i druga ODRA 1305, pracowały przy pełnym obciążeniu zadaniami. Po uzyskaniu zadowalającej sprawności działania przekazano system do normalnej eksploatacji w wymiarze 8 godzin dziennie. Na rys. 5 znajduje się schemat konfiguracji sprzętowej eksploatowanej obecnie w Centrum Obliczeniowym Politechniki Wrocławskiej.

LITERATURA

[1] Battek J., Rudak B., Rudakowa B.: Mnogomasinnaja mnogodostupnaja sistiema, Komunikaty Centrum Obliczeniowego Politechniki Wrocławskiej nr 49 (przekazano do druku w Mat. Konf. IPI PAN).

[2] Enslow P. H. (red).: Systemy cyfrowe wieloprocesorowe, Comtre Corporation, WNT, Warszawa, 1978

[3] Janyszek J.: Adapter międzymaszynowy ADM-305, jego budowa i zastosowania, Raporty Centrum Obliczeniowego Politechniki Wrocławskiej seria PRE nr 3/1979

[4] System Operacyjny GEORGE 3, Publikacja Centrum MERA-ELWRO nr 1 300 203, Wrocław, 1977

[5] Techniczny Podręcznik Programisty. Uzupełnienia. Publikacja CENTRUM MERA-ELWRO nr 137 601, Wrocław, 1978.

Prezentujemy pierwszy z dwóch artykułów zawierających szkic części Drugiej Metody Wiedeńskiej. Przedstawiają one metodę precyzyjnego definiowania języków programowania, metoda ta może być również użyta do specyfikacji oprogramowania: baz danych, translatorów itp.

Artykuły (drugi ukaże się w następnym numerze) wymagają wysiłku od Czytelnika — sądzimy jednak, że warto wysiłek ten podjąć. (Red.)

WŁODZIMIERZ DRABENT
Politechnika Warszawska
Warszawa

Definiowanie języków programowania Druga Metoda Wiedeńska (składnia)

Potrzeba posiadania precyzyjnej metody definiowania języków programowania nie może być kwestionowana.

Jednym z negatywnych skutków niedoskonałości definicji są różnice pomiędzy implementacjami tego samego języka, uniemożliwiające przenoszenie programów z jednego typu komputera na drugi. Brak jednoznaczności utrudnia też programowanie i pogarsza niezawodność programów. Wątpliwości, czy dana konstrukcja językowa jest dozwolona, lub obawy dotyczące szczegółów wykonywania instrukcji, są znane wszystkim programistom.

Raport Algolu 60 wprowadził pewien styl definiowania języków: składnię (bezkontekstową) definiuje się w sposób formalny, a pozostałe aspekty języka opisuje się słownie. Praktyka wykazała, że opis słowny nie jest w stanie spełnić stawianych przed nim wymagań: precyzji, szczegółowości i kompletności z jednej strony, a przejrzystości i zrozumiałości z drugiej.

W ciągu ostatnich 15 lat zaproponowano kilka metod definiowania języków programowania. Jedną z nich, będącą częścią tzw. Drugiej Metody Wiedeńskiej, przedstawiamy w niniejszym i następnym artykule.

Druga Metoda Wiedeńska (Vienna Development Method — VDM) została opracowana w pierwszej połowie lat siedemdziesiątych w Wiedeńskim Laboratorium IBM. Służy ona do projektowania oprogramowania z zapewnieniem poprawności przejścia od specyfikacji do implementacji. Jej nazwa nawiązuje do jej prekursora — tzw. metody wiedeńskiej [2, 3].

Celem artykułów jest zaprezentowanie sposobu definiowania obiektów informatycznych oraz metajęzyka, w którym zapisywane są definicje. Prezentacja ta będzie oparta o przykład, za który posłuży definicja języka programowania. Język ten wybrano dowolnie, jest nim bardzo prosty język „algolopodobny". Ponieważ artykuły mają być jedynie zwięzłym szkicem, nie zostaną podane ścisłe i kompletne definicje nowych pojęć. Wyjaśnienia, podane w sposób odwołujący się do intuicji, ograniczone będą do zagadnień związanych z przykładową definicją. Aby ułatwić czytelnikowi zrozumienie tej definicji zostały podane przykłady, które prezentują prosty program w definiowanym języku i związek tego programu z definicją języka. W literaturze zamieszczono pozycje omawiające w sposób kompletny poruszony tu problem.

Na definicję języka programowania składa się **składnia** (syntaktyka) i **semantyka.** Składnia określa, jakie ciągi symboli są programami w danym języku czyli „programami poprawnymi składniowo". Semantyka określa znaczenie programów, tj. opisuje ich działanie (sposób wykonywania obliczeń). Metoda definiowania semantyki zostanie przedstawiona w drugim artykule.

W przypadku języków programowania o bardziej skomplikowanej notacji niewygodne jest określanie semantyki w sposób bezpośredni. Zwróćmy uwagę, że w typowych językach programowania wiele symboli nie ma bezpośredniego znaczenia dla semantyki (komentarze, odstępy, wszelkiego rodzaju nawiasy, separatory itp). Z drugiej strony używane są liczne skróty, np. deklaracje domyślne, (ang. *defaults*), opuszczanie nawiasów tam, gdzie pozwalają na to priorytety operacji, skrótowe wersje instrukcji i inne. Istnieją również alternatywne sposoby zapisu np. różne postacie słów kluczowych, różna notacja w zależności od dostępnego zestawu znaków itp.

Przy definiowaniu semantyki programu wprowadzamy więc stopień pośredni. Każdemu programowi konkretnemu przyporządkowuje się p r o g r a m a b s t r a k c y j n y, który zawiera istotną dla semantyki treść programu konkretnego i wyraża explicite jego strukturę (zwykle w postaci drzewa).

W ten sposób wprowadzamy j ę z y k a b s t r a k c y j n y, za którego realizację konkretną możemy uznać właściwy język programowania. Ponieważ druga metoda wiedeńska interesuje się przede wszystkim semantyką, uznaje ona język abstrakcyjny za pierwotny. Definicja składni konkretnej i wzajemnego przyporządkowania programów konkretnych i abstrakcyjnych jest zwykle pomijana. Jest też pominięta w naszej przykładowej definicji.

Język abstrakcyjny definiowany jest dwiema grupami reguł. Pierwsza z nich określa „bezkontekstowy" nadzbiór tego języka i nazywana jest **składnią abstrakcyjną.** Druga — to **warunki kontekstowe** określające, czy element powyższego nadzbioru jest poprawnym składniowo programem abstrakcyjnym. Warunki te dotyczą np. zgodności użycia zmiennych (i innych obiektów) z ich deklaracjami, zgodności typów w wyrażeniach, zgodności parametrów aktualnych i formalnych itp.

SKŁADNIA ABSTRAKCYJNA

Metajęzyk Drugiej Metody Wiedeńskiej umożliwia konstruowanie szerokiej klasy typów danych (zwanych tu dziedzinami) w oparciu o dziedziny podstawowe i operacje tworzenia dziedzin złożonych: list, drzew, zbiorów, funkcji oraz tzw. odwzorowań. Zestaw dziedzin podstawowych można dobierać w zależności od potrzeb. W naszej przykładowej składni abstrakcyjnej (tabela 2) są nimi: dziedzina Id będąca zbiorem identyfikatorów oraz kilka stałych (czyli dziedzin jednoelementowych) **BOOL, INT, SKIP** itd.

Tabela 1 zawiera bezkontekstową składnię konkretną przykładowego języka, zapisaną w tradycyjnej notacji Backusa, a tabela 2 — składnię abstrakcyjną. Przykłady P1 i P2 przedstawiają prosty program konkretny i odpowiadający mu program abstrakcyjny.

Tabela 1. Definicja składni konkretnej przykładowego języka (w notacji Backusa

<program> : : = **begin** <lista deklaracji> <lista instrukcji> **end**
<lista deklaracji> : : = <puste> | <typ> <identyfikator> ; <lista deklaracji>
<typ> : : = **boolean** | **integer**
<identyfikator> : : = ...
<lista instrukcji> : : = <puste> | <instrukcja> ; <lista instrukcji>
<instrukcja> : : = **skip** | **begin** <lista instrukcji> **end**
| <identyfikator> : : = <wyrażenie>
| **if** <wyrażenie> **then** <instrukcja> **else** <instrukcja>
| **while** <wyrażenie> **do** <instrukcja>
wyrażenie : : = <identyfikator> | <wyrażenie> <operator> <wyrażenie>
operator : : = + | =

Przykład P1. Program konkretny
begin integer A; **boolean** B;
skip; A: = A + A;
end

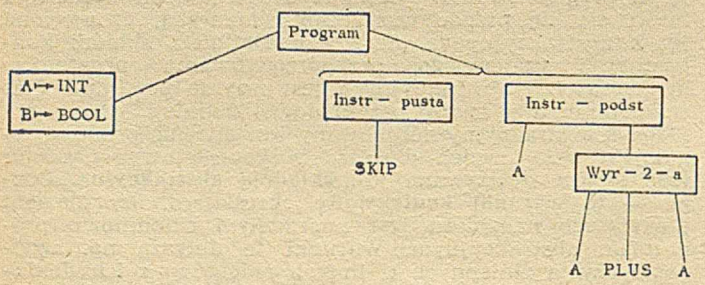
Tabela 2. Składnia abstrakcyjna przykładowego języka (opisana notacją Drugiej Metody Wiedeńskiej)

```
Program : : Dek Instr*                                          (2.1)
Dek = Id ——m→ Typ                                               (2.2)
Typ = BOOL | INT                                                (2.3)
Instr = Instr-pusta | Instr-złożona
      | Instr-podst | Instr-war | Instr-pętli                    (2.4)
Instr-pusta : : SKIP                                            (2.5)
Instr-złożona : : Instr*                                        (2.6)
Instr-podst : : Id Wyr                                          (2.7)
Instr-war : : Wyr Instr Instr                                   (2.8)
Instr-pętli : : Wyr Instr                                       (2.9)
Wyr = Wyr-2-a | Id                                              (2.10)
Wyr-2-a : : Wyr Op Wyr                                          (2.11)
Op = PLUS | EQ                                                  (2.12)
```

Wyjaśnienie skrótów:

Dek — deklaracje
Id — identyfikator
Instr — instrukcja
podst — podstawienia
war — warunkowa
Wyr — wyrażenie
2-a — dwuargumentowe
Op — operator
SKIP — ang. pomin.
Nazwami są wyrazy w liczbie pojedynczej, ale dotyczą nie pojedynczych obiektów, lecz ich zbiorów.

Przykład P2. Program abstrakcyjny odpowiadający programowi P1



Pierwsza reguła z tabeli 2 (2.1) definiuje zbiór programów abstrakcyjnych:

Program : : Dek Instr *

Zapis ten należy rozumieć w sposób następujący. Definiowana jest dziedzina o nazwie *Program*. Symbol : : oznacza, że jej elementami są drzewa (uporządkowane). Składają się one z dwóch gałęzi i korzenia. Pierwsza gałąź należy do dziedziny *Dek* a druga do dziedziny *Instr* *. Korzeń zawiera nazwę definiowanej dziedziny. Symbol * (gwiazdka) jest operatorem tworzenia dziedziny list. *Instr* * oznacza zbiór wszystkich list (o długości $\geqslant 0$), których elementy należą do dziedziny *Instr*. Dziedziny *Instr-pusta* i *Instr-złożona* (por. reguły (2.5) i (2.6) w tabeli 2) są dziedzinami drzew o jednej gałęzi, *Instr-podst* i *Instr-pętli* (por. reguły (2.7), (2.9)) — o dwóch, a *Instr-war* i *Wyr-2-a* (por. reguły (2.8), (2.11)) — o trzech gałęziach.

Wyrażenie $Id \xrightarrow[m]{} Typ$ występujące w regule (2.2) oznacza dziedzinę odwzorowań (z dziedziny *Id* w *Typ*), tj. zbiór wszystkich funkcji określonych na skończonych podzbiorach dziedziny *Id*, a o wartościach z dziedziny *Typ*. Litera m jest pierwszą literą słowa *map* (ang. odwzorowanie). Zbiór argumentów, dla których dane odwzorowanie jest określone, nazywamy dziedziną odwzorowania. Przyjmujemy, że dla argumentów spoza tego zbioru wartością odwzorowania jest stała **undefined**. Cała reguła (2.2) określa *Dek* jako dziedzinę odwzorowań z *Id* w *Typ*. Liście deklaracji z programu konkretnego odpowiada w programie abstrakcyjnym odwzorowanie, przyporządkowujące zadeklarowanym identyfikatorom ich typy.

Symbol | oznacza operację sumowania dziedzin. (Element należy do sumy dwóch dziedzin, gdy należy do pierwszej lub drugiej z nich). Tak więc dziedzina Typ (por. (2.3) w tabeli 2) zawiera dwa elementy, bowiem **BOOL** i **INT** są jednoelementowe.

Zwróćmy uwagę na różnicę w znaczeniu symboli = oraz : : . Reguła z symbolem = nadaje nową nazwę dziedzinie określonej przez prawą stronę reguły. Reguła ze znakiem : : określa nową dziedzinę drzew. Dwie reguły o różnych nazwach po lewej stronie znaku : : definiują różne dziedziny, nawet gdy ich prawe strony są jednakowe.

WARUNKI KONTEKSTOWE

Warunki kontekstowe pozwalają stwierdzić, czy dany program abstrakcyjny (element dziedziny *Program*) jest składniowo poprawny (dla zwięzłości będziemy opuszczać przysłówek — składniowo). Dla każdej dziedziny składniowej (w naszym przykładzie dla *Program*, *Instr* itp.) określamy predykat (*is-wf-program*, *is-wf-instr* itd.), którego wartością jest **TRUE** (prawda) dla poprawnie i **FALSE** (fałsz) dla niepoprawnie utworzonych elementów danej dziedziny. Nazwy „is-wf- ..." są skrótem od ang. *is well formed* ... (jest poprawnie utworzonym). Predykat związany z dziedziną złożoną jest określony w oparciu o predykaty związane z zawartymi w niej dziedzinami. Zastosowana notacja jest oparta na tradycyjnej symbolice matematycznej.

Tablica 3. Warunki kontekstowe dla przykładowego języka (w notacji Drugiej Metody Wiedeńskiej)

```
is-wf-program (mk-program(dek, instrl) =                        (3.1)
  (∀ instr ∈ elemsinstrl) (is-wf-instr(instr,dek)               (3.2)
is-wf-instr-podst(mk-instr-podst(id,wyr),dek) = typ(wyr,dek) = dek(id)
is-wf-id(id,dek) = dek(id) ∈ Typ                                (3.3)
is-wf-instr-war(mk-instr-war(wyr,instrl,instr2),dek) = typ(wyr,dek) = BOOL   (3.4)
is-wf-instr-pętli(mk-instr-pętli(wyr,instr),dek) = typ(wyr,dek) = BOOL      (3.5)
is-wf-wyr-2-a(mk-wyr-2-a(wyr1,op,wyr2),dek)
      = cases op :                                              (3.6)
      (PLUS ⟶ typ (wyr1,dek) = INT & typ(wyr2,dek) = INT,
      EQ ⟶ typ (wyr1,dek) = typ (wyr2,dek)

typ(wyr,dek)                                                    (3.7)
      = cases wyr :
      (mk-wyr-2-a (wyr1,op,wyr2) ⟶ cases op:
                                   (PLUS ⟶ INT,
                                   EQ ⟶ BOOL),
      id ⟶ dek(id) )
```

Omówienie warunków kontekstowych rozpocznie omówienie reguły określającej predykat *is-wf-program* (reguła 3.1. w tabeli 3): Reguła ta oznacza, że program składający się z deklaracji *dek* i listy instrukcji *instrl* jest poprawny, jeżeli dla każdej instrukcji *instr* z tej listy wartością wyrażenia *is-wf-instr(instr,dek)* jest prawda (co znaczy, że *instr* jest poprawną instrukcją względem deklaracji *dek*).

Operator *mk-program* jest operatorem tworzenia drzewa należącego do dziedziny *Program* (mk jest skrótem od *make* — stwórz). Zgodnie z definicją tej dziedziny (reguła (2.1) w tab. 2) ma on dwa argumenty, których wartości winne należeć odpowiednio do dziedzin *Dek* i *Instr* *. W analizowanej regule wyrażenie *(mk-program (dek, instrl)* jest parametrem. Taka postać parametru oznacza, że definiowany przez regułę predykat jest określony tylko dla argumentów z dziedziny *Program*. Jednocześnie wewnątrz reguły identyfikatory metajęzykowe *dek* i *instr* oznaczają odpowiednio pierwszą i drugą gałąź argumentu. Wyrażenie **elems**instrl oznacza zbiór elementów listy *instrl*, a symbol ∀ jest kwantyfikatorem ogólnym „dla każdego...".

Zauważmy, że nie ma żadnych ograniczeń kontekstowych dotyczących deklaracji. Składnia abstrakcyjna gwarantuje, że *dek* jest funkcją, a zatem nie musimy tu sprawdzać, czy nie ma identyfikatorów zadeklarowanych podwójnie. Sprawdzenie to należy do pominiętej w artykule definicji przekształcenia programów konkretnych na abstrakcyjne.

Tabela 3 zawiera wszystkie warunki kontekstowe definicji przykładowego języka. Przy określaniu warunków kontekstowych stosuje się zwykle skróty, opuszczając reguły oczywiste według następujących zasad:
1. Jeżeli w składni abstrakcyjnej występuje reguła postaci

$$D = D_1 | \ldots | D_n \quad (n \geqslant 1)$$

to w warunkach kontekstowych opuszcza się regułę mówiącą, że poprawny element sumy kilku dziedzin musi być poprawnym elementem jednej z nich — co można by zapisać następująco:

is-wf-D(x,dek)
= **if** x∈D_1 **then** is-wf-D_1(x, dek) **else**
...
if x∈D_n **then** is-wf-D_n(x, dek) **else FALSE**

(Użyte powyżej wyrażenie **if** ... jest analogiczne do wyrażenia warunkowego w ALGOLU 60).
2. Jeżeli dziedzina D jest określona w oparciu o dziedziny D_1, ..., D_n ($n \geqslant 1$) w inny sposób niż powyżej, to oczywistym jest wymaganie, że aby dany element dziedziny D był poprawny (względem deklaracji), wszystkie „zawarte w nim" elementy dziedzin D_1, ... D_n muszą być poprawne (względem tych deklaracji). Wobec tego opuszcza się wyrażające to wymaganie reguły lub części reguł.
3. Opuszcza się reguły postaci

is-wf-D (...) = **TRUE**

Zgodnie z podanymi zasadami w tabeli 3 dokonano uproszczeń: np. zgodnie z zasadą 1 — opuszczono regułę określającą *is-wf-instr*, zgodnie z zasadą 2 — w regule (3.5) opuszczono fragment *(is-wf-wyr (wyr,dek) & is-wf-instr(instr,dek)* oraz opuszczono definicję predykatu *is-wf-instr-złożona*, zgodnie z zasadą 3 — opuszczono reguły określające *is-wf-typ*, *is-wf-skip*, itp.

A oto pozostałe komentarze do tabeli 3: W regułach (3.2), (3.4), (3.5), (3.6) drugi znak = jest symbolem predykatu równości. Tak więc np. cała reguła (3.5) oznacza, że instrukcja pętli złożona z wyrażenia *wyr* i instrukcji *instr* jest poprawna względem deklaracji *dek*, gdy typ wyrażenia *wyr* jest równy BOOL (oraz *wyr* i *instr* są poprawne względem *dek*, co wynika z zasady 2). Wyrażenie *dek (id)*, (por. (3.2)) oznacza wartość odwzrorowania *dek* dla argumentu *id*. Zauważmy, że może być nią bądź **undefined**, bądź element dziedziny *Typ*. Pierwsza z tych możliwości oznacza, że identyfikator *id* jest niezadeklarowany. Wtedy wartością predykatu *is-wf-id* (por. (3.3)) jest fałsz (bo undefined nie należy do dziedziny *Typ*). Reguła (3.7) definiuje funkcję pomocniczą pozwalającą określić typ wyrażenia.

Po prawej stronie reguł (3.6) i (3.7) występują wyrażenia wyboru, postaci

cases e_0:
($e_1 \rightarrow es_1$,
...
$e_n \rightarrow es_n$).

W zależności od wartości wyrażenia e_0 (np. w (3.6) jest nim identyfikator *op*) dokonywany jest wybór spośród wyrażeń es_1..., es_n (ang. *cases* — przypadki). Jeżeli e_i jest w ciągu e_1, ..., e_n pierwszym wyrażeniem takim, że wartość e_0 daje się przedstawić w postaci e_i, to wartością całego wyrażenia wyboru jest es_i.

W regule (3.7) w wyrażeniu **cases** wyr.: ... drugi wariant $id \rightarrow dek(id)$ obejmuje wszystkie przypadki nie objęte przez wariant pierwszy, bowiem postać wyrażenia e_2 (a jest nim po prostu identyfikator) nie narzuca żadnych ograniczeń

Przykład P3. Sprawdzenie warunków kontekstowych
Jeżeli deklaracje programu P2 oznaczymy przez d, to

d(A) = **INT**,
a więc zgodnie z (3.3)
is-wf-id (A, d) = (**INT**∈Typ) = **TRUE**,
a zgodnie z (3.7)
typ (A, d) = **INT**.

Teraz można sprawdzić poprawność występującego w programie P2 wyrażenia dwuargumentowego (bowiem wiemy, że jego elementy są poprawne):

is-wf-wyr-2-a(mk-wyr-2-a(A,**PLUS**,A),d) =
(typ(A,d) = **INT**) & (typ(A,d) = **INT**) = **TRUE**.

Mamy też
typ(mk-wyr-2-a(A,**PLUS**,A),d) = **INT**.

Postępując analogicznie, możemy sprawdzić poprawność instrukcji i w końcu stwierdzić, że program P2 jest składniowo poprawny. Czytelnikowi pozostawiamy przekonanie się, że jeżeli zastąpimy operator **PLUS** przez **EQ**, albo identyfikator A po lewej stronie instrukcji podstawienia zastąpimy przez B, to otrzymamy programy niepoprawne składniowo.

LITERATURA

[1] Abstract Software Specifications, Proceedings Copenhagen 1979. Lecture Notes in Computer Science, Springer-Verlag 1980
[2] Łukaszewicz L.: Outline of the Vienna Method for the formal definition of programming languages. Warszawa 1973 COPAN Prace COPAN, nr 107
[3] Wegner P.: The Vienna Definition Langugage. ACM Computing Surveys, vol. 4, nr 1, March 1972
[4] The Vienna Development Method: The Meta-Language, edited by D. Bjørner and C. B. Jones. Lecture Notes in Computer Science nr 61, Spinger-Verlag 1978

Metody zabezpieczeń przeciw przestępczości komputerowej

W dniach 20 i 21 listopada br. odbędzie się w Paryżu międzynarodowe seminarium poświęcone zabezpieczeniom chroniącym od skutków przestępczości komputerowej (ang. *computer crime*). Tematyka seminarium świadczy o tym, że przestępczość komputerowa staje się w krajach rozwiniętej informatyki zjawiskiem coraz powszechniejszym, z którym musi się liczyć praktycznie każdy użytkownik. Szczególnie wrażliwą, a jednocześnie bardzo podatną na tego rodzaju przestępczość jest dziedzina informatyzacji operacji finansowych, zwłaszcza masowych operacji bankowych. Nic więc dziwnego, że organizatorem wspomnianego seminarium jest paryski instytut badawczy zagadnień bankowości INSIG (Institut de Recherche Interbancaire). Głównym tematem seminarium jest przegląd współcześnie stosowanych metod zabezpieczania programów i danych. Szczegółową problematykę obrad charakteryzuje poniższy zestaw tytułów zgłoszonych referatów:

- Zabezpieczenia przeciw nielegalnemu użyciu komputerów
- Analiza skuteczności stosowanych metod kontroli
- Nadzór działalności zabezpieczającej
- Zabezpieczanie oprogramowania użytkowego i podstawowego
- Metody zabezpieczeń w warunkach stosowania baz danych i teleprzetwarzania
- Problemy techniczne zabezpieczeń
- Zabezpieczenia w sieciach teleinformatycznych
- Zastosowanie terminala CP 8 oraz bezpieczeństwo operacji bankowych
- Aspekty poufności w międzybankowej wymianie informacji
- Zabezpieczenia stosowane w przedsiębiorstwie usług informatycznych.

JAN ROCZNIAK
Instytut Systemów Sterowania
Katowice

Projektowanie programów strukturalnych metodą Jacksona

Wśród nowoczesnych metod programowania maszyn cyfrowych poczesne miejsce zajmuje programowanie strukturalne. Jest to sposób programowania pozwalający tworzyć poprawne, proste, zrozumiałe, dające się łatwo modyfikować i konserwować programy, a polegający na ograniczeniu ilości typów struktur, z jakich się te programy buduje. Autorem idei narzucenia takiej dyscypliny w programowaniu jest E. Dijkstra, który na kongresie IFIP-1965 w Nowym Jorku stwierdził, że kwalifikacje programistów są odwrotnie proporcjonalne do ilości instrukcji „GO TO", używanych w ich programach [8].

W precyzowaniu zasad programowania strukturalnego brała udział duża grupa naukowców: H. Mills, O. J. Dahl, C. A. R. Hoare, N. Wirth, F. T. Baker, J. D. Aron, D. E. Knuth, R. W. Floyd, D. Parnas, M. A. Jackson, Z. Manna i wielu innych [1, 6, 7, 8]. Prowadzone prace nie doprowadziły do ustalenia ostatecznej definicji, wskazały jednak zbiór cech, jakie winien spełniać program strukturalny. Można tu wymienić:

- hierarchiczną strukturę systemów programowych
- metody logicznego abstrahowania i dekompozycji problemów, jako podstawy konstrukcji programów
- budowanie programów z sekwencji prostych operacji, z których w zasadzie wyklucza się instrukcję typu „GO TO", oraz operacji złożonych, selekcji i iteracji [5, 7].

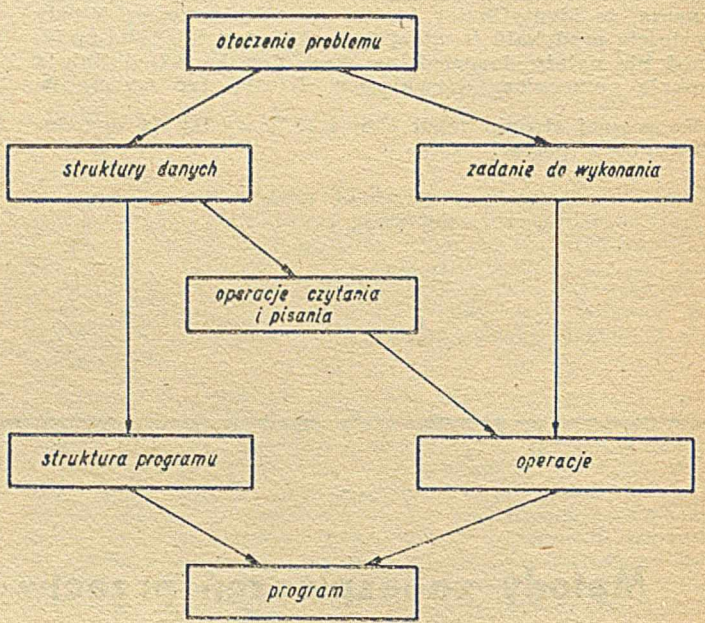
Po długotrwałych i burzliwych dyskusjach, programowanie strukturalne zaczęto wdrażać w szeregu ośrodkach jako obowiązującą metodę konstrukcji oprogramowania [6]. Praktyka wykazuje jednak, że o ile sam proces pisania tekstu programu strukturalnego i uruchamiania go daje znaczne korzyści czasowe w porównaniu z technologią tradycyjną — to projektowanie programów strukturalnych wiąże się z dodatkowymi problemami [4].

Jak oceniają specjaliści, w projektowaniu dekompozycyjnym (ang. *topdown*), technika doskonalenia krok po kroku, proponowana przez Dijkstrę jest za trudna dla przeciętnego programisty. Ma on zbyt wiele możliwości przy wyborze kolejnych przybliżeń struktury programu, nie dysponując jednocześnie jakimś obiektywnym kryterium poprawności swoich decyzji projektowych [5]. Jackson zaproponował, aby nadawać programom strukturę wzorowaną na strukturze przetwarzanych zbiorów. Sposób ten radykalnie ogranicza klasę możliwych struktur danego programu, tak że w konsekwencji uzyskujemy uproszczenie procesu projektowania programów strukturalnych.

W artykule omówione zostaną główne postulaty Jacksona, wprowadzona przez niego symbolika graficzna oraz przykład ilustrujący proces konstrukcji struktur programów. Przedstawiony materiał oparty jest w większości na pracy [3].

Proces projektowania programu Jackson dzieli na następujące etapy:

- definiowanie struktur danych przetwarzanych przez program
- złożenie poszczególnych struktur, o ile program przetwarza więcej niż jeden zbiór, zgodnie z zasadami podobieństwa lub nadrzędności określonego zbioru
- sprecyzowanie zbioru operacji podstawowych, koniecznych dla rozwiązania problemu
- przyjęcie struktury programu bazującej na strukturze zbiorów
- przypisanie poszczególnym blokom, w tak otrzymanej strukturze, elementów ze zbioru operacji podstawowych.

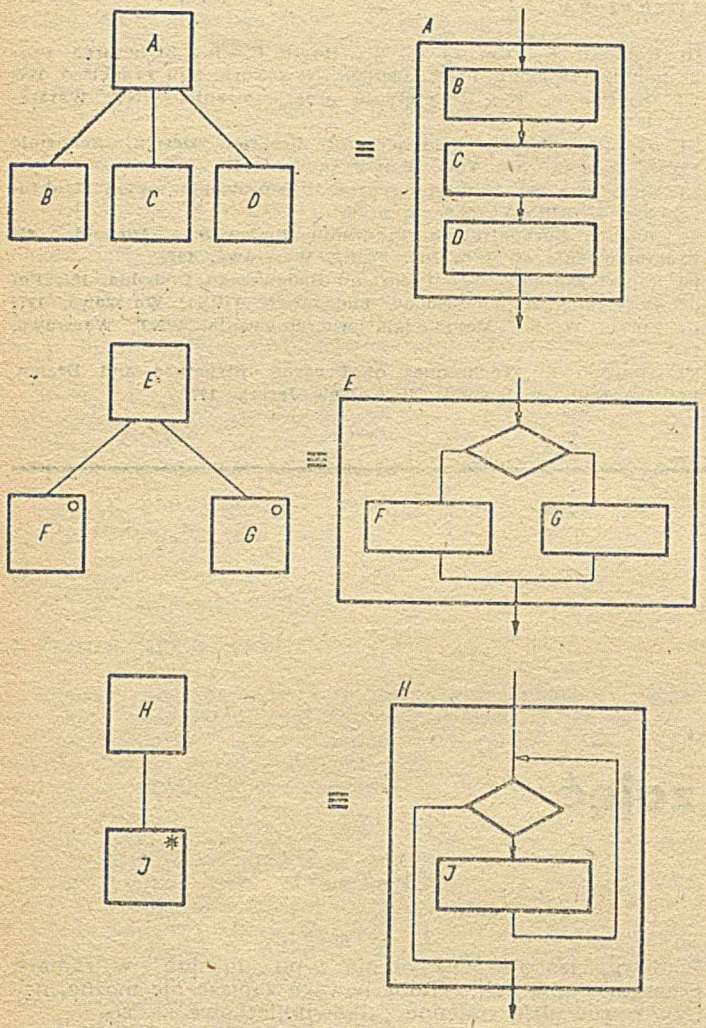


Rys. 1. Schemat projektowania programów metodą Jacksona

Mgr inż. Jan ROCZNIAK ukończył studia wyższe o specjalizacji elektrotechniki i automatyki na Wydziale Górniczym Politechniki Śląskiej (1974) i matematykę teoretyczną na Uniwersytecie Warszawskim (1977). W roku 1974 rozpoczął pracę w Instytucie Maszyn Matematycznych — Oddział Śląski. Obecnie pracuje w Instytucie Systemów Sterowania jako programista maszyn cyfrowych.

Zasadę tej technologii Jackson przedstawia na specjalnym schemacie, uwidoczniając główną myśl — konstrukcja programu strukturalnego opiera się na strukturze przetwarzanych zbiorów. Jackson odrzuca tradycyjny schemat blokowy jako środek zapisu budowy programu, środek, który odzwierciedla co prawda przepływ sterowania, lecz może zupełnie nie nawiązywać do tworzonej struktury. Zgodnie z główną ideą wprowadza on identyczne symbole graficzne dla oznaczenia struktur zbiorów i programów. Sekwencja kolejnych rekordów lub sekwencja kolejnych operacji przedstawia drzewo z wierzchołkiem, oznaczającym zbiór lub program jako całość, oraz gałęziami, kolejno — od lewej do prawej — pokazującymi odpowiednie rekordy czy operacje.

Jeżeli dany element zbioru może przyjmować jedną z kilku postaci lub jeżeli dany fragment programu oznacza instrukcję selekcji, to na schemacie elementy podrzędne wyróżnia się kółkiem. Wielokrotne wystąpienia rekordu w zbiorze lub instrukcję iteracji wyróżnia się gwiazdką.



Rys. 2. Symbole graficzne reprezentacji struktur Jacksona i odpowiadające im symbole tradycyjne

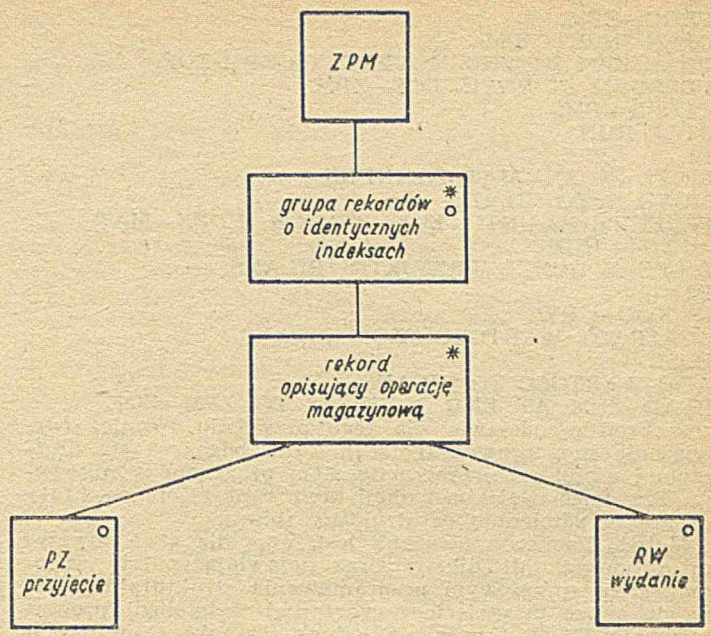
Przykład

Dla zilustrowania procesu konstrukcji programu — prosty przykład. Załóżmy, że mamy zaprojektować program podsumowujący operacje magazynowe. Magazyn wydziału produkcyjnego wydaje i otrzymuje części. Każde wydanie i przyjęcie jest odnotowane na karcie dziurkowanej przez podanie indeksu części, kodu operacji (RW — wydanie, PZ — przyjęcie) oraz wielkości danej operacji.

Następnie karty te są zapisywane do zbioru i sortowane według numerów indeksu. Program powinien przetwarzać tak otrzymany zbiór ZPM, wyprowadzając informację przedstawiającą sumaryczny bilans przeprowadzonych operacji magazynowych dla poszczególnych części.

Proponowany jest następujący format wydruku:

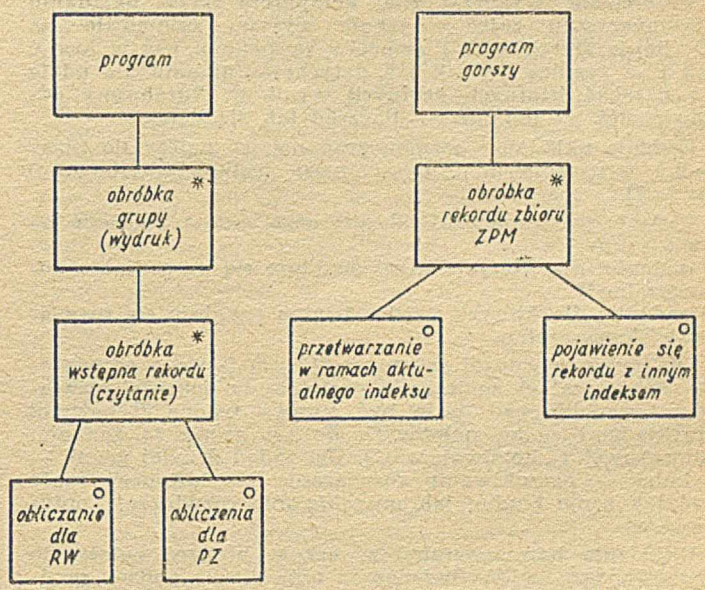
ØØØ1111111ØØØ1	PRZYJETO 49Ø
ØØØ1111111ØØ 37	PRZYJETO —235
ØØØ1222222Ø 7 14	PRZYJETO 14285



Rys. 3. Struktura zbioru ZPM

Strukturę zbioru ZPM pokazuje rysunek 3. Zbiór operacji elementarnych, jakie należy wykonać, to:

- otworzyć zbiór ZPM
- zamknąć zbiór ZPM
- czytać zbiór ZPM
- wyprowadzić informację sumaryczną
- podstawić pod zmienną INDEX indeks kolejnej części
- zerować licznik bilansujący BILANS
- dodać do zmiennej BILANS wielkość operacji magazynowej
- odjąć od zmiennej BILANS wielkość operacji magazynowej.



Rys. 4. Struktury programów realizujących przetwarzanie zbioru ZPM, z lewej — programu bazującego na strukturze zbioru ZPM, z prawej — programu tradycyjnego

Struktura programu (analogiczna do struktury zbioru ZPM) przedstawiona jest na rysunku 4. Przypisanie operacji elementarnych do poszczególnych bloków struktury, przy tak prostym programie, jest natychmiastowe. Odpowiedni program w języku COBOL może wyglądać następująco:

```
PROG—ZPM.
OPEN INPUT ZPM.
READ ZPM AT END MOVE 'E' TO ZPM-EOF.
PERFORM P1 UNTIL EOF—ZPM.
CLOSE ZPM.
STOP RUN.
P1.
MOVE INDEX—MAT TO INDEX.
MOVE ∅ TO BILANS.
PERFORM P2 UNTIL EOF—ZPM OR INDEX—MAT
  NOT = INDEX.
DISPLAY INDEX, 'PRZYJETO', BILANS.
P2
IF KOD = 'RW'
SUBTRACT WARTOSC FROM BILANS
ELSE
ADD WARTOSC TO BILANS.
READ ZPM AT END MOVE 'E' TO ZPM—EOF.
```

Jackson przedstawia dla tego przykładu również rozwiązanie niepoprawne (gorsze), które odznacza się dłuższym i mniej przejrzystym tekstem programu wynikowego. Przyczyna złej konstrukcji tkwi w przyjęciu struktury programu odmiennej od struktury zbioru,

Metoda Jacksona uważana jest, wspólnie z najnowszymi propozycjami Dijkstry oraz pojawiającym się wspomaganiem sprzętowym programowania strukturalnego, za najciekawsze osiągnięcie nowoczesnej technologii programowania [2, 6]. Uważa się jednak, że zakres jej użycia ograniczony jest do prostych zastosowań typu ekonomicznego [5, 6].

Niemniej, propozycje Jacksona warte są szerszego upowszechnienia, choćby z dwóch powodów. Nawet w złożonym systemie programowym znajdują się zwykle elementy proste (np. programy aktualizacji czy wydruków), z których konstrukcją metoda radzi sobie doskonale. Po drugie — ze względu na swoją prostotę nadaje się ona do nauki podstaw programowania strukturalnego, zwłaszcza na tych kierunkach studiów, na których — nie mogąc przekazać szerszego materiału o metodologii programowania — poprzestaje się na nauce języka COBOL.

LITERATURA

[1] Dahl O. J., Dijkstra E. W., Hoare C.A.R.: Structured Programming. Academic Press, London, New York, San Francisco, 1972

[2] Dijkstra E. W.: Umiejętność programowania. WNT, Warszawa, 1978

[3] Jackson M. A.: Principles of Program Design. Academic Press, London, New York, San Francisco, 1975

[4] Niemyjska E.: Praktyczne aspekty stosowania metod strukturalnych. INFORMATYKA nr 2/1980

[5] Raport Europejskiego Programu Badawczego Diebolda, 86: Programowanie strukturalne. OBRI, Warszawa, 1977

[6] Raport Europejskiego Programu Badawczego Diebolda, 103: Poprawa wykorzystania zasobów kadrowych. OBRI, Warszawa, 1979

[7] Turski W. M.: Metodologia programowania. WNT, Warszawa, 1978

[8] Yourdon E.: Techniques of Program Structure and Design. Prentice-Hall, Englewodd Cliffs, New Jersey, 1975.

ANDRZEJ GOSPODAROWICZ
Akademia Ekonomiczna im. Oskara Langego
Instytut Cybernetyki Ekonomicznej
Wrocław

Możliwości automatyzacji układania harmonogramu zajęć

Problemem wykorzystania komputera przy układaniu harmonogramu zajęć w szkole wyższej zajmowało się i zajmuje szereg osób i zespołów zarówno u nas w kraju, jak i za granicą, jednak jak dotychczas nikomu nie udało się uzyskać istotnych ogólnych wyników. Spróbujmy odpowiedzieć na pytanie — dlaczego tak się dzieje?

Zadanie układania harmonogramu zajęć należy do klasy zadań programowania dyskretnego (całkowitoliczbowego) [1, 3, 7].

Zadania programowania dyskretnego można podzielić na dwie grupy:

- zadania, dla których łatwo znajduje się rozwiązanie dopuszczalne [1])
- zadania, dla których bardzo trudno znajduje się rozwiązanie dopuszczalne.

Dla zadań pierwszej grupy (w których łatwo o rozwiązanie dopuszczalne) znalezienie dostatecznie dobrego rozwiązania przybliżonego może być sprawą trudną (typowymi przykładami zadań należących do tej grupy są zadania: „plecakowe" i „komiwojażera"). Dla zadań drugiej grupy — z kolei — poszukiwanie rozwiązania przybliżonego może być tak samo trudne jak poszukiwanie rozwiązania optymalnego.

Układanie harmonogramu w szkole wyższej należy do drugiej grupy zadań. Świadczy o tym choćby podane poniżej zadanie. Układany harmonogram zajęć nie zawiera tutaj szeregu istotnych parametrów i ograniczeń.

Zadane jest n zajęć, które muszą się odbyć w m salach. Czas trwania i-tego zajęcia oznaczymy symbolem t_i.

Należy określić T_{ik}, to jest czas rozpoczęcia i-tego zajęcia w k-tej sali — tak, aby jednocześnie w każdej sali odbywało się tylko jedno zajęcie, czyli dla dowolnej pary zajęć i, j $(i \neq j)$ musi być spełniona jedna z nierówności

$$T_{ik} - T_{jk} \geqslant t_j$$

lub

$$T_{jk} - T_{ik} \geqslant t_i$$

Ponieważ jednak ograniczenia typu „lub-lub" w ramach programowania całkowitoliczbowego zapisać nie można, należy wprowadzić zmienne całkowitoliczbowe — x_{ijk}

$$x_{ijk} = \begin{cases} 1 & \text{jeżeli zajęcie i-te odbywa się w sali k-tej wcześniej niż j-te zajęcie} \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

Teraz ograniczenie typu „lub-lub" można zapisać

$$(W + t_j)\, x_{ijk} + (T_{ik} - T_{jk}) \geqslant t_j$$
$$(W + t_i)\, (1 - x_{ijk}) + (T_{jk} - T_{ik}) \geqslant t_i$$

gdzie: W — dowolnie duża liczba wprowadzona dlatego, aby spełniona była jedna z równości $x_{ijk} = 0$ lub $x_{ijk} = 1$

$$T_{ik} \geqslant 0$$
$$0 \leqslant x_{ijk} \leqslant 1 \qquad x_{ijk} \text{ — całkowite}$$

Liczba zmiennych:

dla T_{ik} — mn

dla x_{ijk} — $\dfrac{mn\,(n-1)}{2}$

Liczba ograniczeń:

$$(W + t_j)\, x_{ijk} + (T_{ik} - T_{jk}) \geqslant t_j - \frac{mn\,(n-1)}{2}$$

$$(W + t_i)\, (1 - x_{ijk}) + (T_{jk} - T_{ik}) \geqslant t_i - \frac{mn\,(n-1)}{2}$$

[1]) Dowolne rozwiązanie dopuszczalne nazywane jest także rozwiązaniem przybliżonym

Łatwo zauważyć, że wygenerowanie rozwiązania dopuszczalnego nie jest już tutaj sprawą tak prostą jak w przypadku zadań pierwszej grupy. Tamte zadania mogą być rozwiązywane tak za pomocą metod dokładnych (tj. metod dających rozwiązanie optymalne), jak i metod przybliżonych. Niemniej z wielu prowadzonych eksperymentów wynika, że metody dokładne mogą być stosowane tylko wówczas, gdy zadanie jest niewielkich rozmiarów (tj. gdy zawiera niewielką liczbę zmiennych i ograniczeń).

Do rozwiązywania zadań drugiej grupy stosuje się wyłącznie metody przybliżone. Zaliczamy do nich metody podziału i ograniczeń, poszukiwania przypadkowego (metody Monte Carlo) oraz heurystyczne. Metoda podziału i ograniczeń oraz metody Monte Carlo znajdują zastosowanie wówczas, gdy zadanie jest małych lub średnich rozmiarów. Metody heurystyczne to w zasadzie jedyne obecnie metody, za pomocą których można rozwiązywać zadania dużych rozmiarów, a więc najczęstsze w praktyce.

O metodach heurystycznych mówi się często, że ich idea jest nieformalna, ponieważ opiera się na intuicji i doświadczeniu człowieka. Stosując metodę heurystyczną nie jesteśmy zatem w stanie zagwarantować, że otrzymamy rozwiązanie dopuszczalne. Proces obliczeniowy w większości tych metod jest jednak ściśle opisany, można go więc zapisać w języku programowania komputera. Niekiedy bywają też takie metody heurystyczne, które zakładają uczestnictwo człowieka na określonych stadiach procesu obliczeniowego, przy podejmowaniu decyzji o dalszych krokach rozwiązywania.

Metody heurystyczne są czasami krytykowane przez matematyków. Wszystko jednak ogranicza się do krytyki, natomiast w zamian nic się nie proponuje. Wydaje się więc, że trzeba akceptować tego rodzaju metody w przypadkach, gdy:

- pozwalają rozwiązywać zadania „niedostępne" obecnie dla metod dokładnych bądź innych metod przybliżonych
- przynoszą lepsze przybliżone rozwiązanie niż to, które daje człowiek rozwiązujący zadanie bez użycia metod formalnych (matematycznych) i komputera.

METODY HEURYSTYCZNE

Metody heurystyczne układania harmonogramu zajęć w szkole wyższej polegają na zastosowaniu w każdym kroku zasad opartych na doświadczeniu i intuicji (tzw. heurystyk), aby w rezultacie doprowadzić do wyboru jednego zajęcia i określenia czasu oraz sali, w której ma się ono odbywać.

Dotychczas opracowane i prezentowane w literaturze metody heurystyczne układania harmonogramu zajęć różnią się w zasadzie tym, że w poszczególnych rozwiązaniach inna jest kolejność stosowania znanych już heurystyk lub wprowadzone są nowe [2, 4, 5, 6, 8]. Rozwiązania otrzymywane za pomocą komputera w wyniku stosowania tych metod są — jak dotąd — często gorsze od rozwiązań „ręcznych", ponieważ zdarza się, że komputer generuje rozwiązania niedopuszczalne (dla szeregu zajęć — mimo występowania okienek — komputer nie przydziela godziny rozpoczęcia, ani sali).

Człowiek, który liczy wiele tysięcy razy wolniej niż komputer, jest w stanie rozwiązać tego rodzaju skomplikowane zadanie często z lepszym skutkiem. Okazuje się bowiem, że człowiek postępuje selektywnie. Nie bada po prostu wszystkich czy większości możliwych przypadków, lecz szybko wybiera jedną z ewentualności. Należy podkreślić, że w kolejnych krokach człowiek z reguły zmienia sposób działania.

Dotychczasowe programy komputerowe układania harmonogramu zajęć nie odzwierciedlają tego postępowania zbyt poprawnie. Niemniej, mimo ich niedoskonałości, pozwoliły ujawnić i lepiej zrozumieć występujące w tym procesie trudności. Nie ulega zatem wątpliwości, że heurystyczne metody układania harmonogramu zajęć muszą być w dalszym ciągu doskonalone. Doskonalenie ich jednak nie powinno polegać na próbie bardziej precyzyjnego zapisu (w języku algorytmicznym) sposobu postępowania człowieka. Wydaje się bowiem, iż sformalizowanie sposobu myślenia człowieka jest obecnie niemożliwe.

Doskonalenie to powinno polegać na tworzeniu pewnego rodzaju barier, które zapobiegałyby generowaniu zupełnie nie akceptowanych rozwiązań, z dużą liczbą okienek oraz zajęć nie umieszczonych w harmonogramie. Po pierwsze — należy dążyć do określenia warunków koniecznych i dostatecznych dla istnienia rozwiązania, a następnie tak wybierać zajęcia, aby nie naruszyć tych warunków spełnionych na początku. Po drugie — dla każdej grupy studenckiej ograniczyć dopuszczalny blok godzin do „obszaru", który na pewno będzie zajęty, gdy zajęcia w grupie będą umieszczane bez okienek. Po trzecie wreszcie — zasoby należy udostępniać kolejno, tj. „wypełniać" poszczególne sale bądź godziny w salach, rozpoczynając od godzin najwcześniejszych.

FORMUŁOWANIE ZADANIA

Formułowane obecnie zadania uwzględniają bardzo często wielką liczbę różnorodnych ograniczeń i parametrów. Wynika to zapewne z przeświadczenia, że za pomocą komputera można rozwiązać każde zadanie i to lepiej, niż może to zrobić człowiek, posługujący się metodami „ręcznymi".

Wydaje się jednak, że sukces w komputerowym rozwiązywaniu zadań układania rozkładów zajęć można osiągnąć dopiero wówczas, gdy:

- zrezygnuje się z części ograniczeń, szczególnie tych, które muszą być spełnione jedynie dla niektórych zmiennych
- ustali się w kroku początkowym wartości niektórych zmiennych
- ustali się wartości niektórych parametrów zadania.

Jest rzeczą oczywistą, że rezygnacja z pewnych ograniczeń może doprowadzić do wypaczenia istoty zadania. Dlatego zasadę tę można zastosować wyłącznie przy współpracy z przyszłym użytkownikiem systemu. Rezygnowanie z pewnych ograniczeń wymaga ponadto stosowania (już po otrzymaniu wyniku) tzw. procedur przestawiania. Zastosowanie tych procedur powinno umożliwić takie przestawienie zajęć, aby mogły być spełnione ograniczenia dotychczas nie uwzględniane. Procedury przestawiania będą miały też zastosowanie w przypadku, gdy — mimo wprowadzenia wyżej omówionych barier — nie uzyska się rozwiązania dopuszczalnego.

Przyjęcie zasady ustalania wartości niektórych zmiennych oznacza w praktyce, że system układania harmonogramu zajęć będzie systemem ręczno-maszynowym. Część zajęć zostanie bowiem umieszczona w harmonogramie „ręcznie", zanim komputer przystąpi do realizacji zadania.

* * *

Uzyskane przez autora doświadczenia pozwalają na sformułowanie poniższych uwag.

- Przedstawione w artykule zasady uproszczenia sformułowania problemu oraz bariery i procedury przestawiania (na etapie tworzenia algorytmu) były weryfikowane na przykładzie rozkładu zajęć dla Wydziału Górniczego Politechniki Wrocławskiej. Przeprowadzone obliczenia komputerowe, chociaż jeszcze nie dają podstaw do formułowania uogólnień, wykazały jednak, że otrzymywane harmonogramy zawierają stosunkowo niewielką liczbę okienek, zarówno w grupach jak i salach.
- Komputer powinien być stosowany do układania harmonogramu zajęć w większej niż dotychczas liczbie szkół wyższych, gdyż umożliwia: szybkie generowanie wielu wariantów rozwiązania, spośród których można wybrać najlepszy, szybkie „ręczne" poprawianie (z urządzenia końcowego) ułożonego przez komputer harmonogramu oraz szybkie uzyskanie ostatecznej wersji rozkładu.
- Prace nad tym problemem to także cenne źródło doświadczeń dla zespołów badawczych, które zajmują się (bądź będą się zajmowały) budową i wdrażaniem informatycznych systemów planowania produkcji w przedsiębiorstwie przemysłowym.

LITERATURA

[1] Bukietyński W.: Niektóre problemy optymalizacji dyskretnej, Wrocław 1975

[2] Chanas S., Florkiewicz B., Kulej M.: Koncepcja automatyzacji planowania rozkładu zajęć w wyższej uczelni, Komunikaty Instytutu Organizacji i Zarządzania Politechniki Wrocławskiej, nr 70, 1975

[3] Finkelsztejn J. J.: Pribliżennyje mietody i prikładnyje zadaczi diskretnogo programmirowanija, Moskwa 1976

[4] Gospodarowicz A.: Zagadnienie układania harmonogramu zajęć za pomocą komputera, Prace Naukowe WSE Wrocław, nr 54, 1974

[5] Kowalski K.: Algorytm układania harmonogramu zajęć dla wydziału wyższej uczelni, Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, nr 7, 1972

[6] Kubale M.: Problemy komputerowego układania rozkładów zajęć dla szkół wyższych, Algorytmy, 1973

[7] Liewner E. W., Giens G. W.: Diskretnyje optimizacjonnyje zadaczi i effiektiwnyje pribliżonnyje algoritmy, Moskwa 1978

[8] Polowczyk J., Runka H., Rzemykowski Z., Sikora W.: KSPZ — komputerowy system planowania zajęć. INFORMATYKA nr 8-9/79.

JERZY BRACHA
JERZY FORTECKI
Zakłady Azotowe „Włocławek"
MAREK KUBALE
Instytut Informatyki
Politechnika Gdańska

ROZA – system układania rozkładów zajęć

System ROZA (ROzkład ZAjęć) zrealizowano w Instytucie Informatyki Politechniki Gdańskiej [1]. Zadaniem systemu jest sporządzanie rozkładów zajęć dla wydziałów szkół wyższych na studiach dziennych i wieczorowych. System został wdrożony na Wydziale Elektroniki PG w roku akademickim 1979/1980.

Za pomocą systemu ROZA można planować układ zajęć, z których każde jest spotkaniem jednej grupy studenckiej z jednym prowadzącym — w jednej sali. ROZA wyznacza tygodniowe rozkłady zajęć dla grup dziekańskich i nauczycieli akademickich oraz dla poszczególnych sal dydaktycznych. System dąży przy tym do spełnienia wymagań zarówno dydaktycznych, jak i organizacyjnych. Ograniczenia zaś są następujące:

- sobota jest dniem wolnym od zajęć
- studenci mają przynajmniej jednogodzinną przerwę na obiad
- zajęcia rozpoczynają się o godz. 7^{00} i mogą trwać do określonej godziny zależnie od typu zajęcia (wykłady i ćwiczenia — do 15^{00}, laboratoria — do 22^{00})
- zajęcia odbywają się w salach odpowiedniej do tych zajęć wielkości i wyposażeniu
- rozkład zajęć zawiera możliwie małą liczbę „okienek"
- wykład i ćwiczenia z jednego przedmiotu odbywają się w różnych dniach
- grupa dziekańska ma nie więcej niż 8 godzin zajęć w ciągu dnia, o ile w tym dniu nie zaplanowano laboratorium
- nauczyciel akademicki nie prowadzi zajęć dłużej niż przez 4 godziny dziennie.

Poza tym system ROZA stwarza możliwości:

- uwzględnienia w rozkładzie życzenia prowadzących odnośnie godzin wolnych od zajęć, sal preferowanych na zajęcia czy maksymalnej liczby zajęć w ciągu dnia
- blokowania sal dydaktycznych w określonych godzinach
- planowania wskazanych zajęć w określonej sali, dniu i godzinie
- planowania zajęć z nieokreślonym prowadzącym bądź w nieokreślonej sali
- wykrywania kolizji w zbiorze zajęć narzuconych (tj. posiadających pewne ograniczenia).

ZARYS METODY

Algorytm sporządzania harmonogramu zajęć oparto na idei multikolorowania grafu, przedstawionej w [3]. Dla zwiększenia efektywności algorytmu proces konstruowania rozkładu zajęć podzielono na trzy etapy:

1. podział zbioru zajęć na podzbiory
2. planowanie zajęć w ramach podzbiorów
3. doplanowywanie zajęć nie zaplanowanych.

W pierwszym etapie zbiór wszystkich zajęć dzielony jest na pięć podzbiorów, odpowiadających poszczególnym dniom tygodnia. Zaszeregowanie zajęć do poszczególnych dni wynika z wcześniej narzuconych ograniczeń. W etapie drugim następuje właściwe planowanie zajęć — kolejno dla każdego dnia. Etap ten realizowany jest w oparciu o przybliżoną metodę multikolorowania grafu, ściślej — zajęcia przeznaczone na dany dzień porządkowane są według kolejności LF lub SL [4] (w systemie zrealizowano oba warianty, zaś wybór jednego z nich pozostawiono operatorowi), po czym następuje wieloprzebiegowy etap alokacji godzin i sal, zgodnie z otrzymaną kolejnością.

Przydział ten przeprowadzany jest według pewnych kryteriów heurystycznych, wynikających z wymagań stawianych rozkładowi zajęć. Jeżeli po tej fazie istnieją jeszcze zajęcia niezaplanowane, to realizowany jest ostatni etap algorytmu. Następuje wtedy trójprzebiegowe planowanie zajęć nie zaplanowanych w ramach jednego z pięciu dni, poczynając od dnia o najmniejszym obciążeniu.

Warto odnotować dużą czułość algorytmu na każdą, nawet pozornie błahą zmianę w danych. Na przykład, zmiana numeru przedmiotu w ramach jednych zajęć może prowadzić do innego podziału zbioru zajęć, a w konsekwencji do całkiem innego rozkładu.

EKSPLOATACJA SYSTEMU

System ROZA może być eksploatowany na komputerach ODRA serii 1300 o następującej minimalnej konfiguracji: dwie jednostki pamięci taśmowej, drukarka wierszowa, czytnik kart. Jest to duży program jednosegmentowy, napisany w języku PLAN 3 za pomocą ok. 5000 zdań.

Istnieje sześć wejść do tego programu, wykorzystywanych zależnie od stadium, w jakim znajduje się proces planowania zajęć. Są to wejścia dla standardowego wczytywania danych, wydruku formularzy pomocniczych, rozszerzonego wczytywania danych, konstruowania rozkładu zajęć, wydruku wyników oraz wczytywania poprawek.

System może być eksploatowany w 16 różnych wariantach, w zależności od stanu wybranych bitów słowa przełączającego: dla wydruków syntetycznych obciążeń nauczycieli, grup i sal; dla przypadku, gdy wydruki obciążeń mają być opuszczone; dla szeregowania zajęć metodą LF lub SL; dla planowania zajęć laboratoryjnych bez pozostawiania okienek i w przypadku przeciwnym; dla wydruków rozkładów zajęć dla prowadzących i obciążeń sal (w postaci tabel) oraz odpowiednich harmonogramów (w postaci listy).

Dane do systemu umieszcza się na kartach dziurkowanych w następującej kolejności:

— nagłówek
— drzewo wydziału
— ograniczenia godzinowe prowadzących
— ograniczenia godzinowe sal
— specyfikacja zajęć
— lista przedmiotów nauczania
— lista sal dydaktycznych
— lista nauczycieli akademickich.

Nagłówek zawiera kartę tytułową rozkładu i parametry wydziału, natomiast **drzewo wydziału** opisuje strukturę grup studenckich.

Najistotniejszą częścią danych jest specyfikacja zajęć. Każda karta zawiera opis jednych zajęć. Na opis ten składają się: numer grupy dziekańskiej, numer nauczyciela akademickiego, numer lub typ sali dydaktycznej, numer przedmiotu nauczania, informacja o typie oraz o czasie trwania zajęć.

Tabulogramy wynikowe ROZA wyprowadza w czterech wariantach. Maksymalny zestaw wydruków obejmuje:

- rozkład zajęć w postaci zakodowanej (obligatoryjnie), a następnie — w przypadku braku zajęć kolidujących ze sobą:
- syntetyczne obciążenia grup, wykładowców i sal w podanej kolejności (opcjonalnie)
- rozkłady zajęć dla grup (rys. 1) w postaci tabel (obligatoryjnie)

PAN DR GORSKI				
PONIEDZIAŁEK				
9 - - 11	SYS. OPER. CZASU RZEC.	SALA 114 NE	GRUPA AUTOMATYKA	ROK IV
13 - - 15	MIKROPROCESORY	SALA AUD. 1 NE	GRUPA	
WTOREK				
8 - - 14	PRAC. PROBLEMOWA LAB.	SALA 507 NE	GRUPA AUTOMATYKA	ROK IV
PAN DR W. JEDRUCH				
PONIEDZIAŁEK				
14 - - 16	STEROWANIE CYFROWE	SALA 7	GRUPA AUTOMATYKA	ROK IV
SRODA				
13 - - 15	UKŁADY CYFROWE	SALA AUD.7 NE	GRUPA INFORM.+ ELEKTR.	ROK II
PIATEK				
10 - - 12	UKŁADY CYFROWE	SALA AUD.2 NE	GRUPA INFORM.+ ELEKTR.	ROK II
PAN DR KOWALSKI				
CZWARTEK				
12 - - 15	WPROW. DO SZTOC. INTEL	SALA 443 NE	GRUPA AUTOMATYKA	ROK IV

Rys. 1. Rozkład zajęć dla słuchaczy (grup)

- rozkłady zajęć dla poszczególnych wykładowców i sal (obligatoryjnie), w postaci tabel lub listy (rys. 2).

GRUPA ELEKTRONIKA 2 ROK I			
GODZ	PONIEDZIAŁEK	WTOREK	SRODA
7- 8	WYCHOWANIE FIZYCZNE SALA AOS		MET. PROBABILISTYCZNE SALA 31 NE
8- 9	WYCHOWANIE FIZYCZNE SALA AOS	FIZYKA SALA AUD MAX	MATEMATYKA SALA AUD. Z 10
9-10		FIZYKA SALA AUD MAX	MATEMATYKA SALA AUD. Z 10
10-11	ETO I METODY NUMER. SALA AUD. 1 NE	EKONOMIA POLITYCZNA SALA AUD. 2 NE	MATEMATYKA SALA AUD. Z 10
11-12	ETO I METODY NUMER. SALA AUD. 1 NE	EKONOMIA POLITYCZNA SALA AUD. 2 NE	TEOR. SYG. I OBWODU SALA AUD. 1 NE
12-13	EKONOMIA POLITYCZNA SALA 436 NE	JEZYK OBCY SALA STUDIUM	TEOR. SYG. I OBWODOW SALA AUD. 1 NE
13-14	EKONOMIA POLITYCZNA SALA 436 NE	JEZYK OBCY SALA STUDIUM	MATER. I ELEM. KONS. P SALA 7
14-15		MET. PROBABILISTYCZNE SALA AUD. 2 NE	MATER. I ELEM. KONS. P SALA 7
15-16			
16-17			FIZYKA LAB SALA 7

Rys. 2. Rozkład zajęć dla prowadzących (w postaci listy)

Rozkłady zajęć dla grup wywieszane są przed dziekanatem oraz w domu studenckim. Rozkłady zajęć nauczycieli akademickich w postaci listy przekazywane są odpowiednim instytutom, a w postaci stabelaryzowanej — wręczane poszczególnym prowadzącym. Obciążenia sal w postaci listy przesyłane są do Działu Nauczania, zaś w postaci tabel umieszczane są na drzwiach odpowiednich pomieszczeń.

PARAMETRY SYSTEMU

System ROZA wymaga ok. 26 K słów pamięci operacyjnej komputera. Algorytm sporządzania rozkładu zajęć jest szybki, świadczy o tym fakt, że ułożenie harmonogramu dla studiów dziennych Wydziału Elektroniki PG [1]) trwało 23 sekundy, zaś wyprowadzenie wyników (w przypadku maksymalnego zestawu) 23 minuty pracy komputera ODRA 1325.

Ważną cechą systemu jest jakość drukowanych rozkładów. Należy zauważyć, iż uzyskanie rozkładu „optymalnego" jest nie tylko praktycznie niemożliwe, lecz również niecelowe (rozwiązanie takie traci swą wartość, ponieważ dane wejściowe nie są w pełni wiarygodne). Dlatego też ROZA daje jedynie tzw. rozkład „surowy", który musi być następnie poprawiony przez człowieka.

Wyznacznikami stopnia poprawności harmonogramów otrzymywanych z maszyny mogłyby być liczby zajęć nie zaplanowanych oraz poprawek do rozkładu surowego, gdyby nie fakt, iż parametry te są silnie obciążone — w pierwszym rzędzie liczbą sal dydaktycznych i rozkładem zajęć narzuconych, a następnie aktualnością życzeń pracowniczych, zgłoszonych uprzednio wobec rozkładu zajęć. Każda zmiana w postulatach wykładowców bądź w obsadzie poszczególnych przedmiotów powoduje zwykle serię poprawek do bieżącej wersji rozkładu. Co więcej, zmiany takie zgłaszane są często już po rozpoczęciu semestru. Dla orientacji, na Wydziale Elektroniki PG około 20% zajęć uległo przesunięciu w stosunku do wersji surowej. Część z nich spowodowana była niedoskonałością pierwotnej wersji rozkładu. Pozostałe były wynikiem interwencji nauczycieli akademickich.

Ważnym parametrem jest również rozmiar wejścia. Dane dla Wydziału Elektroniki umieszczone były na 1100 kartach. Z tego na ograniczenia godzinowe przypadało 69 kart, zaś na opis zajęć — 429. Cała reszta, czyli 622 karty, stanowiła cały zestaw danych. Dane stałe są wykorzystywane w kolejnych semestrach bez konieczności dokonywania większych zmian.

W systemie ROZA obowiązują następujące ograniczenia:

- maksymalna liczba zajęć — 512
- maksymalna liczba przedmiotów — 450
- maksymalna liczba nauczycieli akademickach — 255
- maksymalna liczba sal dydaktycznych — 125
- maksymalna liczba grup dziekańskich — 100
- maksymalny wymiar (czas trwania) przedmiotu — 9.

System ROZA jest systematycznie rozwijany i doskonalony. Obecnie prowadzi się badania eksperymentalne mające na celu znalezienie odpowiedzi na następujące pytania:

— jaka jest optymalna kolejność zajęć w specyfikacji?
— która metoda LF czy SL daje lepsze wyniki?
— jaki należy przyjąć pułap czasowy (godzina zakończenia) zajęć laboratoryjnych?
— jaki wpływ mają ograniczenia czasowe i lokalowe na jakość rozkładu zajęć?

Wstępne rezultaty tych badań sugerują, iż rozkład o najlepszej jakości uzyskuje się stosując metodę SL, z planowaniem laboratoriów bez okienek oraz ograniczeniem czasu tych zajęć do godz. 19^{00}.

Z uwagi na dużą efektywność algorytmu nie przewiduje się prac nad skróceniem czasu jego działania. Przeciwnie, system zostanie uzupełniony o komputerowe poprawianie rozkładu zajęć.

Istotnym kierunkiem prac jest też zmiana składni danych, w celu zmniejszenia udziału danych zmiennych (do ok. 150 kart dla wydziału o liczbie studentów rzędu 1000) kosztem zwiększenia liczby danych stałych.

[1]) — 1002 studentów w 39 grupach dziekańskich, 234 wykładowców, 429 zajęć (w tym 69 narzuconych), 21 sal laboratoryjnych i 14 audytoryjnych (wykorzystywanych w prawie 100%)

LITERATURA

[1] Bracha J., Fortecki J.: System programów do automatycznego układania rozkładów zajęć, praca dyplomowa, Instytut Informatyki PG, Gdańsk 1979

[2] Even S., Itai A., Shamir A.: On the complexity of timetable and multicommodity flow problems. SIAM J. on Comput., vol. 5, str. 691—703, 1976

[3] Kubale M., Szymborski A.: Effective algorithm for construction of University timetables. Proc. 2-nd IFIP Conference "Computers in Education", Marsylia 1975

[4] Kubale M.: Analiza efektywności algorytmów kolorowania grafów. Praca przyjęta do druku w piśmie „Matematyka Stosowana" nr 19.

JERZY ROSZKOWSKI
Resortowy Ośrodek Informatyki ETOCHEM
Warszawa

Określenie i realizacja spójności językowej i informacyjnej systemów informatycznych

Zagadnienie spójności systemów informatycznych jest ostatnio tematem dość ożywionej dyskusji wśród informatyków i ekonomistów [4, 5]. Wskazuje się, że jest to jeden z ważnych czynników, którego skuteczne rozwiązanie warunkuje dalszy rozwój systemów. Dotyczy to systemów różnych typów. Brak sformalizowanej metody badania oraz precyzyjnej definicji spójności przyczynia się do tego, że uzyskuje się stosunkowo niewielkie efekty w porównaniu do ponoszonych nakładów. Wskutek tych niedostatków rozwój pojęć, języków (metajęzyków) służących do opisu zawartości informacyjnej systemów, ich funkcji oraz sprzężeń międzysystemowych odbywa się częściowo w sposób żywiołowy, co doprowadza do nieporównywalności mierzalnych wielkości tych systemów. Jest to jednym z czynników hamujących rozwój już nie tylko systemów informatycznych, ale także metod zarządzania.

Jeżeli będziemy rozwijać systemy dziedzinowe w sposób od siebie niezależny (jak to się dzieje nagminnie obecnie) to oczywiście problem spójności będzie mało istotny. W sytuacji, gdy przechodzimy na systemy bardziej zintegrowane, sprawa spójności staje się pierwszoplanowa. Mówiąc o integracji mam na myśli integrację technologiczną systemów informatycznych typu baza danych, względnie integrację przepływów informacji w wielkich hierarchicznych systemach informatycznych typu systemy branżowe, resortowe, krajowe.

Dotychczas było kilka prób ujednolicenia nazw pojęć w bazach danych (zbiorach danych) systemów informatycznych. Próby te dotyczyły zarówno skali kraju jak i poszczególnych resortów, obejmowały całość zagadnień gospodarczych lub poszczególne grupy tematyczne, np. sprawozdawczość statystyczną. Wymienić tu można np. prace J. Oleńskiego [6] dotyczące definicji wskaźnika ekonomiczno-społecznego, opracowanie Komisji Planowania R. M. pt. „Definicje podstawowych kategorii ekonomiczno-społecznych" oraz zeszyty metodyczne GUS. Opracowania te, jakkolwiek bardzo pożyteczne, nie zmniejszyły wielkiego zamieszania w dziedzinie terminologii i wielkiej redundancji informacji w systemach. Wydaje się, że aby uporać się z tym problemem konieczne są w tej dziedzinie badania podstawowe mieszanych zespołów: matematyków, informatyków, cybernetyków i ekonomistów.

Badania takie powinny mieć charakter systemowy. Podkreślone to zostało szczególnie wyraźnie na V Ogólnokrajowej Konferencji Informatyków [4], dotyczącej organizacji obiektowych systemów informatycznych opartych o bazę danych, gdzie problematyka spójności stanowiła jeden z ważnych tematów podnoszonych zarówno w referatach sesji plenarnych, jak i w dyskusji.

Pojęcie spójności językowej, a w konsekwencji i informacyjnej wymaga — jako pojęcie interdyscyplinarne — ściślejszego zdefiniowania. Zawiera elementy lingwistyki matematycznej, informatyki, ekonomii. Niniejszy artykuł podaje próbę określenia tego pojęcia przy wykorzystaniu aparatu lingwistyki matematycznej. Ponieważ jednak taka definicja byłaby niewystarczająca dla konkretnych zastosowań, podano również techniczną możliwość realizacji spójności językowej systemów informatycznych za pomocą tezaurusa pojęć oraz wskazano na możliwości realizacji spójności informacyjnej za pomocą bazy danych.

Pojęcia (informacje) przetwarzane w systemach informatycznych możemy podzielić na dwie klasy: **opisujące stany** i **opisujące procesy.** Przykładem stanów są wszelkiego rodzaju zaszłości gospodarcze dotyczące np. produkcji, gospodarki materiałami, finansów. Przykładem sekwencji procesów może być np. zbiór nazw kolejnych operacji technologicznych, jakie należy wykonać na danym materiale (półprodukcie), aby otrzymać produkt końcowy.

Do opisu stanów jak i procesów można zastosować formalny aparat lingwistyki matematycznej, budując algorytmiczne języki stanów i procesów oraz definiując spójność w oparciu o te języki. Języki takie zostały opracowane przez autora w oparciu **o gramatykę generacyjną bezkontekstową, monotoniczną typu Chomskiego [1].** Taki wybór gramatyki zapewnia jednocześnie **rozstrzygalność języka,** tzn. istnieje dla takiego języka skończony algorytm pozwalający badać przynależność dowolnego zdania do języka (pozwalający odróżniać w tym języku zdania niepoprawne od poprawnych). Nie można bowiem opisać systemu ani zdefiniować spójności za pomocą zdań, dla których przynależność do języka jest nieokreślona. **Spójność językowa i informacyjna systemów informatycznych może być zatem definiowana wyłącznie dla języków rozstrzygalnych.**

OGÓLNA DEFINICJA SPÓJNOŚCI SYSTEMÓW INFORMATYCZNYCH

Załóżmy, że mamy dany bezkontekstowy język J generowany przez gramatykę G (co zapiszemy symbolicznie jako J (G)) o jednoznacznie określonej semantyce i słowniku A*. Niech będzie dany zbiór systemów informatycznych Z. Niech liczba jego elementów (moc zbioru) będzie $\overline{\overline{Z}} \geqslant 2$. Niech ponadto zawartość informacyjna każdego z systemów zbioru Z da się w pełni opisać przez zdania języka J (G). Przez B* oznaczymy zbiór wszystkich zdań (bez powtórzeń) opisujących zawartość informacyjną wszystkich systemów zbioru Z.

Oczywiście $B^* \subset A^*$. Każde zdanie należące do B* ma jedną określoną i tylko jedną wartość semantyczną w języku J (G). Zbiór B* możemy też nazwać maksymalną bazą danych zbioru Z. Ze względu na możliwość opisu Z za pomocą różnych języków przynależność B* do języka J oznaczymy przez B*(J).

Przy tych założeniach systemy należące do Z będziemy nazywali spójnymi względem siebie, jeżeli istnieje zbiór zdań C*(J), będący podzbiorem B*(J), należący do każdej z baz danych $B_i(J)$, $i = 1, \ldots \overline{\overline{Z}}$. Spójność tę będziemy nazywać spójnością ze względu na zbiór C(J).
Formalny zapis tej definicji ma postać:

$$\begin{bmatrix} Z_i \sim Z_j \\ i \neq j,\ 1 \leqslant i \leqslant \overline{\overline{Z}} \\ Z_i, Z_j \in Z,\ 1 \leqslant i \leqslant \overline{\overline{Z}} \end{bmatrix} <=>$$

$$<=> \begin{bmatrix} \bigvee & \bigwedge & C^*(J) \subset B_i^*(J) \\ C^*(J) \subset B^*(J) & B_i^*(J) & \\ & i = 1, \ldots Z & \end{bmatrix}$$

gdzie:

$Z_i \sim Z_j$ — oznaczenie spójności systemów Z_i oraz Z_j, $i, j = 1, \ldots, \overline{\overline{Z}}$; $Z_i, Z_j \in Z$

$B_i^*(J)$ — maksymalna baza danych systemów Z_i w języku J.

Na tym poziomie ogólności, bez wprowadzenia konkretnej gramatyki języka J, nie możemy odróżnić spójności informacyjnej od językowej w języku J (G). Do tego konieczna jest znajomość struktury zdania (wyznaczona przez konkretną gramatykę konkretnego języka). Wprowadzenie odrębnej definicji spójności językowej i informacyjnej jest już możliwe dla konkretnego reprezentanta języków bezkontekstowych o określonej gramatyce np. omawianego dalej języka L1.

● Własności języka L1.

Przykładem języka umożliwiającego realizację spójności językowej jest zbudowany przez autora język L1. Służy on do opisu stanów. Z ideą takiego języka i opisem własności, które powinien posiadać, wystąpił W. Flakiewicz w pracy [2]. Autorowi artykułu udało się nadać ścisłą formę matematyczną tym ideom. Zdanie w języku L1 ma następującą postać:

$$q_1 (v_s^q) (r_l^{q,b}) \text{ lub } q_1 (v_s^q) \text{ lub } q_1$$

gdzie: q_1 — nazwa obiektu

$$(v_s^q) = (v_1^q, v_2^{q-1}, \ldots, v_s^0)$$

— wektor deskryptorów opisujący obiekt q_1

$$(r_l^{q,b}) = (r_1^{q,b}, r_2^{q,b}, \ldots, r_l^{q,b})$$

— wektor kwantyfikatorów dający wartości liczbowe i jednostki miary mierzalnych deskryptorów

wskaźniki odpowiednio oznaczają:

1 — numer obiektu
q — stopień deskryptora
s — rozmiar wektora deskryptorów
i — rozmiar wektora kwantyfikatorów
b — składowa i-tego kwantyfikatora oznaczająca wartość liczbową
a — składowa i-tego kwantyfikatora oznaczająca jednostkę miary.

Oczywiście język ten ma swoją gramatykę, której własności z braku miejsca nie możemy tu przedstawić.

Poszczególnym symbolom występującym w strukturze zdania można przypisać następujące wartości semantyczne:

1. Nazwy obiektów nadaje się obiektom przez przyporządkowanie im rzeczowników w języku polskim; nazwa obiektu może być jednostkowa lub ogólna; każdy obiekt w systemie musi być nazwany; nazwy mają swój obowiązujący zakres.

Przykład: W przedsiębiorstwie przemysłowym opisywanymi obiektami mogą być np:

● elementy jego struktury organizacyjnej
● środki trwałe
● przedmioty nietrwałe
● półwyroby otrzymane w ramach kooperacji lub półwyroby i wyroby finalne
● normy przedmiotowe, klasyfikacyjne
● ceny i koszty produktów oraz świadczonych usług
● pracownicy
● transakcje (dokumenty transakcyjne)
● organizacje nadrzędne oraz organizacje usługowe
● niektóre elementy bazy normatywnej.

2. Deskryptorami nazywamy wektor $v_s^q = (v_1^q \ldots v_s^0)$ wyrażeń, które opisują własności obiektu oznaczonego nazwą. Mają one następujące własności:

● jeżeli istnieje deskryptor musi istnieć również nazwa, której deskryptor dotyczy
● nazwa obiektu może wystąpić bez deskryptora
● deskryptorem może być każda część mowy (z wyjątkiem czasownika), dowolny wyraz lub ciąg wyrazów określający jedną konkretną własność nazwy
● nazwa deskryptora nie może być jednocześnie nazwą obiektu.

W przypadku gdy liczba deskryptorów jest większa niż 1, to każdy deskryptor otrzymuje kolejny numer (wskaźnik dolny). Wskaźnik górny oznacza natomiast stopień deskryptora. Deskryptory stopnia zerowego są to te wyrazy, które opisują nierozkładalne dalej cechy (własności) danego obiektu.

Nazwa obiektu q_1	v_1^2	v_2^1	v_3^0	v_4^0
tworzywa sztuczne	polichlorek winylu	plan 1980	ilość	wartość

(Przykład zdania w języku L 1)

3. Kwantyfikator jest to uporządkowana dwójka wyrażeń. Poprzednikiem jest tu rzeczownik w pierwszej osobie liczby mnogiej oznaczający nazwę jednostki miary. Następnikiem jest liczba. Tak więc kwantyfikator jest parą uporządkowaną

$$r = (r^a, r^b)$$

WAŻNIEJSZE WYNIKI UZYSKANE W TECHNICZNEJ REALIZACJI SPÓJNOŚCI JĘZYKOWEJ

Jak wynika z ogólnej definicji spójności, spójność językową systemów bada się porównując odpowiednie zbiory pojęć. Wynika stąd, że pojęcia te muszą być porównywalne, a zatem **muszą być opisane za pomocą jednego języka.** Takim językiem, jeżeli chodzi o stany, może być np. L1. Aby zatem doprowadzić do spójności językowej pewnego zbioru systemów informatycznych należy zbudować **tezaurus** pojęć występujących w tych systemach. Wszystkie pojęcia definiowane i wymienione w takim tezaurusie muszą być opisane w określonym języku (np. w języku L1). Ponieważ używanie nazw pojęć nie należących do tezaurusa jest wzbronione, struktury bazy danych systemów informatycznych należy tak przedefiniować, aby posługiwały się tylko terminologią zaczerpniętą z haseł tezaurusa. W ten sposób można otrzymać zbiór systemów spójnych językowo:

OGÓLNA BUDOWA TEZAURUSA ZAPEWNIAJĄCEGO SPÓJNOŚĆ JĘZYKOWĄ, JEGO CZĘŚCI I ICH ROLA

Tezaurus pojęć ekonomicznych w (np. języku L1) powinien zawierać następujące części:

● zestawienie definicji haseł (uporządkowane wg rosnącego kodu hasła)
● rejestr alfabetyczny haseł oraz ich **synonimów** wraz z odpowiednimi odsyłaczami
● rejestr haseł wg wzrastającej wartości kodu hasła.

Zeszyt definicji haseł zbudowany jest z tzw. **kart tematycznych.** Karty tematyczne w zeszycie są uporządkowane wg wzrastającej wartości kodu cyfrowego hasła.

Karta tematyczna powinna zawierać następujące informacje:

● kod literowo cyfrowy hasła
● datę wprowadzenia do zbioru
● datę ostatniej aktualizacji
● nazwę hasła (nazwą hasła może być nazwa grupy tematycznej lub podgrupy tematycznej — obiektu, albo też nazwa deskryptora)
● synonim hasła
● nazwy jednostek miar (jeżeli dla hasła takie istnieją)
● definicję hasła oraz źródło zaczerpnięcia definicji.

Informacje o haśle powinny być uporządkowane w osobnej rubryce karty tematycznej w języku L1 w następującej kolejności:

1) grupa tematyczna (nazwa)
2) podgrupa tematyczna (nazwa obiektu)
3) hasło elementarne (deskryptory)
4) synonim do hasła definiowanego w karcie (opcjonalnie).

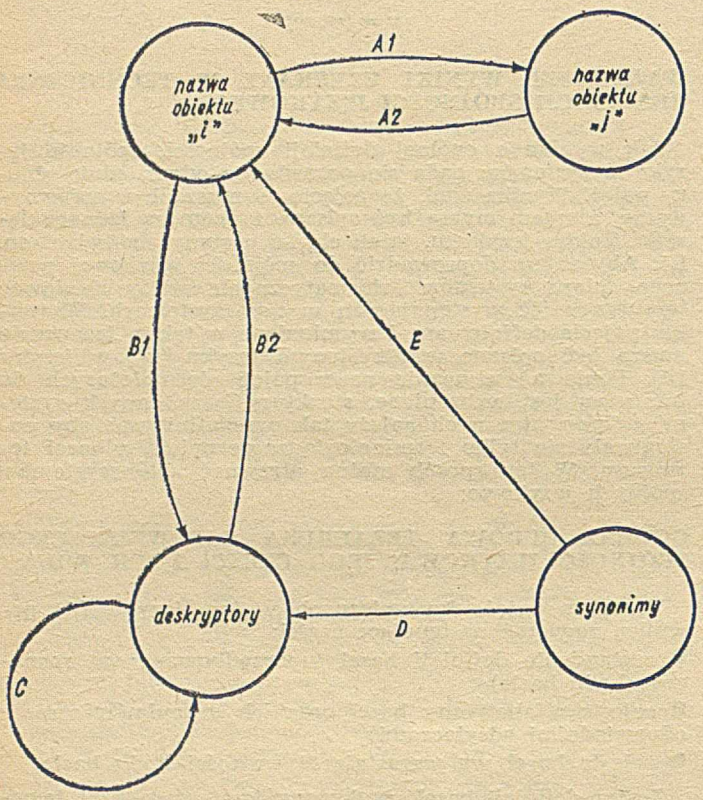
Informacja 1 występuje w każdej karcie tematycznej. Informacje 2, 3 są opcjonalne, w zależności od tego co definiuje karta tematyczna. Informacja 4 jest obiegowym synonimem pojęcia definiowanego w karcie tematycznej. Synonim ten jest używany aktualnie w systemach informatycznych organizacji jako nazwa definiowanego pojęcia.

Taka budowa kart tematycznych pozwala na stosunkowo proste i jednoznaczne przejście z terminologii aktualnie używanej na terminologię w języku L1. Jednoznaczność przejścia wynika z jednoznaczności przyporządkowania nazwy synonimu do nazwy definiowanego pojęcia.

Rejestr alfabetyczny haseł tezaurusa powinien zawierać alfabetyczny wykaz haseł i ich synonimów. Budowa elementarnego wiersza rejestru alfabetycznego powinna być następująca:

nazwa hasła (lub synonimu)	kod hasła (opcjonalnie)	ciąg par uporządkowanych o postaci: nazwa odsyłacza, kod (opcjonalnie)

Charakterystyczne dla tej części tezaurusa jest istnienie tzw. odsyłaczy. Istnieje kilka typów odsyłaczy, co obrazuje graf odsyłaczy.



Rys. 1.

Objaśnienia symboli

A1 i A2 — w strukturę nazwy obiektu i wchodzi nazwa obiektu j lub na odwrót np. „konto środków trwałych" oraz „konto środków trwałych, grupa środków trwałych"

B2 — odsyłacz z deskryptora do podgrupy tematycznej (nazwy obiektu), którą ten deskryptor opisuje

B1 — odsyłacz z podgrupy tematycznej (nazwy obiektu) do wszystkich deskryptorów opisujących ten obiekt

D — odsyłacz z synonimu deskryptora do nazwy deskryptora odpowiadającego temu synonimowi

E — odsyłacz z synonimu nazwy obiektu do tej nazwy

C — odsyłacz z deskryptora do wszystkich pozostałych deskryptorów zależnych opisujących dany obiekt

Funkcja odsyłaczy jest oczywista. Typ A1 i A2 daje związki pomiędzy nazwami obiektów. Typ B1 pozwala odnaleźć wszystkie deskryptory związane z danym obiektem. B2 — umożliwia odnalezienie w tezaurusie obiektu, do którego należy dany deskryptor. C — pozwala znaleźć wszystkie deskryptory zależne (pozostające w relacji podporządkowania — zwierzchnictwa) dla danego obiektu. Typ D i E pozwala znaleźć odpowiadające synonimom nazwy w języku L1.

Konstrukcja tezaurusa pojęć opartego o algorytmiczny język L1 zapewnia osiągnięcie **spójności językowej.** Budując bazy danych systemów informatycznych wyłącznie w oparciu o taki tezaurus umożliwimy osiągnięcie **spójności informacyjnej** tych systemów.

Wstępne prace nad budową takiego tezaurusa rozpoczęto w Resortowym Ośrodku Informatyki Przemysłu Chemicznego ETOCHEM. Wyodrębniono następujące grupy tematyczne:

1. Obrót towarowy, krajowy i zagraniczny
2. Zatrudnienie i płace
3. Produkcja
4. Finanse — koszty
5. Środki trwałe
6. Transport
7. Inwestycje
8. Gospodarka materiałami i wyrobami (GM + wyroby gotowe + przedmioty nietrwałe)

W ramach grup tematycznych ustalono obiekty. I tak np. w grupie „Środki trwałe" są następujące obiekty: środek trwały, transakcja w zakresie środków trwałych, remonty. W grupie „Finanse — koszty": konto księgowe, dekret księgowy, plan kosztów.

Planuje się w pierwszym etapie objęcie tezaurusem całej sprawozdawczości statystycznej w zakresie przemysłu chemicznego, a dopiero potem pozostałych dziedzin.

Użytkownicy pozaresortowi chcąc korzystać z informacji zawartych w bazie danych RSI przemysłu chemicznego będą musieli oprzeć się w zakresie interpretacji informacji na tym tezaurusie.

LITERATURA

[1] Blikle A.: Automaty i gramatyki, PWN, 1971

[2] Flakiewicz W.: Systemy informowania kierownictwa, PWE, Warszawa, 1978

[3] Materiały konferencji ogólnokrajowej „Organizacja obiektowych systemów informatycznych opartych o bazę danych" Tow. Nauk. Org. i Kierownictwa, Bydgoszcz, 1980

[4] Materiały konferencji SPIS'78. Problemy spójności i współdziałania rządowego systemu informatycznego SPIS z resortowymi systemami informatycznymi, OBR SPIS, Warszawa, 1979

[5] Oleński J.: Granice i instrumenty spójności centralnych systemów informatycznych, INFORMATYKA nr 2, 3/1979

[6] Oleński J., Peuker Z.: Podsystem instrumentalny SPIS-SŁOWNIK — projekt organizacji i funkcjonowania, Wiadomości statystyczne nr 3, 1979.

ANDRZEJ KORPAL
MERA-OBREUS
Toruń
ANDRZEJ KUBIAK
Akademia Techniczno-Rolnicza
Bydgoszcz

Architektura systemów wielomikroprocesorowych

Dalszy rozwój systemów komputerowych wymaga ciągłego poprawiania ich gotowości użytkowej, czyli tzw. dyspozycyjności, oraz zwiększania wydajności przetwarzania. Obecnie jest to możliwe dzięki zastosowaniu systemów wielomikroprocesorowych.

Systemem wielomikroprocesorowym jest system cyfrowy, zbudowany z dwóch lub więcej mikroprocesorów o porównywalnych własnościach, w którym wszystkie mikroprocesory mają dostęp do wspólnej pamięci oraz przynajmniej do części urządzeń peryferyjnych, wszystkie też są sterowane jednym, wspólnym systemem operacyjnym, a przesłania informacji pomiędzy modułami funkcjonalnymi odbywają się z pełną szybkością mikroprocesorów.

Systemy takie zbudowane z mikroprocesorów i towarzyszących im układów mikroelektronicznych o wielkiej (LSI) i bardzo wielkiej (VLSI) skali integracji, wyróżniają się niespotykaną dotąd dyspozycyjnością, która wynika z ich dużej elastyczności i wysokiej niezawodności. Ponadto, zastosowanie wielu mikroprocesorów w jednym systemie zapewnia uzyskiwanie wysokiej wydajności, głównie dzięki możliwości jednoczesnego przetwarzania wielu zadań oraz lepszego i bardziej równomiernego obciążenia urządzeń systemu.

Reasumując, systemy wielomikroprocesorowe łączą w sobie możliwości nowoczesnych układów mikroelektronicznych z udoskonaloną organizacją wieloprocesorowego systemu komputerowego. Niniejszy artykuł omawia wybrane zagadnienia architektury systemów wielomikroprocesorowych, stanowiące wprowadzenie do ich analizy i projektowania.

KLASYFIKACJA

Stosowane są trzy podstawowe klasyfikacje systemów wielomikroprocesorowych [1, 2, 9]:

- funkcjonalna, oparta na ilości strumieni danych i strumieni instrukcji oraz sposobie ich współdziałania w systemie
- sprzętowa, oparta na strukturze połączeń wewnętrznych systemu
- topologiczna, oparta na logicznym i fizycznym rozmieszczeniu mikroprocesorów w systemie.

Klasyfikacja funkcjonalna

W systemach wielomikroprocesorowych wyróżnia się dwa rodzaje organizacji funkcjonalnej [2, 9]:

- typu SIMD (Single Instruction — Multiple Data)
- typu MIMD (Multiple Instruction — Multiple Data).

W systemach typu SIMD występuje jeden strumień instrukcji i wiele strumieni danych. Charakteryzuje się to tym, że wykonywana jest równolegle przez wszystkie mikroprocesory ta sama instrukcja, lecz na różnych zestawach danych. Przykładem może być system realizujący operacje wektorowe, np. mnożenie lub odwracanie macierzy, gdzie każdy mikroprocesor wylicza jeden wyraz macierzy wynikowej. W ten sposób całe działanie odbywa się w jednym cyklu przetwarzania.

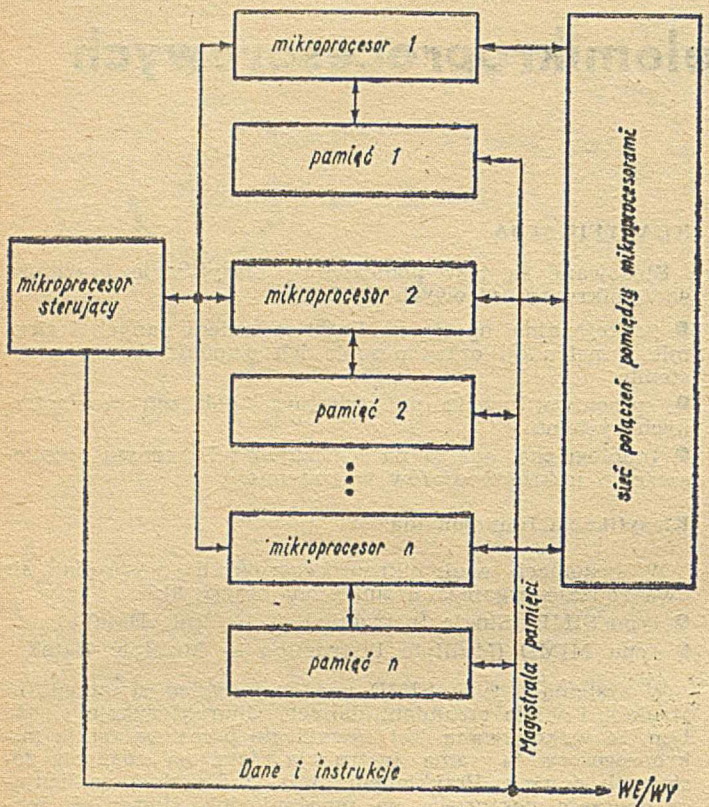
Systemy SIMD nazywane są często równoległoprocesorowymi i dzielone są na trzy podklasy: procesory macierzowe lub tablicowe, zespoły przetwarzające i procesory

Mgr inż. Andrzej KORPAL ukończył w 1968 roku studia na Wydziale Elektroniki i Elektrotechniki Akademii Techniczno-Rolniczej w Bydgoszczy, w specjalności telekomutacji.
Od 1973 roku pracuje w Instytucie Maszyn Matematycznych, Oddział w Toruniu, zajmując się systemami komputerowego sterowania zespołami obrabiarek. Za opracowanie i wykonanie systemu sterowania „Centrum produkcyjnym KOR-1" otrzymał w 1976 roku Nagrodę Ministra Przemysłu Maszynowego I Stopnia oraz Nagrodę Ministra Nauki, Szkolnictwa Wyższego i Techniki. Po przekształceniu Instytutu w Toruniu w Ośrodek Badawczo-Rozwojowy Elektronicznych Układów Specjalizowanych MERA, pracuje od roku 1978 na stanowisku Zastępcy Dyrektora ds. Systemów w Zakładzie Doświadczalnym.

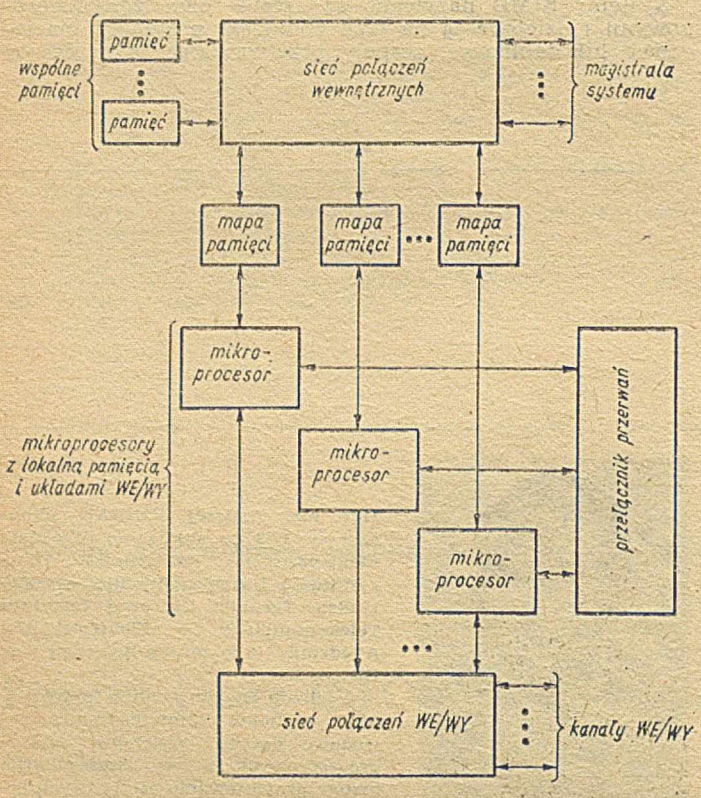
Mgr inż. Andrzej KUBIAK ukończył w 1962 r. studia na Wydziale Elektrycznym Politechniki Poznańskiej. Obecnie pracuje w Zakładzie Techniki Cyfrowej Instytutu Telekomunikacji i Elektrotechniki Akademii Techniczno-Rolniczej w Bydgoszczy.
Specjalizuje się w technice cyfrowej. Jego aktualne zainteresowania zawodowe dotyczą systemów mikroprocesorowych a w szczególności metod projektowania użytkowych systemów mikroprocesorowych.

skojarzeniowe. Podklasy te różnią się między sobą organizacją pamięci i możliwościami mikroprocesora sterującego. Uproszczony schemat systemu typu SIMD przedstawia rys. 1.

W systemach typu MIMD istnieje wiele strumieni instrukcji i wiele strumieni danych (rys. 2). Systemy te charakteryzują się jednoczesnym wykonywaniem wielu różnych zadań na różnych strumieniach danych. Systemy MIMD dzieli się na dwie podklasy: systemy rozdzielone i systemy wieloprocesorowe.



Rys. 1. Schemat blokowy systemu typu SIMD



Rys. 2. Schemat blokowy systemu typu MIMD

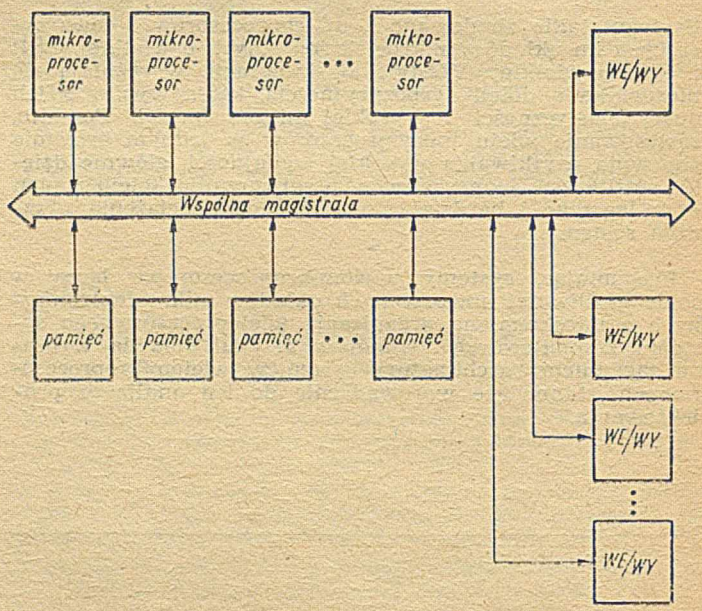
W systemie rozdzielonym, każdy z mikroprocesorów ma na stałe przydzielone zadanie, które wykonuje samodzielnie, podobnie jak w systemie jednoprocesorowym. Zagadnieniem wspólnym dla wszystkich mikroprocesorów jest tu problem korzystania z takich wspólnych zasobów, jak: pamięć, urządzenia WE/WY, magistrale. Taka organizacja systemu jest mało elastyczna, lecz stosunkowo prosta przy oprogramowaniu i uruchomieniu. Przy poprawnym rozdzieleniu zadań na poszczególne mikroprocesory oraz odpowiednio wysokiej niezawodności sprzętu, organizacja rozdzielona (MIMD) zapewnia dobrą wydajność i dyspozycyjność systemu.

W systemie wieloprocesorowym (MIMD) mikroprocesory wspólnie realizują zadania, które są podzielone na operacje. Operacje przydzielane są mikroprocesorom w sposób dynamiczny w zależności od stanu systemu oraz obciążenia i możliwości poszczególnych mikroprocesorów. Taka organizacja systemu zapewnia jego dużą dyspozycyjność i wysoką wydajność. Posiada jednak tę wadę, że oprogramowanie i uruchomienie systemu jest zadaniem trudnym i bardzo pracochłonnym.

Klasyfikacja sprzętowa

W systemach wielomikroprocesorowych stosowane są trzy podstawowe struktury połączeń wewnętrznych pomiędzy modułami funkcjonalnymi [1, 2, 7, 8, 9]:

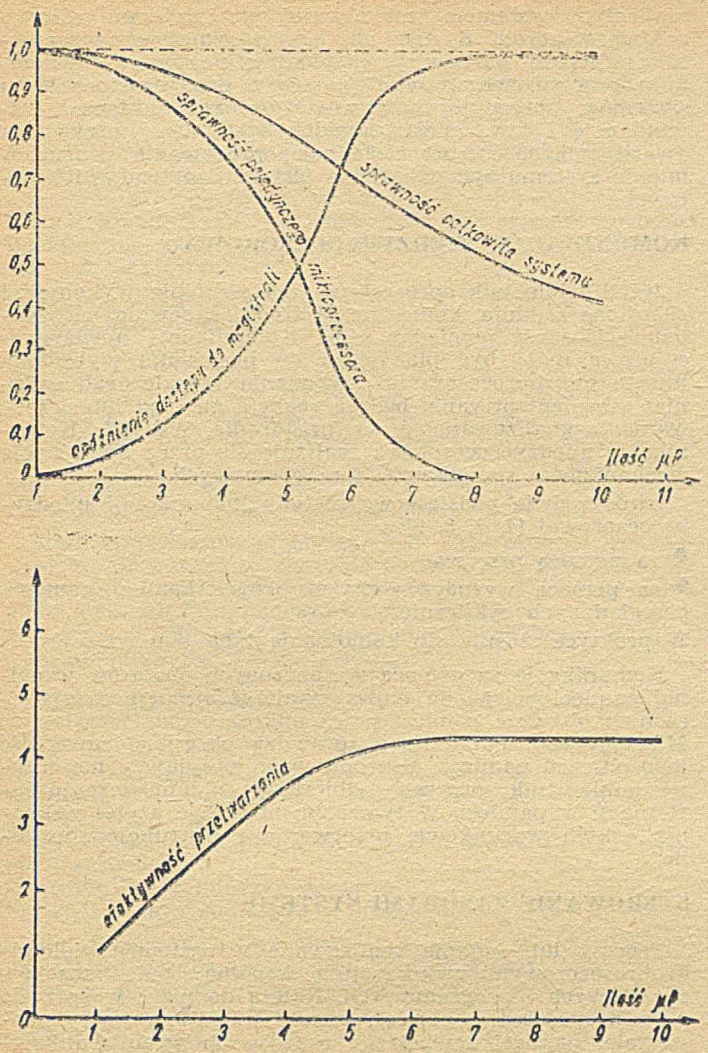
- wspólna magistrala pracująca z podziałem czasu
- przełącznica krzyżowa
- magistrala wieloszynowa-wielobramowa.



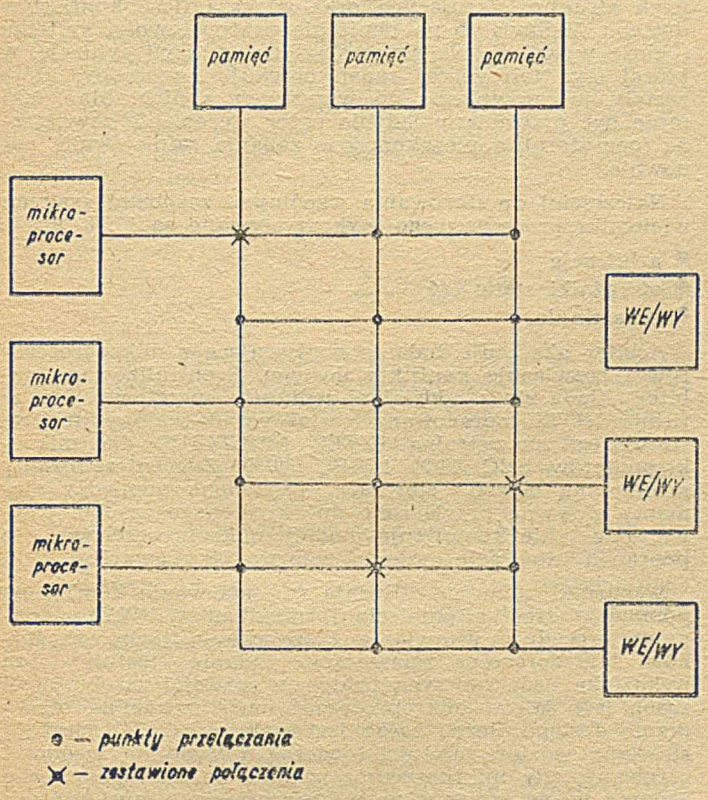
Rys. 3. Organizacja systemu ze wspólną magistralą pracującą z podziałem czasu

System ze wspólną magistralą, przedstawiony na rys. 3, jest najprostszy i najtańszy. Jednak ze względu na to, że liczba modułów włączonych do wspólnej magistrali jest ograniczona jej dopuszczalnym obciążeniem, struktura ta jest odpowiednia dla systemów o mniejszej liczbie mikroprocesorów. Dopuszczalna ich liczba, zależna od ich szybkości, czasu dostępu do magistrali oraz rodzaju wykonywanych zadań, waha się od kilku do kilkudziesięciu. Np. dla systemu zbudowanego z mikroprocesorów INTEL 8080 zwiększenie liczby mikroprocesorów powyżej sześciu nie powoduje już wzrostu wydajności [3]. Ilustruje to rys. 4.

System z przełącznicą krzyżową (rys. 5) umożliwia współpracę większej liczby mikroprocesorów oraz ich lepsze wykorzystanie w systemie. Przełącznica krzyżowa, stanowiąca pewnego rodzaju pole komutacyjne, jest jednak złożona i kosztowna. Zdarza się, że koszt przełącznicy przekracza koszt pozostałej części systemu. Systemy te są jednak często stosowane ze względu na dużą elastyczność połączeń modułów funkcjonalnych i wysoką wydajność.

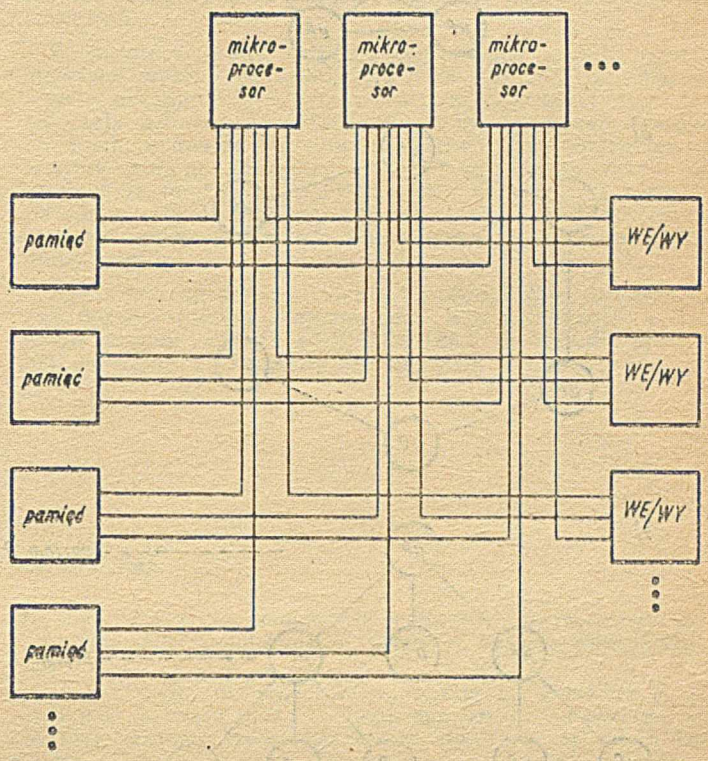


Rys. 4. Przykłady charakterystyk systemu ze wspólną magistralą pracującą z podziałem czasu



Rys. 5. Organizacja systemu z przełącznicą krzyżową

System z magistralą wieloszynową-wielobramową (przedstawiony na rys. 6) posiada najbardziej rozbudowaną sieć połączeń wewnętrznych. Polega ona na tym, że każdy moduł funkcjonalny jest fizycznie połączony z pozostałymi modułami systemu. Taka sieć połączeń nie jest specjalnie kosztowna, wymaga jednak stosowania indywidualnych układów arbitracji dostępu do poszczególnych modułów, przez co stają się one złożone i drogie. Dlatego też magistrala ta stosowana jest w mniejszych systemach.



Rys. 6. Organizacja systemu z magistralą wieloszynową-wielobramową

Klasyfikacja topologiczna

Fizyczne i logiczne rozmieszczenie mikroprocesorów w systemie jest zagadnieniem szczególnie istotnym w tzw. wielkich systemach wielomikroprocesorowych, które zawierają 100 i więcej mikroprocesorów. Rozróżnia się następujące rodzaje topologii systemów [6]:

- zupełną
- pierścieniową
- drzewiastą.

System o topologii zupełnej (rys. 7a) zapewnia bezpośrednią współpracę każdego mikroprocesora z pozostałymi mikroprocesorami działającymi w systemie.

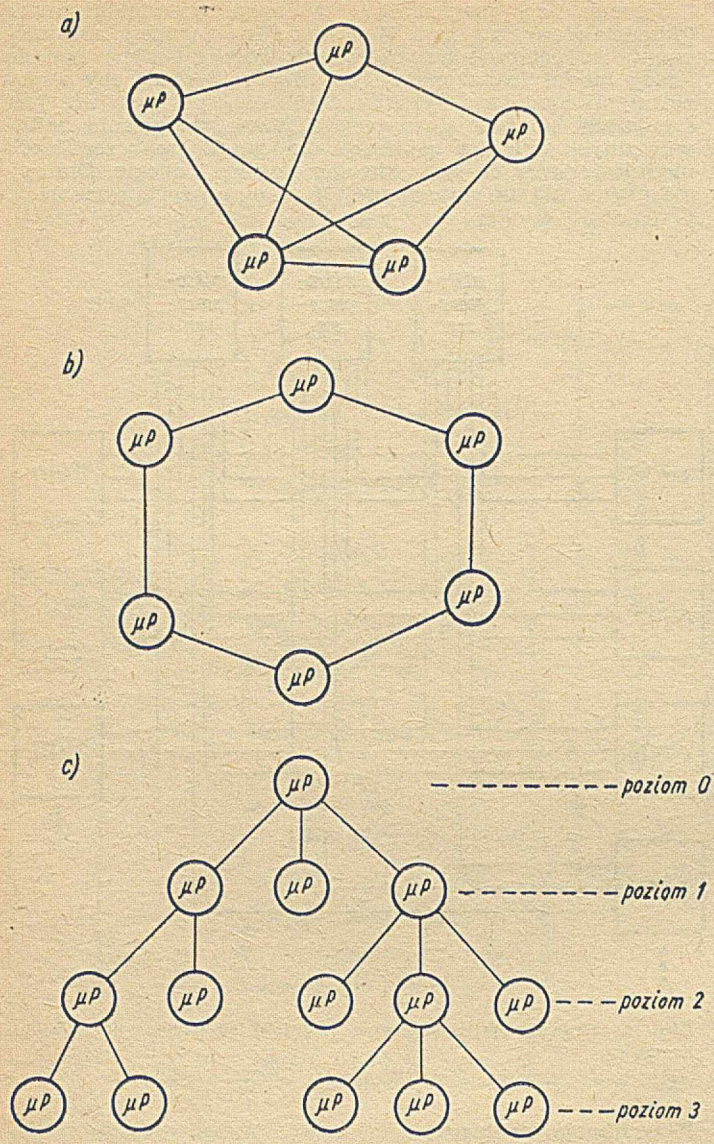
System o topologii pierścieniowej charakteryzuje się cyklicznym komunikowaniem się mikroprocesorów. Każdy z nich może współpracować bezpośrednio z dwoma sąsiadującymi z nim w pierścieniu mikroprocesorami. Ilustruje to rys. 7b.

W systemie o topologii drzewiastej (rys. 7c), zwanej również hierarchiczną, mikroprocesory tworzą strukturę wielopoziomową.

SYSTEMY OPERACYJNE

System operacyjny jest najważniejszym elementem systemu wielomikroprocesorowego. Od jakości systemu operacyjnego zależy optymalne wykorzystanie możliwości nowoczesnego sprzętu — dla osiągnięcia dużej dyspozycyjności i wysokiej wydajności systemu. W systemach wielomikroprocesorowych można zastosować trzy podstawowe rodzaje systemów operacyjnych:

- typu „nadrzędny-podrzędny"
- z oddzielnymi programami zarządzającymi
- symetryczny.



Rys. 7. Podstawowe rodzaje topologii systemów: a) zupełna, b) pierścieniowa, c) drzewiasta

W systemie typu „nadrzędny-podrzędny" program zarządzający wykonywany jest przez jeden wybrany mikroprocesor, zwany mikroprocesorem nadrzędnym; pozostałe mikroprocesory wykonują przydzielone im przez niego zadania. Bezpośredni dostęp do takich zasobów systemu operacyjnego, jak: tablice i procedury systemowe, ma wyłącznie mikroprocesor nadrzędny. Jest to w sumie najprostszy i najtańszy system operacyjny. Posiada jednak tę zasadniczą wadę, że nie zapewnia dużej elastyczności systemu wielomikroprocesorowego, a awaria mikroprocesora nadrzędnego jest katastrofalna.

System operacyjny z oddzielnymi programami zarządzającymi charakteryzuje się tym, że każdy mikroprocesor posiada w pamięci systemu własną, indywidualną kopię programu zarządzającego i realizuje ten program w zakresie niezbędnym dla wykonania przydzielonych mu zadań. Dostęp do zasobów systemowych (tablic i procedur) wymaga stosowania specjalnego arbitra programowego lub programowo-sprzętowego. Ten rodzaj systemu operacyjnego zapewnia dobrą dyspozycyjność i wydajność systemu wielomikroprocesorowego. Jest jednak złożony i dość kosztowny.

System operacyjny symetryczny, charakteryzuje się tzw. symetrycznym lub anonimowym traktowaniem mikroprocesorów. Program zarządzający może być wykonywany przez każdego z nich. W ten sposób każdy mikroprocesor wykonuje funkcje programu zarządzającego związane z aktualnie realizowanym zadaniem oraz te funkcje, które są niezbędne do zainicjowania nowego zadania. Sterowanie całym systemem jest również przekazywane różnym mikroprocesorom. System operacyjny symetryczny zapewnia najlepsze parametry eksploatacyjne systemu wielomikroprocesorowego. Jako jedyny z przedstawionych tu systemów operacyjnych pozwala uzyskać faktyczną redundancję i tzw. miękki upadek systemu w przypadku awarii mikroprocesorów. Jednak opracowanie i uruchomienie systemu symetrycznego jest bardzo trudne i kosztowne.

KOMUNIKACJA MIĘDZYPROCESOROWA

Jeżeli wiele mikroprocesorów ma wspólnie wykonywać określone zadania, konieczne są środki zapewniające wzajemną łączność pomiędzy nimi. Komunikacja międzyprocesorowa może być planowana (w przypadku wywołania innego mikroprocesora do wykonania niezależnego zadania) lub nieplanowana, będąca reakcją na błąd lub przerwanie, sygnalizowane przez urządzenia WE/WY. Komunikację międzyprocesorową realizuje się przez kombinację mechanizmów sprzętowych i programowych.

Istnieją dwie podstawowe metody komunikacji międzyprocesorowej [4, 7, 8, 9]:

- za pomocą przerwań
- za pomocą wyznaczonych obszarów wspólnej pamięci, przeglądanych cyklicznie.

W praktyce stosuje się kombinację tych obu rodzajów.

Komunikacja za pomocą wyznaczonych obszarów wspólnej pamięci polega na umieszczaniu informacji w wyznaczonym obszarze pamięci przez mikroprocesor wysyłający. Mikroprocesor odbierający sprawdza okresowo zawartość tego obszaru pamięci. Mikroprocesor wysyłający może zawiadamiać mikroprocesor odbierający o umieszczeniu informacji w pamięci. W tym celu mogą być użyte: wskaźniki, skoki warunkowe, przerwania i instrukcje specjalne.

STEROWANIE ZASOBAMI SYSTEMU

Procesy lub zadania realizowane w systemie wielomikroprocesorowym wykorzystują wspólną pulę zasobów sprzętowych i programowych. Należą do nich: w sprzęcie — mikroprocesory, pamięć, kanały WE/WY, rejestry, magistrale, pamięci zewnętrzne oraz w oprogramowaniu — programy, zbiory danych, bufory, kolejki, zmienne.

Im bardziej wykorzystywane są dostępne zasoby, tym większa jest kontrola wymagana dla ich przydziału i rozwiązywania konfliktów dostępu. Przesadne wykorzystywanie zasobów powoduje konieczność stosowania złożonych struktur sterujących oraz większą ilość konfliktów dostępu. Wpływa to na obniżenie przepustowości systemu i może wywołać tzw. śmiertelny uścisk, czyli sytuację, w której dwa lub więcej zadań oczekuje na zasoby, które zostały już przypisane do wszystkich z nich. Stan taki może być przełamany jedynie przez sterowanie zewnętrzne (operatora), a niezakończone zadania zainicjowane ponownie.

Najczęściej do sterowania wspólnymi zasobami systemu wielomikroprocesorowego wykorzystywane są [7, 8, 9]:

- arbitracja
- wskaźniki stanu
- przerwania.

Arbiter akceptuje żądania napływające z mikroprocesorów i rozwiązuje konflikty dostępu. Scentralizowany arbiter składa się z oddzielnej jednostki sprzętowej. Np. w firmie INTEL opracowany został układ scalony arbitra, sterującego magistralną INTEL MULTIBUS dla mikrokomputerów SBC 80/20 i SBC 80/30. Zdecentralizowany arbiter ma obwody sterujące rozdzielone pomiędzy elementy aktywne przyłączone do określonego zasobu. Rozwiązanie takie komplikuje moduły mikroprocesorowe, ale poprawia integrację systemu w przypadku awarii.

Wskaźniki stanu umożliwiają rozwiązywanie konfliktów pomiędzy wspólną pamięcią i układami WE/WY — poprzez procedurę ustawiania i testowania tych wskaźników. Mikroprocesor zgłaszający żądanie testuje stan wskaźnika będącego sygnalizatorem zajętości zasobu. Jeśli zasób jest zajęty, mikroprocesoor musi czekać na otrzymanie dostępu. Jeżeli zasób jest wolny, wskaźnik jest ustawiany w stan zajętości (podczas dostępu do zasobu) i zerowany, gdy mikroprocesor zwalnia zasób. Jednoczesne żądanie zasobu przez wiele mikroprocesorów musi być tak rozwiązywane, aby tylko jeden z ich otrzymał dostęp.

Przerwania w systemie wielomikroprocesorowym mają szerokie zastosowanie. Są one używane w przypadku:
- wewnętrznych błędów mikroprocesora, takich jak: błąd parzystości, błędny kod operacji, adres spoza zakresu
- obsługi sygnałów zegarowych, realizujących odmierzanie czasu i zegary czasu rzeczywistego
- obsługi urządzeń zewnętrznych: gotowości, zakończenia transmisji w kanale, odłączenia urządzenia.

Przerwania mogą być również wykorzystane do synchronizacji komunikacji pomiędzy mikroprocesorami (podczas korzystania ze wspólnej pamięci) oraz przypisywania zadań lub urządzeń.

Istnieje kilka metod obsługi przerwań w systemach wielomikroprocesorowych. Urządzenia zewnętrzne mogą być przypisane z góry do określonego mikroprocesora lub dynamicznie kierowane do mikroprocesora najlepiej wyposażonego do obsługi. Przypisanie stałe może być zrealizowane przez nadrzędnego arbitra sprzętowego, czyli tzw. przełącznik przerwań, którym może być bardzo szybki mikroprocesor. Przypisanie dokonywane jest na podstawie zdolności obsługi, dostępności, alokacji zadań i priorytetów programowych każdego mikroprocesora.

Stałe przypisanie przerwań jest najprostszym rozwiązaniem, ale ogranicza znacznie elastyczność systemu i nie zapewnia stopniowego zmniejszenia się wydajności (tzw. miękkiego upadku) podczas awarii mikroprocesorów. Dynamiczne przypisywanie przerwań jest pozbawione tych wad, niemniej jest bardziej złożone.

UKŁADY MIKROPROCESOROWE

Ogólnie, wymagania na układy mikroprocesorowe [2, 5, 7, 8, 9] pożądane dla systemów wielomikroprocesorowych, wynikają z funkcji, jakie realizuje mikroprocesor w takim systemie. Są to:
- współpraca z układami sterującymi dostępem do wspólnych zasobów
- komunikacja międzyprocesorowa
- współpraca z kanałami i urządzeniami WE/WY.

Dla zapewnienia współpracy z układami sterującymi wspólnymi zasobami systemu konieczne jest, aby mikroprocesor posiadał wejście HOLD i wyjście HOLD ACKNOWLEDGE. Wejście HOLD zatrzymuje pracę mikroprocesora oraz ustawia jego magistrale danych, adresów i linii sterujących w stan wysokiej impedancji (po zakończeniu aktualnie wykonywanej instrukcji), aktywizując jednocześnie wyjście HOLD ACKNOWLEDGE. Po zaniku sygnału HOLD mikroprocesor rozpoczyna wykonywanie następnej instrukcji.

Cecha trójstanowości mikroprocesorów jest szczególnie ważna w małych systemach wielomikroprocesorowych, w których magistrale wewnętrzne mikroprocesorów i magistrale systemowe są wspólne. W większych systemach stosowane są układy przekształcające magistralę wewnętrzną w magistralę systemową. Układy takie zawierają odpowiednią logikę sterującą, bufory danych i adresów oraz zapewniają odłączanie i dołączanie mikroprocesora do magistrali systemu.

Do współpracy z urządzeniami WE/WY, a także do współpracy z innymi zasobami systemu, konieczne jest wejście sterujące READY, które umożliwia ustawienie mikroprocesora w stan WAIT (gdy urządzenie, do którego zostało wysłane zgłoszenie nie odpowiada z powodu konfliktu dostępu do magistrali lub zajętości).

W celu zapewnienia efektywnej komunikacji międzyprocesorowej i współpracy z innymi modułami systemu konieczne są sygnały INTERRUPT i INTERRUPT ACKNOWLEDGE, umożliwiające przyjmowanie i obsługę przerwań.

Większość układów mikroprocesorowych spełnia przedstawione wyżej wymagania. Są to wymagania minimalne. Często pożądane jest, aby układy mikroprocesorowe dostarczały informacji o stanie cyklu maszynowego i odpowiednich sygnałów strobu magistrali na wydzielonych wyprowadzeniach. Sygnały te mogą być wykorzystane do sterowania magistralą i wykrywania awarii. Jest również pożądane, aby mikroprocesory posiadały możliwość adresowania indeksowego lub pośredniego, w celu realizacji mapy pamięci.

Jeżeli mikroprocesor pracuje w rozbudowanym module mikroprocesorowym, celowe jest wyprowadzenie z takiego modułu wyjścia BUS REQUEST do układu arbitra magistrali oraz dołączenia wejścia BUS GRANT.

LITERATURA

[1] yun Feng T.: A configurable multiple — microprocessor organization, Microarchitecture of Computer Systems EUROMICRO, 1975

[2] Fuller S. H., Ousterhout J. K., Raskin L., Rubinfeld P. J., Siudhu P. J., Swan R. J.: Multi-microprocessors: An overview an working example, Proceedings of the IEEE, vol. 66, no. 2, 1978

[3] Hoener S., Roehder W.: Efficiency of a multimicroprocessor system with time shared busses, EUROMICRO 1977

[4] Hopkins A. L., Lala J. H., Smith T. B.: FTMP — A highly reliable fault-tolerant multiprocessor for aircraft, Proceedings of the IEEE, vol. 66, no. 10, 1978

[5] Okada Y., Tajima H., Mori R.: A novel multiprocessor array, Second Symposium on Micro-Architecture EUROMICRO 1976

[6] Prener D.: Large multimicroprocessor systems, Microprocessors and microsystems, vol. 3, 1979

[7] Thurber K. J.: Parallel processor architectures Part 1: General purpose systems, Computer Design, vol. 1, 1979

[8] Thurber K. J.: Parallel processor architectures-Part S. Special purpose systems, Computer Design, vol. 2, 1979

[9] Weissberger A. J.: Analysis of multiple — microprocessor system architectures, Computer Design, vol. 1977.

Związek Zawodowy Informatyków

Po wielu latach nieudanych prób powstała wreszcie ogólnokrajowa organizacja informatyków. 9 października br. został zarejestrowany Niezależny Samorządny Związek Zawodowy Pracowników Informatyki, stojący na gruncie porozumień zawartych pomiędzy Komisjami Rządowymi a Międzyzakładowymi Komitetami Strajkowymi na Wybrzeżu. Związek jest powszechną organizacją pracowników informatyki, zrzeszonych na zasadzie dobrowolności. Jest on niezależny od organów administracji państwowej i organizacji politycznych. Związek ma charakter ogólnopolski, siedzibą Zarządu jest Warszawa.

Członkiem Związku może być każdy pracownik zatrudniony w jednostce organizacyjnej zajmującej się informatyką. Członkowstwo Związku nabywa się z chwilą przyjęcia deklaracji członkowskiej uchwałą właściwej Rady Zakładowej.

Związek posiada dwustopniową strukturę organizacyjną. Władzami na szczeblu krajowym są: Zjazd Delegatów, Zarząd Związku, Komisja Rewizyjna; na szczeblu zakładowym: Zakładowe Zebranie Członków, Rada Zakładowa oraz Zakładowa Komisja Rewizyjna. Na obu szczeblach mogą być powoływane sekcje, komisje lub zespoły problemowe. W razie potrzeby mogą być też konstytutowane regionalne zespoły koordynacyjne. Jednym z zadań Związku jest powoływanie do życia stowarzyszeń lub związków twórczych.

12 października odbył się w Warszawie I Walny Zjazd Niezależnego Samorządnego Związku Pracowników Informatyki, na którym omówiono zasady funkcjonowania Związku oraz wybrano jego władze. W najbliższych numerach INFORMATYKI opiszemy szczegółowo sytuację nowo powołanej organizacji związkowej.

REDAKCJA

W poniższym artykule zostały opisane czynniki wpływające na zmniejszenie się szybkości transmisji informacji, porównano też jej realną wielkość z szybkością nominalną. Systemy teleinformatyczne, o których tu mowa, to jeszcze stosunkowo odległa przyszłość polskiej informatyki. Niemniej — przyszłość nieuchronna, warto więc już dzisiaj przedyskutować podstawowe kwestie w tym zakresie. (Red.)

ANDRZEJ BYLICKI
Wojewódzki Urząd Statystyczny
Olsztyn

Szybkość efektywna transmisji synchronicznej

Z punktu widzenia użytkowników systemów teleinformatycznych jednym z najistotniejszych parametrów jest tzw. przepustowość informacyjna, definiowana jako ilość informacji użytkowej przesyłana między dwoma punktami systemu w jednostce czasu.

Można przyjąć, że — ze względu na szybkość działania — podstawowe węzły systemów, jakimi są komputery i minikomputery, a także multipleksery i urządzenia końcowe, nie są źródłem ograniczeń przepustowości informacyjnej. Zasadnicze ograniczenia powodowane są przez najwolniejsze elementy, którymi są środki transmisji — łącza (linie telefoniczne i modemy)

Przyjęcie nominalnej szybkości pracy łącza za miernik jego przepustowości może prowadzić do znacznych błędów oceny przepustowości całego systemu, a w efekcie — do błędnej oceny wydolności i przydatności tego ostatniego, co powoduje rozbieżność pomiędzy osiągniętymi rezultatami a oczekiwaniami użytkowników. Niewłaściwość takiego miernika wynika stąd, że oprócz transmisji łącze obciążane jest szeregiem innych funkcji, które — z punktu widzenia użytkowników — zmniejszają szybkości jego pracy. Ma to związek z technicznymi środkami przesyłania (z ich funkcjonowaniem), a także — z organizacją przesyłania. Te dodatkowe obciążenia powodują, że rzeczywista szybkość przesyłania informacji użytkownika jest znacznie mniejsza od nominalnej. Rzeczywista szybkość transmisji informacji użytkowej nazywana jest szybkością efektywną.

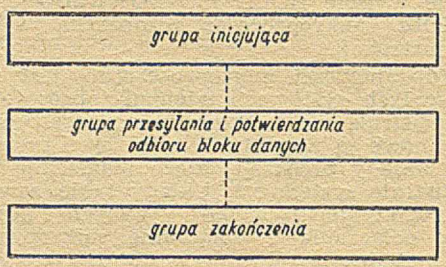
Wszystkie przedstawione poniżej rozważania dotyczą łącza, przez które dane przesyłane są blokowo w trybie synchronicznym półdupleksowym. Ten typ pracy jest obecnie powszechnie stosowany w teleinformatyce. Jest on też najbardziej złożony, zawierający najwięcej różnych ograniczeń. Jego analiza daje zatem możliwie pełny obraz zagadnienia.

Mgr inż. Andrzej BYLICKI, po ukończeniu w 1973 roku studiów na Wydziale Elektroniki Politechniki Warszawskiej (specjalność maszyny matematyczne) podjął pracę w Ośrodku Informatycznym Wojewódzkiego Urzędu Statystycznego w Olsztynie.
W ramach swoich obowiązków służbowych, po odbyciu przeszkolenia specjalistycznego w firmie ICL w Wielkiej Brytanii, odpowiedzialny był za wdrożenie do pracy terminala OCL-7503, a także zajmował się utworzeniem pełnego funkcjonalnie systemu oprogramowania dla prac lokalnych, rozszerzającego możliwości tego terminala. Jego zainteresowania skupiają się na szeroko rozumianych aspektach użytkowych systemów zdalnego przetwarzania danych.

PROTOKÓŁ TRANSMISJI

Jak już stwierdzono, jednym z czynników ograniczających szybkość transmisji jest przyjęta w danym systemie organizacja procesu przesyłania, uwzględniająca stosowane kody, wewnętrzne podziały informacji, znaczenie przesyłań itp. Tak rozumiana organizacja wymiany informacji pomiędzy dwoma punktami nosi nazwę protokołu. Obecnie na świecie używanych jest co najmniej kilkanaście różnych rodzajów protokołów, przy czym z reguły każdy producent zestawów teletransmisyjnych wprowadza własny protokół. Wszystkie jednak rodzaje protokołów niezależnie od szczegółów ich realizacji, można zamknąć w pewne wspólne ramy.

Ogólnie biorąc, możliwe jest przedstawienie każdej kompletnej transmisji jako ciągu przesłań różnych co do funkcji grup komunikatów (rys. 1). Każda z tych grup ma inne zadanie i inną zawartość informacyjną. W ramach grupy wyróżnić można komunikaty przesyłane w obu kierunkach w trybie pytanie—odpowiedź. Każdy komunikat poprzedzany jest zespołem znaków synchronizujących.



Rys. 1. Podział transmisji na grupy przesłań

Grupa inicjująca składa się z ciągu przesyłanych w obu kierunkach komunikatów znakowych, nie niosących żadnej informacji użytkowej, służących do wstępnego zaadresowania odbiorcy, jego urządzeń oraz do ustalania ich stanu. W rozbudowanych systemach transmisji, np. w sieciach komputerowych, w grupie tej mogą być zawarte także cechy adresowe nadawcy, informacje o optymalnej w danej sytuacji drodze przesyłania itp.

Grupa przesyłania i potwierdzania odbioru bloku danych jest zasadniczą w przesłaniu, gdyż w niej zawarty jest blok informacji użytkowej (o akceptowalnej długości, tj. maksymalnej liczbie bitów). Blok ten opakowany jest w tzw. kopertę, czyli znaki wyróżniające jego początek i koniec oraz znaki służące do dokonywanej u odbiorcy kontroli poprawności przesyłania. W odpowiedzi nadawca odbiera komunikat stwierdzający poprawność lub błędność przesyłania bloku danych. W zależności od treści tego komunikatu następnym przesłaniem będzie albo powtórna transmisja błędnego bloku, albo przekazany zostanie kolejny blok danych. Komunikat zwrotny informujący o aktualnym stanie transmisji nie niesie żadnej informacji użytkowej.

Grupa zakończenia służy do wskazania końca operacji przesyłania. Nie zawiera ona żadnej informacji użytkowej.

Jeśli w rozpatrywanym modelu ogólnym liczbę wszystkich bitów w grupach inicjującej i zakończenia oznaczymy przez i, a ilość bitów „koperty" i komunikatu potwierdzenia odbioru w grupie przesyłania bloku przez j, to liczba bitów nie niosąca informacji użytkowej, a konieczna do przesyłania liczby n bloków l-bitowych — przy założeniu bezbłędności transmisji — będzie równa:

$$N_1 = i + n \cdot j$$

Liczba ta w dalszej części artykułu nazywana będzie narzutem bitowym protokołu. W celu oszacowania wielkości tego narzutu dla łącza rzeczywistego, tj. dopuszczającego możliwość przekłamań, należy uwzględnić dodatkowe przesłania korekcyjne. Ich liczbę można określić korzystając z podstawowego parametru charakteryzującego jakość łącza, tj. elementowej stopy błędów. Jest ona definiowana jako stosunek liczby bitów przekłamanych do liczby wszystkich przesłanych bitów.

Zakładając, że przesyłane są bloki informacji o długości l-bitów oraz że błędy w łączu są jednakowo prawdopodobne w czasie i wzajemnie niezależne, otrzymamy, że prawdopodobieństwo poprawnej transmisji bloku jest:

$$p = (1-q)^l$$

gdzie: q — elementowa stopa błędu.

Prawdopodobieństwo pojawienia się bloków błędnych:

$$b = 1 - (1-q)^l.$$

Stąd przy przesyłaniu n bloków l-bitowych, liczbę bloków błędnych, wymagających powtórnego przesyłania, najłatwiej oszacować iloczynem liczby bloków przesyłanych n i prawdopodobieństwa b. O tyle więc przesłań musi być zwiększona sekwencja grupy przesyłania i potwierdzania odbioru bloku, przy czym te dodatkowe przesyłania nie niosą żadnej nowej informacji użytkowej.

Przyjmując, że dostatecznym przybliżeniem jest pojedyncza retransmisja bloku [1]), jako że liczba bloków błędnych przy istniejących wartościach q (od 10^{-4} do 10^{-6}) jest bardzo małym ułamkiem liczby przesyłanych bloków, możemy określić narzut bitowy powtórzeń:

$$N_2 = m \cdot (g + l) = n \cdot (j + l) \cdot [1 - (1-q)^{j+l}],$$

gdyż dodatkowo przesłać trzeba m razy blok l-bitowy wraz z jego j-bitową kopertą i potwierdzeniem.

Należy pamiętać, że elementowa stopa błędów jest w znacznym stopniu uzależniona od szybkości nominalnej i dla kolejnych szybkości 600, 1200, 2400 bit/s rośnie, przyjmując wartości: 2×10^{-6}, 5×10^{-6}, 10^{-5}. Dla dużo bardziej czułych i podatnych na zakłócenia transmisji (z szybkościami powyżej 2400 bit/s) elementowa stopa błędów nie daje się wyrazić jednoznacznie przybliżoną wartością, można ją jedynie lokalizować wewnątrz pewnego przedziału. Jest wtedy bardzo uzależniona od stosowanych rozwiązań technicznych, wyboru typów i metod korekcji, często bardzo złożonych i kosztownych.

Dla szybkości 4800 i 9600 bit/s przedziały te wynoszą odpowiednio: 10^{-5}—10^{-4} i 10^{-4}—10^{-3}. We wszelkich rozważaniach należałoby przyjmować, zgodnie z zasadą najgorszego przypadku, większe stopy błędów, a więc dla 4800 bit/s—10^{-4}, a dla 9600—10^{-3}. Jednak dla łącz istniejących najlepszą podstawą do określenia stopy błędów jest pomiar. Podowane czasami zależności wiążące q z szybkością nominalną są przybliżeniami dość ogólnymi, nie mogącymi uwzględniać różnych rozwiązań technicznych, zwłaszcza dla większych szybkości transmisji.

MODEMY

Modemy są również źródłem zmniejszenia szybkości pracy łącza. Wynika to z faktu, że w trakcie trnsmisji czas pracy modemów zużywany jest na przesyłanie i odbiór oraz na przełączanie kierunków. Jeśli uwzględnimy, że łączny czas przełączenia kierunków „nadawanie-odbiór" i „odbiór-nadawanie" dla modemu jednej strony jest ł oraz że dla danej procedury następuje k przełączeń w grupach inicjującej i zakończenia, a n przełączeń dla przesłania n bloków danych oraz

$$n \cdot [1 - (1-q)^{j+l}]$$

dla powtórzeń z powodu błędów, to czas wszystkich przełączeń jednej strony wyniesie:

$$t = k \cdot \tau + n \cdot \tau [2 - (1-q)^{j+l}] \qquad [1]$$

przy czym, w ogólnym przypadku: $l = l_N + l_O$

gdzie: l_N — opóźnienie nadawania, l_O — opóźnienie odbioru.

Jeśli przyjmiemy, że obie strony łącza pracują z różnymi czasami przełączeń τ_1 i τ_2 oraz że każde przesłanie w jedną stronę wymaga odpowiedzi strony odbierającej, to w wyrażeniu [1] należy uwzględnić łączny czas przełączania $\tau_1 + \tau_2$.

Ponieważ każdemu czasowi t pracy modemu z szybkością nominalną V_N odpowiada przesłanie określonej liczby bitów: $g = V_N \cdot t$, to liczba bitów możliwa do przesłania w czasie przełączeń kierunków wynosi:

$$N_3 = V_N \cdot k \cdot (\tau_1 + \tau_2) + n \cdot V_N \cdot (\tau_1 + \tau_2) \cdot [2 - (1-q)^{l+j}]$$

Wielkość N_3 nazywana będzie dalej narzutem bitowym przełączeń.

Ze wszystkich powyższych stwierdzeń wynika, że dla przesłania n bloków l-bitowych informacji użytkowej trzeba przesłać następujące liczby bitów:

$n \cdot l$ — dane
N_1 — narzut bitowy protokołu
N_2 — narzut bitowy powtórzeń
N_3 — narzut bitowy przełączeń

Stąd efektywna szybkość transmisji wynosi:

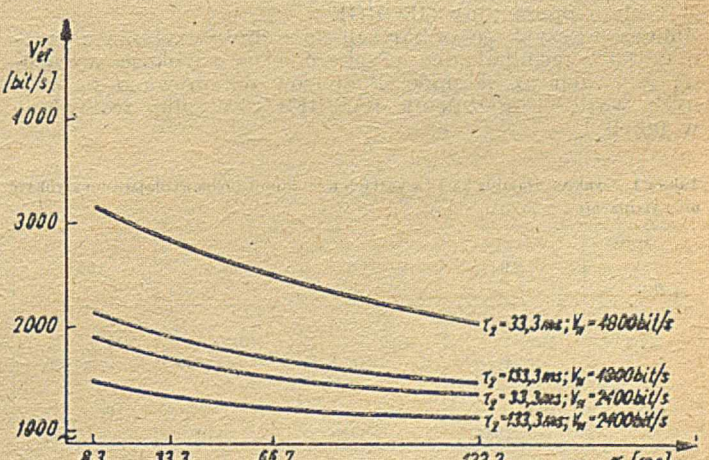
$$V_{ef} = \frac{V_N \cdot n \cdot l}{n \cdot l + N_1 + N_2 + N_3}$$

co po wstawieniu zależności dla N_1, N_2, N_3 prowadzi do zależności na szybkość efektywną. Uwzględniając dodatkowo, że liczba przesyłanych bloków jest zwykle duża, można wartość otrzymanego wyrażenia przybliżyć jego przypadkiem granicznym $n \to \infty$, otrzymując w rezultacie następującą zależność:

$$V'_{ef} = \lim_{n \to \infty} V_{ef} = \frac{V_N \cdot l}{[1 + j + V_N(\tau_1 + \tau_2)] \cdot [2 - (1-q)^{l+j}]} \qquad [2]$$

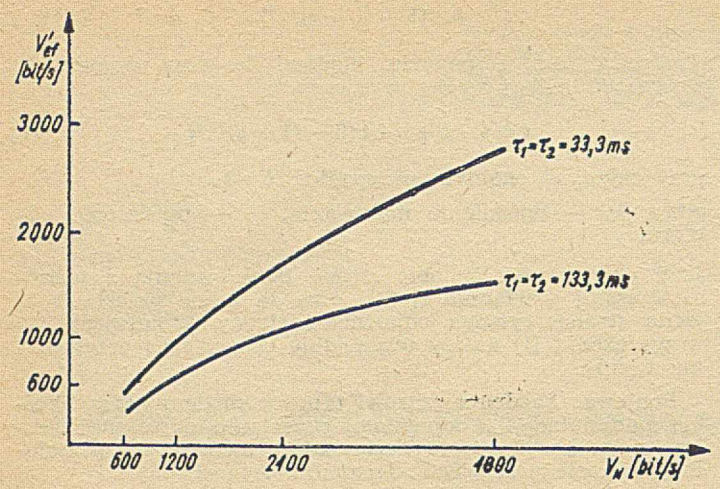
Porównując otrzymane zależności dla V'_{ef} i V_{ef} można poczynić następujące spostrzeżenia:

- dla dużej liczby przesyłanych bloków nieistotne stają się zmniejszenia szybkości pracy powodowane narzutem bitowym grup inicjującej i zakończenia, zatem rozmiar tych grup może być w pewnych granicach swobodnie powiększany, co jest szczególnie ważne dla grupy inicjującej — adresującej; wynika stąd, że cechy adresowe mogą być rozbudowane bez większego wpływu na szybkość efektywną
- szybkość efektywna transmisji zmniejsza się, gdy przesyłana jest mała liczba bloków.

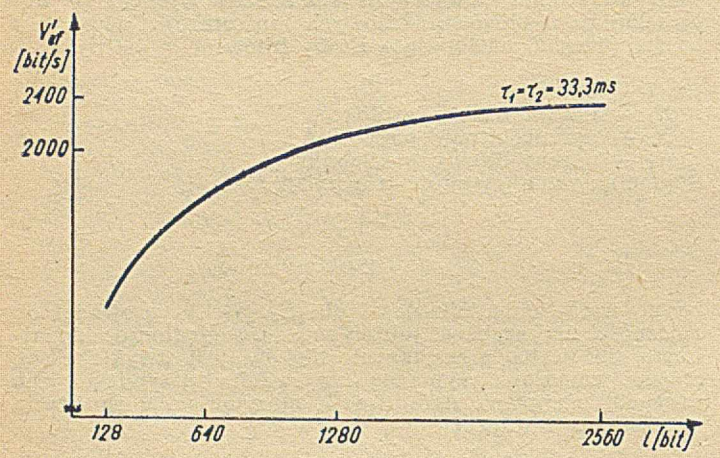


Rys. 2. Wykres zależności szybkości efektywnej od czasu przełączeń

[1]) W rzeczywistości należałoby uwzględnić powtórzenia wielokrotne, co prowadziłoby do liczby powtórzeń określonej iloczynem liczby bloków n i sumy ciągu geometrycznego $\sum_i b^i$ (i — krotność powtórzenia)



Rys. 3. Wykres efektywnej szybkości transmisji w zależności od szybkości nominalnej



Rys. 4. Wykres szybkości efektywnej w zależności od długości bloku

Na rysunkach 2, 3 i 4 przedstawiono określony na podstawie wyrażenia [2] przebieg szybkości efektywnej (jej przypadku granicznego), w zależności odpowiednio do czasów τ_1 i τ_2, szybkości nominalnej i długości bloku.

Celem skonfrontowania zależności [2] z rzeczywistym kształtowaniem się użytkowej szybkości transmisji, dokonano pomiarów tej szybkości. Pomiary wykonane były w następujących warunkach:
— łącze stałe długości ok. 230 km (Olsztyn — Warszawa)
— modemy RACAL MILGO 24 LSI
— szybkość nominalna 2400 bit/s
— system pracy: wielodostęp, komputer ICL 1903A — terminal ICL 7503
— system operacyjny GEORGE 3.
Zmierzono szybkość w kartach na minutę czytania z czytnika kart terminala, co przeliczono na szybkość wyrażoną w bitach na sekundę. Pomiarów dokonywano na różnych wartości opóźnienia RTS/RFS [2]). Wyniki zestawiono w tabeli 1.

Tabela 1. Szybkość czytania kart (w kartach na minutę) i odpowiadająca im szybkość przesyłania bit/s

τ_1 ms / τ_2 ms	8,3	33,3	67,6	133,3
33,3	164/ 1750	152/ 1620	140/ 1500	120/ 1280
133,3	120/ 1345	120/ 1280	112/ 1195	100/ 1065

[2]) RTS/RFS — ang. *request to send/ready for sending* — żądanie informacji (terminal)/gotowość przesyłania (modem)

Szybkość pracy czytnika nie ograniczała szybkości przesyłania, gdyż jest ona prawie dwukrotnie większa od największej wartości zmierzonej. W obliczeniach teoretycznych dla tej samej sytuacji przyjęto dla uproszczenia, że decydującym opóźnieniem jest opóźnienie RTS/RFS.

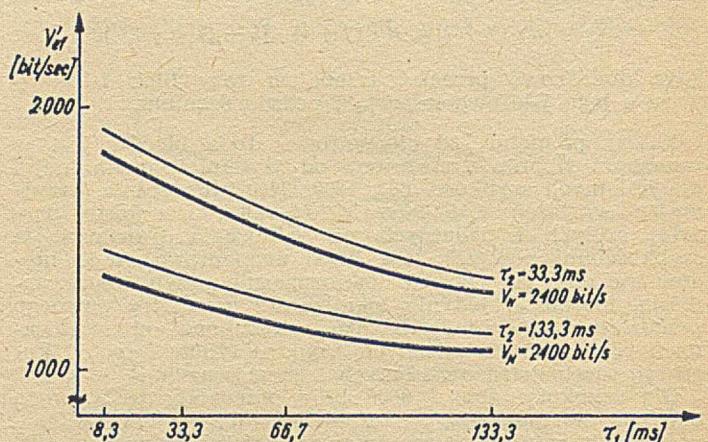
Tabela 2. Procedura transmisji obserwowanego systemu

Wysyłający	Komunikat	Grupa
komputer	SYN SYN EOT 'adri' ENQ	inicjująca
terminal	SYN SYN adr 1'status' ACK	inicjująca
komputer	SYN SYN EOT 'adr 2' ENQ	inicjująca
terminal	SYN SYN SOH/STX' DANE-BLOK' i IS2/DC4-ETB/ETX' bcc' i = 1,2 ... n	przesyłanie i-tego bloku danych
komputer	SYN SYN Status ACK	potwierdzenie odbioru bloku i-tego
terminal	SYN SYN EOT	zakończenie

gdzie:

— SYN, EOT, ENQ ... itd. są standardowymi znakami bitowymi kodu transmisyjnego ISO 7
— 'adr1', 'status', 'bcc' ... są własnymi 8-bitowymi znakami protokołu
— 'DANE — BLOK i'... oznacza 640-bitowy blok danych użytkownika.

Z analizy protokołu otrzymujemy następujące wartości parametrów wchodzących do wyrażenia [2]:
j = 80 bit
l = 640 bit
Przyjęto elementową stopę błędów równą 10^{-5}, a więc wyrażenie $(1-q)^{j+1}$ ma wartość 0,9928.



Rys. 5. Porównanie szybkości efektywnej teoretycznej i praktycznej

Na rysunku 5 przedstawiono przebieg obliczonej funkcji szybkości efektywnej V'ef (linia cienka) oraz wyniki pomiarów w zależności od czasu opóźnienia RTS/RFS. Różnica względna (ok. 5—6%) wynika z przyjętych założeń upraszczających obliczenia modelu teoretycznego i przyjęcia granicznej wersji tego modelu.

Z przedstawionych w artykule rozważań i pomiarów wynika, że teoretyczna zależność [2] odzwierciedla dostatecznie dokładnie rzeczywisty obraz pracy łączy i dlatego można ją przyjąć jako sposób określania efektywnej szybkości transmisji, uwzględniający decydujące czynniki wpływające na tę szybkość.

Należy także zauważyć, że dokonując pewnych modyfikacji, uwzględniających specyfikę innych trybów pracy łącza (np. usuwając składniki zawierające czasy przełączeń dla pełnego dupleksu), można za pomocą zależności [2] określać szybkość efektywną pracy łącz pracujących w trybach: simpleksowym i pełnodupleksowym.

Filmowe konfrontacje

We Wrocławskiej Akademii Ekonomicznej odbyła się dość nietypowa impreza — I Dni Filmu Informatycznego, zorganizowana (marzec br.) przez Instytut Informatyki oraz Koło Naukowe Informatyki. W czasie dwudniowego pokazu zaprezentowano łącznie 15 filmów. Sześć z nich wypożyczono z Ośrodka Postępu Technicznego (OPT) w Katowicach. Były to filmy zagraniczne, prezentowane na VIII Międzynarodowym Festiwalu Filmów Naukowych i Technicznych — Katowice '79 (festiwal ten nie został odpowiednio rozpropagowany, a prezentowanych tam filmów nie udostępniono szerszej publiczności). Pozostałe 9 filmów zrealizowały — na konkretne zamówienie — wytwórnie filmowe w kraju. Filmy te pozostają obecnie w dyspozycji swoich właścicieli; tylko jeden z nich można wypożyczyć za pośrednictwem Okręgowego Przedsiębiorstwa Rozpowszechniania Filmów (OPRF).

Wyboru filmów do prezentacji dokonano na podstawie danych uzyskanych z katalogu OPT, w którym zarejestrowanych jest obecnie 19 filmów krajowych, związanych z informatyką. Głównych kryterium wyboru był rok produkcji filmu oraz krótka informacja o jego treści. Podstawowe dane o prezentowanych filmach zawiera tabela.

Stosunkowo nieliczna pula krajowych filmów informatycznych jest w praktyce wykorzystywana w minimalnym zakresie. Filmy te stanowią głównie pomoc popularyzatorsko-dydaktyczną w działaniach podejmowanych przez ich dysponentów. Okazało się, że niektóre z wyświetlanych podczas imprezy filmów zyskały wielką aprobatę stosunkowo szerokiej widowni (ponad 400 pracowników dydatkycznych i studentów). Filmy te mogłyby się zatem stać istotną pomocą dydaktyczną, zwłaszcza w tych uczelniach i ośrodkach, które nie posiadają odpowiedniego sprzętu komputerowego.

Kilka słów zatem o każdym z filmów:

● **"Komputer w KENTEX"** — film prezentujący zastosowania komputera do wspomagania sterowania i zarządzania procesem produkcyjnym w przemyśle włókienniczym. Film reklamowo-popularyzatorski. Niestety czarno-biała wersja znacznie osłabia efekt.

● **"Cybernetyka"** — w filmie zaprezentowano model cybernetyczny zbudowany na podstawie technologicznego procesu produkcji obuwia. Film jest interesujący. Mankamentem jest jego czarno-biała wersja, a także brak animowanego uogólnienia zaprezentowanego modelu, które ułatwiłoby widzowi zrozumienie wprowadzonych pojęć, związków pomiędzy nimi oraz wyjaśniłoby zasady dynamicznego działania modelu.

● **"Komputery w szkole"** — prezentacja procesu dydaktycznego, w którym wykorzystano komputer. Film zrealizowano w szkołach podstawowych RFN i Szwajcarii. Pokazano w nim m.in. spontaniczne reakcje uczniów na niepowodzenia w pracy z komputerem.

● **"Jak powstaje komputer"** — film zrealizowany w Centrum Komputerowych Systemów Automatyki i Pomiarów prezentuje proces produkcji komputera R-32. To interesująca pozycja, która powinna stanowić obowiązkową pomoc dydaktyczną w procesie nauczania przedmiotów ogólno-informatycznych (wstęp do informatyki, propedeutyka informatyki, organizacja przetwarzania danych itp.).

● **"Zakład Szkoleniowy MERA-ELWRO"** — jest filmem popularyzatorsko-reklamowym, prezentującym zakres i metody szkolenia w ramach kursów prowadzonych przez producenta systemów komputerowych. Film prezentuje nowoczesne podejście do szkolenia, które warto naśladować także w innych formach kształcenia (szkolenia branżowe, kursy doskonalenia zawodowego itp.).

● **"MERA-ELWRO"** — prezentacja zakładu o tej nazwie, jego produkcji, warunków pracy, powiązań z innymi zakładami krajowymi i zagranicznymi. Film typowo popularyzatorski. Może być wyświetlany w szkołach ponadpodstawowych celem zainteresowania uczniów pracą w przemyśle komputerowym.

● **"Komputerem bardziej ekonomicznie"** — film, a właściwie cztery filmy, jako że wydzielono tutaj cztery części, poświęcone prezentacji następujących form wykorzystania komputera: „Informatyka pomaga kierować produkcją", „Minikomputery dla eksperta", „Informatyka w planowaniu", „Bank informacji w gospodarce materiałowej". W sumie jest to bardzo interesująca pozycja — do szerokiego wykorzystania zarówno w wyższych uczelniach (w ramach przedmiotów ogólnoinformatycznych), jak i na kursach kadry kierowniczej, doskonalenia zawodowego itp.

● **„Projektowanie systemu informatycznego"** — film dydaktyczny przeznaczony do wykorzystania na zajęciach z podstawowych przedmiotów informatycznych. Połączenie form filmu „żywego" i animacji ułatwia widzom zrozumienie metodyki prac projektowych.

● **"Eksploatacja systemu informatycznego"** — film dydaktyczny nadający się do wykorzystania we wszystkich formach podstawowego kształcenia informatycznego. Film pokazuje komputery R-32 i MERA 305, prezentując kolejność i zakres czynności związanych z eksploatacją dowolnego zadania realizowanego w technologii wsadowej. Szczególną uwagę poświęcono kontaktom użytkownika z ośrodkiem obliczeniowym.

● **"Potężny RENÉ"** — film popularyzatorsko-reklamowy poświęcony wykorzystaniu systemu komputerowego do sterowania i rozliczania przewozów kolejowych. Wizualnie bardzo efektowny.

● **"Zastosowanie komputera w elektronice"** — przedstawiono tu sposób wykorzystania komputera do wspomagania procesu projektowania obwodów drukowanych oraz nadzorowania technologii przygotowania płyt z obwodami drukowanymi.

● **"Dziecko nieskończoności"** — bardzo interesujący film, pokazujący pracę komputera w różnych badaniach symulacyjno-modelowych, związanych z poszukiwaniem odpowiedzi na pytanie: w jakich warunkach powstało życie na Ziemi i czy istnieje możliwość powtórzenie tego procesu.

● **"Komputer narzędzie uniwersalne"** — film prezentujący różne zastosowania komputera, począwszy od doradztwa agrotechnicznego, poprzez systemy nadzorujące ruch pociągów, systemy dydaktyczne, aż po medyczną diagnostykę.

Filmy prezentowane w ramach konfrontacji

Tytuł filmu	Dysponent	Taśma	Czas trwania	Kolor	Produkcja	Rok produkcji
"Komputer w KENTEX"	OPRF	16 mm	17 min.	czarno-biały	polska	1973
"Cybernetyka"	DOSKO-Łódź	16 mm	15 min.	czarno-biały	polska	1975
"Komputery w szkole"	Politechnika Poznańska	16 mm	20 min.	barwny	polska	1976
"Jak powstaje komputer"	Zakład Szkoleniowy MERA-ELWRO	16 mm	20 min.	barwny	polska	1976
"Zakład szkoleniowy MERA-ELWRO"	MERA-ELWRO	16 mm	20 min.	barwny	polska	1976
"MERA-ELWRO"	MERA-ELWRO	16 mm	20 min.	barwny	polska	1976
"Komputerem bardziej ekonomicznie"	PTE-Warszawa	16 mm	40 min.	barwny	polska	1978
"Projektowanie systemu informatycznego"	AE-Wrocław	16 mm	17 min.	barwny	polska	1979
"Eksploatacja systemu informatycznego"	AE-Wrocław	16 mm	17 min.	barwny	polska	1979
"Potężny RENE"	OPT-Katowice	32 mm	30 min.	barwny	francuska	1978
"Zastosowanie komputera w elektronice"	OPT-Katowice	32 mm	30 min.	barwny	węgierska	1978
"Dziecko nieskończoności"	OPT-Katowice	16 mm	30 min.	barwny	amerykańska	1978
"Komputer narzędzie uniwersalne"	OPT-Katowice	16 mm	30 min	barwny	amerykańska	1978
"Omega READ BACK"	OPT-Katowice	16 mm	11 min.	barwny	szwajcarska	1978
"Kryształy ciekłe"	OPT-Katowice	16 mm	20 min.	barwny	włoska	1978

● "Omega READ-BACK" — film przedstawiający system szczegółowej informacji o lotach pasażerskich oraz obsługę systemu informacyjnego lotniska za pomocą komputera oraz zestawu tablic informacyjnych.

● "Kryształy ciekłe" — film łączący technikę filmu „żywego" z animacją — wyjaśnia problemy wykorzystania kryształów ciekłych w różnych dziedzinach nauki i techniki, m. in. do budowy wyświetlaczy cyfrowych, mających duże znaczenie w elektronice i technice komputerowej.

Mimo skąpego materiału filmowego, jaki został zaprezentowany na wspomnianej imprezie, można pokusić się o wnioski ogólniejsze.

Mianowicie:
— z uwagi na coraz powszechniejszą obecność komputerów w różnych dziedzinach życia gospodarczego, filmowe prezentacje sprzętu komputerowego i obsługującego go personelu jest dla widzów mało atrakcyjne (z wyjątkiem niektórych nietypowych urządzeń zewnętrznych)
— dla celów dydaktycznych korzystne jest łączenie filmu „żywego" z animacją, która pozwala na przedstawienie procesów i zjawisk zachodzących w dłuższych okresach czasu lub niemożliwych do sfilmowania
— niezbędna jest szersza popularyzacja dostępnych aktualnie filmów dydaktycznych zarówno przez instytucje trudniące się wypożyczaniem filmów, jak i poprzez tworzenie w ramach określonych instytucji dydaktycznych (np. wyższych uczelni) własnych działów unowocześniania dydaktyki
— wydaje się celowym zakup wybranych filmów zagranicznych, które mogłyby stanowić pomoc dydaktyczną, a często służyć jako wzorzec ciekawego rozwiązania problemu prezentacji informatyki
— w ramach krajowej produkcji filmowej z dziedziny informatyki należałoby postulować podejmowanie tematów związanych z nowoczesnymi technologiami przetwarzania, budową i eksploatowaniem powtarzalnych systemów informatycznych, a ponadto z różnymi specjalistycznymi zastosowaniami komputerów.

Barbara ŁUKASIK-MAKOWSKA

Kopie filmów o informatyce

Uprzejmie informuję, że w notatce o filmie „Eksploatacja systemu informatycznego", która ukazała się na łamach INFORMATYKI w nr 1/80, wkradła się pewna nieścisłość. Akademia Ekonomiczna we Wrocławiu posiada bowiem dwa filmy dydaktyczne o informatyce, tj. wyżej wzmiankowany oraz drugi pt. „Projektowanie systemów informatycznych". Oba filmy zrealizowane zostały w 1979 roku przez wytwórnię POLTEL w Katowicach przy współpracy pracowników Instytutu Informatyki naszej Uczelni. Koszt sprzedaży kopii filmowych jest nieco wyższy niż to podano, bowiem do technicznego kosztu kopii (3 tys. zł) dochodzi, zgodnie z obowiązującymi przepisami, koszt rozpowszechniania równy 10% kosztu produkcji filmu — czyli około 39 tys. zł. Łącznie zatem koszt jednego filmu wynosi 42 tys. zł. Informuję ponadto, że obecnie kończone są prace przy opracowywaniu obcojęzycznych wersji tych filmów, tzn. rosyjskiej, niemieckiej i angielskiej.

Jednocześnie podaję do wiadomości Redakcji, że w przypadku indywidualnych zakupów filmów problemy odpłatności są uzgadniane indywidualnie.

B.Ł.M.

Metody układania harmonogramu zajęć

W czerwcu br., z inicjatywy Oddziału Gdańskiego Polskiego Towarzystwa Cybernetycznego, odbyło się w Politechnice Gdańskiej seminarium poświęcone komputerowemu układaniu rozkładów zajęć dydaktycznych. Seminarium to zgromadziło ponad trzydziestu specjalistów z siedmiu ośrodków akademickich w kraju.

Dr A. Gospodarowicz [1]) z Akademii Ekonomicznej we Wrocławiu przedstawił stan obecny i perspektywy komputerowego układania harmonogramów zajęć; mgr R. Kubiak z Uniwersytetu Gdańskiego podzielił się swymi doświadczeniami z wdrażania programu PLAN, służącego do układania rozkładów zajęć w szkołach i wyższych uczelniach; kpt. mgr K. Ficoń z Wyższej Szkoły Marynarki Wojennej przedstawił system programów do sporządzania rozkładów zajęć w wyższych szkołach wojskowych. Następnie zaprezentowano systemy eksploatowane; dr M. Kubale z Politechniki Gdańskiej przedstawił system ROZA [2]), który posłużył do ułożenia harmonogramów na Wydziale Elektroniki PG, zaś mgr Z. Rzemykowski z Akademii Ekonomicznej w Poznaniu zaprezentował system KSPZ, wdrożony w tej uczelni w 1977 roku.

Problem komputerowego sporządzania rozkładów zajęć znajduje się w sferze zainteresowań informatyków i dydaktyków od ponad 20 lat. W tym czasie w literaturze światowej opublikowano na ten temat ponad 200 artykułów. Niemniej, wygłoszone referaty, a także przebieg dyskusji, wykazały, iż nie należy oczekiwać rychłego zastąpienia człowieka-planisty przez maszynę cyfrową. Trudności związane z zaprogramowaniem skutecznych metod automatycznego planowania zajęć mają bowiem pewne uzasadnienie teoretyczne.

Od 1976 r. wiadomo, że układanie harmonogramu zajęć jest problemem tzw. „NP — zupełnym", co oznacza, że nie istnieje efektywny (wielomianowy) algorytm konstruowania rozkładu zajęć w przypadku ogólnym. Istniejące algorytmy niewielomianowe są natomiast tak złożone, iż otrzymywanie rozwiązania kompletnego (tj. takiego, w którym zaplanowano wszystkie zajęcia) wymagałoby dla dużej liczby zajęć najprawdopodobniej kilku miesięcy pracy komputera. A zatem pojawia się paradoksalna sytuacja: czas potrzebny na ułożenia rozkładu zajęć jest dłuższy od okresu na jaki planuje się ten rozkład (jeden semestr).

Jednocześnie od algorytmów konstruujących harmonogramy zajęć dydaktycznych oczekuje się kompletnych, optymalnych rozkładów, wyprodukowanych możliwie szybko. Jako kryteria optymalności podaje się tutaj rozliczne wymagania, często ze sobą sprzeczne, komplikując dodatkowo cały problem. Próbując wyjaśnić zagadnienie miar optymalności rozkładów zajęć J. Jirku przeanalizował w swej pracy doktorskiej 88 kryteriów,

[1]) Artykuł dr. A. Gospodarowicza zamieszczamy w pierwszej części numeru na str. 12

[2]) System ROZA przedstawiamy w osobnym artykule na str. 14

obowiązujących w wyższych szkołach wojskowych na terenie Czechosłowacji (Brno 1977).

Skoro nie jest możliwe sporządzenie rozkładów optymalnych, to musimy zrezygnować z kryterium optymalności na rzecz rozwiązań kompletnych lub prawie kompletnych. Jest to wskazane tym bardziej, że planowanie zajęć jest typowym przykładem problemów optymalizacyjnych dla których dane mają charakter przybliżony, zaś kryterium optymalności jest — do pewnego stopnia — wybrane dowolnie. Tą drogą poszli również autorzy systemów zaprezentowanych w trakcie gdańskiego seminarium. Niestety, wyniki uzyskane w oparciu o algorytmy heurystyczne okazały się gorsze od rozwiązań otrzymywanych metodami tradycyjnymi.

Dogodnym sposobem pokonania powyższych trudności jest odciążenie maszyny (częściowe lub całkowite) od podejmowania pewnych decyzji planistycznych i powierzenie ich człowiekowi. I tak, w zależności od stopnia zautomatyzowania kluczowego procesu (alokacji godzin i sal), wyróżniamy trzy typy systemów informatycznych do układania harmonogramów zajęć.

- **Systemy półautomatyczne,** w których rozkład dokonywany jest manualnie, poza komputerem. W systemach tych maszyna używana jest do weryfikowania i testowania rozkładu ułożonego przez człowieka, a następnie do drukowania go we wszystkich możliwych przekrojach (dla grup, nauczycieli oraz sal) łącznie z pewnymi zestawieniami statystycznymi.
- **Systemy konwersacyjne,** w których przydział odbywa się we współpracy człowieka i maszyny. W systemach tych komputer przygotowuje możliwe warianty zaplanowania zajęć, człowiek zaś dokonuje wyboru najlepszego z nich, sprawując ciągły nadzór nad pracą maszyny. Zaletą tego sposobu jest łatwość stopniowego korygowania otrzymywanego rozkładu.
- **Systemy wsadowe,** w których harmonogram układany jest całkowicie przez komputer. Po wydrukowaniu rozkład zajęć musi być przeanalizowany przez użytkownika, który winien go skorygować i jeśli występują zajęcia nie zaplanowane — doplanować je.

Ogromna większość istniejących systemów informatycznych to systemy konwersacyjne lub wsadowe. Tym niemniej istnieją szkoły, w których zdecydowano, iż układanie planów lekcji najlepiej pozostawić człowiekowi, wyręczając go w tych pozostałych czynnościach, które łatwo zautomatyzować. Warto wszak zauważyć, że nawet w wariancie maksymalnie zautomatyzowanym osoba odpowiedzialna za rozkład zajęć odgrywa istotną rolę.

W trakcie seminarium wyrażono opinię, że o przyszłości komputerowego planowania zajęć zadecydują systemy konwersacyjne. **(M.K.)**

ETOB

Postulaty 1980

Artykuł ten jest pisany dokładnie w połowie września, a więc w kalendarzowym końcu „Polskiego Lata 1980". Ukaże się natomiast tuż przed początkiem zimy, a więc w czasie, gdy temperatura i realia ocen nabiorą zapewne stosowanego dystansu do świeżych jeszcze teraz wydarzeń.

Sprawy informatyki w Polsce nie należą z pewnością do najistotniejszych w gospodarczym rachunku sumienia za minione dziesięciolecie. Dla naszego jednak środowiska zawodowego mają przecież z natury rzeczy znaczenie podstawowe, a znana powszechnie sytuacja wymaga rzetelnej analizy, aby sformułować konstruktywne wnioski na najbliższą i nieco dalszą przyszłość. Jednak nie wedle deklamacji z początku lat siedemdziesiątych o „kluczu do dobrobytu", bo nawet złośliwe kontrhasło o „wytrychu dla niektórych" nie całkiem się sprawdziło. A także nie wedle decyzji z końca minionego dziesięciolecia, sprowadzających Polskę z czołowej europejskiej pozycji w opracowaniach i produkcji komputerów do roli eksportera peryferii licencyjnych z zaniedbaniem potrzeb własnego kraju, bo całkiem nie informatyczną profesję na myśl to przywodzi.

Ostatnio najpopularniejszym bodaj w Polsce słowem stały się **postulaty**. Obawiać się nawet można, że wkrótce zdezawuuje się ono podobnie jak wiele innych haseł, choćby **pomyślność**, żeby nie sięgać dalszych czasów. Składają zatem postulaty także pracownicy przedsiębiorstw i ośrodków informatycznych.

Znam wiele tych postulatów, zwłaszcza z sieci ETOB, ale nie tylko. Na czoło wrażeń z tej lektury wysuwa się pewna jednostronność oraz — co tu ukrywać — niewysoki stopień troski o własną dziedzinę zawodową. Dużo jest postulatów o płacach, sporo o organizacji pracy, trochę personaliów, ale prawie nic w kwestii podstawowej, za jaką uważam stan i sposób traktowania informatyki w polskim życiu gospodarczym. A sytuacja jest tu wysoce niezdrowa.

Szukając konstruktywnych dróg wyjścia z obecnego stanu rzeczy, należy nie tylko wnikliwie analizować jego objawy, ale bodaj przede wszystkim analizować przyczyny i stopniowo je usuwać, aby nie krępowały naprawy.

W oparciu o doświadczenia sieci ETOB oraz o obserwację innych organizacji informatycznych, pragnę poddać rozwadze następujący obraz podstawowych przyczyn złej sytuacji informatyki w kraju.

Ostatnie dziesięciolecie można umownie podzielić — nie tylko zresztą w sprawach informatyki — na dwa mniej więcej równe okresy: rozwoju i hamowania. Okres pierwszy charakteryzował się wysokim stopniem entuzjazmu (nierzadko ponad zdrowy rozsądek), względną obfitością środków (w tym dewizowych) oraz stosunkowo łatwą dostępnością potrzebnego sprzętu. Za tymi możliwościami nie nadążył jednak stopień przygotowania kadry — zarówno po stronie użytkowników, jak po stronie ośrodków obliczeniowych, które — zwłaszcza w tym czasie — rosły jak przysłowiowe grzyby po deszczu.

Sytuacja taka spowodowała m.in. powstanie wielkiej liczby pakietów oprogramowania użytkowego, zwykle o niskim poziomie rozwiązań i pozornie od siebie odmiennych. Sprzyjała też rozwojowi łatwych karier kierowniczych, przy czym ogólna zaradność, tupet, a nierzadko i demagogia znaczyły w wielu przypadkach więcej niż rzetelna odpowiedzialność i umiejętności zawodowe. Zemściło się to w drugim okresie, kiedy wielu owych „oficerów informatyki" skutecznie hamowało jej rozwój. Oni nawet nie chcieli źle, ale po prostu nie rozumieli tego, czym zarządzają.

Trwałym natomiast i pozytywnym dorobkiem tego okresu było wyposażenie kraju w dość nowoczesny — jak na owe czasy — sprzęt oraz powstanie dość licznej, wartościowej kadry zawodowej, która okrzepła w następnym pięcioleciu. Mówiąc po prostu: bez tamtego entuzjazmu, a nawet pewnej rozrzutności, obecny stan informatyki byłby znacznie gorszy. I nawet takie wyjątki jak np. sieć ETOB, która najbujniej rozwinęła się w ostatnim pięcioleciu (unikając przez to wielu potknięć), nie naruszają tej reguły.

Uważam, że radykalne pogorszenie się sytuacji w drugim okresie wynikło przede wszystkim z błędnej koncepcji zarządzania informatyką. Na początku, przy braku kadry i względnym nadmiarze środków, uzasadniona była pewna centralizacja zarządzania rozwojem, choćby dla kontrolowania rozdziału tych środków. Co by można zaprzeczyć, że próbowano tam działać kompetentnie i nie w sposób biurokratyczny.

Potem jednak zdecydowano, że informatykę będą nadzorować osobiście: premier (w sferze zastosowań) oraz przewodniczący Komisji Planowania (w sferze zakupów i produkcji sprzętu). Skutki są znane. Komitet Informatyki zdziałał niewiele, nawet jeśli do działań zaliczyć oficjalne zatwierdzanie raczej znanych faktów, że należy rozwijać rządowe systemy informatyczne i że ZETO jest główną siecią usługową w Polsce. Polityka Komisji Planowania doprowadziła natomiast m.in. do ogromnych strat wynikających z niedokończenia programu produkcji maszyn ODRA i do odczuwanych dotkliwie skutków przesadnej orientacji proeksportowej. Wątpić należy, czy suma zysków z eksportu w liczącym się stopniu rekompensuje straty z tego tytułu.

Nadmierna centralizacja zarządzania informatyką jest także jednym z powodów wysokiego stopnia zbiurokratyzowania ocen wykorzystania potencjału informatycznego i działalności ośrodków obliczeniowych. Jest to — moim zdaniem — trzeci z najważniejszych hamulców rozwoju, zwłaszcza w sferze zastosowań informatyki.

Posłużę się przykładem. Nikomu odpowiedzialnemu nie przyszło by już chyba na myśl, aby wykorzystanie specjalistów, np. w biurze konstrukcyjnym, mierzyć li-tylko liczbą godzin ich przebywania w pracy. A z komputerami tak właśnie się robi i GUS oficjalnie (corocznie) to publikuje. Antybodziec do racjonalnego ich wykorzystywania jest tu oczywisty.

Podobnie rzecz się ma z tzw. wskaźnikiem wydajności w ośrodkach. Jest nim wynik podzielenia wartości sprzedaży usług przez liczbę zatrudnionych. Aby nie narazić się na obniżenie funduszu płac, wskaźnik ten musi z każdym rokiem być wyższy. Nadmiernie ograniczona dostępność produkowanych w kraju nowoczesnych urządzeń do rozproszonego przygotowania nośników danych skutecznie uniemożliwia rzeczowy rozwój usług. Aby więc wzrastała sprzedaż (z reguły tylko w imię wzrostu owej „wydajności") ośrodki najchętniej podejmują się wykonywania dodatkowych wersji tabulogramów, o wątpliwej często przydatności użytkowej, ale nie wymagających wprowadzania dodatkowych danych źródłowych. Rośnie zużycie deficytowego papieru, dezawuuje się informatykę u użytkowników. Przypomina mi to Antoniego Słonimskiego, który usłyszawszy przed laty o „Tygodniu Pisania Listów", zaproponował „Tydzień Spuszczania Wody". Z nagrodami oczywiście. Łatwo się domyśleć, że w tej sytuacji próby obniżania cen za usługi mają w wielu przypadkach posmak działalności samobójczej dla ośrodka.

W tej krótkiej i z natury rzeczy spłyconej próbce analizy nie dotykam wielu innych czynników hamujących rozwój zastosowań informatyki, choćby w tak ważnych jej aspektach, jak technologia przetwarzania, wielodostęp, produkcja oprogramowania itp. Chciałem przede wszystkim podzielić się etobowskimi i własnymi poglądami na praźródła obecnych niedomagań.

Niech więc apel o przemyślenie tych poglądów będzie moim **Postulatem 1980.**

Wincenty ŁADA

Systemy informatyczne w Kombinacie Górniczo-Hutniczym Miedzi

Na Kombinat Górniczo-Hutniczy Miedzi w Lubinie składa się 17 zakładów o silnym zróżnicowaniu z punktu widzenia zastosowań informatyki; obok Zakładów Górniczych i Hut Miedzi jest olbrzymi Zakład Transportu i Zakłady Naprawcze Maszyn, a wreszcie — Zakłady Badawcze i Projektowe Miedzi CUPRUM. W tak zróżnicowanej technologicznie strukturze Kombinatu występuje wiele różnych systemów zarządzania. Niewiele elementów tych wielorakich systemów zarządzania ma charakter powtarzalny, natomiast co krok napotyka się zindywidualizowane potrzeby i wymagania uzasadnione rzeczywistą odrębnością zakładów.

Rozwojem zastosowań informatyki w zakładach Kombinatu zajmuje się Główny Informatyk KGHM. Do jego zadań należy przede wszystkim określanie i koordynacja przedsięwzięć informatycznych.

W zakładach KGHM znajdują się cztery systemy komputerowe serii ODRA 1300, z których trzy wyposażone są w zestawy wymiennej pamięci dyskowej. Większość zakładów posiada sprzęt do przygotowania danych na nośnikach papierowych. Zapotrzebowanie na moc obliczeniową stwarza ponadto potrzebę korzystania z mocy komputerowych w ZETO — w Jeleniej Górze oraz Wrocławiu (stacja wsadowa ICL-7503 powiązana z systemem POLRAX-2). Na potrzeby systemu „Kontrola Ruchu Załogi" w ZG Polkowice zainstalowany jest minikomputer RC-3600. 70% kadry informatycznej zatrudnionej w KGHM zajmuje się obsługą techniczną i operatorską sprzętu. Pozostali opracowują, wdrażają i zapewniają eksploatację systemów informatycznych — od strony użytkowej.

Większość systemów informatycznych wdrożonych w zakładach KGHM powstało w ZETO Wrocław lub przy jego współpracy. Współpraca ta narodziła się wraz z wejściem na rynek serii ODRA 1300 i trwa po dziś dzień. Mimo tak długiego już okresu współpracy nie pojawiły się żadne problemy, które mogłyby rzucić cień na bardzo poprawne i owocne kontakty.

Systemy informatyczne eksploatowane w zakładach KGHM można sklasyfikować następująco:

- systemy ewidencyjne
- systemy obsługi produkcji i utrzymania ruchu
- system o działaniu bezpośrednim w ZG „Polkowice".

SYSTEMY EWIDENCYJNE

Do grupy dużych systemów informatycznych typu ewidencyjnego eksploatowanych we wszystkich zakładach KGHM z częstotliwością miesiąca należą:

- **gospodarka materiałowa w pełnym zakresie** — od funkcji ewidencyjno-rozliczeniowych, przez kontrolę stanu zapasów, prowadzenie norm zużycia, wykazywanie materiałów zbędnych, aż do rozliczania rozdzielni dołowych części zamiennych do maszyn samojezdnych; w 1979 zakłady KGHM przeszły w całości z kodu SWW na KTM. System został zrealizowany, wdrożony i jest konserwowany przez ZETO Wrocław;
- **płace** — dla wszystkich zatrudnionych w pełnym zakresie;
- **ewidencja osobowa** — system stanowi, obok spełniania swoich własnych funkcji w pełnym zakresie, bazę informacyjną dla innych systemów informatycznych, w których występują dane o pracowniku (płace, gospodarka materiałowa, gospodarka środkami trwałymi, kontrola ruchu załogi). Oprogramowanie systemu zostało przygotowane do pracy w trybie konwersacyjnym i tak będzie eksploatowane od momentu uruchomienia systemu operacyjnego GEORGE — 3 na komputerach Kombinatu. System opracowano we współpracy KGHM i ZETO Wrocław;
- **gospodarka środkami trwałymi** — w pełnym zakresie ewidencyjnym i rozliczeniowym;
- **rozliczanie kosztów oddziałowych** — na podstawie danych z systemów: gospodarka materiałowa, płace i gospodarka środkami trwałymi.

Wymienione systemy, przystosowane do wymagań poszczególnych zakładów, eksploatowane są z powodzeniem; z ich wdrożenia osiągnięto takie korzyści, jak poprawa jakości i kompletności ewidencji, zaufanie do danych oraz dysponowanie wielką ilością danych historycznych. Rezultaty te dają dobrą podstawę do rozpoczęcia prac następnego etapu w zakresie systemów ewidencyjnych — opracowanie oprogramowania umożliwiającego stworzenie banku danych, przede wszystkim na potrzeby zarządu Kombinatu, ale także na potrzeby zarządów poszczególnych zakładów.

SYSTEMY OBSŁUGI PRODUKCJI I UTRZYMANIA RUCHU

W tej grupie systemów znajdują się zastosowania wynikające z potrzeb poszczególnych zakładów i zindywidualizowane ze względu na proces produkcyjno-technologiczny. Eksploatowane są następujące systemy informatyczne:

- Geolog, obejmujący swym zakresem ewidencję i analizę złóż wraz z bilansowaniem zasobów i przygotowaniem danych do kierowania produkcją górniczą. System opracowano i wdrożono siłami ZETO Wrocław, założenia do systemu sformułowano w KGHM;
- Technolog, obejmujący swym zakresem ewidencję i rozliczanie produkcji Zakładów Wzbogacania Rudy wraz z prowadzeniem analiz bieżących i wyliczaniem wskaźników technologicznych. Analizy i wyliczanie wskaźników odbywa się także narastająco (dekadowo), co umożliwia śledzenie stopnia rytmiczności produkcji. System opracowano i wdrożono siłami ZETO Wrocław na bazie założeń sformułowanych w KGHM;
- Elektrorafinacja, obejmujący swym zakresem układanie harmonogramu ciągnięć elektrod i rozliczanie produkcji hutniczej;
- Separ, obejmujący swym zakresem ewidencję, rozliczanie, planowanie i analizę pracy górniczych maszyn samojezdnych wraz z rozliczaniem pracy operatorów maszyn i wskaźnikową oceną pracy maszyn. System opracowano i wdrożono siłami ZETO Wrocław, a założenia sformułowano w KGHM;
- Sepat, obejmujący swym zakresem analizę pracy i postojów przenośników taśmowych;
- Sokur, obejmujący swym zakresem funkcje utrzymania ruchu stacjonarnych ciągów technologicznych (planowanie przeglądów, remontów i wymian części we wszystkich urządzeniach ciągu produkcyjnego wraz z kontrolą wykonania tych prac).

Obok podstawowych, wyżej wymienionych, systemów informatycznych wspomagających produkcję i utrzymanie ruchu, a eksploatowanych z częstotliwością od trzech do dziesięciu razy na miesiąc, w eksploatacji znajdują się także, opracowane w ZETO Wrocław, mniejsze zastosowania z tej dziedziny. Należą do nich:

- BHP dla służb bezpieczeństwa i higieny pracy w zakładach górniczych i hutach
- SEWIG — rejestrujący i przede wszyskim kontrolujący wyposażenie w gaśnice przeciwpożarowe dołowych maszyn samojezdnych
- system kontrolujący stan zabezpieczenia w części zamienne dla maszyn dołowych w zakładach górniczych.

SYSTEM O DZIAŁANIU BEZPOŚREDNIM

System „Kontrola Ruchu Załogi" opracowany i wdrożony przez firmę RC na minikomputer RC-3600 powstał na bazie założeń sformułowanych przez KGHM i ZETO Wrocław. Działanie systemu polega na automatycznej rejestracji każdego wejścia lub wyjścia do lub z zakładu oraz każdego zjazdu lub wyjazdu pracowników dołowych, przygotowywanie raportów dla dozoru w ciągu 15 min od rozpoczęcia zmiany oraz na dostępie do zbiorów — w razie potrzeby — natychmiastowym.

System współpracuje z systemem ewidencji osobowej dla wyprowadzenia szczegółowych danych o pracownikach (w razie potrzeby). Zjazdy i wyjazdy są kontrolowane systemowo, tak że po zakończeniu zmiany listowane są informacje dla dozoru w celu zyskania pewności, że nikt nie został na dole. Efektem wdrożenia systemu miało być przede wszystkim podniesienie poczucia bezpieczeństwa brygad dołowych oraz zwiększenie dyscypliny pracy; efekt ten osiągnięto.

ZAMIERZENIA ROZWOJOWE

Trudności w pozyskaniu sprzętu, koniecznego dla rozbudowy systemów teleprzetwarzania, nie pozwoliły na rozwój i wdrożenie innych systemów niż eksploatowane cyklicznie lokalne systemy wsadowe (niektóre nawet z częstotliwością raz na dobę).

KGHM posiada program wejścia w zdalne przetwarzanie transakcji. Pierwotnie przewidywał on budowę systemów wielodostępnych typu gwiaździstego (jak POLRAX bądź WASC). Jednakże wspomniane trudności sprzętowe spowodowały opóźnienia w realizacji tego programu i konieczność jego modyfikacji. Obecnie ocenia się, że rozwiązaniem o dłuższej perspektywie rozwojowej, bardziej uzasadnionym ekonomicznie będzie budowa w KGHM sieci komputerowej, do której będą dołączone komputery będące w dyspozycji Kombinatu. Na podstawie podpisanego prozumienia ZETO Wrocław podejmie prace związane z tworzeniem sieci komputerowej dla KGHM — tak projektowe, jak i znaczną część oprogramowania. Planuje się realizację oprogramowania powiązań zbiorów danych z dotychczas eksploatowanych systemów. Utworzenie banku danych i jego obsługa będzie najpoważniejszym zadaniem z dziedziny budowy oprogramowania dla KGHM, wykonywanym przez ZETO Wrocław.

Znaczne już nasycenie Kombinatu systemami ogólnego przeznaczenia oraz systemami specjalizowanymi jest przyczyną zawężenia frontu prac. Do nowych systemów należy np. opracowany system SZLAK, obejmujący planowanie i kontrolę przewozów kolejowych w Kombinacie. Niemniej główny wysiłek skierowany został na doskonalenie, modyfikację i powiązanie systemów znajdujących się w eksploatacji.

Mgr inż. Tomasz TYMEŃSKI
Kombinat Górniczo-Hutniczy Miedzi
Mgr Waldemar ZAGRAJEK
ZETO Wrocław

SETR — system dla przedsiębiorstw transportu samochodowego

Jednym z czynników, decydujących o wynikach finansowo-gospodarczych przedsiębiorstw lub gospodarstw transportowych, jest sprawne dokonywanie rozliczeń, kontroli i analiz eksploatacji jednostek transportowych np. zużycia paliwa, wykorzystania pojazdów itp. W dużych przedsiębiorstwach samochodowych organizacja procesu przetwarzania danych staje się w tym zakresie coraz bardziej zawodna, zwłaszcza, gdy wykorzystuje tradycyjne, ręczne metody. Jej usprawnieniem może być system informatyczny ewidencji i rozliczania pracy taboru samochodowego SETR, opracowany przez Zakład Elektronicznej Techniki Obliczeniowej w Białymstoku.

W systemie tym automatyzacją objęto następujące prace:

- sprawozdawczość z wykonania pracy taboru samochodowego (osobowego i ciężarowego) w miesiącu, kwartale i roku.
- rozliczenie zużycia paliwa w układzie pojazdów i kierowców
- rozliczenie kosztów transportowych związanych z eksploatacją pojazdów, wg listy użytkowników
- obliczanie wskaźników techniczno-ekonomicznych.

Ponadto system przygotowuje wydawnictwa, z których informacje służą analizie, planowaniu i sprawozdawczości. Dotyczą one wykorzystania pojazdów, przyczyn i wielkości postojów oraz przewożonych ładunków. System może być w tym zakresie z powodzeniem stosowany przez dowolne przedsiębiorstwa transportowe oraz gospodarstwa samochodowe, nie posiada bowiem zasadniczych ograniczeń.

Podstawowymi dokumentami źródłowymi są powszechnie obowiązujące karty drogowe: SM-101 i SM-102, wypełniane wg instrukcji obowiązujących w tradycyjnym systemie przetwarzania. Na potrzeby systemu uzupełniane są one tylko dodatkowymi informacjami: symbolem pojazdu, kodem kierowcy, nr zmiany, symbolem konta kosztów itp. Wymagana symbolika nie stanowi żadnych ograniczeń w zastosowaniu systemu, bowiem założono dużą swobodę w jej ustaleniu, w zależności od indywidualnych potrzeb użytkownika.

SETR opracowany został w dwóch wersjach oprogramowania: dla maszyn serii ODRA 1300 oraz JS RIAD pod nadzorem systemu DOS, w obu przypadkach w zestawach standardowych. Nośnikiem informacji są karty dziurkowane. Organizacja technologiczna systemu akceptuje również — bez konieczności dokonywania zmian — wszystkie zbiory na taśmie magnetycznej, tworzone w systemach MERA 9150.

SETR oparty jest przede wszystkim na oprogramowaniu własnym, częściowo sparametryzowanym. Składa się z następujących jednostek przetwarzania:

- zakładanie i aktualizacja kartoteki pojazdów (wraz z cennikiem na paliwo i normami jego zużycia) i kartoteki kierowców (imiona i nazwiska)
- kontrola danych z kart drogowych oraz miesięcznych kart przyczep
- analiza wykorzystania i pracy taboru samochodowego w miesiącu (analiza postojów, wskaźniki techniczno-ekonomiczne)
- rozliczanie kosztów transportu (informacje o liczbie przebytych kilometrów i godzin pracy oraz dane dla faktur za usługi transportowe)
- rozliczanie paliw (wyliczenie ilościowe i złotówkowe dla pojazdów i kierowców)
- sprawozdawczość kwartalna i roczna.

System może być realizowany w dowolnych cyklach obliczeniowych w zależności od indywidualnych potrzeb użytkownika. Budowa modułowa systemu umożliwia zaś dalszą jego rozbudowę, bez naruszenia dotychczasowej struktury, a także łączenie go z systemami już wdrożonymi w przedsiębiorstwie (finansowo-księgowy, gospodarka materiałowa itp.).

Wąska baza indeksowa, niewielkie kartoteki podstawowe (pojazdów i kierowców) oraz przyjęcie jako źródłowych dokumentów powszechnie obowiązujących kart drogowych — to elementy, które powodują, że wdrożenie systemu SETR nie wiąże się z generalną reorganizacją systemu tradycyjnego, a wdrożenie nie jest pracochłonne z punktu widzenia użytkownika; jest też stosunkowo krótkie (3 miesiące).

Jednak zarówno w okresie próbnego wdrożenia, jak i bieżącej eksploatacji systemu najwięcej trudności pojawia się na etapie przygotowania — zgodnie z wymogami systemu — kart drogowych. Wymaga się tutaj dużej dyscypliny ze strony służb przedsiębiorstwa, które odpowiedzialne są za wypełnianie kart, za ich kontrolę i weryfikację, jak również terminowy przepływ informacji. System dokładnie kontroluje wszystkie dane wprowadzane z kart drogowych i nie dopuszcza do przetwarzania błędnych.

Wyniki obliczeń — emitowane w systemie SETR — w dużej części zastępują pracę tradycyjnych urządzeń ewidencyjnych i sporządzane zestawienia rozliczeniowe, statystyczne i analityczne, a więc w znacznym stopniu odciążają służby przedsiębiorstwa. W systemie tradycyjnym, te same dane źródłowe występują wielokrotnie w urządzeniach ewidencyjnych, rozproszonych organizacyjnie i czasowo. System informatyczny natomiast zapewnia zaspokojenie potrzeb informacyjnych w oparciu o jedną bazę źródłową, według żądanego stopnia selektywności i agregacji.

System wprowadza zatem zmiany jakościowe w pracy służb przedsiębiorstwa; pracochłonne i mechaniczne czynności ewidencyjno-rozliczeniowe zastępowane są pracami zautomatyzowanymi, podnoszącymi jakość danych źródłowych i wynikowych oraz dających możliwość głębszej analizy wyników, związanych ze zdarzeniami gospodarczymi. W dużych przedsiębiorstwach transportowych, posiadających więcej niż jedną bazę, system SETR zmniejsza — jak się ocenia — pracochłonność i rozliczenia pracy taboru samochodowego do 50%. Osiąga się to m.in. poprzez scentralizowanie:

- spływu dokumentów, ich weryfikacji i korygowania
- opracowywania wszelkich sprawozdań, planów i obliczania wskaźników techniczno-ekonomicznych
- rozliczania paliw wraz ze sporządzeniem list premii za oszczędność i list potrąceń za „przepał".

Doświadczenia z dotychczasowej eksploatacji systemu wskazują ponadto na większą efektywność jego zastosowania w przypadku tworzenia maszynowych nośników informacji przez użytkownika. Uzasadnia się to: ograniczeniem liczby błędów przy przenoszeniu informacji z dokumentów na nośniki oraz zwiększeniem terminowości (uniezależnieniem się od kolejki użytkowników w jednostkach sieci ZETO).

Dotychczas system SETR upowszechniono w dziesięciu jednostkach organizacyjnych w kraju, należących do różnych branż gospodarki narodowej, a wśród nich: w Zakładzie Energetycznym, Zakładzie Transportu Maszyn Drogowych, ZWS oraz CHEMITEX-ELANA. System upowszechniany jest na ogólnie obowiązujących zasadach w sieci ZETO (cena systemu: 166 425 zł). Razem z systemem przekazuje się bibliotekę programów i dokumentację eksploatacyjną. Wdrożenia systemu dokonywane są na odrębne zlecenia.

Mgr Danuta MILEWSKA
Barbara WASIL
ZETO Białystok

Jak nas poinformowano, w dniu 21 października 1980 r. odbyło się posiedzenie Zespołu Roboczego powołanego przez Sekretariat Komitetu Informatyki dla opracowania projektu nowych zasad wynagradzania pracowników zatrudnionych w jednostkach organizacyjnych informatyki. W ww. posiedzeniu udział wzięli, włączając się do dyskusji merytorycznej, przedstawiciele:

- Niezależnego Samorządnego Związku Zawodowego Pracowników Informatyki
- Niezależnego Samorządnego Związku Zawodowego Pracowników Energetyki (Sekcja Informatyki)
- Niezależnego Samorządnego Związku Zawodowego Pracowników Państwowych i Społecznych
- Niezależnego Samorządnego Związku Zawodowego „Solidarność" (Sekcja Informatyki).

Redakcja

Międzynarodowa Konferencja nt. zastosowań ETO w budownictwie

W dniach 23—25 lutego 1982 r. odbędzie się w Berlinie (NRD) Międzynarodowa Konferencja „Zastosowania ETO w Budownictwie, Architekturze i Planowaniu Przestrzennym" IBA-DAT'82, organizowana przez Instytut Projektowania i Standaryzacji Akademii Budownictwa NRD. Przy organizacji konferencji współpracują jednostki wiodące w zastosowaniach komputerów w projektowaniu budowlanym w krajach Socjalistycznych. Ze strony Polski z organizatorami współdziałała COBPBP BISTYP.

Konferencja jest pomyślana jako początek serii konferencji o tej tematyce, organizowanych co 2—3 lata w krajach socjalistycznych.

Konferencja obejmie przede wszystkim zastosowania informatyki w następujących dziedzinach:

- planowanie przestrzenne osiedli i kompleksów przemysłowych w fazie koncepcji: modele i metody analityczne; uwzględnienie zasobów naturalnych i istniejącej zabudowy; wykorzystywanie istniejących zbiorów danych; kryteria i modele decyzyjne; określanie zadań inwestycyjnych;
- prace projektowo budowlane, instalacyjne i in.; wybór miejsca budowy; zagospodarowanie placu budowy i infrastruktura: sieci, drogi, roboty ziemne i makroniwelacja; metody i systemy budownictwa; konstrukcje i wyposażenie; użytkowanie katalogów projekty architektoniczne; tworzenie i porównywanie wariantów; kryteria wyboru, decyzje, prezentacja wyników w procesie przetwarzania;
- projektowanie konstrukcyjne, ekonomiczne i technologiczne; obliczenia i wymiarowanie konstrukcji; wyposażenie budynków i instalacje; budowle inżynierskie; optymalizacja rozwiązań; harmonogramy; związki interdyscyplinarne; dokumentacja;
- organizacja wykorzystania ETO; sprzęt do różnych zadań: technologia ETO: grafika komputerowa; tryb dialogowy; praca interaktywna; zdalne przetwarzanie danych; ergonomia.

Celem Konferencji jest prezentacja osiągnięć (przede wszystkim praktycznych) w dziedzinie zastosowań ETO w krajach RWPG oraz wybranych osiągnięć innych krajów, a także wymiana doświadczeń, która być może stworzy merytoryczne postawy późniejszej współpracy. Pożądane jest więc możliwie szerokie przedstawienie polskich prac.

Językami oficjalnymi Konferencji będą: angielski, niemiecki i rosyjski. Termin wstępnego zgłaszania uczestnictwa i nadsyłania skrótów referatów lub komunikatów: 30 październik 1980 r. Skrót referatu o objętości do 1 strony maszynopisu winien być nadesłany w jednym z języków Konferencji. Autorzy zostaną zawiadomieni (do 15 lutego 1981 r) o przyjęciu bądź odrzuceniu referatów przez Komisję Programową. Pełny tekst referatu lub komunikatu (o objętości: referat do 10 stron A4 wraz z rysunkami, komunikat do 5 stron) należy nadesłać do 30 czerwca 1981 r. W tymże terminie należy zgłaszać ew. uczestnictwo bez referatu. Opłaty konferencyjne wyniosą ok. 150 DM.

Zgłoszenia (imię i nazwisko zgłaszającego się, tytuł i stopień służbowy, miejsce pracy z adresem i telefonem, język — w którym ma być wygłoszony referat), skróty referatów i korespondencję należy kierować na adres polskiego współorganizatora konferencji: **COBPBP BISTYP ul. Parkingowa 1, 00-518 Warszawa, Zakład BO** z adnotacją **IBA-DAT'82.**

Automatyczny skład wzorów matematycznych

Skład drukarski wzorów matematycznych był jednym z ostatnich, trudnych do pokonania, bastionów ręcznej technologii przemysłu poligraficznego.

W poprzednich latach podejmowano szereg prób zautomatyzowania tej czynności z zastosowaniem wspomaganego komputerem fotoskładu, przy czym uzyskane wyniki nie były zadowalające. Różnorodność występujących w praktyce wzorów, znaków graficznych oraz reguł zapisu potwierdziła pogląd wielu specjalistów, że ten zakres prac zecerskich w ogóle nie jest możliwy do zautomatyzowania. Drukarnia H. Heenemann KG w Berlinie Zachodnim poszukiwała systemu fotoskładowego dostosowanego do potrzeb zautomatyzowanego druku książek, czasopism i akcydensów. Żądanie programu składania wzorów matematycznych, eliminującego w dodatku kosztowne urządzenie przygotowania danych, znacznie ograniczyło krąg systemów fotoskładu.

Wybór, dokonany po wnikliwej analizie dostępnych na rynku systemów, padł na fotoskład DIGISET 400 T2, chociaż w momencie podejmowania decyzji dla urządzania tego nie było jeszcze odpowiedniego programu komputerowego. Okazało się jednak, że standardowa lista rozkazów systemu automatycznego składu firmy SIEMENS DOSY, zorientowanego na urządzenie DIGISET, umożliwia składanie również wzorów matematycznych. Wielkość znaku pierwiastka, całki lub nawiasów wymagały jednak uprzedniego określenia przez zecera, podobnie jak parametrów przesunięć pionowych i poziomych niezbędnych do ustawienia na płaszczyźnie poszczególnych znaków wzoru. Znaków pierwiastka jednakże brakowało i należało je ręcznie wykreślać. Metoda ta wiązała się z bardzo pracochłonnym przygotowaniem danych wejściowych, znacznym nakładem czasu na testowanie, podatnością na błędy oraz związanymi z tym dużymi nakładami na prace korektorskie.

Dla rozwiązania wspomnianych zadań opracowano dodatkowy moduł systemu DOSY, umożliwiający zapis na taśmie dziurkowanej lub ekranie monitora wzorów wstawianych w tekście. Dzięki nowemu elementowi standardowego oprogramowania można składać w sposób całkowicie zautomatyzowany nawet bardzo złożone wzory matematyczne.

$$F(x) = \frac{\sqrt{1+g(x,0)-\frac{1}{2\pi}\cdot\sqrt{\frac{g(x,0)}{h(x,0)}}}}{1+\left\{g(x,0)-\frac{1}{2\pi}\lim_{a\to\infty}\int_{z=-a}^{a}\left[\frac{\alpha_0}{\sqrt[3]{1+g^2(x,z)}}-\frac{\beta_0}{\sqrt[3]{1+h^2(x,z)}}\right]dz\right\}}$$

Przylad złożonego automatycznie wzoru matematycznego

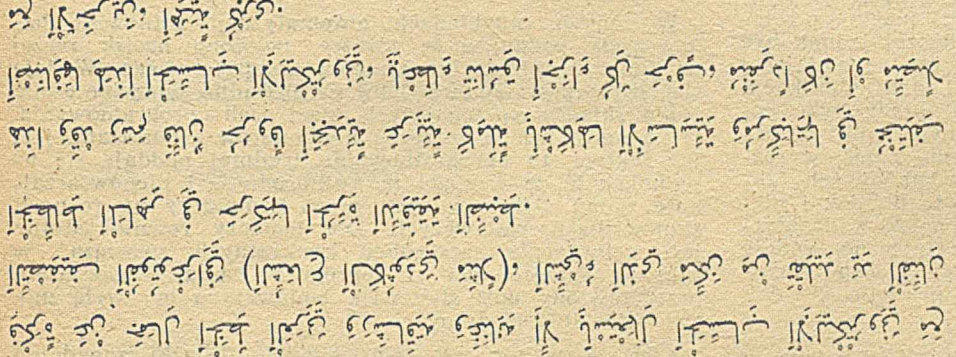
Na podstawie czasopisma Data Report 1/79 opracował WK

Pismo arabskie w automatycznym składzie

Jedna z najnowocześniejszych drukarni RFN — firma Ernst Klett w Stuttgarcie — podjęła próbę zaspokojenia rosnących potrzeb rynku w zakresie zautomatyzowania składu publikacji w języku arabskim. We współpracy z podległym sobie przedsiębiorstwem INTERPART, specjalizującym się w eksporcie technologii wydawniczej i poligraficznej do krajów Trzeciego Świata, firma ta opracowała dla stosowanego już fotoskładu DIGISET 400 T2 program komputerowy umożliwiający automatyczny skład jednego z najtrudniejszych pism (ze względu na olbrzymią różnorodność znaków i reguł zapisu), jakim jest niewątpliwie pismo arabskie. Program ten nazwano kryptonimem KITAB (Klett /Interpart/. Typographie für Arabisch/ Basis — DV — Programm), co w języku arabskim oznacza także książkę. Wspomniany program dostosowany jest do składu wszystkich rodzajów publikacji, a więc tekstów literatury religijnej, pięknej, technicznej, dziecięcej oraz podręczników i katalogów.

W piśmie arabskim podstawę słów stanowią spółgłoski. Kształt tych spółgłosek zależy od tego, czy znajdują się one na początku, w środku lub na końcu słowa. Kształt ten zależy również od tego, jaki znak poprzedza daną spółgłoskę oraz jaki za nią następuje. Wynikają z tego różne możliwości dołączania lub przenoszenia słów. Samogłoski podawane są poprzez akcenty lub inne znaki diakrytyczne, ustawiane częściowo ponad lub pomiędzy spółgłoskami, w zależności od tego, ile przestrzeni zajmuje pojedyncza spółgłoska lub ich zestaw tworzący tzw. ligatury. Dzielenie głosek, jakie występują w językach europejskich, język arabski nie dopuszcza.

W tej sytuacji szczególnego znaczenia nabiera wyrównywanie kolejnych wierszy. Kaligrafowie opierają się tu na swym wyszkolonym wzroku oraz poczuciu harmonii. Do wyrównywania wierszy stosują oni — znów wg ści słych reguł — specjalne formy szerokości liter oraz tzw. kaszydy, elementy przedłużające, za pomocą których w zależności od sytuacji przestrzennej wiersze mogą być „rozciągane". Każdy wiersz staje się więc jakby indywidualnym dziełem piszącego.



Fragment automatycznie złożonego tekstu w języku arabskim, ilustrujący złożoność tego problemu

Liczba znaków pisma arabskiego jest odpowiednio duża. Chociaż znaków podstawowych jest niewiele więcej niż w alfabecie łacińskim, a mianowicie 29, to występują tu setki możliwości (oraz konieczności), aby te 29 znaków modyfikować i wzajemnie wiązać zgodnie z kanonami arabskiej kaligrafii i gramatyki.

Przy opracowaniu pisma opartego na programie KITAB współdziałali specjaliści z zakresu lingwistyki, typografii i informatyki, celem stworzenia produktu zarówno nowoczesnego w formie, jak i spełniającego wymagania elegancji stylu pisma Nashki z XII w., jaki występuje w niektórych najpiękniejszych z zachowanych rękopisów Koranu.

Podstawowy zestaw znaków pisarskich zapamiętanych w pamięci komputera fotoskładu DIGISET 400 T2 można dzięki opracowanemu programowi składającemu tak wariantować, że operator ma do dyspozycji ponad 600 różnych modyfikacji i kombinacji liter, akcentów i znaków diakrytycznych w wymaganym zestawie ponad 400 możliwości ligatury.

Tekst rękopisu do automatycznego złożenia jest przygotowywany na urządzeniach dziurkujących taśmę papierową lub na maszynach do pisania w taki sposób, aby mógł być zinterpretowany przez komputer fotoskładu. Wspomniany program komputerowy analizuje dane każdego wiersza, wykrywając m.in. różne zależności kontekstu, np. położenie poszczególnych liter w każdym słowie, a także kombinacje liter wymagających uwzględnienia ligatur. Komputer uwzględnia również reguły kaligraficzne obowiązujące poszczególne znaki diakrytyczne i ortograficzne.

Na podstawie czasopisma Data Report nr 1/79 opracował WK

Bank danych dla obsługi rynku mieszkaniowego

Największe europejskie przedsiębiorstwo sprzedaży i wynajmu mieszkań NEUE-HEIMAT w Hamburgu (RFN) przeniosło całość swego systemu przetwarzania danych na komputer serii SIEMENS 7700. W pierwszej fazie inwestycji (do połowy 1979 r.) zainstalowano komputer SIEMENS 7760 z rozbudowanymi urządzeniami peryferyjnymi wartości ok. 4,2 mln $. W następnej fazie inwestowania, którego koszt szacuje się na ok. 3 mln $, jest realizowana sieć zdalnego przetwarzania danych, złożona ze 140 terminali ekranowych, 50 terminali drukujących i 8 minikomputerów komunikacyjnych, które zostaną zainstalowane w oddziałach terenowych firmy na obszarze całego kraju. Poprzez tę sieć oddziały uzyskają bezpośredni dostęp do centralnego banku danych „Rynek mieszkaniowy", zawierającego aktualne informacje o wolnych mieszkaniach. Dostęp do informacji będzie się odbywał w trybie konwersacyjnym z uwzględnieniem kryteriów określanych przez zgłaszającego się klienta (położenie, wielkość, wyposażenie mieszkania itp.). Do realizacji tego zamierzenia zastosowano firmowe oprogramowanie producenta: system zarządzania bazą danych UDS oraz system operacyjny BS 2000.

Opracował W.K. na podstawie czasopisma DATA REPORT 1/79

Rozwój przemysłu komputerowego RWPG

Jak doniosła prasa codzienna, w toku kolejnej, XXXIV Sesji Rady Wzajemnej Pomocy Gospodarczej (17—19 czerwca br. w Pradze), przewodniczący delegacji krajów członkowskich RWPG podpisali porozumienie o wielostronnej międzynarodowej specjalizacji i kooperacji w opracowaniu i produkcji nowych środków techniki obliczeniowej. W porozumieniu tym podkreślono konieczność zwiększenia w zbliżającym się pięcioleciu zakresu wzajemnych dostaw mniej więcej dwukrotnie w porównaniu z latami 1976—1980. Należy sądzić, że najbliższe miesiące przyniosą informacje dotyczące konkretyzacji powyższych ustaleń, a zwłaszcza szczegółowego podziału zadań pomiędzy poszczególne kraje. **(W.K.)**

O zastosowaniu minikomputerów w badaniach eksperymentalnych

Badania eksperymentalne stały się dziedziną, w której nie sposób obejść się bez komputera. Dlatego z radością należy powitać ukazanie się pierwszej książki w języku polskim na ten temat. Z tego względu praca Ryszarda Szczepana Ożarowskiego i Stanisława Kornackiego „Minikomputery w pracach eksperymentalnych" [1]) warta jest omówienia, choć z pewnością nie wypełnia ona luki w tej dziedzinie; jest raczej kroplą w morzu potrzeb zajmujących się komputeryzacją eksperymentatorów.

Po krótkim wprowadzeniu (dwa pierwsze rozdziały) autorzy omawiają w sposób elementarny budowę i działanie jednostki centralnej, zasady połączenia komputera ze środowiskiem i wykorzystywane do tego układy oraz podstawy oprogramowania minikomputerów. W drugiej części — po zwięzłej charakterystyce prac eksperymentalnych i wprowadzeniu do analizy sygnałów — autorzy przedstawili szereg przykładów komputeryzacji badań doświadczalnych, głównie w mechanice (charakteryzując krótko także techniki symulacji) oraz przeglądowo w wielu innych dziedzinach nauki i techniki. Książkę zamyka rozdział podejmujący zagadnienie projektowania i eksploatacji minikomputerów w systemach pomiarowych.

Opis działania komputera oraz zasad współpracy z eksperymentem zawarty w pierwszej części książki, jest podany na poziomie elementarnym — ktoś mógłby powiedzieć, że uproszczenie jest przesadne. Inaczej jednak być nie mogło. Nie było bowiem dotąd pracy omawiającej wnikliwie zagadnienia komputeryzacji badań, publikacja o charakterze podstawowym musi więc precyzować pojęcia i ustalać fakty, stanowiąc tym samym punkt odniesienia dla następnych prac. Wprowadzenie pojęć, wyróżnienie głównych zagadnień i podanie podstawowych faktów było obowiązkiem autorów. Zrobili to dobrze; i niewiele można tu dopowiedzieć, aby nie popaść w drobiazgowość. Cenne, choć chyba jeszcze niedostateczne, wydaje się podkreślenie roli, jaką we współpracy z eksperymentem odgrywają standardowe systemy sprzęgające, jak CAMAC czy magistrala IEC.

Warto zwrócić uwagę, że ta część książki jest, z jednym wyjątkiem, bez zarzutu pod względem terminologicznym, co świadczy o wyjątkowej staranności jej autorów i redaktorów. Ów wyjątek to termin **assembler**. Należy z całym naciskiem podkreślić, że — wbrew temu, co podaje omawiana publikacja — nie oznacza on języka adresów symbolicznych (str. 105). Assembler, to program tłumaczący programy z języka symbolicznego na język maszyny. Pomyłka, dosyć częsta w naszej literaturze fachowej, wywodzi się prawdopodobnie stąd, że niektórzy producenci określają języki symboliczne swoich komputerów nazwą ASSEMBLER, lecz wówczas jest to tylko nazwa własna, a nie termin ogólny.

Charakterystykę prac eksperymentalnych autorzy rozpoczynają od omówienia zagadnień analizy sygnałów, tj. zarówno torów pomiarowych, jak i metod przetwarzania sygnałów. Jest to temat sam w sobie dostatecznie rozległy, jednak z punktu widzenia nie tylko informatyka, ale nawet eksperymentatora należało pójść dalej, niż to zrobili autorzy. Mam na myśli fakt, że badacz jest człowiekiem praktycznym i potrzebuje zazwyczaj gotowych narzędzi. Choć na ogół eksperymentatorzy dają sobie radę z teorią transformacji sygnałów czy z teorią filtracji, w większości przypadków jest to dla nich przykra konieczność. Badacze potrzebują raczej gotowych programów analizy. Znacznie cenniejsze zatem byłoby dla nich podanie np. podprogramów szybkiej transformacji Fouriera lub Walsha oraz programów filtracji cyfrowej przy ograniczeniu niezbędnej teorii do minimum lub wręcz zrezygnowaniu z niej na rzecz opisu zastosowań do analizy konkretnych sygnałów. Słowem brakuje tu realizacji programowej algorytmów. Inne uwagi dotyczące analizy sygnałów pomijam, gdyż wykraczałyby poza ramy informatyki.

Rozdział dotyczący komputeryzacji badań eksperymentalnych mógłby stać się centralnym punktem książki. Jeżeli tak nie jest, wynika to po części z faktu, że w obecnej fazie rozwoju technologii badania mechaniczne nie są w pełni reprezentatywne dla przedstawienia technik komputeryzacji laboratoriów, ponieważ nie narzucają na system komputerowy warunków krytycznych (ze względu na szybkość przetwarzania i pojemność pamięci), tak jak to się dzieje np. w fizyce czy chemii. Zamiast rozdziału pt „Eksperymentalne badanie maszyn" w tej postaci, wystarczyłby przykład, z którego wynikałoby także wiele problemów szczegółowych. Należy jednak podkreślić, że przykładowy system komputerowy omówiony w tym rozdziale jest dobrany bardzo trafnie, ponieważ obejmuje wszystkie cechy nowoczesnego systemu komputeryzacji badań (szczególnie, jeśli chodzi o architekturę). Reprezentatywny będzie jednak dopiero wówczas, gdy dojdzie do jego pełnej realizacji.

W innych przykładach automatyzacji stanowisk badawczych, podanych w książce, położono nacisk raczej na funkcje aplikacyjne systemu komputerowego. Wydaje się, że korzystniejsze byłoby tu dokładne omówienie nie tylko układów sprzęgających, ale i cech oprogramowania. O ile normalizacja sprzężeń komputerów z obiektami jest już faktem, to oprogramowanie trudniej poddaje się ujednoliceniu ze względu na różnorodność i niepowtarzalność zadań wykonywanych przez komputery na różnych stanowiskach badawczych. Dla badaczy równie istotne jest uchwycenie szczegółów konkretnych rozwiązań w zakresie sprzętu, jak i oprogramowania. Tymczasem przedstawiony opis stanowisk badawczych ma wyłącznie charakter informacyjny.

Kontrowersyjne jest włączenie do książki zagadnień techniki symulacji. Nie ulega wątpliwości, że symulacja jest zgoła odmienną techniką badawczą niż przetwarzanie danych w czasie rzeczywistym i, choć nie odmawiam autorom prawa do łącznego ujęcia obu zagadnień, nie sądzę, aby w tym przypadku było to potrzebne i celowe.

O ile pewna ogólność sformułowań podanych w poprzednich rozdziałach może razić eksperymentatorów i inżynierów przyzwyczajonych z natury do operowania konkretami, to w rozdziale ostatnim stanowi ona cenną zaletę książki. Zawarto tam kilka wartościowych uwag dotyczących projektowania i eksploatacji systemów minikomputerowych.

Projektowanie systemu powinno być — według autorów — prowadzone przez późniejszych jego użytkowników a dokładniej przez osoby, które będą eksploatować ten system. Dodajmy, że wynika z tego konieczność takiej normalizacji sprzętu i oprogramowania, aby badacze, nie będący na ogół specjalistami w dziedzinie informatyki, mogli bez trudności taki system zaprojektować. Jednakże, autorzy zwracają uwagę, że kompletowanie systemu i jego oprogramowanie we własnym zakresie jest zadaniem bardzo trudnym i ryzykownym, o czym — dodajmy — muszą pamiętać szczególnie eksperymentatorzy planujący tworzenie własnych systemów mikroprocesorowych.

Zdaniem autorów — sprawą o kapitalnym znaczeniu jest zapewnienie właściwej konserwacji sprzętu komputerowego. Nabiera to szczególnej wagi wobec istniejącego poziomu zawodności urządzeń krajowych. Uwagi te są bardzo ważne, ponieważ badacze nie zawsze zdają sobie sprawę ze wszystkich problemów czekających ich w związku z nadejściem ery techniki komputerowej.

[1]) Ryszard Szczepan Ożarowski, Stanisław Kornacki: Minikomputery w pracach eksperymentalnych. Seria „Informatyka", WNT, Warszawa, 1980. Wydanie I, nakład 4000 + 260, stron 244, cena 48 zł

Warto byłoby również podkreślić, że w koszt systemu należy wliczyć (oprócz ceny komputera) także czas oczekiwania na dostawę oraz fundusze pochłaniane przez prace nad oprogramowaniem. Podobnie, zakup prostszych jednostek sterujących tylko pozornie zmniejsza koszty, gdyż potem wydatkuje się kilkakrotnie więcej na oprogramowanie.

Reasumując, o ile elementarny poziom pierwszej części książki jest uzasadniony, to w części aplikacyjnej autorzy mogli postawić sobie ostrzejsze wymagania i pójść znacznie dalej. Jednakże uwagi krytyczne nie mają na celu zdezawuowanie książki; mają za zadanie wskazać kierunki, jakie powinny uwzględnić dalsze publikacje wydawniczę z tego zakresu.

Pod względem systematyczności i klarowności wykładu niewiele prac może się równać z książką Ożarowskiego i Kornackiego. Widać tu akademicką praktykę autorów, którzy — jako wykładowcy uczelni technicznej — są szczególnie predysponowani do pisania podręczników. Gdyby informatyka wkraczała do szkół średnich, książka stanowiłaby bardzo dobry podręcznik ze względu na swój podstawowy charakter (pod warunkiem dodania znacznej liczby ćwiczeń).

A może Wiedza Powszechna mogłaby spróbować popularyzacji tej problematyki, publikując książeczkę w swojej serii OMEGA. Rozszerzenie kręgu odbiorców i przekazanie im systematycznej wiedzy wcale nie musiałoby się odbyć kosztem obniżenia poziomu publikacji, o czym świadczą doświadczenia innych autorów (por. W. Sobczak „Uczenie się automatów" 1977; M. Hołyński „Sztuczna inteligencja" 1979).

Jak już powiedziano, książka ta w żadnym wypadku nie wypełnia luki (powiedziałbym nawet — próżni) w krajowej literaturze dotyczącej tak dynamicznie rozwijającej się dziedziny, jak komputeryzacja badań doświadczalnych. Dobrze by się stało, gdyby ta prawda dotarła także do świadomości wydawców, którzy nie raz już udowadniali, że starają się być na bieżąco z osiągnięciami nauki światowej. Należy jednak dodać, że nawet w literaturze światowej takich prac jest stosunkowo mało i nie wszystkie są znaczące.

Kończąc przedstawione uwagi stwierdzam, że książka jest warta, aby ją polecić jako lekturę podstawową wszystkim, którzy zamierzają tworzyć komputerowe systemy pomiarowe i komputeryzować laboratoria badawcze lub już to robią, a więc — zarówno początkującym, jak i zaawansowanym.

Janusz ZALEWSKI

O jednolitą terminologię

„Czas rzeczywisty"

Czas rzeczywisty, jako termin, istnieje i nie można temu zaprzeczyć. Problem w tym, czy istnieje czas rzeczywisty, tzn. czy można używać wymienionego określenia na oznaczenie sytuacji, w której na działanie systemu komputerowego narzucone są z zewnątrz istotne ograniczenia czasowe, np. związane z koniecznością wysyłania sygnałów lub z możliwością strat odbieranej informacji. Wielu specjalistów twierdzi, że nie można. Prawdopodobnie dlatego powstało kilka polskich odpowiedników angielskiego terminu real time.

Do najbardziej rozpowszechnionych należy zaliczyć następujące: **natychmiastowość** (ang. real time), **na bieżąco** (ang. in real time), **system uwarunkowany czasowo** (ang. real time system) oraz **przetwarzanie** lub **praca w trybie nadążnym** (ang. real time processing, operation). Choć każde z tych określeń jest zasadniczo poprawne, to nie na tyle, aby wyprzeć z użycia wszystkie pozostałe. Ostatnie określenie, mające na to najwięcej szans jako najbardziej uniwersalne, jest już używane w innym znaczeniu w automatyce. Nie można także zaręczyć, że nie pojawią się nowe określenia, które zdaniem ich autorów będą jeszcze lepiej ujmowały istotę rzeczy.

Znaczna dowolność, będąca konsekwencją tego stanu nie sprzyja precyzyjnemu wyrażaniu się, a co za tym idzie — zrozumiałości wypowiedzi, i nie świadczy najlepiej o dbałości informatyków o swój język.

Choć podstawowym zarzutem stawianym terminowi **czas rzeczywisty** jest prymitywizm metody jego utworzenia (tzw. kalkowanie), jestem przekonany, że w opisanej sytuacji nie warto odchodzić zbyt daleko od źródłosłowu angielskiego. Następujące odpowiedniki angielskich określeń: **w czasie rzeczywistym** (ang. in real time), **system czasu rzeczywistego** (ang. real time system), **przetwarzanie w czasie rzeczywistym** (ang. real time processing) i **programowanie systemów czasu rzeczywistego** (ang. real time programming) są powszechnie zrozumiałe i to kryterium powinno być w tym przypadku wystarczające, aby zaakceptować ich użycie.

Innych sposobów pracy systemu komputerowego dotyczą angielskie określenia on-line i off-line. Można wyróżnić dwie istotne cechy obu określeń w odniesieniu do przetwarzania danych. Pierwsza pochodzi od urządzenia będącego pierwotnym źródłem danych i zależna jest od tego, czy urządzenie to jest dołączone do komputera elektrycznie, czy nie. Druga, która niejako wzmacnia pierwszą, odnosi się do nośnika danych. Nośnik pośredni między pierwotnym źródłem danych, a komputerem może być wyeliminowany w tym sensie, że nie ma pośredniej rejestracji danych, choć może następować ich zapamiętywanie. W przypadku, gdy nośnik pośredni nie jest wyeliminowany, rejestracja występuje.

Mając na uwadze drugą z wymienionych cech, mówi się: **przetwarzanie bezpośrednie** (ang. on-line processing), **praca w trybie bezpośrednim** (ang. on-line operation) lub **system o działaniu bezpośrednim** (ang. on-line system).

Jeżeli chodzi o termin off-line, to jako polski odpowiednik rzadko podaje się **pośredni**, częściej zaś — **autonomiczny**. Termin ostatni nie jest zbyt szczęśliwy, ponieważ ma wiele innych znaczeń, obcy źródłosłów, daleki od określenia angielskiego, które ma on zastąpić, a ponadto (co jest także istotne) określa sposób pracy urządzenia zewnętrznego, a nie samego komputera. Mimo że występuje w wielu słownikach, nie jest używany.

W książce J. Bańkowskiego i K. Fiałkowskiego „Wprowadzenie do informatyki" (PWN, Warszawa, 1978) wzięto pod uwagę pierwszą z cech, jakie musi mieć ten termin, nazywając tryb off-line trybem **rozłącznym**. Wydaje się to bardzo logiczne, ponieważ po odłączeniu urządzenia od

komputera nie można przekazywać danych inaczej jak tylko pośrednio. Zatem wystarcza nazwać ten rodzaj pracy komputera **pracą w trybie rozłącznym** (ang. off-line operation).

Wynika stąd, że pewna niekonsekwencja terminologiczna w przyjęciu określeń: **bezpośredni** (ang. on-line) i **rozłączny** (ang. off-line) byłaby tylko powierzchowna i usadniona przyczynami tkwiącymi głębiej.

Niekiedy spotyka się odwrócenie rozumowania i nazwanie pracy on-line — **pracą w trybie podłączonym.** Pomijając ocenę poprawności tego określenia, należy stwierdzić, że istnienie jednego odpowiednika terminu on-line (mianowicie: **bezpośredni**) eliminuje potrzebę wprowadzenia innego.

Wracając do określeń związanych z czasem trzeba stwierdzić, że istnieje kilka innych terminów, które nie mają jednolitej interpretacji, a w konsekwencji nie zawsze są używane precyzyjnie. Przykładowo, polski termin **zegar** jest używany w technice cyfrowej co najmniej w dwóch znaczeniach.

Zegarem (ang. clock) nazywa się urządzenie odmierzające czas w systemie komputerowym przez generowanie regularnych impulsów. Jeżeli funkcje tego urządzenia są bardziej rozbudowane, np. istnieje rejestr nastawny lub licznik, który przy przepełnieniu wysyła sygnał przerwania, to w języku angielskim zegar określa się mianem timer. Wydaje się, że celowe byłoby odróżnienie tych pojęć także w języku polskim. W takim przypadku na określenie drugiego z nich bardzo dobrze nadawałby się wyraz **czasomierz.**

Angielskie określenie timing — **odmierzanie czasu** — ma kilka odpowiedników. Mając na uwadze, że odmierzanie czasu w systemie cyfrowym zawsze czemuś służy, tzn. w każdej wyróżnionej chwili istnieje wyraźne przeznaczenie i określony odbiornik sygnału (co najmniej jeden), można bardziej precyzyjnie wyrazić ten fakt w języku polskim. Spośród polskich określeń odpowiadających terminowi timing wyróżniłbym dwa główne: **taktowanie** i **synchronizacja.** Jednakże wahałbym się wykorzystywać drugi z nich, ponieważ musi on pozostać odpowiednikiem angielskiego słowa synchronization.

Za podstawę rozróżnienia można by przyjąć liczbę źródeł sygnału. Podczas taktowania (ang. timing) w zasadzie istnieje jedno nadrzędne źródło sygnału o dokładnie zdefiniowanym przeznaczeniu, które odmierza czas bez oczekiwania na odpowiedź, a nawet bez zamiaru uzyskania odpowiedzi. Natomiast o synchronizacji w ścisłym sensie można by mówić dopiero wtedy, gdy istnieje wymiana sygnałów między dwoma składnikami systemu cyfrowego (a więc istnieją co najmniej dwa źródła sygnałów). Niemniej w wielu przypadkach, szczególnie przy tłumaczeniu tekstów obcojęzycznych, **synchronizacja** (ang. synchronization) nie da się odróżnić jednoznacznie od **taktowania** (ang. timing) i nie ma potrzeby, aby dążyć do takiego rozróżniania za wszelką cenę.

Terminem, który pojawia się coraz częściej w pracach z dziedziny techniki komputerowej jest angielskie określenie time out, oznaczające **upływ czasu.** Wydaje się, że polskim odpowiednikiem mógłby być właśnie **upływ czasu,** ponieważ jego znaczenie w miarę dokładnie odpowiada treści pojęcia i — co bardzo ważne — określenie to jest bardzo wygodne przy użyciu różnych konstrukcji zdaniowych, np. **nastąpił upływ czasu** (ang. time out has occured), czy **błąd spowodowany upływem czasu** (ang. time out error).

Poruszając zagadnienia związków terminologicznych wyrazu czas nie sposób pominąć angielskiego terminu time sharing, używanego w wielu różnych znaczeniach. **Podział czasu,** bo taki powinien pozostać polski odpowiednik określenia angielskiego, oznacza (wg normy ISO 2381): „sposób pracy systemu komputerowego, umożliwiający przeplatanie w czasie dwóch lub więcej procesów[1]) na jednym procesorze". Natomiast „tryb pracy, w którym dwóm lub więcej procesom przydziela się kwanty czasu tego samego procesora", nazywa się w jęz. angielskim time slicing (brak odpowiednika polskiego), a nie — time sharing. Według wymienionej normy niepoprawne jest także używanie terminu podział czasu na oznaczenie konwersacyjnego (interakcyjnego) trybu pracy.

W świetle definicji, zawartych w wymienionej normie, niewłaściwe jest określenie podane w pracy J. Marońskiego i M. Rupińskiej „Computer Networks Terminology" (PWN, Warszawa, 1980), gdzie **podział czasu** oznacza „metodę pracy, w której zasoby komputera (ang. computer facility) są dzielone przez wielu użytkowników w różnych celach w (pozornie) tym samym czasie". Powyższe określenie ma natomiast wszystkie cechy terminu **wielodostęp (wielodostępność)** zdefiniowanego prawie identycznie w tej samej pracy i pozostaje w zgodzie z używanymi odpowiednikami terminu time sharing system w znaczeniu **system wielodostępny, system abonencki** (M. Bazewicz: „Sieci komputerowe", INFORMATYKA, nr 2/1977).

Janusz ZALEWSKI

[1]) Według tej samej normy proces jest to „bieg zdarzeń następujący zgodnie z zamierzonym celem lub skutkiem".

LISTY DO REDAKCJI

Pochwała minikomputera

Podkreślanie osiągnięć przemysłu krajowego jest rzeczą naturalną i słuszną. W latach siedemdziesiątych nastąpił u nas duży wzrost produkcji urządzeń komputerowych — ilościowy i asortymentowy, żeby wymienić tylko znane drukarki wierszowe z Błonia, pamięci taśmowe czy urządzenia do klawiaturowej rejestracji danych na taśmie magnetycznej. W przyszłej pięciolatce połowa produkcji ma być przeznaczona na potrzeby odbiorców zagranicznych. Jest to olbrzymi procent, dobrze świadczący o możliwościach technicznych i handlowych branży.

O korzyściach automatyzacji prac biurowych, a także automatyzacji procesów zarządzania oraz sterowania produkcją w zasadzie nie ma potrzeby przypominać, trzeba natomiast pamiętać o dostarczaniu odpowiednich środków do ich osiągania, co jest głównym zadaniem przemysłu. I tu należy powiedzieć jedno: dla masowej informatyzacji, a taka jest potrzebna, żeby podnieść poważnie wydajność pracy w zakresie prostych czynności biurowych oraz zarządzania na szczeblu przedsiębiorstw (zwłaszcza średnich i małych) ważniejsze jest dysponowanie odpowiednim MINIKOMPUTEREM, niż KOMPUTEREM DUŻYM.

Przyjmijmy umownie, że za duży komputer będziemy uważać konfigurację sprzętową jaką reprezentuje R-20 i wyżej, a za „odpowiedni minikomputer" urządzenie o orientacyjnych parametrach: nie mniej niż 32 KB pamięci operacyjnej, nie mniej niż 1 MB pamięci zewnętrznej, monitor ekranowy, drukarkę, język programowania wyższego rzędu — co najmniej BASIC. Na taki właśnie minikomputer czeka cały kraj od wielu lat.

Pięć lat temu pytaliśmy, kiedy otrzymamy krajowy minikomputer o podobnych parametrach, seryjnie produkowany (a więc łatwo dostępny) z dobrym serwisem. To samo pytanie jest jak najbardziej aktualne i dziś. „Odpowiedniego minikomputera" bowiem nie ma nadal. Duże komputery nie rozwiązują natomiast problemu powszechnej informatyzacji.

Nasz bardzo już rozwinięty przemysł komputerowy dotychczas nie może dostarczyć np. niezmiernie potrzebnego urządzenia klawiaturowego z ekranikiem (niechby nawet jednowierszowym) do wprowadzania danych na najprostszy nośnik magnetyczny — taśmę kasetową, ani drugiego prościutkiego urządzenia — wczytującego dane z takiej taśmy

na taśmę półcalową lub dysk, co pozwoliłoby na stopniową eliminację ważących kilkaset kilogramów, nieekonomicznych dziurkarek i sprawdzarek kart. Produkujemy jedynie na licencji urządzenie wielostanowiskowe, bardzo drogie i wymagające w dodatku (przynajmniej dotychczas) wkładu dewizowego. A tymczasem chodzi o masowość, a więc o sprzęt mały, tani, dostępny dziesiątkom tysięcy jednostek organizacyjnych. Tym bardziej, że nie mogąc liczyć na prędki rozwój transmisji danych jesteśmy skazani przez wiele jeszcze lat na przetwarzanie lokalne.

Ton niektórych wypowiedzi na temat produkcji sprzętu komputerowego przypomina jako żywo enuncjacje w stylu: „wyprodukowaliśmy tyle a tyle tysięcy ton i tyle a tyle setek czy tysięcy sztuk" — tak, jakby najważniejsza była tylko ilość. W roku 1980 takie podejście do oceny działalności gospodarczej jest już anachronizmem. Nie pora imponować tonami i ilością sztuk produkcji; podstawowym kryterium jest dziś efektywność zaspokajania potrzeb społecznych. A tymczasem odpowiedniego minikomputera i prostego urządzenia do wprowadzania danych na nośnik magnetyczny nie ma.

Przemysł zapewne stwierdzi, że otrzymał kilkaset zamówień na minikomputer MERA 400 i zaspokaja potrzeby w 60—70%; takie jest bowiem rzeczywiste zapotrzebowanie społeczne i jest ono zaspokajane w dość wysokim stopniu. Na stwierdzenie to można odpowiedzieć, że już przed kilku laty zainteresowanie krajowymi minikomputerami było olbrzymie. Idę o zakład, że w rzeczywistości sięgało tysięcy zestawów. Gdy jednak nabywcy lub potencjalni klienci stwierdzili, że mija rok, dwa, pięć, a tymczasem nie ma dysków, albo system operacyjny jeszcze niekompletny, albo brak języka programowania wyższego rzędu, albo serwis z epoki kamienia łupanego, albo termin dostawy 2—3 letni, albo wreszcie — zawodność sprzętu nie pozwala na pracę ciągłą — to zainteresowanie spadło rzeczywiście do kilkuset konfiguracji.

Trzeba wreszcie przypomnieć producentom sprzętu, że płomień nie szerzy się tak szybko wśród suchej trawy, jak komputeryzacja w przedsiębiorstwach — gdy dostępny sprzęt jest dobrze rozwiązany technicznie (lekki, prosty, pewny) i kompletny, gdy stwarza możliwość łatwej rozbudowy i jest stosunkowo tani.

Mgr Andrzej KAMIEŃSKI

Jeszcze o projektowaniu algorytmów

SZANOWNA REDAKCJO

W ślad za moim artykułem pt. „Usprawnienie projektowania algorytmów" zamieszczonym w numerze 8-9/79 ukazała się w numerze 1/80 w Trybunie Czytelnika na str. 29 wypowiedź dr. Bogdana Stefanowicza związana z jego treścią. Pragnę ustosunkować się do postawionych zarzutów:

● Ogólna formuła algorytmu podana w artykule jest następująca: „...algorytm jest to oparty o odpowiednie reguły ściśle określony formalnie sposób postępowania...", a nie jak przytacza dr Stefanowicz: ...algorytm jest oparty o odpowiednie reguły... W związku z powyższym rozważania na ten temat podane w wypowiedzi stają się bezprzedmiotowe.

● Algorytm można traktować jako pojęcie oraz jako metodę posiadającą swoją postać użytkową. O charakterze postaci użytkowej algorytmu decyduje odpowiedni opis stanowiący jego integralną część. Przy projektowaniu oraz użytkowaniu algorytmu ważna jest przede wszystkim jego postać użytkowa, która może posiadać różne wersje uzależnione od specyfiki rozwiązywanego problemu. Chcąc ująć razem zawartość postaci użytkowej oraz treści przedstawione w ogólnej formule algorytmu, określono jego postać ogólną. Moim zdaniem na ogólną postać algorytmu obliczeniowego lub decyzyjnego składają się elementy podane w rozdziale „Ogólna postać algorytmu". Przy określeniu tejże postaci brałem również pod uwagę doświadczenia praktyki projektowania algorytmów oraz programowania, które często wymuszają konieczność komentowania wybranych operacji algorytmu.

● Istota zastosowania tablic decyzyjnych do określenia algorytmów uzależniona jest od stopnia złożoności rozwiązywanych problemów. Z określonych i oczywistych powodów niekiedy prościej jest zastosować sieć czynności dla rozwiązania problemu posiadającego do trzech warunków i do trzech czynności. Problemy takie występują dość często w praktyce. Obserwując jednak literaturę zagraniczną — dotyczącą informatycznych metod rozwiązywania problemów — oraz informacje krajowe — o opracowanych systemach informatycznych — można stwierdzić, że coraz częściej napotyka się na konkretne zastosowania tablic decyzyjnych. W kraju świadczą o tym opracowania przede wszystkim wrocławskich ośrodków informatyki. Wydaje mi się, że niemałe osiągnięcia mają również w tej dziedzinie ośrodki pomorskie, warszawskie i śląskie. Konkretnym przykładem niech będą niektóre referaty II konferencji naukowej „Automatyzacja projektowania'80" (Białystok — 15-17.05.1980). Proponuję jednak zamienić kwestionowaną wypowiedź „...obecnie często spotykaną postacią użytkową algorytmu..." na „...obecnie coraz częściej spotykaną postacią użytkową algorytmu...".

● Dr Stefanowicz kwestionuje sens zdania „...jednym z głównych celów projektowania algorytmu jest utworzenie schematu blokowego..." — twierdząc, cytuję: „...nie jest to celem tego procesu lecz znalezienie rozwiązania określonego problemu...". O tym co jest celem algorytmu zaznaczyłem w pierwszym zdaniu artykułu oraz w uprzednio wspomnianej ogólnej formule algorytmu. W tym miejscu należy dodać, że algorytm powinien być optymalnym rozwiązaniem określonego problemu, co sugeruję, na przykład, przez wybór metody rozwiązania w opisie matematycznym problemu (punkt b) dla algorytmu obliczeniowego w rozdziale „Ogólna postać algorytmu". Nawiązując do kwestionowanego zdania przypominam, że mówię w nim o istnieniu kilku głównych celów projektowania algorytmu, a nie o jednym — sprowadzonym w wypowiedzi dr. Stefanowicza do utworzenia schematu blokowego. W rozdziale „Wytyczne projektowania schematów blokowych" jest akurat mowa o jednym ze wspomnianych celów, związanym z treścią i tytułem rozdziału, jakim jest utworzenie schematu blokowego. Znane jest stwierdzenie, że schemat blokowy mimo swojej określonej pracochłonności jest najprzejrzystszym i najbardziej poglądowym obrazem algorytmu rozwiązania badanego problemu. Dlatego w moim odczuciu należy ponownie przywrócić schematom blokowym jedną z głównych rang przy projektowaniu algorytmów. Wydaje się, że postawione zarzuty przez dr. Stefanowicza wynikają z niedokładnego przeczytania artykułu, stąd — nieporozumienie. Trudno jest akceptować uwagi krytyczne związane bardziej z formą i stylem zdań, niż z zawartością merytoryczną artykułu. Uwagi te oparto o wyrwane z tekstu fragmenty zdań, cytat niezgodny z oryginałem oraz o subiektywną interpretację treści. W związku z tym pierwotnie nie zamierzałem odpowiadać na powyższe uwagi. Czynię to teraz dlatego, że nie można przecież pozostawić bez echa takiej wypowiedzi.

Mgr inż. Wiesław BĄBA

Kilka uwag o Informatyce

„Łamy INFORMATYKI otwarte dla wszystkich"! Chciałabym odpowiedzieć na to hasło i podzielić się swoimi refleksjami na temat samego pisma.

Dział „Nasze recenzje". Wielokrotnie zastanawiałam się, jaki może być cel prowadzenia tego działu w jego obecnej postaci i sądzę, że jego forma nie odpowiada potrzebom. Jaki jest bowiem sens recenzji opublikowanej co najmniej rok po ukazaniu się książki w sprzedaży? Nie wpłynie ona przecież na decyzję Czytelnika o zakupie omawianego tytułu, ponieważ najczęściej książka znika z półek w ciągu kilku tygodni. Wytykanie braków edytorskich czy błędów autora na łamach miesięcznika nie ma zaś sensu, gdyż interesuje niewielkie grono osób. Uwagi te powinny być kierowane do wydawcy, autora lub tłumacza.

Ze swej strony proponuję sygnalizować książki dopiero przygotowywane do druku, przedstawiając zwięźle ich zawartość (np. tytuły rozdziałów czy inne fragmenty spisu treści) wraz z komentarzem. Sądzę, że można by było to tak zorganizować, aby wiadomość w INFORMATYCE wyprzedzała spodziewane pojawienie się książki na rynku. Tylko wtedy Czytelnicy będą mogli rzeczowo rozstrzygnąć problem zakupu. Równocześnie nasuwa się myśl, aby sygnalizować sprowadzane do kraju książki radzieckie. Zwłaszcza wydawnictwa „MIR", które przygotowuje coraz więcej doskonałych przekładów z literatury zachodniej. Zgłaszam też postulat omawiania na Waszych łamach tych tytułów książek zachodnich, zakupionych przez różne instytuty, które mogą zainteresować szersze grono specjalistów.

Nie chciałabym się tu rozwodzić nad sprawą odnotowywania ważniejszych artykułów z czasopism zagranicznych. Sądzę jednak, że obszerne streszczenia obcojęzycznych prac poświęconych ocenie aktualnego stanu wiedzy czy techniki, bądź nowym technikom, prognozom itp., są nie mniej ważne niż oryginalne prace „krajowe", omawiające problemy szczegółowe. Powstaje pytanie, kosztem którego z działów można byłoby rozszerzyć takie aktualności? Otóż uważam, że zbyt dużo miejsca poświęca się na popularyzację sylwetek „zasłużonych dla informatyki"...

Zamysł skądinąd słuszny, nie mniej realizacja zamienia się często w malowanie laurek, a niektóre teksty nasuwają skojarzenia z tanim reklamiarstwem. Równocześnie budzi pewien niepokój fakt, że stosunkowo dużą część numeru zajmują różne „branżowe kąciki". Odnoszę często wrażenie, że działy te zapełnia się na siłę — byle tylko wykorzystać przydział stron.

Brakuje na łamach INFORMATYKI informacji o nowym produkowanym w kraju sprzęcie informatycznym. Ogólny artykuł dyrektora Zjednoczenia MERA, aczkolwiek potrzebny, nie załatwia sprawy bieżącej informacji. Wiadomo przecież, jak trudno dostać aktualny katalog! (Przeczytałabym chętnie np. o mikrokomputerze MERA 60 i to zarówno o architekturze systemu, jak i o jego oprogramowaniu).

Wreszcie sprawa olbrzymiej wagi: docenianie znaczenia mikroprocesorów dla krajowej informatyki. Wyrażam wielkie uznanie dla Redakcji za szybkie podjęcie tego ważnego tematu. Zagadnienia systemów mikrokomputerowych powinny być otoczone szczególną troską Redakcji i powinny regularnie pojawiać się na łamach. Oczywiście, materiały powinny uwzględniać informatyczny punkt widzenia. Publikacje mogłyby wykorzystać doświadczenia czasopism zachodnich. Można wprawdzie powiedzieć, że mamy pilniejsze sprawy związane z użytkowaniem maszyn cyfrowych, a mikroprocesor jest w kraju rarytasem. Sądzę jednak, że nie zwalnia to nas z obowiązku szerokiego przygotowania ludzi do wykorzystania mikroprocesorów. Pamiętajmy bowiem, że zaistniała obecnie nowa sytuacja: części systemu mikroprocesorowego są przeważnie dostępne „z półki" i tanieją z roku na rok. Warto też chyba zamieścić w INFORMATYCE obszerniejszy materiał o nowych, tzw. „silnych" mikroprocesorach.

Ostatnio ukazuje się wiele książek poświęconych różnym aspektom oprogramowania, w większości związanych z problemami szczegółowymi. Brakuje natomiast książek poświęconych ogólnym, „filozoficznym" zagadnieniom inżynierii oprogramowania. Sądzę, że wielu osobom uprawiającym zawodowo informatykę potrzeba głębszej refleksji nad programowaniem. Czy INFORMATYKA może w istotny sposób wpłynąć na kulturę programowania i na kształtowanie nowego spojrzenia zawodowego, czy też ma tylko służyć wymianie informacji fachowych — jest sprawą dyskusyjną.

Na zakończenie chciałabym zauważyć, że na łamach INFORMATYKI rzadko znajduję materiały poświęcone tzw. „automatyzacji prac zawodowych", czy — jak kto woli — „projektowania wspomaganego komputerowo". Być może jest to tylko wrażenie, spowodowane „skrzywieniem zawodowym". Sądzę jednak, że bardzo prawdopodobny jest w najbliższych latach wzrost zainteresowania w kraju tą problematyką. Będzie to głównie wynikało z większej dostępności sprzętu, spowodowanej niższą ceną wynikającą z zastosowania mikroprocesorów (ostatnio firma Hewlett-Packard poinformowała o masowym wprowadzeniu na rynek superkalkulatora stołowego typu HP 85A po raz pierwszy w cenie poniżej 5000 $ — z małym monitorem ekranowym, „mocnym" BASICIEM i wszechstronną biblioteką programów). Warto byłoby może przedstawiać na łamach INFORMATYKI małe, ciekawe programy w BASICU, poświęcone różnym zagadnieniom inżynierskim.

Krystyna ŻEBROWSKA

A jednak pomogło!

DO REDAKCJI

W INFORMATYCE (nr 4/80) opublikowaliście artykuł, w którym zajęliście się naszymi kłopotami związanymi z uruchomieniem dostarczonego we wrześniu 1979 r. minikomputera MERA 400. Z przyjemnością informujemy, że w dniu 23 maja br. serwis techniczny producenta zainstalował posiadany przez nas System MERA 400 wraz z urządzeniami zewnętrznymi. Uruchomiony minikomputer jest sprawny.

Obecnie posiadamy pełny skład zespołu do obsługi MERY oraz dwa komplety programów do obliczeń inżynierskich: PROBUS i ZELBET 80-M. Nie możemy jednak korzystać z tych programów, gdyż brak nam papieru do drukarek. Na nasze zamówienie tak producent — Kieleckie Zakłady Wyrobów Papierniczych — jak i Zjednoczenie Przemysłu Papierniczego odpowiadają, że produkcja papieru do drukarek wierszowych jest niewystarczająca i nie pokrywa nawet potrzeb dotychczasowych odbiorców. MERA nie jest więc w pełni wykorzystana, co wpływa ujemnie na wydajność i pewnego rodzaju rozgoryczenie wśród naszych pracowników.

Inż. Stanisław JAROCZYŃSKI
dyrektor Biura Projektów
Budownictwa Ogólnego
MIASTOPROJEKT w Nowych Tychach

O planowaniu w informatyce

Ustalone już dość dawno, bo około sześć lat temu, zasady koordynacji rozwoju informatyki (zaktualizowane w czerwcu br. uchwałą o Komitecie Informatyki) zakładają, że za zastosowanie informatyki odpowiadają resorty. Podważenie tego założenia nie byłoby chyba rozsądne, informatyka bowiem jest jedynie narzędziem, jednym ze środków realizacji zadań gospodarczo-społecznych.

Gdy jednak spojrzeć wstecz na rozwój informatyki w Polsce daje się zauważyć, że była ona najczęściej rozwijana jako przedsięwzięcie samo dla siebie, realizowane zresztą z olbrzymim zapałem przez informatyków, którym patronowała moda i których chroniła ślepa wiara decydentów. Wiara ta miała zresztą zwykle jakieś uzasadnienie, choćby przykłady cytowane z literatury czy rozwiązania oglądane u innych.

Niestety, zbyt często się jednak okazywało, że wysiłek oraz niemałe koszty przynosiły efekty znacznie poniżej rozbudzonych nadziei. Ciekawy jest przy tym fakt, że niezaspokojone nadzieje nie prowadziły z reguły do plajty przedsięwzięcia. Podpierano je nadal, sztukowano, tłumacząc niepowodzenia w różny sposób; niestety, bardzo rzadko zdarzało się, aby poszukiwanie przyczyn kończyło się wskazaniem tej najważniejszej, tj. błędnej metody postawienia zadania, realizowanego przez informatyków. Na marginesie warto wspomnieć, że nie był to błąd specyficznie polski, podobnie działo się na całym świecie.

Na czym ten błąd polegał? Wykorzystajmy pewną analogię. Przy konstruowaniu silnika elektrycznego pomysłowość i zapał konstruktorów spotyka się z jasna określoną potrzebą; każdy łatwo może sobie wyobrazić przyszłe jego zastosowania czy też warunki, jakie musi spełniać. Żaden decydent, choćby nawet wierzył w nieograniczoność możliwości silnika elektrycznego, nie byłby w stanie przekonać konstruktora o potrzebie wykorzystania tego silnika do — powiedzmy — odpinania guzików marynarki. Z informatyką rzecz ma się niestety, zupełnie inaczej.

Z jednej strony — gros informatycznych zastosowań leży w usprawnianiu procesów zarządzania, które z natury rzeczy nie są ani oczywiste, ani łatwe (o czym przekonuje doświadczenie); z drugiej — możliwości oprogramowania komputera nie są oczywiste, tym bardziej, że można je znacznie rozwinąć. Nic też dziwnego, że decydenci wierzyli, wierzą i — na szczęście — będą wierzyć w zapewnienia informatyków o zaletach komputerów. Jednej rzeczy natomiast nie wolno im przekazać w ręce informatyków — postawienia konkretnego zadania, choćby zadania minimum, ale za to: jasno określonego i sprawdzalnego. I właśnie d e c y d e n t powinien przede wszystkim rozumieć sens przedsięwzięcia. Oddanie w tym zakresie inicjatywy w ręce informatyków doprowadziło już do kryzysu zaufania w możliwości informatyki.

Takie postawienie sprawy oznacza, iż zadanie musi być sformułowane w kategoriach innych niż informatyczne. Na przykład, przy podejmowaniu decyzji o komputeryzacji oddziału PKO, decydent winien jasno postawić zadanie. Dajmy na to:

a) skrócić dzięki komputerowi czas obsługi klienta do maksimum jednej minuty lub
b) upiększyć każde stanowisko obsługi klienta klawiaturą i innymi urządzeniami, zgodnie z modą na nowoczesność. Nie powinien zaś nigdy akceptować zadania, które brzmiałoby:
c) zainstalować na każdym stanowisku obsługi klienta teletransmisyjny system oparty na bazie danych oraz kanałach łączności typu X15/123.

Czytelnik może się wprawdzie dziwić, że przypadek b) uznaliśmy za właściwe sformułowanie zadania, jednak prosimy zauważyć, iż jest ono jednoznaczne, a opiera się na poczuciu estetyki decydenta i nic też innego poza estetyką mu nie obiecuje. Można by jedynie mieć do niego pretensję, iż ma zbyt kosztowne gusta.

Może się zdarzyć, że również sformułowanie c) jest zrozumiałe dla decydenta, wówczas jednak mamy do czynienia z niecodziennym przypadkiem „wielospecjalnościowego" fachowca, albo po prostu — niewłaściwego człowieka na tym stanowisku. Gdyby nawet był to tego rodzaju unikalny fachowiec, to i tak dla powodzenie przedsięwzięcia musiałby on uświadomić swój personel, co w jego codziennej pracy oznacza sformułowanie c), a więc tak czy inaczej określić zadanie podobnie jak w przypadkach a) lub b).

Ten przydługi i być może banalny wywód miał przedstawiać i uzasadnić tę tylko tezę, że nie ma sensu planować i rozliczać przedsięwzięć informatycznych bez sprecyzowania jasnego celu pozainformatycznego, oczywistego dla tych, którzy za to przedsięwzięcie płacą.

Przedstawiona zasada winna obowiązywać na każdym szczeblu: przedsiębiorstwa, resortu czy wreszcie — państwa. Jeśli tak nie jest, to trudno się dziwić, że nigdy dotychczas nie zdarzyło się, aby na posiedzeniach Rady Ministrów stawiano problem niedostatku komputerów. Nic też dziwnego, że eksport urządzeń komputerowych sięga niemal 95%; skutek gospodarczy jest tu bowiem jasno określony. Kierownicy innych resortów co najwyżej zazdroszczą zarobionych w ten sposób dewiz, zamiast martwić się o planowane u siebie przedsięwzięcia gospodarcze, którym z a b r a k n i e sprzętu komputerowego. Prezentowane im bowiem przedsięwzięcia informatyczne nie są przekonywające, a w konsekwencji kierownictwa resortów nie wiążą z nimi sukcesu.

Czy sam Komitet Informatyki jest w stanie tutaj czegoś istotnego dokonać? Otóż — jest! Po pierwsze, musi doprowadzić do tego, aby plany rozwoju informatyki były zawsze elementem mierzalnych przedsięwzięć gospodarczych czy społecznych i wówczas będzie można rozważać co też się bardziej krajowi opłaca. Wskazane są tu przedsięwzięcia możliwie duże; małe bowiem winny być w gestii dyrektora zakładu bądź szefa instytucji.

O planie rozwoju samej informatyki, a więc o planie, który opracowują, zatwierdzają i kontrolują sami informatycy bez odwoływania się i oglądania na odgórną akceptację (a tym samym na rozgrzeszenie, jeśli coś sknocą), można mówić dopiero na tle zidentyfikowanych potrzeb gospodarczych. Dopiero wtedy można żądać, aby producenci sprzętu komputerowego — oprócz kryterium korzyści eksportowych — kierowali się również w swojej polityce kryterium zaspokojenia potrzeb krajowych.

Na zakończenie chcielibyśmy dodać, że przedstawiony pogląd ma nie tylko teoretyczny charakter. Trwają już wstępne prace zespołu kierowanego przez Sekretariat Komitetu Informatyki, mające na celu sformułowanie szczegółów metodologicznych tak rozumianego rozwoju informatyki, w efekcie — także sprawozdawczości. Sądzimy, że planowanie wychodzące z zadań ogólnogospodarczych będzie nie tylko skuteczniejsze, ale również prostsze, gdyż w założeniach bardziej zrozumiałe.

Sądzimy, że najwyższy już czas, żeby zacząć wyciągać wnioski z doświadczeń kilkunastoletniej historii informatyki w Polsce. Brak mocnej relacji między nakładami na informatykę a jej efektami gospodarczymi jest nie do utrzymania. Nikt już w Polsce nie wyda, szczególnie w obecnej sytuacji, ani złotówki na przedsięwzięcia, których główną wartość stanowi jedynie satysfakcja intelektualna poszczególnych osób lub grup. Trzeba wreszcie zacząć robić to, co jest niezbędne, nie zaś to tylko, co sprawia przyjemność.

Janusz Gwiazda
Janusz Kwiek

35 lat Naczelnej Organizacji Technicznej

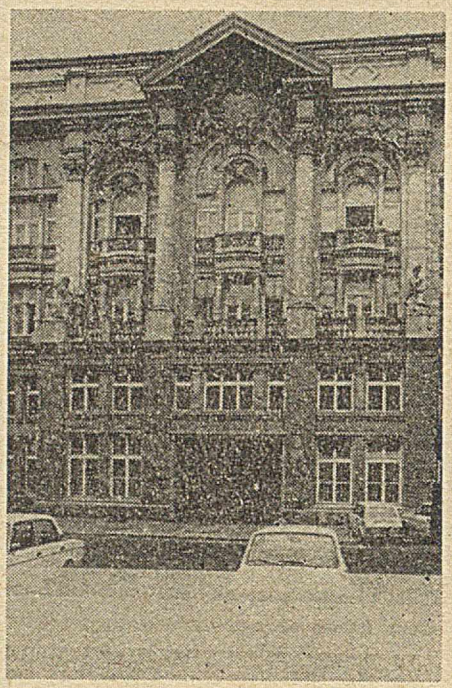
20 Stowarzyszeń naukowo-technicznych, 24 Komitety naukowo-techniczne, 49 Oddziałów Wojewódzkich, 77 Komitetów Miejskich (Miejsko-Gminnych), 14 Komisji Głównych — to potencjał merytoryczny i organizacyjny Naczelnej Organizacji Technicznej dzisiaj — w trzydziestym piątym roku jej istnienia. Początki sięgają pierwszych miesięcy po wyzwoleniu.

Po reaktywowaniu w lutym 1945 r. swej działalności przez przedwojenne stowarzyszenia techniczne w Warszawie, Poznaniu, Krakowie i Łodzi, w drugiej połowie tegoż roku, powstała w warszawskim środowisku technicznym myśl utworzenia jednej organizacji technicznej, która zjednoczyłaby w swych szeregach inżynierów i techników z całego kraju. Nowa organizacja miała w powojennych warunkach społecznych stworzyć jednolitą podstawę organizacyjną dla całego środowiska technicznego, miała być ważnym społecznym czynnikiem planowania oraz kontroli produkcji.

Myśl ta została zrealizowana 12 grudnia 1945 roku — w dniu tym odbyło się inauguracyjne zebranie i od tej daty liczy się historia Naczelnej Organizacji Technicznej. W grudniu br. mija 35 lat od tej inauguracji.

Pierwsze lata istnienia NOT minęły pod znakiem wytężonej pracy organizacyjnej. W styczniu 1946 roku na zebraniu plenarnym Komitetu Organizacyjnego NOT zatwierdzono deklarację ideową i statut NOT, w maju tego roku zapadła decyzja o przejęciu od Ogólnopolskiego Towarzystwa Technicznego jego dotychczasowego organu — „Przeglądu Technicznego" oraz utworzeniu Spółdzielni Wydawniczej, zadaniem której byłoby publikowanie i rozpowszechnianie wydawnictw technicznych.

Ruch społeczny inżynierów i techników w Polsce od początku swego istnienia podejmował różnorodne działania na rzecz doskonalenia zawodowego kadr technicznych oraz kształtowania zaangażowanych postaw społecznych inżynierów i techników. Działania te należą do statutowych zadań NOT. Powtórzmy je w nieco innym ujęciu: organizowanie i aktywizowanie wszelkich środowisk technicznych w celu szerzenia postępu technicznego, tworzenia warunków dla rozwoju wynalazczości i racjonalizacji, zdobywania przez kadrę specjalistów wyższych kwalifikacji oraz ich pełnego wykorzystania, uczestnictwo w tworzeniu koncepcji rozwoju poszczególnych działów gospodarki i gałęzi przemysłu, podnoszenie poziomu etyki zawodowej.



Nie mniej ważnym zadaniem NOT jest wymiana wiedzy i doświadczeń zawodowych oraz kształtowanie kultury technicznej w społeczeństwie, popularyzowanie nowej techniki i jej twórców. Istotną rolę w tej działalności odgrywa prasa techniczna.

Nawiązując do stuletnich tradycji wydawniczych polskich zrzeszeń technicznych oraz wychodząc naprzeciw dążeniom rozwijających się stowarzyszeń Naczelna Organizacja Techniczna w 1949 roku powołała do życia agendę wydawniczą pod nazwą Administracje Czasopism Technicznych, przekształconą w 1962 roku w Wydawnictwa Czasopism Technicznych NOT, a w r. 1979 w Wydawnictwo Czasopism i Książek Technicznych SIGMA. W ciągu 30 lat swej działalności Wydawnictwo NOT stało się największym w kraju wydawcą czasopism fachowych przeznaczonych dla inżynierów, techników i robotników wykwalifikowanych oraz czasopism popularnotechnicznych dla wszystkich zainteresowanych techniką — również poza zawodowo.

Szybkiemu przekazowi informacji i wymiany doświadczeń służą również organizowane na szeroką skalę przez Naczelną Organizację Techniczną i stowarzyszenia naukowo-techniczne konferencje, sympozja, narady, odczyty, pokazy filmów, wycieczki do zakładów pracy. W imprezach tych uczestniczą każdego roku setki tysięcy inżynierów i techników.

Istotne znaczenie dla podnoszenia poziomu wiedzy o rozwoju i osiągnięciach nauki i techniki w świecie, poznania działalności organizacji technicznych w innych krajach, prezentowania dorobku polskiej myśli technicznej na forum międzynarodowym — ma działalność NOT i zrzeszonych w niej stowarzyszeń w dziedzinie kontaktów i współpracy z zagranicznymi i międzynarodowymi organizacjami oraz środowiskami naukowo-technicznymi.

Podstawowe zadania i sposoby ich realizacji przez polskich inżynierów i techników oraz kształtowanie coraz lepszych form działania ruchu stowarzyszeniowego wyznaczane były i są nadal przez kolejne Kongresy Techników Polskich. Mają one w historii Polski Ludowej bogate tradycje. Jako generalne debaty techniczno-gospodarcze, poprzedzane dyskusjami środowiskowymi i specjalistycznymi, odgrywają ważną rolę w życiu gospodarczym i społecznym kraju. Siedem powojennych kongresów techników — przygotowanych i przeprowadzonych przez NOT i SNT — obejmowało tematycznie kolejne fazy gospodarczego rozwoju kraju.

Rok 1980, trzydziesty piąty rok istnienia Naczelnej Organizacji Technicznej to okres intensywnych przygotowań do VIII Kongresu Techników Polskich, oceny dorobku i dalszych zamierzeń całej federacji.