OŚRODEK BADAWCZO-ROZWOJOWY INFORMATYKI

OPROGRAMOWANIE SYSTEMOWE

Europejski Program Badawczy Diebolda

70

Warszawa 1975

OŚRODEK BADAWCZO-ROZWOJOWY INFORMATYKI

OPROGRAMOWANIE SYSTEMOWE

Europejski Program Badawczy Diebolda

Wyłącznie do użytku na terenie PRL

70

Warszawa 1975

Tytuł oryginału: Software - systems
Document No. E 125 M

Przekład: Józef Niedźwiecki
Redakcja: Zdzisław Zapolski

Opracowanie stylistyczne: Małgorzata Frontczak
Opracowanie techniczne: Kazimierz Walczyński
Korekta: Mirosława Gałecka-Guzik

Wydawca

OBRI - Dział Wydawnictw, 00-017 Warszawa, ul. Marszałkowska 104/122

Warszawa 1975. Nakład: 850 + 135 egz. Objętość: ark. wyd.2,25; ark. druk. 6. Format A4. Papier offsetowy kl. III, 80g, 61x86

zam. 228/75 Cena zł 92.-

SPIS TREŚCI

STRESZCZENIE

Systemy komputerowe rozwinęły się poważnie od czasu, kiedy użytkownik musiał rozumieć i kontrolować każdy aspekt systemu, jeśli chciał za jego pomocą zrealizować niewielkie nawet, lecz przydatne dla siebie zadanie. Obecnie sprzęt i oprogramowanie wykonują automatycznie wiele powtarzających się funkcji, które dawniej wymagały najwięcej czasu i wysiłku od programistów i analityków systemów.

W początkowym okresie programowania użytkownik musiał wyrażać swoje problemy w wewnętrznym języku maszyny, musiał sterować pracą wszystkich urządzeń wejścia/wyjścia, zajmować się przydzielaniem miejsc w pamięci oraz na innych nośnikach pamięciowych, tworzyć i utrzymywać zbiory danych, opracowywać własne programy usługowe. Jednakże, gdy wzrosła szybkość i złożoność komputerów stało się jasne, że użytkownicy potrzebują nowych instrumentów do sterowania systemami i do skrócenia czasu potrzebnego na definiowanie problemów.

Jednym z rezultatów tej ciągłej ewolucji jest rozwój rozmaitych "pakietów programowych" /programów standardowych/ przystosowanych do realizacji wielu funkcji, które dawniej były wykonywane przez personel zajmujący się programowaniem u użytkownika. Te pakiety znane są pod wspólną nazwą "systemów operacyjnych"; tworzą one, jeśli idzie o ich złożoność, szeroki wachlarz od najprostszych systemów do systemów ezoterycznych.

W skład systemów operacyjnych wchodzą na ogół następujące elementy:

. system sterowania,
. system zarządzania danymi,
. translatory językowe /programy tłumaczące/,
. programy usługowe,
. programy komunikacyjne /programy obsługujące teletransmisję danych/.

W niniejszym opracowaniu omówiono rozwój oprogramowania systemowego w trzech kolejnych okresach pięcioletnich: 1970-75,

1975-80 oraz 1980-85. Każdemu z tych okresów poświęcony jest jeden rozdział, w którym opisane są przewidywane zmiany i rozwój oprogramowania systemowego w danym okresie czasu.

Okres 1970-75

W okresie 1970-75 nie nastąpiła poważniejsza zmiana w dziedzinie oprogramowania systemowego; zostały wprowadzone jedynie usprawnienia techniczne o mniejszym znaczeniu. Te usprawnienia w dziedzinie oprogramowania, które przede wszystkim dały korzyści ekonomiczne, polegały na zastosowaniu znanych już technik, a w szczególności techniki pamięci wirtualnej. Koncepcja pamięci wirtualnej nie jest nowa, jednakże jej wdrożenie odbywało się wyjątkowo wolno.

Okres 1975-80

W latach 1975-80 zacznie zanikać znana nam obecnie koncepcja "centralnego procesora", a wraz z nią także dzisiejsze systemy operacyjne. Dominująca stanie się koncepcja rozdzielonej architektury jednostki centralnej, której zarysy zaczęły się kształtować już w latach 1970-75. Translatory językowe /programy tłumaczące/, programy obsługujące zbiory itp. zostaną wbudowane do sprzętu. Wraz z rozszerzeniem się zakresu czynności inteligentnych terminali upowszechni się stosowanie zdalnego przetwarzania oraz tzw. "rozdzielonej inteligencji".

System automatycznego przetwarzania danych będzie miał całkowicie modułową strukturę, a większość funkcji z zakresu oprogramowania będzie przejęta przez sprzęt. Tak więc system APD stanie się systemem przystosowanym do teletransmisji danych i sterowanym przez dane.

Aktualnie prowadzone prace badawcze w dziedzinie oprogramowania, których tematem jest sprawdzanie poprawności programów oraz automatyzacja programowania, umożliwią wdrożenie osiągnięć w tym zakresie do praktyki dnia codziennego. W związku z rozwojem "proto-systemów" w sposób radykalny zmieni się rola programistów, zajmujących się opracowywaniem programów zastosowaniowych. "Proto-systemy" będą zaopatrywały użytkowników w pomoce przydat-

ne przy rozwiązywaniu ich problemów w dziedzinie zastosowań. Systemy te będą projektowane dla specyficznych zastosowań lub funkcji i spełniać będą rolę właściwego pośrednika między użytkownikiem a systemem. Umożliwią one użytkownikowi działającemu w ramach funkcji lub tematu, do których "proto-system" będzie przystosowany, użycie komputera do rozwiązania problemu, bez formalnej znajomości programowania.

Architektura tzw. "rozdzielonej inteligencji" doprowadzi do systemów "fail soft", a także do zdalnej naprawy jednostek, które uległy awariom.

Okres 1980-85

Ten okres będzie można nazwać "erą konsolidacji", ponieważ wiedza nagromadzona w okresach poprzedzających zostanie wykorzystana do rozwinięcia systemu totalnego. Ten okres będzie świadkiem ewolucji systemu APD w kierunku systemu będącego uzupełnieniem rozumu ludzkiego, z właściwym mu przetwarzaniem informacji.

I. OPROGRAMOWANIE SYSTEMOWE

Przegląd rozwoju oprogramowania systemowego

Systemy komputerowe rozwinęły się poważnie od czasu, kiedy użytkownik musiał rozumieć i kontrolować każdy aspekt systemu, jeśli chciał za jego pomocą zrealizować niewielkie nawet, lecz przydatne dla siebie zadanie. Obecnie sprzęt i oprogramowanie wykonują automatycznie wiele powtarzających się funkcji, które dawniej wymagały najwięcej czasu i wysiłku od programistów i analityków systemów.

W początkowym okresie rozwoju programowania użytkownik musiał wyrażać swoje problemy w wewnętrznym języku maszyny, musiał sterować pracą wszystkich urządzeń wejścia/wyjścia, zajmować się przydzielaniem miejsc w pamięci oraz na innych nośnikach pamięciowych, tworzyć i utrzymywać zbiory danych, opracowywać własne programy usługowe. Jednakże, gdy wzrosła szybkość i złożoność komputerów, stało się jasne, że użytkownicy muszą mieć nowe instrumenty do sterowania systemem i dla skrócenia czasu potrzebnego na definiowanie problemów.

Jednym z rezultatów tej ciągłej ewolucji jest rozwój rozmaitych "pakietów programowych" /programów standardowych/ przystosowanych do realizacji wielu funkcji, które dawniej wykonywał personel zajmujący się programowaniem u użytkownika. Pakiety te znane są pod wspólną nazwą "systemów operacyjnych"; tworzą one jeśli idzie o ich złożoność, szeroki wachlarz od systemów najbardziej prostych do systemów ezoterycznych.

A. Rodzaje systemów operacyjnych

Systemy operacyjne są na ogół tak projektowane, aby działać w jednym lub kilku następujących układach:

- w układzie jednoprogramowym /Monoprogramming/ - jedno zadanie jest rozpoczynane i zrealizowane, po nim zostaje wprowadzone zadanie następne;
- w układzie wieloprogramowym /Multiprogramming/ - kilka zadań realizowanych jest równocześnie;
- w układzie wieloprocesorowym /Multiprocessing/ - zadanie zostaje podzielone między dwa lub kilka procesorów. Umożliwia to szybsze przetwarzanie oraz pozwala na zastąpienie procesora, który uległ awarii.

Jednoprogramowość

W warunkach jednoprogramowości występują dwie formy działania:

- przetwarzanie partiowe /Batch Processing/ - wszystkie dane wejściowe niezbędne dla danego programu są zebrane przed przystąpieniem do jego realizacji. Dane wejściowe mogą być wprowadzone do komputera na miejscu lub w sposób zdalny, co oznacza, że urządzenie wejściowe może znajdować się fizycznie w tym miejscu, w którym znajduje się komputer lub może być połączone z systemem za pomocą publicznych środków łączności;
- transmisja danych /Communications/ - dane są w sposób ciągły przyjmowane od terminali wejścia/wyjścia, a następnie przetwarzane i przesyłane z powrotem. Terminale mają bezpośrednie połączenia z komputerem poprzez publiczne środki łączności.

Wieloprogramowość/Wieloprocesorowość

W warunkach wieloprogramowości lub wieloprocesorowości można przetwarzać programy w obu układach, w układzie przetwarzania partiowego oraz w układzie transmisji danych. W dodatku warunki te dopuszczają także programy interaktywne. W programie interaktywnym oddalony użytkownik prowadzi "dialog" z komputerem w tym sensie, że po każdym wejściu lub "zapytaniu" z terminalu, następuje odpowiedź lub potwierdzenie z komputera. Przetwarzanie interaktywne stanowi poważne rozszerzenie systemu komputerowego, ponieważ dzięki niemu opracowywanie programów, rozwiązywanie problemów, manipulowanie zbiorami oraz zadawanie pytań może odbywać się z oddalonych punktów, w odpowiednio krótszym czasie i przy niższych kosztach.

B. Składniki systemu operacyjnego

Niezależnie od sposobu przetwarzania, systemy operacyjne zawierają na ogół następujące składniki:

- system sterowania /Control system/ - dla zapewnienia sprzężenia między biegnącym programem /producenta lub klienta/ a konfiguracją sprzętu. Zadaniem tego sprzężenia jest automatyczne planowanie, alokacja /przydzielanie/ oraz sterowanie urządzeniami wykorzystywanymi w systemie;
- system zarządzania danymi /Data management system/ - dla sterowania bazami danych masowych poprzez prowadzenie automatycznej alokacji obszarów zbiorów oraz przez wykorzystanie sprawnych technik przechowywania i wyszukiwania zapisów w tych zbiorach;
- translatory językowe /Language processors/ - dla tłumaczenia programów pisanych w języku symbolicznym, naukowym lub w języku typu angielskiego na język binarny maszyny;
- programy usługowe /Service programs/ - dla realizacji powtarzających się często czynności usługowych /takich jak przenoszenie danych z kart dziurkowanych na taśmę magnetyczną/;
- programy komunikacyjne /Communications programs/ - dla obsługi publicznych linii łączności oraz oddalonych terminali.

System sterowania

Rozwój techniczny przyniósł w tych latach tak znaczne postępy w zakresie szybkości działania sprzętu komputerowego, że wiele elementów tego sprzętu jest właściwie bezczynnych w czasie wykonywania pojedynczego programu. Wynika to stąd, że większość programów zajmuje się głównie albo wykonywaniem obliczeń, albo wykonywaniem operacji wejściowych i wyjściowych. Tak więc w większości przypadków albo jednostka centralna jest zajęta a urządzenia peryferyjne są bezczynne, lub na odwrót.

W celu bardziej efektywnego wykorzystania sprzętu, niektórzy programiści próbowali, na początku, włączać elementy z różnych zadań do jednego programu. Ta metoda okazała się jednakże niepraktyczna, była bowiem zbyt trudna dla większości programis-

tów. Prace badawcze w zakresie programowania, prowadzone u różnych producentów, doprowadziły w końcu do rozwoju metody, która pozwala na równoczesne i efektywne użytkowanie sprzętu komputerowego przez niezależne programy w procesie wieloprogramowym. Wdrożenie tej metody wymagało wyłączenia z indywidualnych programów elementów, zajmujących się planowaniem przetwarzania poszczególnych zadań /programów/, sterowaniem pracą urządzeń wejścia/wyjścia, zarządzaniem pamięcią oraz przerywaniem biegu przetwarzania, i ulokowania tych elementów we wspólnym pakiecie "kierowania ruchem" zwanym systemem sterowania. System sterowania jest właśnie tym sposobem, który umożliwia, w oparciu o programy i za pomocą sprzętu, realizację tych zadań z zakresu sterowania, które nazywane są często "zarządzaniem środkami".

Kierowanie zadaniami /Task Management/

Jednostka pracy, która ma być wykonana przez komputer, określana jest jako zadanie /task/. Posługując się programami kierowania zadaniami system sterowania ustala porządek /kolejność/, w jakim poszczególne zadania będą wykonywane i steruje rozdziałem czasu przetwarzania pomiędzy współbieżne zadania, w oparciu o priorytety ustalone przez użytkownika. Dodatkowo, programy kierowania zadaniami realizują funkcje łączności międzyzadaniowej, funkcje łączności z operatorem oraz funkcje z zakresu rozliczenia wykorzystania zastosowanego sprzętu.

Kierowanie wejściem/wyjściem /Input/Output Management/

Programy kierowania wejściem/wyjściem zajmują się przydzielaniem urządzeń we/wy i planują porządek, w jakim rozkazy we/wy mają być realizowane. Do obowiązków programów kierowania wejściem/wyjściem wchodzi także analizowanie wyników każdego rozkazu we/wy, sprawdzanie czy operacje zostały wykonane bezbłędnie oraz poprawianie w razie potrzeby błędów.

Zarządzanie pamięcią /Memory management/

Programy zarządzania pamięcią zajmują się alokacją oraz dealokacją obszaru pamięci komputera dla rozmaitych zadań, które mają być realizowane. Alokacja oznacza przydział miejsca w pa-

mięci celem wykorzystania go przez określony program. Dealokacja oznacza zabranie przydzielonego programowi, a już mu niepotrzebnego, miejsca w pamięci, po to aby je przydzielić innemu programowi.

Kierowanie przerywaniami /Interrupt Management/

Programy kierowania przerywaniami sterują reakcjami systemu na wymagania związane z obsługą programów. Analizują one różnego rodzaju przerwania w pracy komputera /zakończenie wejścia/ wyjścia, błąd w programie, awaria sprzętu, itp./ i przekazują sterowanie temu elementowi systemu sterowania, który jest odpowiednio zaprogramowany, aby podjąć działanie w danej sytuacji.

System operowania danymi /Data Management System/

Użytkownicy komputerów domagali się zwiększenia roli systemu operacyjnego w operowaniu zbiorami danych, gdy zbiory te stawały się coraz większe i bardziej złożone. Programy operowania danymi stworzyły użytkownikom możliwość jednorazowego przygotowania i przekazania danych do dyspozycji komputera, po czym system operacyjny przejmuje całkowitą kontrolę nad tymi danymi. Tak więc system operowania danymi przejmuje odpowiedzialnośc za umieszczenie zbiorów w pamięci, za udostępnianie ich zgodnie z wymaganiami użytkownika, za utrzymywanie zbiorów zastępczych /kopii zbiorów/ oraz za ostateczną eliminację zbiorów, gdy nie są one już potrzebne.

Translatory językowe /Language Processors/

Pisanie programów w języku wewnętrznym maszyny jest dla większości ludzi bardzo trudne i wymaga długotrwałego specjalistycznego szkolenia. Z tego też powodu producenci komputerów wprowadzili do użytku klientów wiele specjalnych języków programowania. Języki te umożliwiają formułowanie problemów, które mają być rozwiązywane przez komputer, w postaci bardziej zrozumiałej dla programistów niż dla komputera.

Pierwszym krokiem w tym kierunku było opracowanie języków

typu ASSEMBLERÓW, które przekształcały język symboliczny stosowany przez programistę na język maszynowy. Potem nastąpiło wprowadzenie kompajlerów COBOLu, FORTRANu, ALGOLu, PL1. Były to bardziej skomplikowane programy, które przekształcały wyrażenia angielskie lub wyrażenia matematyczne na kod maszynowy.

Wraz z rozwojem języków programowania wyższego rzędu, programowanie stało się w zasadzie niezależne od maszyny. Oznacza to, że programista może posiadać mniej technicznych wiadomości dotyczących komputera, aby móc opracowywać programy, które mają być przez ten komputer realizowane.

Programy usługowe /Service Programs/

Programami usługowymi są rutyny opracowywane dla wykonywania, w sposób standardowy, czynności często powtarzających się. Oto typowe przykłady rutyn usługowych:

. programy typowe /Utility routines/,
. programy diagnostyczne /Diagnostic routines/,
. programy konwersyjne /konwersji zbiorów z jednego nośnika na inny/ /Conversion routines/,
. programy biblioteczne /Library maintenance routines/,
. programy sortowania i łączenia /Sort/merge routines/.

Programy obsługujące teletransmisję danych /Communications Programs/

Możliwość podłączania odległych urządzeń wejścia/wyjścia bezpośrednio do komputera za pomocą linii telefonicznych, doprowadziła do rozwinięcia wyspecjalizowanych systemów obsługi jedynych w swoim rodzaju potrzeb użytkowników teletransmisji danych. Systemy te dzielone są na ogół na trzy kategorie:

. Pytanie-Odpowiedź /Inquiry-Response/
  To zastosowanie jest przede wszystkim systemem przekazu wiadomości "jedna za jedną". Po każdej wiadomości przesłanej z oddalonego terminalu do komputera, następuje przesłanie odpowiedzi z komputera do terminalu;
. Zbieranie i rozprowadzanie danych /Data Collection-Dissemination/

Te dwa rodzaje zastosowań są dzisiaj najbardziej wykorzystywane. Zbieranie danych zakłada wykorzystywanie oddalonych terminali, jako środków szybkiej i taniej transmisji danych do ośrodka komputerowego, dla przetwarzania. W systemach rozprowadzania danych informacje wychodzące z komputera /raporty, rachunki, itp./ rozsyłane są do odbiorców via obwody telekomunikacyjne, dla zaoszczędzenia czasu i dla obniżenia kosztów transportowych i pocztowych;

- Przełączanie informacji /Message Switching/
  W tego rodzaju systemach komputer wykorzystywany jest jako centrum przełącznikowe, przechowujące informacje przeznaczone dla wszystkich terminali przyłączonych do sieci teletransmisyjnej i rozsyłające je do nich.

Ogólnie biorąc, oprogramowanie dla teletransmisji danych jest wysoce wyspecjalizowaną formą sterowania wejściem/wyjściem oraz zarządzania danymi. Różni się ono od konwencjonalnego przetwarzania danych głównie sposobem wprowadzania informacji do systemu i wyprowadzania ich z niego. Informacje mogą napływać z terminali w dowolnym porządku oraz w nieustalonych interwałach czasowych. Zależnie od rodzaju systemu, informacje te mogą często "zaprzątać uwagę" systemu w określonym okresie czasu. Tak więc, w celu ułatwienia tego przepływu danych, niezbędna jest specjalna linia oraz urządzenia sterujące terminalem, aby zwiększyć normalne funkcje systemu w zakresie przetwarzania danych.

W dalszej części niniejszego opracowania omawiany zostanie rozwój oprogramowania systemowego w trzech następujących po sobie pięcioletnich okresach czasu: 1970-75, 1975-80 oraz 1980-85. Każdemu z tych okresów poświęcony jest rozdział, w którym przedstawione są przewidywane zmiany lub rozwój w dziedzinie oprogramowania systemowego w danym okresie.

II. OPROGRAMOWANIE SYSTEMOWE W LATACH 1970-1975

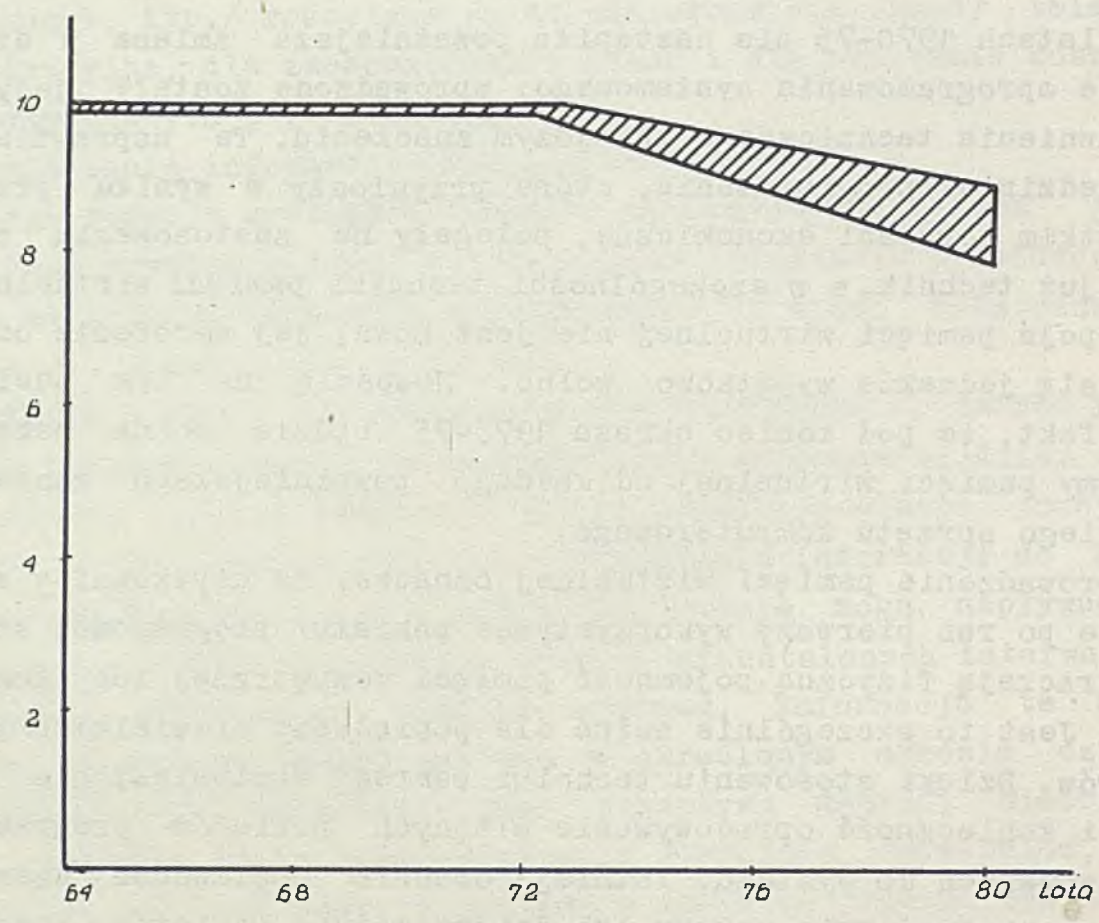
W latach 1970-75 nie nastąpiła poważniejsza zmiana w dziedzinie oprogramowania systemowego; wprowadzone zostały jedynie usprawnienia techniczne o mniejszym znaczeniu. Te usprawnienia w dziedzinie oprogramowania, które przyniosły w wyniku przede wszystkim korzyści ekonomiczne, polegały na zastosowaniu znanych już technik, a w szczególności techniki pamięci wirtualnej. Koncepcja pamięci wirtualnej nie jest nowa, jej wdrożenie odbywało się jednakże wyjątkowo wolno. Nowością na tym odcinku jest fakt, że pod koniec okresu 1973-75 będzie można nabywać systemy pamięci wirtualnej od każdego poważniejszego dostawcy wielkiego sprzętu komputerowego.

Wprowadzenie pamięci wirtualnej oznacza, że użytkownicy są w stanie po raz pierwszy wykorzystywać pakiety programowe, które przekraczają fizyczną pojemność pamięci wewnętrznej ich komputera. Jest to szczególnie ważne dla posiadaczy niewielkich komputerów. Dzięki stosowaniu techniki pamięci wirtualnej nie zachodzi konieczność opracowywania własnych pakietów programów, dostosowanych do systemu. Istnieją obecnie możliwości uzasadnionego ekonomicznie zakupu lub dzierżawienia pakietów programów, które mogą być z łatwością przystosowane do systemu komputerowego, posiadanego przez użytkownika.

Oprócz powyższego, systemy pamięci wirtualnej będą stymulowały stosowanie przygotowanych z góry /prefabrykowanych/ pomocy do programowania, w wyniku czego wzrośnie wydajność pracy programistów. W tym samym czasie wzrośnie wykorzystanie programów interaktywnych.

Rysunek 1 przedstawia prawdopodobny koszt wiersza kodu, który będzie wynikiem tego rozwoju.

Doprowadzi to w końcu do uproszczenia płaszczyzny styku między użytkownikiem a systemem /po części drogą przejęcia funkcji programowych przez urządzenia/. Będzie to podstawą wzrostu wydajności z punktu widzenia personelu zajmującego się rozwiązy-



RYS. 1. Koszt wiersza kodu

waniem problemów /nie programistów/, personelu zajmującego się programowaniem, a także operatorów i konserwatorów.

A. Pamięć wirtualna

Koncepcję pamięci wirtualnej można najłatwiej wyjaśnić przez przeciwstawienie systemów pamięci wirtualnej bardziej znanym systemom pamięci rzeczywistej.

W obu przypadkach program zajmuje jakiś obszar programowy. Przez obszar programu rozumie się całkowitą liczbę bajtów pamięci komputera, która została przydzielona danemu programowi; w obszarze tym znajdują się instrukcje programu, stałe programowe, miejsca danych i ewentualnie jakieś miejsca niewykorzystane.

Obszar programowy rezyduje zawsze na jakimś fizycznym nośniku pamięci. Odróżniamy dwa rodzaje fizycznych nośników pamięci: pamięć główną /na ogół rdzenie magnetyczne/ oraz pamięć pomocniczą /na ogół dysk lub bęben/. Określenie pamięć główna oznacza pamięć centralną komputera. W pamięci głównej muszą znajdować się instrukcje realizowane przez jednostkę centralną oraz dane, do których te instrukcje się odwołują. Pamięć pomocnicza służy do przechowywania programów oraz tych części systemu operacyjnego, które nie biorą aktualnie udziału w przetwarzaniu. Program, który ma być przetwarzany, musi zostać w całości lub częściami przeniesiony z pamięci pomocniczej do pamięci głównej. Program znajdujący się w pamięci głównej może być czasowo zawieszony i oczekując na dalsze przetwarzanie może być przeniesiony w całości lub częściami do pamięci pomocniczej.

Główne różnice pomiędzy systemami pamięci rzeczywistej a systemami pamięci wirtualnej polegają na metodach, które stosowane są przy alokacji obszaru programowego i przy sterowaniu nim. W systemie pamięci rzeczywistej obszar programowy określonego programu zajmuje przylegający /ciągły/ zespół miejsc w pamięci głównej. Gdy program jest przetwarzany, cały obszar programowy musi rezydować w pamięci głównej i musi zajmować w niej zwarte miejsce.

W systemie pamięci wirtualnej obszar programowy określonego programu zajmuje nie sąsiadujące ze sobą miejsca w obu pamięciach, tak w pamięci głównej jak i w pamięci pomocniczej. Gdy dany program jest przetwarzany, znaczne jego części mogą się wciąż znajdować w pamięci pomocniczej. Części programu rezydujące w określonym momencie czasu w pamięci głównej, zawierają przeważnie tylko te elementy programu, w których znajdują się aktualnie przetwarzane instrukcje oraz dane, do których instrukcje te odwołują się. Określenie pamięć wirtualna odnosi się do całości obszaru programowego czy obszaru adresowego, którym potencjalnie dysponuje indywidualny program.

Opracowano rozmaite metody sprzętowe i programowe, które umożliwiają wspólne wykorzystanie dysponowanej pamięci przez kilka przetwarzanych zadań. Ponieważ wymagania odnośnie miejsca w pamięci zmieniają się z programu na program /zmieniają się one nawet w trakcie realizowania jednego programu/, dlatego alokacja pamięci powinna być realizowana w sposób dynamiczny. Ponadto, z punktu widzenia użytkownika czy programisty, system powinien wyglądać tak, jak gdyby posiadał nieograniczoną pamięć główną i jak gdyby nie było ograniczeń odnośnie rozmiarów pisanych przez nie programów. W przeciwnym bowiem przypadku programiści poświęcają wiele czasu na "upychanie" programu lub na wymyślanie sposobów podziału programu na segmenty i składanie tych segmentów w czasie realizacji programu. Pomysł pamięci wirtualnej usuwa to skrępowanie programistów, zwiększając tym samym ich wydajność.

B. Pomoce dla programistów

Trzecia generacja komputerów zaznajomiła środowisko informatyków z oprogramowaniem systemowym. Wśród celów oprogramowania systemowego wymienić należy:

. uproszczenie płaszczyzny styku z użytkownikiem,
. zwiększenie wydajności programisty,
. poprawienie stosunku koszty/wydajność w systemach.

Koszty oprogramowania trzeciej generacji komputerów osiągnęły stałą wartość, zbliżającą się do 10 dolarów za wiersz kodów.

I choć koszt wiersza kodów pozostał bez zmian, to jednak koszt programowania podniósł się znacznie, a działalność w dziedzinie automatycznego przetwarzania danych stała się pracochłonna. Z drugiej strony zmniejszył się poważnie względny koszt urządzeń.

Akceptacja pamięci wirtualnej stwarza użytkownikom nowe możliwości znacznego zwiększenia wydajności personelu, zajmującego się automatycznym przetwarzaniem danych. Wielu użytkowników rozważa obecnie stosowanie prefabrykowanych pakietów programowych ze względu na konieczność utrzymania kosztów programowania na ustalonym poziomie, a także wobec konieczności zwiększenia opłacalności ekonomicznej nowych zastosowań. Ta tendencja umocni się, gdy z jednej strony użytkownicy nabiorą zaufania do dostawców, a z drugiej nabiorą doświadczenia w posługiwaniu się pamięcią wirtualną. Powyższemu trendowi towarzyszy wzrastające stosowanie, niezależnie dostarczanych pomocy programowych, których zadaniem jest zwiększenie wydajności programistów. Tendencja ta będzie się umacniała wraz z przechodzeniem użytkowników do stosowania systemów pamięci wirtualnej. Wzrastające stosowanie pomocy programowych wiąże się z przejściem użytkowników do stosowania języków programowania wyższego rzędu, co samo przez się przyniosło wzrost wydajności pracy programistów. Przy przyzwyczajaniu się do stosowania programów usługowych, dostępnych w ramach systemów operacyjnych, przeszkodą w akceptacji pomocy programowych mogły być jedynie wymagania wewnętrznej pamięci komputera /pamięci na rdzeniach magnetycznych/. Ponieważ przeszkoda ta została skutecznie usunięta wraz z wprowadzeniem pamięci wirtualnej, należy się spodziewać znacznego wzrostu stosowania pomocy programowych oraz rozwoju programów w systemie on-line.

Rozwój programów pracujących w systemie on-line nakłada nowe wymagania na oprogramowanie systemowe. W warunkach idealnego systemu dialogowego pożądanym staje się kompajler działający w sposób narastający. Takie kompajlery układają składnię analizując i tłumacząc każdy napływający wiersz kodu. Aktualnie kompajlery te są niezmiernie powolne i nie stosuje się ich na większą skalę. Kompajlery dialogowe /interaktywne/ oraz programy redagujące teksty w układzie on-line będą jednakże coraz sze-

rzej akceptowane. Obecnie w powszechnym użyciu są interaktywne programy redagujące teksty, które pomagają w dokonywaniu zmian i poprawek w programach, w wyznaczaniu nowych numerów sekwencyjnych, w przesyłaniu bloków kodów, we wprowadzaniu rutyn, itp. Większość interaktywnych kompejlerów oraz programów redagujących teksty w układzie on-line, dostarczanych jest przez producentów komputerów, w kategorii programów tłumaczących, w ramach systemu operacyjnego. Podczas, gdy producenci komputerów dostarczają interaktywne kompajlery oraz programy redagujące teksty, dodatkowe pomoce dla programistów, potrzebne przy opracowywaniu programów w układzie on-line, można uzyskać od wytwórców oprogramowania. Przygotowując się do opracowywania programów w układzie on-line, należy zbadać przydatność tych pomocy jako dodatkowego środka, umożliwiającego usprawnienie pracy programistów.

C. Diagnostyka on-line - niezawodność systemu

Przy budowie oprogramowania systemowego musi się zwracać szczególną uwagę na trwałość systemu. Jest to warunek kluczowy zadowolenia użytkowników, tym bardziej, że coraz więcej systemów automatycznego przetwarzania danych staje się "mózgami" korporacji. Jeśli pamięć lub urządzenia zarządzające pamięcią dostosowane są do zadań, jak to ma miejsce w systemach pamięci wirtualnej, wówczas można uzyskać trwały system. Trwały system powinien posiadać następujące właściwości:

. detekcję i korekturę błędów,
. diagnostykę on-line,
. testy lokalizujące uszkodzenia,
. możliwość izolacji,
. możliwość rekonfiguracji,
. możliwość restartu,
. bezpieczeństwo systemu.

Każdy z wymienionych wyżej elementów wymaga nie tylko wydatku na sprzęt i oprogramowanie, lecz także zmian technicznych, które pozwoliłyby uzyskać opłacalną ekonomicznie trwałość sys-

temu. Wymienione właściwości są obecnie technicznie osiągalne, przy zastosowaniu odpowiednich urządzeń i odpowiedniego oprogramowania. To oprogramowanie musiałoby jednakże być umieszczone w pamięci głównej, co podniosłoby w poważnej mierze koszt systemu.

Tym niemniej, dwa osiągnięcia dają nadzieję uzyskania udoskonalonej trwałości systemu; są nimi:

. zmniejszający się koszt urządzeń,
. pamięć wirtualna.

Przy obniżającym się koszcie sprzętu, staje się obecnie ekonomicznie opłacalne i możliwe wbudowanie do systemu komputerowego urządzeń dla detekcji i korektury błędów. Pierwsze kroki w tym kierunku zostały poczynione w systemach trzeciej generacji, w przyszłości dokona się więcej.

Wprowadzenie systemów pamięci wirtualnej usuwa skrępowanie związane z pojemnością pamięci głównej. W rezultacie, w bezpośredniej już przyszłości będziemy świadkami szerokiego stosowania testów lokalizujących uszkodzenia /FLT`s/ oraz diagnostyki on-line, jako pomocy przy odnajdywaniu uszkodzonych elementów w systemie. Te testy lokalizujące uszkodzenia /FLT`s/ oraz diagnostyka on-line realizowane będą pod kontrolą systemu operacyjnego. Metody restartu systemu także będą rozwinięte i znacznie usprawnione oraz wdrożone jako uzupełnienia systemu operacyjnego. Jest to szczególnie ważnym problemem w instalacjach wielodostępnych, gdzie wielu użytkowników będzie korzystało równocześnie z jednego systemu.

W wyniku tego rozwoju wzrośnie dyspozycyjność systemu, obniży się średni czas jego reperacji i niższy będzie poziom fachowy wymagany od personelu, zajmującego się konserwacją systemu.

Wzrost dyspozycyjności systemu przyniesie w rezultacie:

. zwiększoną wydajność,
. usprawnione planowanie przetwarzania,
. przyspieszoną rotację zadań,
. uproszczenie zadań operatora,
. zmniejszenie kosztów konserwacji,
. niższe wymagania kwalifikacyjne odnośnie personelu zajmującego się konserwacją.

Wymienione wyżej korzyści wynagrodzą użytkownikom wzrost wydatków, związany z uzyskaniem zwiększonej sprawności systemu. Te

narastające usprawnienia są zresztą niezbędnym polem doświadczalnym, prowadzącym do systemów typu "fail soft", które rozwiną się w latach 1975-80.

D. Efekty sprzętowe /hardware'owe/ w oprogramowaniu systemowym

W komputerach trzeciej generacji producenci usiłowali połączyć w jednym systemie wszelkie możliwe funkcje oraz udogodnienia. Te usiłowania dały dwa wyniki, a mianowicie:

. poważne opóźnienia w rozwoju oprogramowania,
. narażenie użytkowników na niepotrzebne wydatki, związane z wysokimi kosztami ogólnymi.

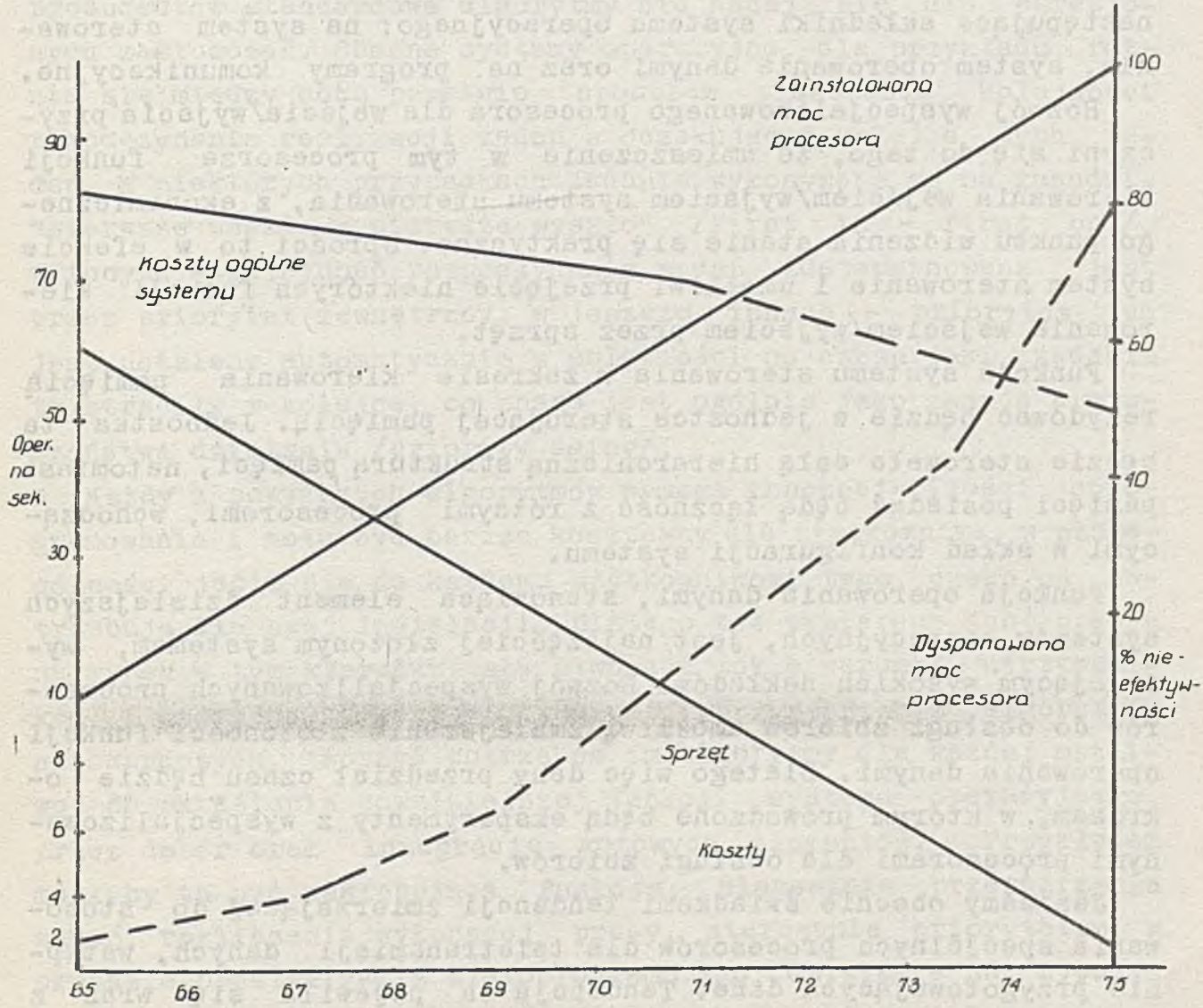
Usiłując rozwinąć systemy operacyjne przydatne dla systemów obsługujących wielu użytkowników, musiano podejmować nowe i skomplikowane zadania projektowe w oparciu o ograniczony zasób wiedzy. Spowodowało to opóźnienia w realizacji założonych harmonogramów nowych opracowań, a także niewyprodukowanie na czas zapowiedzianych już na rynku urządzeń. Wynikająca stąd niska efektywność systemu okazała się także kosztowna dla użytkowników.

Ostatnio proponuje się architekturę systemową, wprowadzającą specjalizowane "czarne skrzynki"; zespół takich "czarnych skrzynek" stanowi normalny system. Takie wyspecjalizowane elementy mają tę zaletę, że wprowadzane na bieżąco udoskonalenia techniczne w sprzęcie umożliwiają osiąganie wyższej wydajności przy niższych kosztach. Konwersja funkcji oprogramowania na funkcję sprzętu jest bardziej praktyczna z ekonomicznego punktu widzenia. Zalecane do szerszego stosowania piszące pamięci sterujące zapewniają niezbędną elastyczność.

Na rysunku 2 przedstawiono graficznie wzajemne zależności między cenami sprzętu, malejącymi kosztami ogólnymi systemu oraz osiągalną przez użytkowników mocą dysponowaną /przetwarzania/.

Można powiedzieć, że systemy najbliższej przyszłości będą się składały z następujących elementów:

. z wyspecjalizowanych procesorów dla wejścia/wyjścia,
. z hierarchii pamięci,



RYS. 2 Rewolucja sprzętowa

. z wyspecjalizowanych procesorów dla teletransmisji danych,
. z ultraszybkich procesorów dla arytmetyki oraz sterowania.

Zmiany w architekturze czy strukturze systemu także będą miały wpływ na organizację oprogramowania systemowego. Wymienione zmiany w architekturze systemu wywrą najmniejszy wpływ na następujące składniki systemu operacyjnego: na system sterowania, system operowania danymi oraz na programy komunikacyjne.

Rozwój wyspecjalizowanego procesora dla wejścia/wyjścia przyczyni się do tego, że umieszczenie w tym procesorze funkcji kierowania wejściem/wyjściem systemu sterowania, z ekonomicznego punktu widzenia stanie się praktyczne. Uprości to w efekcie system sterowania i umożliwi przejęcie niektórych funkcji kierowania wejściem/wyjściem przez sprzęt.

Funkcja systemu sterowania w zakresie kierowania pamięcią rezydować będzie w jednostce sterującej pamięcią. Jednostka ta będzie sterowała całą hierarchiczną strukturą pamięci, natomiast pamięci posiadać będą łączność z różnymi procesorami, wchodzącymi w skład konfiguracji systemu.

Funkcja operowania danymi, stanowiąca element dzisiejszych systemów operacyjnych, jest najczęściej złożonym systemem, wymagającym wysokich nakładów. Rozwój wyspecjalizowanych procesorów do obsługi zbiorów umożliwi zmniejszenie złożoności funkcji operowania danymi. Dlatego więc dany przedział czasu będzie okresem, w którym prowadzone będą eksperymenty z wyspecjalizowanymi procesorami dla obsługi zbiorów.

Jesteśmy obecnie świadkami tendencji zmierzającej do stosowania specjalnych procesorów dla teletransmisji danych, wstępnie przygotowujących dane. Tendencja ta pojawiła się wraz z przechodzeniem użytkowników do szerszego stosowania terminali. Nie jest bowiem ekonomicznie uzasadnione stosowanie uniwersalnego systemu APD do efektywnej obsługi dużej ilości terminali. W związku z obniżeniem się kosztów sprzętu, bardziej opłacalną obecnie jest instalacja procesora, zaprojektowanego specjalnie dla obsługi transmisji danych. Wobec tego, że takie procesory są obecnie osiągalne, realne stało się przeniesienie programów obsługujących teletransmisję danych z jednostki centralnej do specjalnego procesora komunikacyjnego, co przyczynia się dodatkowo do uproszczenia systemu operacyjnego.

E. Modułowość oprogramowania systemowego

Gdy wzrasta ilość i doświadczenie użytkowników komputerów, a także gdy oprogramowanie obejmuje coraz bardziej złożone i odpowiedzialne funkcje, zdarza się często, że opracowywane przez producentów standardowe algorytmy nie nadają się dla określonych zastosowań. Obecne systemy operacyjne, dla przykładu, różnią się między sobą poważnie sposobem planowania kolejności rozpoczynania realizacji zadań w oczekującej kolejce tych zadań. W niektórych przypadkach zadania wykonywane są na zasadzie "pierwsze weszło - pierwsze wyszło" /first in - first out/, w innych - kolejność rozpoczynania zadań zdeterminowana jest przez priorytet zewnętrzny, w jeszcze innych - priorytet ten jest ustalany automatycznie w zależności od czasu jaki zadania te straciły w kolejce, co znane jest ogólnie jako zasada pierwszeństwa działania /priority aging/.

Każdy z powyższych algorytmów wymaga znacznej ilości oprogramowania i może być bardzo kosztowny dla użytkownika, w szczególności jeśli nie da każdemu użytkownikowi tego, czego on potrzebuje dla swej instalacji. Dlatego też wywierana jest presja na rynek w tym kierunku, aby stworzyć takie warunki użytkownikom, w których mogliby oni, zamiast korzystania z algorytmów standardowych, tworzyć potrzebne im algorytmy dla każdej możliwej do określenia poważniejszej funkcji systemu operacyjnego, przez dobór oraz integrację gotowych elementów. Przykładem mogłyby tu być następujące funkcje: planowanie przetwarzania zadań, rozliczanie wykonanej pracy, sterowanie priorytetem w procesie przetwarzania wieloprogramowego, działanie na wypadek błędu, itp.

Jeśli użytkownik ma mieć możliwość doboru algorytmów standardowych, czy też możliwość zastępowania algorytmów dostarczanych przez producentów swoimi własnymi, to system musi mieć strukturę modułową. Oznacza to, że każda funkcja i każdy algorytm powinny być dobrze zdefiniowane odnośnie tego co one wykonują, a także tego z czym współpracują. Wszyscy dostawcy oprogramowania, tak producenci oprogramowania jak i producenci komputerów, notują zapotrzebowanie użytkowników na oprogramowanie o strukturze modułowej.

Przyczyną opóźnień w wytwarzaniu oprogramowania systemowego o strukturze modułowej był brak opracowanej i dostępnej metodologii, a więc takiej dyscypliny w zakresie oprogramowania, która spełniałaby rolę podobną do nauki o konstruowaniu. Tym niemniej dyscyplina ta rozwija się szybko. Dojdzie do takiej modułowości oprogramowania systemowego, że w efekcie powstanie jedyny system operacyjny, pozwalający użytkownikowi na wybór stopnia złożoności jego funkcji w oparciu o potrzeby operacyjne użytkownika. Wynika z tego, że każda instalacja do automatycznego przetwarzania danych będzie posiadała unikalne oprogramowanie systemowe, które będzie odpowiadało jej charakterystyce operacyjnej w każdym momencie czasu. Mogłoby to spowodować wzrost kosztów konserwacji systemu.

Oto niektóre właściwości modułów oprogramowania:

- moduł wykonuje specyficzną i jedyną funkcję, co znaczy, że jest on elementarną jednostką funkcyjną,
- moduł charakteryzuje się niewielką i określoną ilością wzajemnych oddziaływań z innymi modułami,
- wyjście modułu jest całkowicie zależne od funkcji danego modułu i absolutnie niezależne od czynności innych modułów,
- zmiana w określonej funkcji systemu powinna wymagać zmiany w jednym tylko module,
- moduł posiada dwa punkty styczne: jeden dla wejścia, poprzez który inne moduły komunikują się z nim, oraz jeden dla wyjścia, poprzez który on komunikuje się z innymi modułami,
- moduł może być testowany automatycznie, bez wywierania wpływu na pozostały system.

Stosowanie oprogramowania o strukturze modułowej przynosi następujące wyniki:

- daje więcej specjalizowanych, elastycznych systemów dla specyficznych użytkowników,
- łatwiejsze ustalanie błędów w systemie,
- możliwość łatwiejszego umacniania funkcji,
- możliwość przekazywania funkcji oprogramowania sprzętowi /wmontowywania funkcji oprogramowania do sprzętu/.

F. Przejmowanie funkcji oprogramowania przez sprzęt /realizacja software'u w hardware'rze/

Koszt jednostki centralnej w stosunku do kosztu całego systemu obniżał się z generacji na generację; ten zniżkowy trend będzie postępował dalej. Integracja wielkoskalowa spowodowała, że stosowanie logiki w sprzęcie /hardware logic/ zamiast oprogramowania /software/ jest coraz bardziej usprawiedliwione.

Z drugiej strony, oprogramowanie systemowe stało się bardziej skomplikowane bez jakichś poważniejszych przełomów w zakresie metodologii czy techniki oprogramowania. Gdy wzrosła złożoność oprogramowania systemowego, wzrosły także koszty oraz czas potrzebny na jego opracowywanie. Zanotowano tendencję zmierzającą, jak to przedstawiono w poprzednim rozdziale, do wprowadzania zmian funkcjonalnych w architekturze systemu, których celem jest obniżenie kosztów oprogramowania oraz podniesienie jego wydajności. W dodatku, wiele funkcji oprogramowania będzie przejętych przez sprzęt /hardware/ lub przez oprogramowanie techniczne /firmware/. Oto niektóre spośród funkcji oprogramowania systemowego, które będą prawdopodobnie przejęte przez sprzęt /które zostaną wmontowane do sprzętu/:

. lokalizacja uszkodzeń lub logika detekcji i korektury błędów,
. makrosy z zakresu zarządzania danymi,
. makrosy z zakresu kompilacji,
. zarządzanie pamięcią wirtualną,
. bezpośrednie redagowanie tekstów,
. funkcje zarządzania środkami,
. funkcje dopasowywania styków,
. makrosy z zakresu zarządzania,
. listy rozkazów specjalnych,
. funkcje wejścia/wyjścia.

Możliwość opracowywania oprogramowania o strukturze modułowej w połączeniu ze sprzętem o strukturze funkcjonalno-modułowej pozwala producentom na wmontowywanie oprogramowania do sprzętu lub do oprogramowania technicznego /mikroprogramowanie/. Największą korzyścią, jaką daje stosowanie oprogramowania technicznego /firmware/ zamiast oprogramowania /software/ jest po-

prawa wydajności, szczególnie wówczas, gdy funkcje oprogramowania:

- związane są z jednostką centralną i nie są zależne od wejścia/wyjścia. Możliwe jest stosowanie mikrokodów dla zwiększenia szybkości działania jednostki centralnej;
- dają wiele wyników pośrednich. Jeśli funkcja oprogramowania generuje wiele wyników pośrednich, które są wykorzystywane dla przetwarzania, lecz nie muszą być zachowane po zakończeniu przetwarzania, to istnieje możliwość przejęcia tej funkcji przez mikrokod. Eliminuje to potrzebę adresowania pamięci i zachowania niepotrzebnych wyników pośrednich;
- są funkcjami często powtarzalnymi. Jeśli jakaś funkcja oprogramowania jest wielokrotnie powtarzana w programie, czy też w wielu programach, to funkcja ta nadaje się idealnie do przekształcenia jej na mikrokod, aby w ten sposób podnieść wydajność przetwarzania;
- nie mogą być wykonywane za pomocą rozkazów. Funkcja, która albo jest trudna do wykonania, albo nie może być wykonana za pomocą istniejącego zestawu rozkazów, może być wbudowana do sprzętu /może być przejęta przez sprzęt/ lub przekształcona w mikrokod;
- wymagają kosztownych rozkazów. Funkcja oprogramowania, która wykorzystuje rozkazy o wysokim stopniu zużycia czasu, może być włączona do mikrokodu dla poprawy wydajności. Rozkazami zużywającymi wiele czasu są te rozkazy, które odwołują się często do pamięci. Wewnętrzna szybkość działania komputera była zawsze wyższa od szybkości działania pamięci głównej.

III. OPROGRAMOWANIE SYSTEMOWE W LATACH 1975-1980

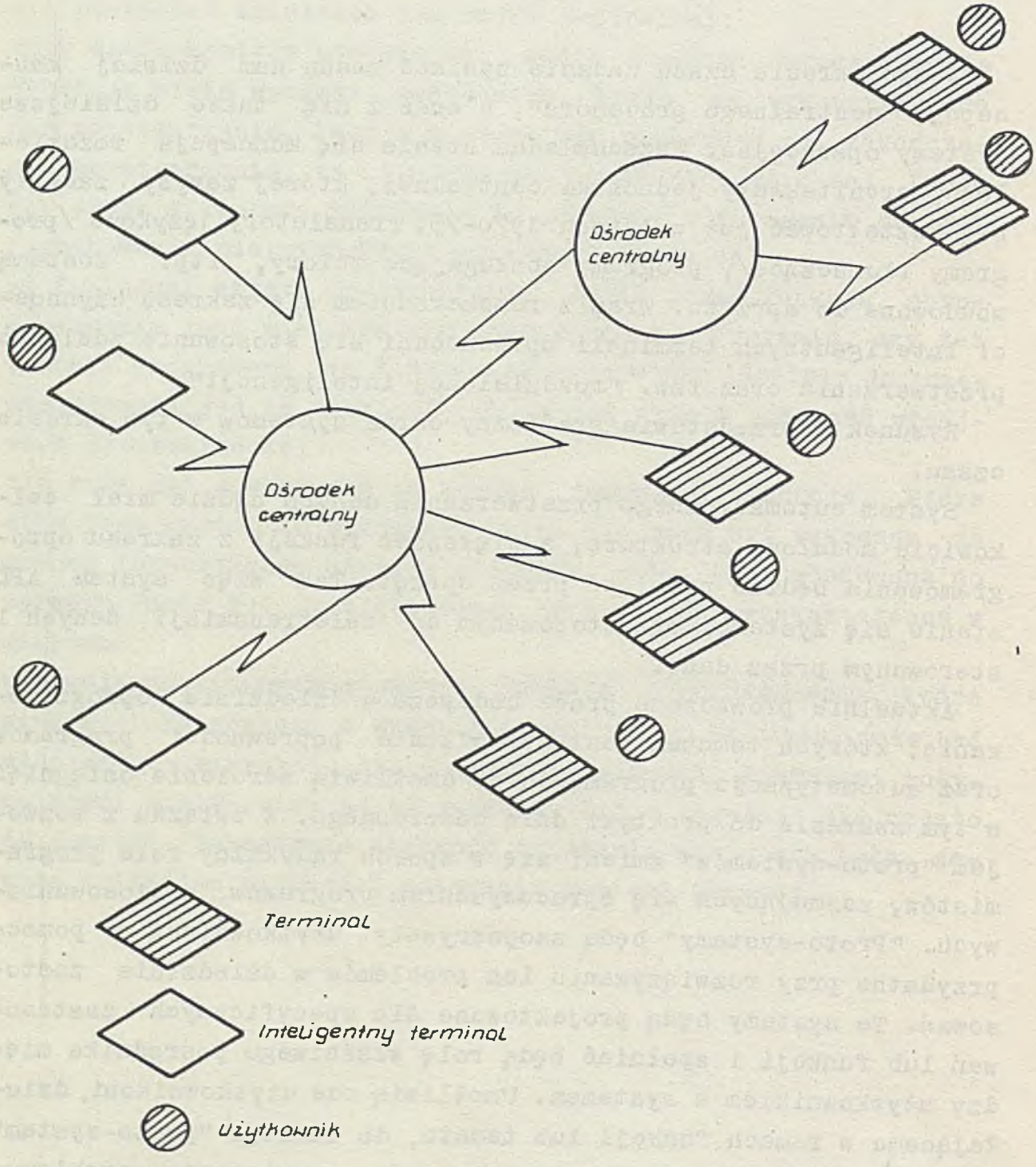
W tym okresie czasu zacznie zanikać znana nam dzisiaj koncepcja "centralnego procesora", a wraz z nią także dzisiejsze systemy operacyjne. Wszechwładna stanie się koncepcja rozdzielonej architektury jednostki centralnej, której zarysy zaczęły się kształtować już w latach 1970-75. Translatory językowe /programy tłumaczące/, programy obsługujące zbiory, itp. zostaną wbudowane do sprzętu. Wraz z rozszerzaniem się zakresu czynności inteligentnych terminali upowszechni się stosowanie zdalnego przetwarzania oraz tzw. "rozdzielonej inteligencji".

Rysunek 3 przedstawia graficzny obraz systemów w tym okresie czasu.

System automatycznego przetwarzania danych będzie miał całkowicie modułową strukturę, a większość funkcji z zakresu oprogramowania będzie przejęta przez sprzęt. Tak więc system APD stanie się systemem przystosowanym do teletransmisji danych i sterowanym przez dane.

Aktualnie prowadzone prace badawcze w dziedzinie oprogramowania, których tematem jest sprawdzanie poprawności programów oraz automatyzacja programowania, umożliwią wdrożenie osiągnięć w tym zakresie do praktyki dnia codziennego. W związku z rozwojem "proto-systemów" zmieni się w sposób radykalny rola programistów, zajmujących się opracowywaniem programów zastosowaniowych. "Proto-systemy" będą zaopatrywały użytkowników w pomoce przydatne przy rozwiązywaniu ich problemów w dziedzinie zastosowań. Te systemy będą projektowane dla specyficznych zastosowań lub funkcji i spełniać będą rolę właściwego pośrednika między użytkownikiem a systemem. Umożliwią one użytkownikowi, działającemu w ramach funkcji lub tematu, do którego "proto-system" jest przystosowany, użycie komputera do rozwiązania problemu, bez formalnej znajomości programowania.

Architektura "rozdzielonej inteligencji" doprowadzi do systemów "fail soft", a także do zdalnej naprawy jednostek, które



RYS. 3 Wiek zdalnych obliczeń

uległy awariom. Powszechne staną się nie tylko pojedyncze komputery o strukturze modułowej, lecz całe grupy komputerów o takiej strukturze będą łączone razem w sieci w tym celu, aby bardziej efektywnie wykorzystywać środki i w razie potrzeby wymieniać dane między systemami.

A. Przejmowanie funkcji oprogramowania przez sprzęt

W poprzednim okresie czasu /1970-75/ obserwowaliśmy rozwój funkcjonalnie rozdzielonej architektury sprzętu komputerowego oraz architektury oprogramowania. Poszczególne systemy komputerowe złożone są z:

. procesorów dla wejścia/wyjścia,
. hierarchii pamięci,
. procesorów dla teletransmisji danych, wstępnie przygotowujących dane,
. procesorów dla arytmetyki i sterowania.

Oprogramowanie systemowe zostało także podzielone na segmenty i rezyduje częściami w różnych funkcjonalnych jednostkach sprzętu.

Kontynuując tę tendencję, okres 1975 do 1980 roku będzie świadkiem rozwoju następujących jednostek:

. procesorów dla obsługi programów tłumaczących /translatorów językowych/,
. procesorów dla obsługi zbiorów,
. procesorów o strukturze modułowej,
. sieci złożonych z procesorów o strukturze modułowej.

Procesory dla obsługi programów tłumaczących

Kiedy tylko program tłumaczący /kompajler/ przekłada wyrażenia opisujące problem na rozkazy maszynowe, to przetwarzanie przebiega w sposób nieefektywny, ponieważ każde wyrażenie napisane w języku programowania wyższego rzędu generuje około dziesięciu operacji wyrażonych w wewnętrznym języku maszyny. Dochodzi do tego jeszcze fakt, że wykonywanie programu nie może się rozpocząć przed zakończeniem jego kompilacji. Dlatego też daw-

nym marzeniem było zaprojektowanie takiej "maszyny", która mogłaby wykonywać bezpośrednio instrukcje wyrażone w języku programowania wyższego rzędu. Pozwoliłoby to bowiem nie tylko na wyeliminowanie nisko efektywnych procesów przetwarzania oraz na podniesienie wydajności w stosunku do konwencjonalnych programów tłumaczących, lecz także na zmniejszenie zapotrzebowania na miejsce w pamięci.

We wczesnych latach siedemdziesiątych, językiem programowania wyższego rzędu, najbardziej powszechnie używanym w dziedzinie przetwarzania danych gospodarczych, był COBOL, za nim następował FORTRAN i PL1. Zachętą do używania języków programowania wyższego rzędu była dążność do zwiększenia wydajności programistów i do zwiększenia przydatności programów do ich rozpowszechniania. W związku z tym, że wielkie środki zostały zainwestowane w istniejące programy pisane w COBOLu, pierwszymi procesorami dla obsługi programów tłumaczących, dostępnymi na rynku, będą procesory dla COBOLu, po nich przyjdą procesory dla FORTRANu i PL1, a także dla całej gamy będących w użyciu programów tłumaczących /translatorów językowych/.

Procesory dla obsługi zbiorów

Równocześnie z procesorami dla obsługi translatorów językowych pojawią się procesory dla obsługi zbiorów, realizujące funkcje zarządzania danymi, które dzisiaj spełniane są przez oprogramowanie systemowe. Większość tych funkcji zarządzania danymi zostanie przejęta przez sprzęt komputerowy oraz oprogramowanie techniczne /hardware/firmware/. Użytkownik nie będzie musiał troszczyć się o organizację swych zbiorów, ani gdzie te zbiory rezydują, ponieważ tymi sprawami zajmować się będzie sam system.

Prekursorem opisanych systemów jest istniejący już obecnie system SYMBOL. Pierwszym celem tego eksperymentalnego systemu było wykazanie, że można wmontować /włączyć/ bezpośrednio do sprzętu komputerowego uniwersalny język programowania wyższego rzędu, a także wyrafinowany system operacyjny, obsługujący wielodostępny system komputerowy i, że dzięki temu, można uzyskać znaczną poprawę w zakresie wydajności systemu. Drugim celem by-

ło doprowadzenie sprzętu komputerowego oraz oprogramowania do takiego rozwoju, aby mogły one okazywać bezpośrednie wsparcie użytkownikowi, gdy tym użytkownikiem jest programista, czy też projektant rozwiązujący problem. Filozofia architektury systemu doprowadziła także do wmontowania do sprzętu komputerowego /do przejęcia przez sprzęt/ pewnej ilości funkcji, które są realizowane przez oprogramowanie w systemach aktualnie stosowanych w dziedzinie gospodarczej. Oto niektóre spośród tych funkcji:

. dynamiczna alokacja pamięci,
. dynamicznie zmieniająca się długość pola oraz zmieniająca się struktura,
. automatyczne kierowanie pamięcią wirtualną,
. automatyczna konwersja typu danych,
. bezpośrednie adresowanie symboliczne,
. bezpośrednia kompilacja realizowana przez sprzęt,
. bezpośrednie redagowanie tekstów.

Językiem wewnętrznym maszyny jest sam język źródłowy wyższego rzędu. Cały system operacyjny komputera SYMBOL zawiera tylko 8000 bajtów, pozostałość wmontowana jest w sprzęt w postaci oddzielnych sześciu modułów przetwarzania. Program dyrygent systemu aktywizuje odpowiednie jednostki przetwarzania, zgodnie z wymaganiami programu.

Działający obecnie system SYMBOL wykazał, że:

. kompajler może być wmontowany w sprzęt komputerowy /sprzęt komputerowy może realizować funkcje kompajlera/,
. większość funkcji systemu sterującego może być wmontowana w sprzęt komputerowy /sprzęt komputerowy może realizować większość funkcji systemu sterującego/,
. zarządzanie danymi może być wmontowane w sprzęt komputerowy /sprzęt komputerowy może realizować funkcje zarządzania danymi/,
. możliwa jest modułowa struktura sprzętu komputerowego oraz oprogramowania,
. wyspecjalizowane systemy wieloprocesorowe, w których poszczególne procesory wykonują określone funkcje, dają w wyniku lepszy stosunek wydajności do kosztów.

Eksperyment ten wykazał także, że przejęcie funkcji oprogramowania przez sprzęt komputerowy daje w wyniku:

. lepszy stosunek efektów do kosztów,
. większą niezawodność,
. wyższą wydajność programistów.

Oprócz SYMBOLu inne systemy eksperymentalne zrealizowane zostały w:
. maszynie SPL firmy CIRAD,
. maszynie 6500 ALGOL firmy BURROUGHS,
. maszynie eksperymentalnej APL firmy IBM.

Opierając się na wymienionych wyżej systemach oraz na wielu innych strukturach minikomputerowych, złożonych z większej ilości jednostek centralnych, prawdopodobne wydaje się, że jeszcze w latach siedemdziesiątych sprzęt komputerowy oraz oprogramowanie rozwiną się w rzeczywiste procesory o strukturze modułowej, które będą z kolei wzajemnie łączone w sieci procesorów o strukturze modułowej.

B. Systemy "Fail Soft"

W tym samym czasie, w którym wiele funkcji oprogramowania zostanie przejętych przez sprzęt komputerowy; będziemy świadkami rozwoju systemów "fail soft". W latach 1970-75 obserwowaliśmy wzrastające stosowanie pomocy służących do detekcji i korektury błędów, a także do diagnostyki oprogramowania. Wraz z rozwojem systemów komputerowych o strukturze modułowej, /które w efekcie zawierają jakiś stopień selektywnej redundancji/ systemy "fail soft" staną się wreszcie rzeczywistością.

System "fail soft" jest tak zbudowany, aby kontynuował wykonywanie swego zadania na niższym poziomie wydajności, mimo wadliwego działania sprzętu, czy to jednostki centralnej, czy urządzenia peryferyjnego. Dla osiągnięcia tego celu proponowane były różne metody, jedną z najbardziej efektywnych okazało się zastosowanie redundancji w systemie. Redundancja istnieje w systemie wówczas, gdy dwie lub więcej jednostek wchodzących w skład systemu jest funkcjonalnie ekwiwalentnych, co oznacza, że każda z nich może stać się urządzeniem zastępczym dla drugiej. Spośród kilku możliwych form redundancji interesują nas w danym

przypadku tylko dwie formy: redundancja masywna oraz redundancja selektywna. Pod pojęciem masywnej redundancji rozumie się system całkowicie zdublowany, w którym oba jego człony realizują to samo zadanie w tym samym czasie, po to aby system kontynuował przetwarzanie na wypadek awarii w którymś z jego członów. Natomiast w przypadku redundancji selektywnej system dysponuje tylko duplikatami wybranych urządzeń. Te duplikaty urządzeń mogą być nieobciążone lub zajęte przy innym zadaniu.

Redundancja selektywna, aby była efektywna, wymaga:

- wyszukanej detekcji błędów oraz diagnostyki, umożliwiającej lokalizację wadliwej jednostki w systemie;
- zdolności do tolerowania bardzo krótkich przerw w przetwarzaniu, które umożliwiłyby izolację wadliwej jednostki, włączenie jednostki redundantnej drogą rekonfiguracji oraz restart systemu;
- małej, wysoce niezawodnej jednostki logicznej, wmontowanej do sprzętu, dzięki której niezbędne procedury powrotu do normalnego działania mogłyby być podejmowane precyzyjnie, zgodnie z wymaganiami.

Aby system "fail soft" działał tak jak został zaprojektowany, średnie załadowanie instalacji zadaniami nie powinno przekraczać jej pojemności, co oznacza, że należy dbać o to, aby w systemie istniał zawsze jakiś "luz" operacyjny. Jednym z możliwych sposobów utrzymywania takiego operacyjnego "luzu" jest odkładanie części załadunku /zadań/ na czas późniejszy, jednakże w takich przypadkach system musi być zdolny do "odrobienia" tej zaległości na czas.

Istnieje minimalne "krytyczne załadowanie systemu zadaniami", które powinno być dotrzymywane, jeśli system ma się mieścić w kategorii systemów "użytecznych". Tak więc, jak długo system, mimo zdarzających się w nim awarii, przetwarza swoje minimum "pracy użytecznej", tak długo można uważać go za system sprawny. Należy wskazać, że systemy "fail soft" są odporne nie tylko na pojedyncze awarie, mogą one stawiać czoła także wielokrotnym awariom i są sprawne tak długo, jak długo czynna jest minimalna konfiguracja, niezbędna do wykonania "krytycznego" zadania.

Wykrycie usterki w jednostce /jednostką taką może być moduł lub submoduł/, wyłączenie tej jednostki z systemu, rekonfiguracja systemu oraz przejęcie zadania przez inne jednostki w systemie /restart/, wszystkie te funkcje muszą być w pełni opanowane, jeśli system ma być systemem typu "fail soft".

Detekcja i diagnoza usterek

Poważnym problemem, który powstaje po skonstruowaniu systemu komputerowego, jest możliwość wykrycia defektu fizycznego w komputerze. Defekt może być przyczyną wadliwego działania komputera i jego awaryjności. Usterka może mieć charakter przemijający lub trwałego błędu. W obu przypadkach niezbędna jest detekcja usterki i ustalenie wadliwie działającej jednostki. Detekcja może być prowadzona na bieżąco, periodycznie lub po zaistnieniu błędu. Wybór jednej z tych metod zależy od charakteru zastosowania.

Systemy "fail soft" wymagają na ogół detekcji prowadzonej na bieżąco. Bieżące wykrywanie usterek stosowane jest przy kontroli działania jednostki w czasie wykonywania przez nią zadania, wymaga to na ogół włączenia do jednostki dodatkowego wyposażenia. Detekcja periodyczna oraz detekcja po zaistnieniu błędu należą do kategorii diagnostyki prowadzonej w układzie on-line, której zadaniem jest ustalenie, czy jednostka działa poprawnie. Jednak niezależnie od sposobu wykrywania błędu przeprowadza się na ogół serię badań diagnostycznych w układzie on-line, po pierwsze dla ustalenia wadliwie działającej większej jednostki, którą należałoby wyłączyć z systemu i po drugie, dla sprecyzowania niesprawnego elementu, który wchodzi w skład tej jednostki.

Dla uzyskania, w tym okresie czasu, sprawnych systemów "fail soft", niezbędne będzie także dodatkowe wyposażenie jednostek w układy logiczne, potrzebne do korektury błędów, obok logicznych układów dla detekcji błędów, które to układy zdolne będą do wyprowadzania tych jednostek z chwilowych warunków niesprawności. W ten sposób będzie poważnie ograniczona częstotliwość wydarzeń, w których wymagana byłaby rekonfiguracja systemu oraz restart. Wśród postawowych przeszkód, które trzeba będzie prze-

zwyciężyć, aby mogły być rozwinięte systemy "fail soft", znajduje się detekcja oraz diagnoza błędów.

Izolacja jednostki

Po wykryciu wadliwie działającej jednostki lub modułu, zachodzi konieczność izolacji z systemu tej niesprawnej jednostki. Izolacja jest złożonym problemem, co wynika chociażby z tego, że mimo wyłączenia awaryjnej jednostki z będącego do dyspozycji zestawu środków nie powinna ona być całkowicie, fizycznie usunięta z systemu; takie usunięcie nie jest nawet wskazane. Łączność z systemem powinna być utrzymana po to, aby można było wykorzystać /właściwą systemowi/ precyzyjną diagnostykę przy reperacji izolowanej jednostki i aby przyspieszyć ostateczny powrót tej jednostki do systemu. Należy jednakże zauważyć, że w tym czasie szwankująca jednostka powinna być w takim stopniu odizolowana od systemu, aby nie mogła wprowadzać do niego ubocznych sygnałów. Co więcej, powinna istnieć także fizyczna możliwość usunięcia jednostki z systemu bez konieczności unieruchomienia całego systemu. W podobny sposób, powinna istnieć możliwość dodawania jednostek do konfiguracji bez zakłócania działalności systemu.

Rekonfiguracja

Rekonfiguracja jest reorganizacją systemu na konfigurację wolną od usterek i zdolną do kontynuowania przetwarzania, z mniejszą ogólną wydajnością /zakładając, że jednostki redundantne były już zajęte przy innych pracach/. W kolejności zdarzeń rekonfiguracja ma miejsce albo równocześnie z izolacją, albo następuje po izolacji.

Istnieją dwie podstawowe okoliczności, w których wykorzystuje się jednostki redundantne; w każdej z nich czynności rekonfiguracyjne są nieco odmienne. Są to okoliczności następujące:
. jednostka redundantna jest bezczynna,
. jednostka redundantna jest zajęta.

W pierwszym przypadku wystarczy włączenie "bezczynnej jednostki" do konfiguracji oraz restart zadania na tej jednostce. Sprawa jest bardziej skomplikowana, jeśli jednostka redundantna

zajęta jest inną pracą. Wówczas system sterowania powinien określić, która z tych prac posiada wyższy priorytet. Jeśli praca wykonywana przez awaryjną jednostkę posiada wyższy priorytet, to rezerwa zostaje przekazana do wykonywania zadania, które zostało zdekompletowane. Może oczywiście zaistnieć sytuacja, w której jednostka "szwankuje", lecz zadanie, które ta jednostka wykonywała posiada niższy priorytet w stosunku do zadania wykonywanego równocześnie przez jednostkę redundantną. W takich przypadkach wszystko co można uczynić, to zawiesić "szwankujące" zadanie.

Problem może być jeszcze bardziej skomplikowany, jeśli niesprawna jednostka jest jednostką "przełączaną", to jest taką, która zależnie od warunków może być użyta w różnych procesorach danego systemu. W takim przypadku procesory powinny wzajemnie porozumieć się i określić, która praca jest najważniejsza. Jeśli jednostka ma być zwolniona od swego dotychczasowego zadania, to powinien zostać ustalony punkt kontrolny, od którego zacznie się przetwarzanie po powrocie do tego zadania. I dopiero teraz może mieć miejsce rekonfiguracja.

Nie wspomniano dotąd o sytuacji, w której nie ma redundancji, albo w której wszystkie redundantne /wzajemnie zastępujące się/ jednostki są niesprawne. Wyrafinowany system posiada w takiej sytuacji dwa warianty postępowania. Jeśli awarii uległo urządzenie wyjścia, na przykład drukarka, to system mógłby przekazać informacje wyjściowe za pomocą procesu "spool"[1/] do urządzenia pamięciowego do późniejszego ich wydruku. Alternatywnie, system mógłby mieć także zdolność rzeczywistej substytucji urządzeń, co oznacza, że system miałby możliwość wzajemnego zastępowania urządzeń podobnych /pamięci dyskowej przez taśmę lub taśmy przez dysk/ w granicach urządzeń pamięci zewnętrznej. Można wysunąć zastrzeżenie, że nie jest to forma rekonfiguracji. Jednakże to postępowanie pozwala na kontynuowanie przetwarzania. Żadna z opisanych wyżej metod nie może być stosowana do urządzenia wejścia, albowiem w tym przypadku konieczne jest fi-

---

1/ Określenie "spool" jest skrótem słów "simultaneous peripheral operations on line"

zyczne zdjęcie nośników informacji wejściowych z urządzenia, które uległo awarii.

Restart

Problem posiadania zdolności do restartu /do ponownego podjęcia przetwarzania po przerwie awaryjnej/ w pełni włączonego do systemu operacyjnego na wszystkich poziomach, tak na poziomie zarządzania programem jak i na poziomie zarządzania danymi, ma w systemie "fail soft" większe nawet znaczenie niż kiedykolwiek przedtem.

Zadaniem zarządzania programem jest dostarczenie użytkownikowi sposobów na ustalanie zaplanowanych punktów kontrolnych[1/] lub, jeśli idzie o systemy bardziej rozwinięte, automatyczne ustalanie punktów kontrolnych. Ustalanie punktu kontrolnego polega jak gdyby na zrobieniu "zdjęcia migawkowego" stanu programu. Może ono być potem wykorzystane do czasowego usunięcia zadania albo do minimalizacji czasu straconego z winy awarii maszyny, albo z winy błędu zewnętrznego. Dalszym zadaniem zarządzania programem jest sterowanie restartem zadania po tym, gdy przerwa w przetwarzaniu stała się przyczyną czasowego usunięcia tego zadania z zespołu zadań.

Zadaniem zarządzania danymi jest utrzymywanie ciągłej świadomości o stanie danych i o potrzebach w tym względzie każdego zadania, a także włączania tych informacji do danych punktu kontrolnego. W czasie restartu /podjęcia pracy nad przerwanym zadaniem/ zarządzanie danymi powinno umiejscowić potrzebne zbiory, tak jak to się dzieje przy normalnym rozpoczynaniu zadania, i powinno "ustawić" te zbiory do dalszego przetwarzania tak, jakby nie było przerwy. Proces ten jest łatwy do zrozumienia, gdy się ma do czynienia z zapisem odcinkowym i ze zbiorami na taśmie. Jednakże, w przypadku zbiorów o dostępie dowolnym problem jest bardziej skomplikowany, ponieważ aktualizacja "miej-

---

1/ Checkpoint - punkt kontrolny lub punkt restartowy, tj. punkt, z którego należy rozpocząć przetwarzanie, przerwane przez awarię, po jego ponownym podjęciu.

sca" została już zwykle przeprowadzona w następstwie ostatniego punktu kontrolnego i należy jej albo "nie przeprowadzać,lub ją ominąć, gdy po restarcie przetwarzania od punktu kontrolnego przebiega się znów przez zbiory wejściowe naprzeciw zbiorowi o dostępie dowolnym.

Transmisja danych jeszcze bardziej komplikuje problem restartu, ponieważ konieczna jest w tym przypadku jakaś forma łączności z każdym z odległych terminali, które brały udział w wykonywaniu zadania, w celu ponownego podjęcia ich aktywności /w punkcie kontrolnym lub za punktem kontrolnym/.

Dalszym problemem komplikującym restart jest ustalanie punktów kontrolnych oraz wznawianie działania samego systemu operacyjnego. System operacyjny, tak jak każdy inny program, narażony jest na niesprawność urządzeń oraz na błędy jednostki centralnej lub pamięci. W tym przypadku restart systemu z punktu kontrolnego oznacza konieczność ponownego startu każdego zadania, wchodzącego w danym czasie w skład zespołu zadań, co prowadzi do wielu powikłań niewątpliwie mało dziś jeszcze zrozumiałych.

Problemy restartu dla zbiorów o dostępie dowolnym, dla urządzeń do teletransmisji danych oraz dla samego systemu operacyjnego nie zostały dotąd zadowalająco rozwiązane, dlatego też zbudowanie efektywnego systemu "fail soft" wymagać będzie jeszcze dużo pracy na tym polu.

Opracowanie systemu "fail soft" w latach 1975-80 stanowi logiczne rozwinięcie zmian wprowadzanych uprzednio do oprogramowania systemowego. Rekapitulując, w latach 1970-75 byliśmy świadkami wzrastającego stosowania:

. układów logicznych dla detekcji i korektury błędów,
. diagnostyki on- line,
. testów lokalizujących awarie.

Powyższe techniki stosowane były w celu usprawnienia niezawodności składników systemu. Wraz z pojawieniem się w latach 1975-80 systemów z procesorem o strukturze modułowej, techniki stosowane do detekcji i korektury błędów, diagnostyki, izolacji jednostek oraz restartu zadań zostaną rozwinięte, aby utworzyć systemy "fail soft". Modułowa architektura procesorów umożliwia

powstanie systemu "fail soft", albowiem jednostki redundantne istnieją w ramach struktury tego systemu.

Systemy "fail soft" usprawnią pracę jak następuje:

- operatorzy nie będą się zajmowali rekonfiguracją systemów,
- diagnostyka niesprawnych jednostek będzie całkowicie zautomatyzowana. Dzięki temu prace konserwacyjne będą mogły być wykonywane przez niżej wykwalifikowany personel, a koszty konserwacji będą niższe.

C. Sprawdzanie poprawności programów

W późnych latach sześćdziesiątych oraz we wczesnych latach siedemdziesiątych troska o niezawodność oprogramowania zastąpiła w dużej mierze troskę o niezawodność sprzętu. Przyczyną zmiany zainteresowań był fakt, że oprogramowanie stało się najdroższą i najbardziej złożoną pozycją w systemie automatycznego przetwarzania danych. Wzrastała złożoność zastosowań oraz systemów operacyjnych, co powodowało, że do opracowywania i do wdrażania projektów tych systemów niezbędne były wielkie zespoły programistów. To z kolei było przyczyną wzrostu kosztów oprogramowania, a także zwiększania się ilości błędów w oprogramowaniu. Błędy powodowały dalszy wzrost kosztów i obniżały dyspozycyjność systemów. Gdy rozwinięto wieloprogramowość oraz systemy wielodostępne, wzrosła ilość użytkowników systemu, komplikując sprawę dodatkowo i wywołując problem zabezpieczenia danych w systemie. Kiedy skoncentrowano uwagę na sprawach oprogramowania, stało się oczywiste, że wydajność oprogramowania oparta jest na względnie słabej podbudowie.

Tak więc, zaistniała nieodparta potrzeba rozwinięcia naukowych metod opracowywania niezawodnego oprogramowania. Aktualnie dokonuje się poważnego wysiłku w celu zautomatyzowania kontroli poprawności programów, która jest kluczowym elementem niezawodności programów. Na przeszkodzie stoi, po części, trudność precyzyjnego zdefiniowania celu programu. Bez takiej definicji niemożliwe jest określenie, czy program spełnia swe zadanie. Nie należy tu oczywiście udowadniać, że sprawdzanie błędów,

prowadzone przez kompilator, wychwytuje wiele błędów oraz, że ilość błędów wychwytywanych przy kompilacji będzie wzrastała wraz z udoskonaleniem analizatorów składni. Pozostałe błędy powinny być wychwytywane w procesie manualnego, skrupulatnie przeprowadzanego, sprawdzania programów. Problemem pokrewnym jest sprawa braku sformalizowanego opisu języków programowania.

Warunkiem rozpowszechnienia się systemów "proto" jest całkowite zautomatyzowanie kontroli poprawności programów. Jednakże "proto-systemy" pojawią się w praktyce i przed spełnieniem powyższego warunku. Chociaż "proto-systemy" omawiane są w zeszycie dotyczącym oprogramowania zastosowaniowego /Dokument Nr E 102M/, wspomina się o nich i tutaj, ponieważ ich rozwój wpłynie na rozwój oprogramowania systemowego.

Automatyczne programowanie

Równolegle, z opisanym wyżej kierunkiem rozwoju, prowadzone będą badania w dziedzinie automatyzacji opracowywania programów. Tego rodzaju system będzie posiadał dużą "świadomość" odnośnie wbudowanego weń sprzętu /możliwości/ i będzie na żądanie przygotowywał określoną odpowiedź. Odpowiedź będzie kształtowana przy pomocy dialogu prowadzonego z systemem w języku pseudonaturalnym /naturalnym dla danej specjalności/. Automatyzacja programowania będzie procesem ewolucyjnym, który wyłoni się dzięki rozpowszechnieniu specjalizowanych, inteligentnych terminali, rozwijających się w warunkach, które są wspólne /specyficzne/ dla języka i użytkownika. Języki specjalizowane są nie tylko łatwiejsze w użyciu, lecz mogą być także bardziej efektywne. Można więc rozpatrywać automatyczne programowanie jako próbę wprowadzenia łatwego w użyciu języka, który umożliwiłby prowadzenie dialogu z systemem APD. Natomiast system APD miałby wbudowaną wiedzę/informację, która pozwoliłaby mu na zadawanie pytań użytkownikowi odnośnie jego specyficznych wymagań oraz na przygotowanie właściwej odpowiedzi.

Można by utrzymywać, że "proto-systemy" i automatyczne programowanie należą do oprogramowania zastosowaniowego /użytkowego/, natomiast kontrola poprawności programów, to pomoc w diagnostyce lub pomoc dla programistów i dlatego należy ona właści-

wie do oprogramowania systemowego. Sprawy klasyfikacyjne nie mają tu jednakże znaczenia. Ważne jest to, że te nowe osiągnięcia zmienią oprogramowanie systemowe niezależnie od tego, czy zostaną one wprowadzone do sprzętu czy do oprogramowania.

IV. OPROGRAMOWANIE SYSTEMOWE W LATACH 1980-1985

W poprzednich okresach czasu stwierdzono, że
- funkcje oprogramowania były przejmowane przez sprzęt,
- powstały systemy procesorów o strukturze modułowej,
- zaczęto tworzyć systemy typu "fail soft",
- zautomatyzowano kontrolę poprawności programów,
- zaczęto kształtować "proto-systemy",
- zaczęto realizować automatyczne programowanie.

Ten okres czasu /1980-1985/ może być nazwany "erą konsolidacji", ponieważ wiedza nagromadzona w okresach poprzedzających będzie użyta do rozwinięcia systemu totalnego. Ten okres czasu będzie świadkiem ewolucji systemu APD w kierunku systemu będącego uzupełnieniem rozumu ludzkiego, z właściwym mu przetwarzaniem informacji.

Sieci procesorów o strukturze modułowej

Celem komputerów jest podbudowanie inteligencji ludzkiej; komputer ma być rozszerzeniem zdolności człowieka do myślenia i rozwiązywania problemów. Cel ten obejmuje inteligencję pojedynczych ludzi jak i inteligencję całego człowieczeństwa. Tak więc, w późnych latach sześćdziesiątych mówiono o systemach MIS oraz o ich zdolności do rozszerzania umiejętności kierownictwa. Systemy MIS nie zostały urzeczywistnione tak szybko, jak się tego spodziewano, lecz komputery mogły być używane do rozwoju bardzo wielkich systemów i do sterowania nimi. Wiele spośród tych systemów postało jako wynik postępu technicznego w dziedzinie teletransmisji oraz syntezy komputerów z teletransmisją.

Rozwój systemów procesorów o strukturze modułowej oraz sieci procesorów o strukturze modułowej w okresie 1975-80 będzie oparty na dostępności odpowiednich kanałów teletransmisyjnych. Rolę mechanizmu sterującego siecią spełnia program - dyrygent sieci, którego główną funkcją jest rozprowadzanie i regulowanie

przepływu zadań przez system. Gdy zadanie wchodzi do systemu, program - dyrygent /program nadzorujący/ sieci ustala jakie centrum komputerowe ma je wykonać. Zadanie zostaje wówczas przydzielone do określonego obszaru lub centrum procesorowego o strukturze modułowej, które z kolei wyznacze niezbędne środki do wykonania danego zadania. Należy zauważyć, że program nadzorujący sieci będzie w większości swej wmontowany do sprzętu.

Szeroko stosowane będą "proto-systemy" i trudno będzie dostrzec różnice między oprogramowaniem systemowym a oprogramowaniem użytkowym. Systemy będą posiadały wbudowane monitory, aby można było śledzić jakie funkcje "proto-systemu" są wybierane przez różnych użytkowników i aby na podstawie tej statystyki można było udzielać porad rozmaitym użytkownikom. Wiele spośród bardziej powszechnie stosowanych "proto-systemów lub modeli sytuacji użytkowników będą realizowane przez logiczne układy pamięci.

Czytelnik przypomina sobie chyba, że pierwsze logiczno-adaptacyjne mikroukłady pamięciowe lub SLAMS'y /Stored Logic Adaptive Micro Circuits/ zostały opracowane na początku 1971 roku w brytyjskim uniwersytecie w Kent w projekcie Minerva. Minerva jest pakietem elektronicznym zawierającym 1000 SLAMSów, zdolnym do zapamiętania i przywołania takiej ilości informacji, jaką mózg ludzki może przyjąć w okresie ponad 50 lat. SLAMSy mogą akceptować dane tak jak każda pamięć komputera, lecz posiadają tę właściwą dla siebie zdolność, że można je nauczyć tego rodzaju odpowiedzi, które powinny być udzielane danym. Przy stosowaniu techniki typu SLAMS koszt "proto-systemów" spadnie, a ich użycie upowszechni się. Sieci poliprocesorowe będą mogły wymieniać informacje na użytek różnych "proto-systemów". W ten sposób system będzie do pewnego stopnia nabywał "inteligencję", tworząc w ten sposób podstawę dla prawdziwego systemu przetwarzania informacji. Okres ten będzie także świadkiem powstawania, w inny sposób, prawdziwych systemów przetwarzania informacji; system zacznie przyswajać sobie dane lub profile każdego z użytkowników. W oparciu o bodźce, które system będzie otrzymywał, czy to w formie danych zewnętrznych,

wprowadzanych przez użytkownika, czy też w formie danych pochodzących z "sensorów", system wezwie niezbędny program dla przetworzenia tych danych. Tak więc system będzie w pełni kierowany przez dane tak, że wydarzenia pochodzące z zewnątrz systemu będą go pobudzały do działania. Te kierowane przez dane systemy będą kombinowane z profilową zdolnością użytkownika tak, aby każdy użytkownik odbierał informacje a nie tylko dane, z których on ma wydobyć informacje.

Wraz z nadejściem całkowicie dojrzałych "proto-systemów" będziemy także świadkami ewolucji przetwarzania języków naturalnych. Będzie to punktem szczytowym na drodze rozwojowej automatycznego programowania, w którym wymiana /dialog/ odbywa się w wyspecjalizowanym, profesjonalnym języku. Pod koniec tego okresu systemy słuchowe rozwiną się do tego stopnia, że będą akceptowały bezpośrednio większość wymawianych słów - nie będą to systemy, które rozpoznają każde angielskie słowo, lecz będą to systemy, które rozpoznają określone zestawy słów. Większość ludzi będzie się uczyła tych zestawów słów jako przedmiotu dodatkowego w ich edukacji zawodowej. Należy do tego dodać, że gdy dialog z systemem zostanie ustalony, użytkownik końcowy będzie posiadał możliwość ustalania zakresu słownika.