TRZECIA WIOSENNA SZKOŁA PTI

KOMPUTEROWE WSPOMAGANIE TWORZENIA SYSTEMÓW INFORMATYCZNYCH

Organizowana przez POLSKIE TOWARZYSTWO INFORMATYCZNE

Do użytku wewnętrznego

Świnoujście, 14-18 maja 1990 r.

SŁOWO WSTĘPNE

Polskie Towarzystwo Informatyczne w ramach prowadzonej działalności statutowej spopularyzowało mało dotychczas wykorzystywaną formułę spotkań - Szkołę. Obecna trzecia edycja Szkoły Wiosennej PTI jest organizowana przez Koło PTI w Szczecinie oraz Sekcję Informatyki Stosowanej w Zarządzaniu. Tradycyjnie Szkoła Wiosenna jest monotematyczna.

Komputerowe wspomaganie tworzenia systemów informatycznych dopiero od kilku lat zdobywa rynki zastosowań informatyki. Pojawiające się pod wspólną nazwa CASE, produkty programowe wspomagające inżynierię systemów oraz inżynierię oprogramowania, wspomagają prace informatyka. Najczęściej są to stosunkowo drogie produkty programowe, stąd obecność ich na polskim rynku należy do wyjątków. Brak prawnej ochrony oprogramowania, o czym tak wiele się mówi w naszym Towarzystwie, zniechęca wielu utalentowanych polskich twórców do angażowania się w wytwarzanie oryginalnych produktów.

Problematyka CASE pojawiła się sygnalnie w Pierwszej i Drugiej Wiosennej Szkole PTI. Dopiero obecna Trzecia edycja Szkoły poświęcona jest w całości tej problematyce. Kompletując wykładowców Szkoły i uzgadniając tematykę referatów, staraliśmy się dotrzeć do większości zajmujących się tą tematyką Kolegów.

W programie Szkoły, jak również w materiałach wyróżnione zostały:
- wykłady, wygłaszane w sesji przedpołudniowej,
- referaty, uzupełniające tematykę w sesji popołudniowej.

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

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

Zdzisław Szyjewski

Szczecin, kwiecień 1990 r.

SPIS TREŚCI

Systemy CASE - problemy, techniki, rozwiązania

Piotr Fuglewicz
40-521 Katowice
Łabędzia 1/2

1. Wprowadzenie

Wspomagana Komputerowo Inżynieria Oprogramowania - Computer Aided Software Engeneering (CASE) jest pewnego rodzaju hasłem reklamowym, pozwalającym sprzedawać bardzo różne produkty informatyczne. Takimi hasłami były w historii informatyki: strukturalność, spolegliwość (ang. human friendliness) czy inne powtarzające się w ogłoszeniach reklamowych cechy proponowanego oprogramowania. Z drugiej strony CASE jest synonimem automatyzacji tworzenia programów, z zastosowaniem metod i technik, będących dorobkiem dotychczasowej praktyki wykonywania systemów informatycznych.

Zakres komputerowego wspomagania inżynierii programowania obejmuje:

- identyfikację obiektu, dla którego należy skonstruować system informatyczny,
- analizę i dekompozycję problemu na składowe odpowiadające elementom tego systemu,
- dobór najwłaściwszych narzędzi i metod realizacji tych składowych,
- syntezę systemu informatycznego.

Od samego początku, a w każdym razie od momentu opracowania pierwszego assemblera, praca programistów była wspomagana komputerowo. Dla tworzenia programów aplikacyjnych używano zawsze oprogramowania narzędziowego. W latach 60-tych i pierwszej połowie lat 70-tych oprogramowanie narzędziowe koncentrowało się przede wszystkim na etapie kodowania programu. Lata te były okresem gwałtownego rozwoju języków programowania. Zaniedbywany był rozwój narzędzi wspomagających inne fazy cyklu życia produktu programowego. Obecnie, dzięki rozwojowi inżynierii programowania, opracowanych zostało wiele różnorodnych narzędzi wspomagających wszystkie fazy cyklu życia produktu programowego. Niestety nie istnieje jeden, ogólnie przyjęty model cyklu życia produktu. Jednym z używanych jest model kaskadowy. Cykl życia projektu programowego składa się w tym modelu z następujących faz [30]:

. Specyfikacja założeń
. Projekt struktury
. Projekt szczegółowy
. Kodowanie
. Testowanie i weryfikacja
. Integracja
. Pielęgnowanie

Model ten dość naturalnie odpowiada działaniom informatyka tworzącego produkt programistyczny. Należy jednak zwrócić uwagę, że rzeczywista praca nie przebiega prawie nigdy w sposób sekwencyjny, i że praktycznie z każdej fazy cyklu może nastąpić powrót do dowolnej z poprzednich. To spostrzeżenie leży u podstaw filozofii CASE będącej próbą stworzenia

spójnej metody identyfikacji obiektów poprzez wykorzystanie komputera jako narzędzia ułatwiającego opisywanie ich struktury oraz odpowiedników tej struktury wyrażonej w kategoriach informatycznych.

Podstawową sprawą w CASE są techniki i metodologie. Istnieje co najmniej kilkanaście metod opisu struktur obiektów i odpowiadających im systemów informatycznych. Jedne wsparte mocną teorią matematyczną, jak np. sieci Petriego, inne bardziej intuicyjne, jak podejście Yourdona. Nie ma jednak w CASE, jak na razie żadnych standardów. Producenci firmują określeniem CASE każdy produkt, który w dowolny sposób automatyzuje prace nad oprogramowaniem. Ambitne zadanie określenia zasad i norm stawiają przed sobą między innymi: Software Engeneering Institute w Carnegie-Mellon University, oraz zwłaszcza CAIM (Center for Advanced Information Menagement - Centrum Zaawansowanego Zarządzania Informacją).

CAIM jest niedochodową organizacją afiliowaną przy Uniwersytecie w Auburn. Celem jej istnienia są badania w zakresie inżynierii oprogramowania. W skład CAIM wchodzą członkowie zarówno ze środowiska akademickiego, jak i z przemysłu. Zadanie CAIM polega na stworzeniu forum dla prac badawczych nad potrzebami i metodami ich zaspokajania w zakresie szeroko rozumianego przetwarzania informacji. Głównymi obszarami zainteresowania tej instytucji są:

. planowanie organizacyjne,
. analiza i projektowanie systemów zarządzania,
. rozwój i utrzymanie systemów informatycznych,
. standaryzacja dokumentacji,
. systemy czasu rzeczywistego,
. zarządzanie zasobami informacyjnymi,
. wspomaganie projektowania.

Długofalowym celem działania CAIM jest sformułowanie międzynarodowych standardów dla różnych aspektów inżynierii oprogramowania. Jego realizacji ma służyć powołanie czasopisma: Journal of Information Engeneering, które ma się stać platformą wymiany informacji między zainteresowanymi tą problematyką. Centrum ma zamiar współdziałać ściśle z ISO (International Standarts Organization) oraz USASI (United States of America Standards Institute) w określaniu standardów dla różnych gałęzi inżynierii oprogramowania.

2. Narzędzia CASE

Dla celów informatyzacji zarządzania powstają liczne programy narzędziowe typu CASE. Można wśród nich wyróżnić dwie klasy:

- pakiety narzędziowe (ang. toolkits),
- pakiety zintegrowane o charakterze zamkniętym (ang. workbenches).

Pakiety narzędziowe tworzone w jednej firmie, jeśli stają się w jakimś zakresie standardem, są rozszerzane przez dodatkowe programy, dostępne u wielu dostawców. Użytkownik musi wybierać między jednolitością stosowanych narzędzi, czemu sprzyja ich zakup u jednego producenta, a różnorodnością udostępnianych środków, czemu z kolei sprzyja urozmaicenie źródeł dostaw. Zintegrowane pakiety CASE zawierają z reguły następujące środki programowe:

- narzędzia specyfikacji i interpretacji opisu systemu,

- generatory struktur baz danych,

- generatory programów wykonawczych,

- programy dokonujące modyfikacji wersji systemu.

Zarówno programy narzędziowe jak i pakiety CASE stosują bardzo szeroko graficzne środki wymiany informacji z użytkownikiem. Większość metodologii stosowanych w CASE opiera się o graficzną prezentację własności projektowanego systemu informatycznego. W systemach zintegrowanych jądrem, wokół którego organizuje się cały system, jest swoista baza danych, nazywana słownikiem danych (ang. data dictionary) lub słownikiem opisu aplikacji. W takiej bazie przechowuje się informacje tekstowe zarówno dotyczące nazw elementów danych i struktury systemu, jak i informacje czysto opisowe.

Zależnie od rozmiarów docelowego systemu informatycznego, stosuje się do jego wyprodukowania komputery różnej mocy: od personalnych, poprzez komputerowe stanowiska robocze (ang. workstations), komputery średniej mocy obliczeniowej (jak np. VAX firmy DEC) po duże komputery (ang. mainframes) i składające się z nich sieci.

Pakiety CASE są dość drogimi narzędziami programowymi, przy ich zakupie należy dokładnie przeanalizować potrzeby i cele stosowania narzędzia, oraz dokładnie zbadać jakie opcje oferuje dany pakiet. Należy sprawdzić czy wyczerpuje on znane potrzeby, a także jakie daje dodatkowe możliwości. Do takiej analizy można zastosować formularze takie jak formularz zaproponowany w artykule [26] i dołączony do niniejszego opracowania.

2.1. CASE w zarządzaniu

Dla potrzeb zarządzania w przemyśle i w handlu istnieje bardzo bogate oprogramowanie narzędziowe typu CASE, używane do wspomagania prac nad rozwojem oprogramowania. Oczywiście oprogramowanie tego typu używane jest również i w innych obszarach, jednak właśnie w tym jest najbardziej rozbudowane, złożone i dzieli się na największą liczbę warstw. W artykule [15] proponowany jest podział na trzy poziomy: niski, średni i wysoki (ang. upper, middle, lower CASE). Podział arbitralny, ale nie pozbawiony podstaw w rzeczywistości. Poszczególne poziomy mają następujące przeznaczenie:

- wysoki, nazywany też komputerowo wspomaganym planowaniem (ang. Computer Aided Planning – CAP), pozwala na przechowywanie w pamięci komputera i analizowanie informacji na temat struktury przedsiębiorstwa, jego celów, warunków osiągania tych celów itd.,

- średni, służący analizie i projektowaniu systemów informatycznych, w szczególności określeniu ich najbardziej efektywnej i odpowiadającej rzeczywistości struktury,

- niski, ukierunkowany na wspomaganie tworzenia programów realizujących funkcje opisane na wyższym poziomie w sposób co najmniej półautomatyczny, tzn. eliminujący konieczność pisania

przez programistę fragmentów kodu realizujących powtarzalne w wielu systemach funkcje,

- i wreszcie pod CASE, czyli całe coraz bogatsze instrumentarium programowe, wspomagające pracę analityka, projektanta i programisty.

Jak widać całe programowanie, czyli to co przeciętny programista zwykł uważać za swoje główne zajęcie, stanowi zaledwie najniższy poziom, w dodatku nie wiadomo na pewno, czy w ogóle wymagający udziału człowieka.

2.1.1. Poziom wysoki

Osoby odpowiedzialne za zarządzanie różnymi przedsiębiorstwami spędzają wiele czasu na próbach zrozumienia wszystkich mechanizmów działania i określaniu planu przedsięwzięć firmy. Plany działania opisują cele, strategie ich osiągania, metody ogniskowania wysiłków na realizacji tych strategii oraz określają zasady oceny efektów realizacji zamierzeń. Do tych założeń dobiera się strukturę administracyjną, wyznacza się terminy realizacyjne i następstwo realizacji w czasie. Wyodrębnia się czynniki determinujące powodzenie, konieczne rozmieszczenie zasobów. Przewiduje się wpływ czynników zewnętrznych i pojawiające się w związku z tym problemy dla przedsiębiorstwa. Opracowuje się scenariusze postępowania rozpatrując sytuacje najgorsze i najlepsze.

Wysoki poziom CASE dostarcza oprogramowania umożliwiającego opis przedsiębiorstwa i jego planów, na ogół z wykorzystaniem diagramów lub innych graficznych metod opisu. Przy użyciu tych metod dokonuje się dekompozycja ogólnego opisu przedsiębiorstwa na coraz bardziej szczegółowy opis jego składowych. Jednym z podstawowych elementów jest opis poszczególnych komórek organizacyjnych, ich funkcji i sposobów działania. Innymi elementami są: cele, odpowiedzialność, zasoby, otoczenie całego przedsiębiorstwa i jego składowych. Oprócz informacji graficznej, związanej ze strukturą analizowanego obiektu, użytkownik oprogramowania musi mieć dostęp do informacji tekstowej, opisującej na ogół własności elementów tej struktury.

Tak stworzone opisy mogą wspomagać tworzenie strategicznych planów działania. Istniejący opis na ogół pozwala na prezentacje zasobów oraz zadań przedsiębiorstwa. Narzędzie z obszaru wysokiego CASE dostarcza opisu struktur, które użytkownik uszczegóławia podając atrybuty takie jak cele, ograniczenia, zasoby itd. Dla różnych planów struktura opisu przedsiębiorstwa pozostaje w zasadzie niezmieniona, zmieniają się natomiast parametry.

Tworzenie takich modeli wymaga sporego nakładu urzędniczej pracy, jednakże w miarę korzystania z systemu okazuje się że raz wprowadzony opis może być wielokrotnie i w różnych celach wykorzystany. Bardzo ważne w systemach wysokiego poziomu CASE jest rozwiązanie sposobu komunikacji z użytkownikiem, zwłaszcza elastyczny i pozwalający na wielokrotne wykorzystanie elementów dostęp do słownika danych. Niektóre z systemów zawierają fragment umożliwiający wykorzystanie znanych od dość dawna metod planowania przedsięwzięć, takich jak metoda ścieżki krytycznej czy schematy Gantta.

Mogło by się wydawać, że systemy z wysokiego poziomu CASE przeznaczone są głównie dla personelu kierowniczego przedsiębiorstwa, w którym je

wdrożono, i że nie są zbytnio przydatne dla osób odpowiedzialnych za tworzenie i utrzymanie systemów informatycznych. Nic bardziej błędnego. Filozofia CASE obejmuje cały cykl życia oprogramowania, niezależnie od tego jak nazwiemy poszczególne etapy tego cyklu. W szczególności CASE pozwala uzyskać wsparcie komputera w najważniejszej fazie tego cyklu: specyfikacji wymagań, projektu struktury, założeń technicznych, czy jak jeszcze można nazwać fazę zapoznawania się projektanta z obiektem. Decyzje podjęte w tej fazie stają się dość prędko nieodwracalne lub odwracalne bardzo dużym kosztem. Niezrozumienie wszystkich zasad działania obiektu musi prowadzić do stworzenia nieprawidłowego systemu informatycznego. Wysoki poziom CASE daje możliwość nie tylko dostępu do pełnego i kompetentnego opisu obiektu, lecz również symulacji zmian niezbędnych do przeprowadzenia przed informatyzacją części lub całej firmy. W dotychczasowej praktyce takich zmian na ogół się nie przeprowadzało, ponieważ zadaniem projektanta było dopasowanie systemu do zastanej rzeczywistości, a nie zmiana tej rzeczywistości.

2.1.2. Poziom średni

Średni poziom CASE wspomaga analizę problemów związanych z przetwarzaniem danych i projektowanie rozwiązań tych problemów. Większość oprogramowania tego poziomu korzysta z graficznych metod projektowania i słownika danych podobnego do tych, które występują na wyższym poziomie. W odróżnieniu od narzędzi wyższego poziomu, programy średniego poziomu wspierają pracę analityka systemów.

Oprogramowanie tego poziomu istotnie skraca czas rozwoju projektu oprogramowania użytkowego, przy czym wiedza, która w przypadku stosowania narzędzi tradycyjnych posiadali tylko członkowie zespołu projektującego system, jest w sposób pełny i systematyczny zapisana w odpowiedniej bazie danych.

Niestety tylko nieznaczna ilość informacji zapisanej w systemie wysokiego poziomu CASE jest użyteczna na poziomie średnim. O ile 'wysoki' CASE zawiera informacje na temat strategii, o tyle w strukturach średniego poziomu możemy przechowywać odpowiedzi na pytania:

- jakie operacje są wykonywane w poszczególnych działach, i dlaczego są to operacje ważne?
- dlaczego operacje wykonuje się w ten a nie inny sposób?
- jakie informacje są potrzebne do wykonania tych operacji, i w jaki sposób się je wykorzystuje?
- dlaczego określone warunki mają wpływ na wykonanie operacji i jakie informacje o tych warunkach są potrzebne i po co?
- kto odpowiada za poszczególne operacje?

Z drugiej strony tylko około 25% wiedzy zapamiętanej w bazach średniego poziomu CASE, jest bezpośrednio przydatne na poziomie niższym, na którym dopiero tworzy się faktyczne specyfikacje programów. Narzędzia średniego poziomu pozwalają określić strukturę systemu informatycznego. Wobec względnie niewielkiego powiązania z sąsiednimi warstwami neguje się niekiedy istnienie średniego poziomu CASE, rozdzielając jego funkcje na narzędzia poziomów niższego i wyższego. Należy jednak pamiętać, że o ile

CASE wysokiego poziomu służy do opisu obiektów rzeczywistych, o tyle 'średni' CASE opisuje odpowiadające tym obiektom abstrakcje - struktury systemów informatycznych.

2.1.3. Poziom niski

Powrót od abstrakcji do konkretu mamy na niskim poziomie CASE, którego narzędzia służą do wspomagania projektowania i tworzenia programów komputerowych. Narzędzia tego poziomu działają najogólniej rzecz biorąc w oparciu o jedno z dwóch podejść. Pierwsze z nich zakłada, że podstawą tworzonej aplikacji jest jej struktura oraz funkcje. W drugim za pierwotne uważa się dane. Szerszą prezentację tych podejść można znaleźć w punkcie 3. niniejszego opracowania.

Osobiście uważam podejście od strony danych za bardziej naturalne i prostsze koncepcyjnie. Pierwszą czynnością przy projektowaniu jakiegokolwiek programu, przy takim podejściu, jest określenie zestawu danych wejściowych, wyjściowych a także, w trakcie projektowania, danych pośrednich programu. Określenie algorytmów działania programu dokonuje się w oparciu o wiedzę o obiektach, na których te algorytmy działają. Duża część fachowej wiedzy projektanta systemu informatycznego składa się z informacji o sposobach i możliwościach wprowadzania, przechowywania oraz przetwarzania danych o obiektach, na których działa program. Również projektant systemu informatycznego podejmuje w oparciu o analizę (a czasem intuicyjnie) decyzje o sposobie agregowania obiektów różnych typów (na przykład pól rekordów w rekord), lub tego samego typu (rekordów w plik bazy danych). Często decyzje podjęte w tym momencie mają zasadniczy wpływ na parametry eksploatacyjne finalnego produktu.

Narzędzia niskiego poziomu CASE pozwalają budować programy aplikacyjne, przy wykorzystaniu środków graficznych, na ogół z wykorzystaniem słownika opisu aplikacji. Kolejność czynności wykonawcy (programisty?), przy tworzeniu takich aplikacji jest następująca:

1) Stworzenie słownika danych. Słownik danych zawiera opis wszystkich danych, struktur i dokumentów z których będzie korzystać aplikacja. Opis danej składa się między innymi z określenia: nazwy, typów (według różnych kryteriów: wejściowa-wewnętrzna-wyjściowa, tekstowa-numeryczna, jedno-wielokrotna, itd.), maksymalnej liczby znaków potrzebnych do wprowadzenia lub wyświetlenia danej, wartości dopuszczalnych (dziedziny), itd.

2) Określenie formatów według jakich dane będą wprowadzane i wyświetlane (formatki ekranowe) i drukowane (raporty).

3) Opisanie w sposób formalny algorytmów przetwarzania danych.

4) Określenie struktury aplikacji (systemu menu).

5) Stworzenie makiety systemu, którą można przedstawić odbiorcy na przykład do ćwiczeń.

6) Generacja dokumentacji projektu i lektura tejże wraz z uwagami z testowania makiety przez użytkownika.

7) Powtarzanie operacji 1-6 w trybie aktualizacji wcześniej

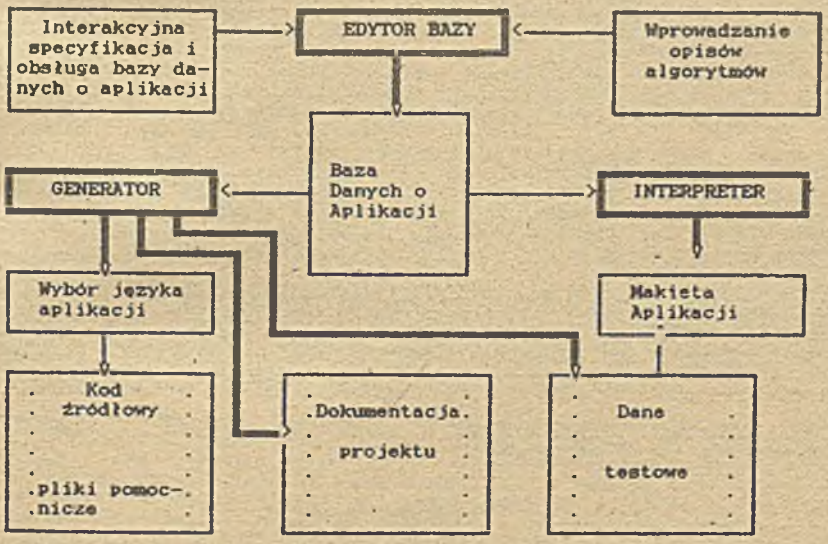
wprowadzanych danych do czasu uzyskania zadawalającej makiety.

8) Generacja kodu aplikacji w języku docelowym oraz plików pomocniczych (na przykład, dla programu typu MAKE, czy innych narzędzi klasy pod CASE).

Takie podejście pozwalała połączyć zalety metody makietowania (patrz p.3.6.) z korzyściami ze stosowania specjalizowanych języków bardzo wysokiego poziomu (fourth generation languge). W wyniku jego stosowania uzyskuje się automatycznie:

- makietę aplikacji,
- znaczny procent kodu źródłowego ostatecznej postaci aplikacji,
- dokumentację konstrukcyjną,
- powstałe w wyniku analizy powiązań pomiędzy danymi oraz układu procedur pliki dla programów systemowych, pozwalające na wygenerowanie optymalnej, według pewnych kryteriów, postaci wykonywalnej aplikacji.

Przykładowa struktura narzędzi programowych dla realizacji takich celów przedstawiona jest na poniższym rysunku.



Centralnym punktem tego rysunku jest Baza Danych o Aplikacji. Opis aplikacji możemy przedstawić jako drzewo, którego gałęziami są kolejne operacje wykonywane przez program, a liśćmi dane wykorzystywane na tych poziomach. Z uwagi na to, że te same dane mogą być użyte w różnych gałęziach, w drzewie pojawiają się powiązania czyniące z niego sieć. Dane do Bazy Danych o Aplikacji wprowadzane są przy użyciu

interakcyjnych możliwości Edytora Bazy, przez osobę pracującą nad aplikacją.

Dla opisu struktur danych i funkcji projektowanej aplikacji stosowane są narzędzia programistyczne, działające w oparciu o techniki i metody, które przedstawimy w kolejnej części opracowania.

3. Metody

Przemysłowa produkcja oprogramowania wymaga stosowania metod i technik projektowania i implementacji systemów przez duże zespoły programistów. Techniki takie, rozwijane równolegle z tworzeniem produktów programowych są sukcesywnie włączane w narzędzia CASE. Poniżej przedstawionych zostanie pobieżnie kilka wybranych metod, nazywanych czasem dumnie metodologiami. Problemem wykraczającym poza zakres niniejszego opracowania, ale równie istotnym, są normy, dotyczące dokumentowania wszystkich faz pracy nad produktem programowym [18]. Dopiero wspólne stosowanie opisanych technik i systematyczne dokumentowanie wszystkich etapów pracy, daje cień szansy na powodzenie naprawdę dużych projektów.

Metodologie w inżynierii programowania powstają niejako przy okazji rozwiązywania problemów konkretnych. Analitycy i programiści rozwiązują problemy, obserwują co robią inne grupy i w trakcie pracy, przechodząc od konkretu do abstrakcji, tworzą swoje metody pracy. Kiedy okazuje się (a ze względu na sposób działania ludzkiego mózgu musi się okazać), że kilka grup stosuje podobne metody, powstaje metodologia, której nazwa pochodzi od nazwiska osoby, która opublikowała opis metody [5, 11, 20, 22, 23, 24, 25, 27, 33]. Inni informatycy czytając taką publikację utwierdzają się w przekonaniu o słuszności swoich metod i, odprężeni, działają jeszcze efektywniej. Również programiści, którzy w ogóle nie czytają literatury fachowej stosują nieświadomie techniki projektowania wstępującego lub zstępującego (albo wariant pośredni) bo inaczej po prostu się nie da. Praktyka dowodzi, że programista na ogół wie że program działa, zanim (o ile w ogóle) uświadomi sobie dlaczego .

Z metodologiami związane są techniki prezentacji informacji. Z uwagi na największą przepustowość wzroku jako kanału informacyjnego u człowieka, większość technik oparta jest o różnego rodzaju diagramy. Inną metodą, szeroko stosowaną jest rysowanie schematów działania programu, struktur danych, przepływu sterowania, tablic przejść stanów itd. Wszystkie te techniki połączone ze sobą stanowią podstawę działania narzędzi CASE.

Poniżej zostanie przedstawionych kilka metodologii i technik, stosowanych w takich narzędziach

3.1. Metoda Warniera/Orra

Opublikowany w 1972 roku, w IBM System Journal artykuł Terry Bakera 'Chief Programmer Team Operation' był pierwszą próbą połączenia kilku stosowanych w praktyce technik i metod: programowania strukturalnego, projektowania zstępującego, zespołu głównego programisty oraz bibliotekarza projektu. Artykuł ten zapoczątkował powszechne stosowanie technik strukturalnego projektowania i tworzenia aplikacji w USA.

W tym czasie Ken Orr, który również stosował techniki strukturalnego

projektowania doszedł do wniosku, że hierarchiczna struktura programów odzwierciedla w dużej mierze hierachię danych tego programu, będących z kolei odpowiednikiem struktury opisywanej rzeczywistości. Równocześnie Jean Warnier przedstawił nie tylko podobne podejście [36], ale również zbudował systematyczną metodologię programowania opartego o struktury danych. Orr w swoich pracach opierał się również o wyniki Michaela Jacksona [19]. Syntezą tych prac stała się technika nazwana DSSD (Data-structured Systems Developement - tworzenia systemów w oparciu o struktury danych), która stanowi podstawę metodologii znanej jako podejście Warniera/Orra [16, 28].

Podejście Warniera/Orra oparte jest o przyjęcie struktury danych jako szkieletu dla budowy programów oraz systemów. Programy budowane są hierarchicznie, w oparciu o ograniczoną liczbę prostych struktur danych. Taka metoda pozwala tworzyć poprawnie działające programy dla całych klas zastosowań, w których struktury danych wejściowych i wyjściowych są identyczne lub podobne. Dla rozwiązywania złożonych, wielopoziomowych problemów następuje ich dekompozycja, w taki sposób, żeby zawsze mieć do czynienia co najwyżej z jednopoziomową relacją n:n (wiele do wielu). Jest to podejście zgodne z tym, co robią od lat matematycy, rozbijając złożone problemy, na elementarne, dla których istnieją jasne, jednoznaczne rozwiązanie.

W przypadku projektowania opartego o struktury danych mamy do czynienia kolejno z transformacją fizycznej struktury danych wejściowych w logiczną strukturę, logicznej struktury danych wejściowych w logiczną strukturę danych wyjściowych, i wreszcie, logicznej struktury danych wyjściowych, w ich strukturę fizyczną. Samo projektowanie odbywa się w tej metodzie w kierunku odwrotnym, to znaczy najpierw określa się fizyczny format danych wyjściowych (na ogół wiadomo, czego żąda się od systemu), aby przez określenie logicznej struktury danych wewnętrznych i ich transformacji dojść do wymogów na fizyczną strukturę wejścia.

W przypadku wykonywania bardziej złożonych systemów okazuje się dość szybko, że główne problemy leżą nie w samym programowaniu, ale na poziomie projektowania systemu. Pojawia się pytanie: jak projektować zbiór programów działających na wspólnych danych? Stosowanie techniki SDDS jest pomocne w próbach odpowiedzi na takie pytanie. Jej stosowanie wspomaga projektowanie struktur bazy danych, określanie wymogów na system, określanie architektury systemu. Na tym poziomie opisywana technika zachowuje swoją podstawową cechę: pracę 'w tył' od wyjścia. Różnica polega na tym, że z fizycznych struktur danych wyjściowych przechodzi się nie do idealnych struktur danych, ale do określenia bazy danych, wspólnej dla całego systemu. Taka baza ma w tej metodzie postać znormalizowanej bazy relacyjnej.

Określenie rezultatów działania programu poprzez zdefiniowanie danych wyjściowych oraz algorytmów stanowi dobrą podstawę do projektowania systemu, ale nie jest wystarczające do sformułowania wymagań. W miarę rozwoju metody tworzenia systemów w oparciu o struktury danych przyjęto określanie najpierw kontekstu, potem funkcji a na końcu rezultatów działania systemu. Techniki wspomagające te działania to: schematy bytów (ang. entity diagram), diagramy Warniera/Orra (assembly-line diagrams), pozwalające określić zależność między funkcjami systemu, logiczne projektowanie danych i inne.

Przykładem narzędzia opartego o tę metodologię jest Design Machine, opracowana w firmie Kena Orra - Optima Inc.; 1300 Woodfield Road;

Schaumburg; IL 60173; USA.

3.2. Metoda Gane/Sarsona

Zaproponowane przez Gana i Sarsona podejście do projektowania systemów nazywane bywa modelowaniem logicznym [14]. Jego celem jest szybkie uzyskanie precyzyjnych definicji elementów systemu z niejasnych i nieprecyzyjnych początkowo informacji o przyszłych danych i funkcjach projektowanego systemu. Istotny wpływ na efektywność metody ma zastosowanie technik graficznych, które pozwalają na przedstawienie struktury danych i sterowania w systemie bez uciekania się do fizycznej implementacji lub makietowania systemu.

W pracy [13] modelowanie logiczne jest przedstawione jako proces, składający się z siedmiu kroków.

Pierwszym krokiem jest opracowanie diagramów przepływu danych (ang. data-flow diagram - DFD). Schemat przepływu danych jest grafem, którego węzłami są:

- byty (ang. entities) np. klienci, dostawcy, działy przedsiębiorstwa,
- składnice danych (w końcowym systemie pliki danych),
- procesy (czynności do wykonania na danych)

Krawędziami grafu łączącymi byty z procesami i składnicami danych są linie ze strzałkami określającymi kierunek przesyłania informacji. Linie są zawsze jednostronnie skierowane. Jeśli przekazywana jest informacja zwrotna używa się kolejnych linii. Informacja przepływa od bytów do procesów, od procesów do składnic informacji, od składnic do procesów i od procesów do bytów.

Przedstawiona technika ma trzy poważne zalety:

- pozwala ściśle określić zakres systemu - byty zewnętrzne (klienci, banki, dostawcy) są z definicji poza zakresem systemu, procesy nie przedstawione na diagramie nie są częścią projektu,
- diagram jest czytelny dla osób nie zajmujących się informatyką, a więc dla większości zleceniodawców,
- w sposób przejrzysty ukazuje zarówno dane systemu, jak i procesy przekształcające te dane, unaoczniając zależności między danymi i procesami.

Drugim krokiem jest zestawienie listy elementów danych, wraz z informacją, w którym magazynie danych dana powinna być przechowywana. Staranne wykonanie tego kroku pozwala uniknąć redundancji danych i zagregować dane powiązane ze sobą w jednym miejscu.

Trzecim krokiem jest przeprowadzenie analizy relacji pomiędzy danymi i uszczegółowienie struktur logicznych baz danych.

W czwartym kroku dane przedstawia się w postaci połączonych, dwuwymiarowych tablic i określa się klucze, według których powinien

odbywać się dostęp do wierszy tych tablic. Podobnie jak w poprzedniej metodzie wykorzystuje się relacyjny model danych.

Piątym krokiem jest aktualizacja schematu przepływu danych, w oparciu o znormalizowane relacje, uzyskane w kroku czwartym. Ten krok jest faktycznie ostatnim krokiem modelowania logicznego. Dwa pozostałe służą już projektowaniu systemu informatycznego, działającego na wyspecyfikowanych danych.

Szóstym krokiem jest wydzielenie procesów i związanych z nimi danych, tak aby można było określić jednostki proceduralne projektowanego systemu. W tym etapie należy odpowiedzieć na pytania:

- kiedy wykonywana jest procedura?
- jaki obszar diagramu przepływu danych jest zaangażowany w jej realizacji?
- czy ten obszar może być zrealizowany przy użyciu jednej procedury, a jeśli nie to dlaczego?

Siódmym krokiem jest fizyczne projektowanie samych procedur, wspierane sformalizowanymi informacjami zebranymi w krokach poprzednich.

Współtwórca metody - Chris Gane, jest konsultantem firmy Bachman Information Systems; Four Cambrige Center; Cambrige MA 02142; USA.

3.3. Podejście relacyjno-bytowe

Jednym z głównych czynników opóźniających wdrażanie zintegrowanych systemów informatycznych jest to, że na ogół systemy takie projektowane i wykonywane są stopniowo, a nie w sposób globalny dla danego przedsiębiorstwa. Stosowanie zintegrowanej bazy danej, wokół której buduje się kolejne podsystemy jest jakimś rozwiązaniem, z tym, że sam system bazy danych nie zapewnia jeszcze poprawnego porządkowania danych. Metoda relacji bytów (ang. entity-relationship approach) jest techniką pozwalającą, w sposób systematyczny, przetwarzać wymagania użytkownika w schematy prawidłowo zaprojektowanych baz danych [6, 9, 21, 31, 32].

Dostęp do elementów danych, zapisanych w bazie odbywa się na ogół poprzez nazwę. Dość często przy tworzeniu systemu informatycznego mamy do czynienia z przypadkiem rozbudowy systemu, w oparciu o już istniejące bazy danych. W sytuacji tworzenia systemów działających na podobnych danych, przez więcej niż jeden zespół projektantów, pojawia się niebezpieczeństwo wystąpienia następujących problemów:

- różne nazwy dla tego samego fizycznie elementu danych,
- te same nazwy dla różnych elementów danych,
- niezgodność formatów danych,
- redundancja danych,
- nieprawidłowości aktualizacji.

Równocześnie zawsze występuje problem liczby typów rekordów (w przypadku modelu relacyjnego - relacji), jaki powinny wystąpić w bazie. Nie zawsze oczywiste jest które pole lub kombinacja pól winny być kluczem. Metoda

relacji bytów (stosowana nieformalnie i nieświadomie przez praktycznie każdego projektującego struktury danych) jest obecnie najpopularniejszą techniką projektowania takich relacji.

Graficzną techniką wspomagającą projektowanie jest w tej metodzie Diagram relacji bytów (ang. Entity-Relationship Diagram - ERD). Byty w tym diagramie reprezentują obiekty, organizacje, zdarzenia i inne elementy opisu rzeczywistości, które mają być przechowywane w bazie systemu. Byty połączone są liniami zaopatrzonymi w strzałki. W tym schemacie strzałki oznaczają relacje między bytami. Relacje te mogą być dwustronne. Musi istnieć graficzny wyróżnik, informujący czy relacja jest jedno- czy wielokrotna.

Analiza diagramu relacji bytów pozwala na opisanie zawartości bazy w kategoriach relacyjnego modelu baz danych. Przykładowo, strukturę danych przedsiębiorstwa tworzącego programy można opisać następująco:

```
PRACOWNIA(ID_PRACOWNI,OBŁOZENIE)
PROGRAMISTA(ID_PROGRAMISTY,NAZWISKO,WIEK,ID_PRACOWNI)
PROJEKT(NR_PROJEKTU,NAZWA_PROJEKTU)
PRACUJE_W(ID_PROGRAMISTY,NR_PROJEKTU,%CZASU)
```

Słowa przed nawiasami odpowiadają albo bytom (PRACOWNIA, PROGRAMISTA, PROJEKT) albo relacjom (PRACUJE_W). Słowa w nawiasach są atrybutami, pozwalającymi tworzyć klucze, umożliwiające dostęp do danych. Z punktu widzenia teorii relacyjnych baz danych byty są tu również relacjami. Wszystkie relacje występują w trzeciej postaci normalnej.

Trzecim krokiem jest opracowanie aplikacji, z wykorzystaniem narzędzi zbliżonych do języka zapytań systemowych (ang. System Query Language - SQL). SQL jest narzędziem związanym z relacyjnymi bazami danych, natomiast metodę ER można stosować dla różnych modeli bazy danych. W przypadku jeśli mamy do czynienia z innym modelem, wykorzystywane mogą być metody zapytań, właściwe dla danego modelu.

Istnieje norma ANSI [2], dotycząca słowników zasobów informacyjnych oparta o relacyjny model bazy danych i diagramy relacji bytów.

3.4. Metoda Yourdona

Metoda Yourdona jest efektem dwudziestoletnich doświadczeń wielu ludzi współpracujących z Edwardem Yourdonem, między innymi w firmie Yourdon Inc. Techniki składające się na tę metodę są nazywane: strukturalną analizą, strukturalnym projektowaniem i strukturalnym programowaniem. Metodologia Yourdona stale ewoluuje, pod wpływem idei innych informatyków, poczynając od modnych w latach siedemdziesiątych technik strukturalnych, poprzez ciągle rozwijane techniki diagramów po elementy podejścia obiektowego [17, 29, 35, 37, 38].

Metodologia Yourdona składa się z dwóch części: narzędzi i technik. Narzędziami są różnego rodzaju diagramy graficzne, używane do przyrostowego modelowania wymogów i architektury projektowanych systemów.

Najpopularniejszym narzędziem jest diagram przepływu danych (DFD), opisany wcześniej, przy okazji omawiania metody Gane/Sarsona. O ile diagram przepływu danych jest doskonałym narzędziem opisu funkcji

systemu, o tyle niewiele mówi on o zależnościach między danymi i uwarunkowaniach czasowych systemu. Dla uwzględnienia tych czynników stosuje się w metodzie Yourdona diagramy relacji bytów (ERD) oraz diagramy zmian stanów (ang. state-transition diagram - STD). Różnego rodzaju diagramy dostarczają czytelnej i bogatej wiedzy o projektowanym systemie, jednak wiedza ta nie jest pełna. Dla kompletnego opisu konieczna jest również informacja tekstowa, między innymi o strukturach elementów danych oraz o działaniu procesów, opisywanych w DFD pojedyńczym bloczkiem.

O ile opisane narzędzia ułatwiają istotnie analizę systemu, o tyle techniki zaproponowane przez Yourdona mają ułatwiać proces tworzenia aplikacji. Podstawową techniką jest rozdział zdarzeń (ang. event partitioning). Polega ona na stworzeniu diagramu kontekstu (ang. context diagram) wysokiego poziomu. Diagram taki służy do określenia kontekstu systemu, jego granic oraz wzajemnych oddziaływań ze światem zewnętrznym. Następnie tworzy się wykaz zdarzeń, które zachodzą w otoczeniu systemu, a na które system powinien reagować. Praktycznie takie zdarzenia są na ogół transakcjami bazy danych systemu. Na podstawie wykazu opracowuje się diagram przepływu danych. Z kolei diagram ten redukuje się, łącząc na przykład zdarzenia, które powodują korzystanie z takich samych danych. Redukcja diagramu prowadzi w końcu do określenia fizycznej struktury aplikacji.

3.5. Metody tworzenia systemów czasu rzeczywistego

W przypadku stosowania technik CASE w systemach czasu rzeczywistego mamy do czynienia z problemami o mniejszym na ogół statycznym stopniu złożoności, natomiast bardzo skomplikowanymi w związku z wystąpieniem czasu, jako podstawowego parametru w systemie. Istnieje kilka sposobów podejścia do tworzenia systemów czasu rzeczywistego [12].

Pierwszy z nich każe traktować system, jako jeden wielki proces, ulegający w trakcie pracy dekompozycji na coraz mniejsze podprocesy. Takie podejście określa się mianem funkcjonalnej dekompozycji. Z drugiej strony znane podprocesy podstawowe mogą być kompletowane w większe. Wynikowy model może być przedstawiony jako hierarchia diagramów przepływu danych, na przykład w rodzaju zaproponowanych przez DeMarco [10]. Takie podejście od strony hierarchii funkcji, zostało rozwinięte dla systemów czasu rzeczywistego między innymi przez Hatley'a. Pierwotne dla określenia struktury systemu informatycznego są procesy, potem zależności między nimi, a w końcu dane. U podstaw tej metody leżą dobrze opracowane techniki projektowania i programowania strukturalnego.

Diametralnie różne podejście zaproponowali w 1984 Ward i Mellor [35]. System jest przez nich widziany jako 'czarna skrzynka', która dostarcza ustalonych odpowiedzi na zewnętrzne sygnały. Projektant systemu definiuje zdarzenia, które zachodzą na zewnątrz systemu (zewnętrzne pobudzenia) i określa odpowiedzi, definiując oddzielne procesy, odpowiedzialne za wywołanie tych reakcji. W tym podejściu pierwotne są zdarzenia zewnętrzne, procesy są efektem zajścia tych zdarzeń, natomiast dane procesów są całkowicie pochodne w stosunku do funkcji procesu. Zaletą podejścia Ward/Mellor'a jest szybkie przejście osoby realizującej system do określania jego działań. Kiedy już istnieje model, stosunkowo prostą rzeczą jest jego rozbudowa o nowe zdarzenia, lub zmiana reakcji na już zdefiniowane zdarzenia. Podstawową wadą jest duża podatność innych procesów na zmiany w jednym - wprowadzenie nowego procesu może

nieść konieczność zmian w danych i działaniach innych.

W trzecim wreszcie podejściu, stosowanym w produktach firmy Project Technology z Berkeley, świat jest traktowany jako nieograniczony i wypełniony obiektami. Reguły określające zależności między obiektami są ściśle określone. Każdy obiekt i każda relacja, w której obiekt bierze udział, mają określony czas życia. Formalizacją cykli życia obiektów są tablice przejść stanów, które dokładnie opisują stany, w których mogą znajdować się obiekty, oraz zdarzenia, które powodują zmianę stanu obiektu. Definiowanie aplikacji zaczyna się od określenia danych, w oderwaniu od ich fizycznego znaczenia. Z kolei definiuje się sterowanie poprzez odwzorowanie fizycznych własności obiektu. Procesy definiowane są automatycznie na podstawie dwóch poprzednich kroków. Takie podejście pozwala na prostą rozszerzenie liczby obiektów lub procesów, bez zaburzeń w innych częściach systemu. W systemie projektowanym, w oparciu o metody obiektowe, nie jest sprawą prostą i oczywistą precyzyjne określenie co dzieje się, kiedy system poddany jest ekstremalnym i złożonym obciążeniom. W odróżnieniu od podejścia poprzedniego, zachowanie 'czarnych skrzynek' nie jest opisane w jednym miejscu projektowanego systemu.

Innym narzędziem, coraz częściej wykorzystywanym przy projektowaniu systemów czasu rzeczywistego jest SDL (Specification and Description Language), będący standartowym językiem specyfikacji i opisu systemów. Został opracowany i poddany standaryzacji przez CCITT (International Telegraph and Telephone Consultative Comitee). Rozpoczęte w 1972 roku prace zakończyły się, jak na razie, przyjęciem w 1987 roku wersji SDL88. Początkowo SDL stosowany był do opisu systemów telekomunikacyjnych. Obecnie znajduje coraz szersze zastosowanie we wspomaganiu dowolnych systemów czasu rzeczywistego. Przykładem narzędzia opartego o język SDL, wykorzystującego wszystkie możliwości współczesnych systemów CASE, jest program SDT szwedzkiej firmy TeleLOGIC.

3.6. Makietowanie

Makietowanie (ang. rapid prototyping) [1, 3, 4, 7, 8, 34] nie jest oddzielną metodologią, a raczej sposobem względnie szybkiego uzyskiwania działającego modelu aplikacji. W odróżnieniu od proponowanego na przykład przez Gane'a modelowania działania aplikacji, makietowanie związane jest z generacja lub z interpretacją opisu elementów aplikacji, szczególnie dotyczących interakcji programu z użytkownikiem. Programy stosujące technikę makietowania pozwalają modyfikować opisy elementów interakcji (menu, formatki, podpowiedzi) i natychmiast przedstawić efekt tego działania na ekranie. Taka technika pozwala na udział końcowego użytkownika w procesie tworzenia aplikacji, bez konieczności przygotowywania na papierze projektów ekranów, raportów, menu. Technika makietowania jest stosowana w wielu narzędziach korzystających z opisanych metodologii.

W pewnym sensie narzędzia używane do makietowania są podobne do tzw. języków czwartej generacji, w których obok normalnych elementów języka programowania, wbudowane są silne narzędzia dostępu do informacji w bazach danych. Podobnie narzędzia wspomagające makietowanie są swego rodzaju interpreterami języków opisu interakcji z użytkownikiem. Po uzyskaniu właściwej postaci aplikacji, generowany jest pseudokod lub wręcz programu w języku programowania, który następnie może być, w różnym stopniu - zależnym od możliwości narzędzia makietującego -

uzupełniany ręcznie o fragmenty dotyczące przetwarzania, które to fragmenty z kolei mogą pochodzić z innych narzędzi generacji kodu.

4. Przykłady narzędzi CASE

Liczba narzędzi, zaliczanych do kategorii CASE wynosi już w świecie co najmniej kilkaset i szybko rośnie. Dobry przegląd tych narzędzi zawierają, dostępne w Polsce, publikacje [12, 26] . Poniżej omówione zostaną cztery programy, będące pierwszym krokiem w stronę 'polskiego CASE'. Dwa z nich oparte są o język zgodny z dBase III, dwa kolejne o własne biblioteki metod dostępu.

4.1. TurboGen

Generator aplikacji TurboGen został zaprogramowany w oparciu o system FoxBASE i powstał w zespole programistów PKiWSI "WEKTOR" w Warszawie. Projektanci generatora mieli na uwadze dwa podstawowe cele:

- stworzenie narzędzia umożliwiającego generowanie w krótkim czasie dowolnego (w określonej klasie) systemu użytkowego;
- zapewnienie możliwie dużej pewności działania systemów użytkowych przez budowę tych systemów na bazie stałych, dobrze przetestowanych modułów biblioteki generatora.

Generator składa się z następujących modułów:

* moduł tworzenia i modyfikacji struktury systemu użytkowego w zakresie:
  - specyfikacji baz danych, ich struktur, budowy i wzajemnego powiązania;
  - określenia sposobu kontroli poprawności formalnej i merytorycznej pól przy wprowadzaniu danych;
  - tworzenia i inicjacji stałych systemowych;
* moduł budowy i modyfikacji "menu" systemu użytkowego;
* moduł obsługi fizycznych baz danych w zakresie:
  - dodawania rekordów
  - przeglądania
  - korekty
  - usuwania

* moduł obsługi logicznych baz danych w zakresie:

  - organizacji ekranu (położenie pól rekordu logicznego na ekranie)
  - dodawania rekordów
  - przeglądania
  - korekty
  - usuwania

  logiczna baza danych jest jedną bazą danych z punktu widzenia użytkownika, chociaż może składać się z kilku fizycznych logicznie powiązanych baz

* moduł obsługi informacyjnej systemu użytkowego tzw. "help" systemowy;

* moduł generowania różnego rodzaju raportów systemu, z możliwością agregacji danych na różnych poziomach;

* moduł zarządzania systemu w zakresie:

  - kopiowania baz danych
  - bieżącej kontroli nad bazami, które uległy zmianie w czasie sesji systemu
  - zabezpieczenia wykonywania poszczególnych operacji z menu systemu poprzez rozbudowany system haseł.

W generatorze, oprócz typów pól charakterystycznych dla standardu dBASE wyróżniono jako dodatkowy typ pole atrybutowe. Pole to różni się od pozostałych sposobem wprowadzania danych, a mianowicie jest to pole, które określa atrybut, mogący przybierać wartości ze skończonego zbioru, przy czym każdej wartości przypisana jest określona wartość atrybutu. Wprowadzanie danych odbywa się poprzez wybór zorganizowany w postaci "menu".

Generator dzieli bazy danych na dwie rozłączne klasy. Do pierwszej należą tzw. bazy pojedyncze, do drugiej tzw. bazy logiczne. Baza pojedyncza jest bazą, która fizycznie reprezentuje jeden plik. Baza logiczna składa się z co najmniej dwóch baz pojedynczych związanych ze sobą logicznie poprzez określone wspólne pola. Baza logiczna umożliwia budowę hierarchicznych struktur baz danych.

Menu główne generatora ma następującą postać:

```
1.  Specyfikacja baz danych systemu
2.  Definiowanie wszystkich pól wszystkich baz systemu
3.  Definiowanie treści menu pól atrybutowych
4.  Określenie struktur baz danych
5.  Zakładanie fizycznych plików baz danych
6.  Specyfikacja baz logicznych
7.  Automatyczna budowa programów obsługi baz danych
8.  Budowa ekranów dla baz logicznych
9.  Budowa menu systemu
10. Ustalenie położenia poszczególnych tablic (okienek) menu
11. Tworzenie dodatkowych plików indeksowych
12. Budowa raportów
```

4.2. NanoX

Program Tworzenia Dialogowych Systemów Przetwarzania Danych Nanox został zaprogramowany w oparciu o system Clipper i powstał w zespole programistów PTH BISTER w Katowicach. NanoX jest narzędziem projektanta i programisty tworzących dialogowy system przetwarzania danych w języku Clipper '87. Wspomaganie pracy przez ten program obejmuje:

- definiowanie dialogu tworzonej aplikacji. Przyjęto, że aplikacja będzie miała dialog realizowany metodą hierarchicznych menu. W trakcie generacji dialogu kształt i efekty tego dialogu są symulowane,

- definiowanie nazw funkcji systemu (opisywanych w menu) oraz ich atrybutów tj.: typu, poziomu ochrony, oraz nazw procedur kontrolnych,

  - określanie mechanizmu i treści podpowiedzi systemu aplikacyjnego,

  - projektowanie w sposób dialogowy, z natychmiastową symulacją ekranów - formatek służących do wyświetlania informacji i pobierania danych od użytkownika aplikacji oraz wydawnictw - raportów prezentujących dane systemu,

  - udostępnianie gotowych, standardowych funkcji systemów przetwarzania danych (edycji danych, prezentacji danych, archiwacji danych),

  - udostępnianie typowych elementów oprogramowania zebranych w bibliotekę procedur, szczególnie mechanizmów:

    . słownikowego wspomagania edycji pól kodowanych,
    . kalkulatora wspomagającego edycję pól numerycznych,

- uwzględnienia w indeksowaniu właściwości dowolnie zdefiniowanego alfabetu.

- automatyczne tworzenie dokumentacji technicznej i użytkowej systemu,

- nadzór nad scalaniem systemu.

Efektem użycia programu NanoX jest zapis danych o projektowanym systemie aplikacyjnym w postaci Opisu Aplikacji. Pozwala to na:

- symulację, pod kontrolą programu NanoX, pracy systemu aplikacyjnego w trakcie jego projektowania,

- tworzenie makiety systemu aplikacyjnego. Makieta realizuje wszystkie funkcje już zdefiniowane przez projektanta oraz symuluje wykonanie funkcji, których definicji jeszcze nie ukończono,

- generację ostatecznej postaci systemu aplikacyjnego,

- automatyczne tworzenie dokumentacji technicznej systemu aplikacyjnego.

Proces tworzenia aplikacji nie musi kończyć się w momencie jej wygenerowania. W dowolnej chwili (o ile przechowany zostanie Opis Aplikacji) można powtórzyć wszystkie fazy projektu dopasowując aplikację do zmieniających się warunków jej stosowania. Dodatkową zaletą takiego rozwiązania jest możliwość użycia fragmentów gotowych systemów aplikacyjnych do generowania kolejnych aplikacji (o ile te systemy zostały stworzone przy pomocy programu NanoX).

NanoX został pomyślany jako model systemu wspomagającego tworzenie aplikacji. Przyjęto że zakres tworzonych przez ten model aplikacji ogranicza się do dialogowych systemów gromadzenia i przetwarzania danych. Stosunkowo proste struktury danych, których obsługę umożliwia Clipper występują na tyle często, że zakres stosowania narzędzia jest dość szeroki.

Wyodrębniono z funkcji systemów użytkowych te, które występują najczęściej w podobnej postaci:

- dialog z użytkownikiem,
- edycję i prezentację danych zapisanych w systemie,
- przetwarzanie danych,
- zabezpieczenia funkcji (i danych) przed niepowołanym dostępem i utratą.

Przetwarzanie uznano za najbardziej indywidualną część każdego z systemów i przyjęto, że nie będzie wchodziła w zakres wspomagania tym narzędziem.

4.3. mixPACK

Biblioteka Relacyjnej Bazy Danych mixBASE i Generator Aplikacji mixUTIL są produktami PTH "KaNet" z Katowic. Biblioteka mixBASE zawiera zestaw procedur realizujących funkcje relacyjnej bazy danych. Właściwości bazy mixBASE ująć można najogólniej w następujące grupy:

- mixBASE umożliwia tworzenie baz relacyjnych. Wszystkie dane przechowywane są w relacjach pamiętanych w postaci indeksowo-sekwencyjnych plików dyskowych. Jednocześnie właściwości mixBASE w tym zakresie wykraczają poza możliwości tradycyjnych baz relacyjnych dzięki udostępnieniu programiście struktury danych (nazywanej "plikiem wirtualnym"), łączącej informacje zawarte w wielu różnych relacjach, oraz dopuszczeniu wszystkich operacji (przeglądanie, dodawanie, modyfikacja i kasowanie) na takiej konstrukcji logicznej.

- Słownik Opisu Aplikacji, obsługiwany przez wewnętrzne mechanizmy oprogramowania bazy, składa się z następujących słowników:

  - słownik pól systemowych, zawierający opisy pól występujących w

relacjach (plikach dyskowych) oraz pól pamięciowych,

- słownik relacji (plików dyskowych), opisujący zawartości i sposoby udostępniania informacji przechowywanych w plikach fizycznych dla potrzeb aplikacji,

- słownik plików wirtualnych, definiujący zawartości i sposoby udostępniania agregatów danych, podlegających faktycznemu przetwarzaniu w aplikacji,

- słownik formatek ekranowych, zawierający opisy sposobów wizualizacji konstrukcji zdefiniowanych w słowniku plików wirtualnych,

- słownik menu sterowania aplikacją, opisujący wygląd i działania wszystkich menu aplikacji,

- słownik formatów raportów, opisujący sposoby drukowania informacji udostępnianych przez konstrukcje zdefiniowane w słowniku plików wirtualnych,

- słownik opisu struktury aplikacji, zawierający informacje na temat powiązań pomiędzy funkcjami danej aplikacji, plikami wirtualnymi i menu sterowania,

- słownik operatorów (użytkowników) aplikacji, opisujący system autoryzacji dostępu do poszczególnych funkcji danej aplikacji.

Struktury danych zdefiniowane w słownikach w dużej mierze są niezależne od programu realizującego aplikację, tzn. modyfikacja np. jednej z formatek ekranowych nie powoduje konieczności rekompilacji programu.

Aplikacje zbudowane w oparciu o mixBASE, wykorzystują następujące cechy biblioteki:

- mixBASE zawiera rozbudowany system podpowiedzi, zarówno tekstowych jak i konstruowanych w oparciu o informacje zawarte w bazie danych,

- każda aplikacja zrealizowana w standardzie mixBASE posiada wbudowane elementy języka zapytań o właściwościach zbliżonych do QUERY BY EXAMPLE,

- komunikacja dowolnej aplikacji wykonanej w standardzie mixBASE z użytkownikiem realizowana jest zawsze przy pomocy stałego zestawu klawiszy funkcyjnych (tzn. określony klawisz w każdej aplikacji powoduje wykonanie zawsze takiej samej akcji) oraz z wykorzystaniem techniki rozwijanych menu i okien,

- mixBASE jest z założenia bazą przeznaczoną do pracy w systemach wielodostępnych oraz sieciowych (wbudowane mechanizmy blokady dostępu do pliku wirtualnego - a więc grupy plików fizycznych, oraz rozwiązywanie konfliktów przy próbie modyfikacji rekordu wirtualnego - a więc grupy powiązanych ze sobą rekordów fizycznych),

- mixBASE nie zawiera istotnych ograniczeń konstrukcyjnych (możliwość jednoczesnego otwarcia 127 plików fizycznych, maksymalna długość rekordu 64 kB) pozwalając na tworzenie bardzo dużych aplikacji,

- istnieje możliwość śledzenia użytkowania aplikacji (journal), z wykorzystaniem informacji z przebiegu śledzenia do ewentualnego odtworzenia zawartości bazy w oparciu o ostatnią kopię archiwalną.

Podane w powyższym zestawieniu istotne właściwości funkcjonalne aplikacji wykonanych w standardzie mixBASE, takie jak:

- język zapytań,
- rozbudowany system podpowiedzi,
- standardowy sposób komunikowania się z użytkownikiem,
- rozwiązywanie konfliktów modyfikacji tych samych danych przez różnych użytkowników,

są realizowane wewnętrznie przez procedury biblioteki mixBASE w oparciu o dane zawarte w słownikach i nie wymagają od programisty aplikacji żadnego nakładu pracy.

Poza omówioną biblioteką procedur relacyjnej bazy danych, w skład pakietu mixBASE wchodzą standardowe programy wspomagające, dołączane do każdej aplikacji:

mixVIEW - umożliwia przeglądanie zawartości plików fizycznych z uwzględnieniem ich struktury (m.in. wyświetlane są nazwy pól rekordu),

mixUSER - program pozwalający na dokonywanie przez użytkowników ograniczonych (przewidzianych przez projektanta aplikacji) zmian w zawartości słowników.

mixJOUR - program obsługi dziennika (dostarczany wraz z aplikacjami wykorzystującymi dziennik)

W oparciu o bibliotekę mixBASE opracowano generator aplikacji mixUTIL, który jest pakietem programów przeznaczonych do tworzenia i pielęgnacji aplikacji, realizowanych z wykorzystaniem biblioteki. Podstawowy zestaw programów pakietu mixUTIL przeznaczony jest do tworzenia i pielęgnacji słowników przechowujących opisy struktur danych tworzonej aplikacji. Definiowanie struktur danych odbywa się w maksymalnym zakresie metodą wykorzystywania podpowiedzi oraz bieżącej kontroli poprawności wprowadzanych danych. Cały zestaw programów ma jednolitą formę komunikacji z użytkownikiem, wykorzystującą technikę rozwijanych menu i okien. W przypadku projektowania formatek ekranowych, menu sterowania przebiegiem aplikacji oraz formatów wydruków zastosowano technikę makietowania. Pozostałe części składowe pakietu mixUTIL podzielić można na następujące grupy:

- moduł generacji programów: składa się z generatora tworzącego w oparciu o informacje zawarte w słownikach szkielet programu aplikacji zapisany w specjalnej, uproszczonej notacji oraz preprocesora przetwarzającego tę notację na program w języku C,

- moduł dokumentatora aplikacji, tworzący szkielet dokumentacji konstrukcyjnej poprzez odpowiedni wydruk informacji zawartych w słownikach opisu struktury danych,

- moduł analizatora spójności słowników, pozwalający także na

tworzenie tablic powiązań poszczególnych elementów struktur danych, opisanych w słownikach,

- moduł integratora aplikacji pozwalający na łączenie słowników opisujących struktury danych dwóch lub więcej aplikacji w jeden komplet lub też na dekompozycję jednego kompletu na kilka mniejszych (co może mieć zastosowanie przy niezależnym tworzeniu fragmentów jednej aplikacji przez różnych programistów).

4.4. SUPER

Generator systemów użytkowych SUPER powstał w warszawskiej firmie MacroSoft.

Systemy użytkowe tworzone z użyciem generatora SUPER, są charakteryzowane przez następujące cechy:

* relacyjność.
  Występujące bazy danych są strukturami tabelarycznymi, przetwarzanymi przez zamknięty język programowania. Język ten rozumiany jest przez jądro wykonawcze (engine) tak, że polecenia tego języka na zbiorze wierszy bazy relacyjnej, są wykonywane bez tłumaczenia na język proceduralny

* utrzymywanie integralności danych.
  Zapewniono kontrolę dzięki tabelom indeksów, przechowywanym w plikach .NDX. Utrzymywana jest także unikalność kluczy w wybranych tabelach indeksów

* obsługę przekrojów międzybazowych.
  Możliwa jest prezentacja i redakcja danych z więcej niż jednej bazy danych, tworzących identyfikowany przez nazwę przekrój międzybazowy. Podczas redakcji kontrolowana jest integralność referencyjna, przy odpowiedniej definicji relacji

* komunikowalność.
  Zapewniono możliwość importu i eksportu danych. Każdy system użytkowy jest wyposażany w mechanizm kontekstowego samouczka z możliwością jego redakcji przez Użytkownika. Do aktualizacji danych służą zautomatyzowane funkcje przeglądania i redakcji, wspomagane mechanizmami obsługi menu. Możliwe jest działanie interaktywne z użyciem oryginalnego języka, jak również tworzenie wydruków w języku generatora sprawozdań

* korzystanie z katalogu systemu.
  Rolę katalogu systemu pełni tu tzw. plik sterujący .DEF. Zawiera on opisy wszelkich struktur baz danych, indeksów, relacji i komunikacji z użytkownikiem

* odtwarzanie utraconych danych.
  Realizowane jest dla tabel indeksów (plików .NDX). System ma funkcję odtwarzania indeksów uszkodzonych w wyniku nieprawidłowego przebiegu pracy lub na życzenie użytkownika

* brak kontroli transakcji.
  Nie istnieje możliwość opisania i automatycznej kontroli transakcji

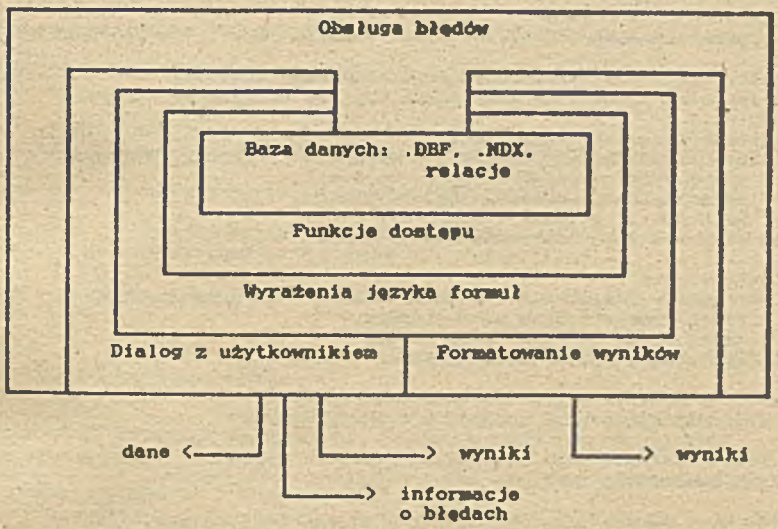
Systemy użytkowe, tworzone przy użyciu generatora pracują w ten sposób, że program wykonawczy .EXE (egzekutor) działa na relacyjnych bazach danych (plikach .DBF), czerpiąc informacje opisujące dany system użytkowy z katalogu systemu (pliku sterującego .DEF). Plik ten jest wynikiem generowania systemu, na podstawie projektu. Do takiego generowania służy SUPER, będący ponadto edytorem katalogu systemu. Do tworzenia algorytmów obliczeniowych dla systemów użytkowych, stosuje się język FORMULA, którego wyrażenia są interpretowane podczas pracy systemu przez wbudowany w .EXE interpreter. Kontaktowanie się poziomu algorytmu z rzeczywistym światem bazy danych realizujemy przez zespół funkcji dostępu MacroBase. Umożliwiono także sporządzanie sprawozdań, które wykonują się według zapisu sterującego zawartego w plikach tekstowych, czytanych i interpretowanych przez język Report.

Program wykonawczy systemu użytkowego zawiera:

- procedury inicjalizacji pracy systemu użytkowego
- procedury obsługi błędów czasu wykonania (urządzeń zewnętrznych, pamięci, przepełnienia arytmetycznego, itp.)
- procedury obsługi urządzeń zewnętrznych (klawiatury, ekranu, drukarki, itd.)
- funkcje dostępu do baz danych
- jądro wykonawcze (engine, system przetwarzania relacyjnej bazy danych)
- interpreter języka formuł FORMULA
- interpreter języka sprawozdań Report
- dodatkowe procedury właściwe dla systemu użytkowego, wykonywane poza interpreterem formuł i funkcjami dostępu

Użytkownik dostaje do dyspozycji uniwersalny program wykonawczy o nazwie MASTER.EXE, który czyta plik sterujący .DEF z dysku i wykonuje polecenia Użytkownika z użyciem wbudowanych interpreterów. Aby system użytkowy rozpoczął pracę wystarczy, jako argument programu MASTER w linii polecenia systemu operacyjnego podać nazwę pliku sterującego .DEF. Po wstępnej fazie projektowej, pierwszym krokiem musi więc być redakcja takiego pliku i jego zapisanie na dysk, tj. wygenerowanie systemu.

W systemach użytkowych MacroSoft wyróżnia się następujące warstwy:



Informacje przepływają od świata zewnętrznego do bazy danych przez poszczególne warstwy, wyspecjalizowane w podanych na schemacie funkcjach. Warstwy te są realizowane przez odpowiednie moduły programowe, co podane jest poniżej:

Warstwa	Moduł	SUPER
baza danych	jądro wykonawcze	definiowanie struktur baz danych, ich indeksów i relacji
funkcje dostępu	MacroBase	
wyrażenia języka formuł	MacroFormula	definiowanie formuł sterujących systemu
dialog z użytkownikiem	MacroWindows	definiowanie okienek i formuł dialogu
formatowanie wyników	MacroReport	
obsługa błędów	MacroError	

W ostatniej kolumnie podane są czynności niezbędne do utworzenia systemu użytkowego dokonywane przy użyciu generatora SUPER.

Przetwarzanie informacji zachodzi dzięki wykonywaniu formuł języka FORMULA, przy czym mogą one być wywoływane jako skutek:

. wyboru funkcji z menu
  Formuła związana jest z pozycją w danym menu (formuła sterująca).

. wyboru funkcji z linii podpowiedzi (formuła dialogu)
  Formuła, związana z klawiszami obsługi okienka wertowania.

. zmiany położenia w bazie danych
  Formuły wykonywane w czterech wypadkach: (formuły dialogu)
  przed wyświetleniem okienka (Display)
  przed redakcją okienka (Before edit)
  po redakcji okienka (After edit)
  w wyniku dołączenia rekordu (Blank).

. redakcji pola
  Formuły wykonywane w trzech wypadkach: (formuły dialogu)
  przed wyświetleniem pola (Display)
  przed redakcją pola (Before edit)
  po redakcji pola (After edit).

. użycia w tekście wzorca sprawozdania .RPM
  Formuły wykonywane zgodnie z treścią wzorca.

Definicję formuł dla pierwszych trzech wypadków przeprowadza się z użyciem generatora SUPER.

Główne menu generatora SUPER pozwala wybrać następujące funkcje:

```
┌──────── GŁÓWNE MENU ────────┐
│ definicja Systemów          │
│ definicja Baz danych        │
│ definicja Relacji           │
│ definicja wolnych Okienek   │
│ definicja okienek Menu      │
│ definicja wolnych Pól       │
│ edycja Formuł               │
│ ratowanie Danych po awarii  │
│ Tworzenie zbiorów           │
│ sCalanie danych             │
└─────────────────────────────┘
```

Przedstawione powyżej cztery przykłady są dowodem na to, że również w Polsce pojawiła się potrzeba stosowania narzędzi tej klasy. Warto zwrócić uwagę, że żadne z przedstawionych narzędzi nie powstało w środowiskach akademickich. Wszystkie opracowano w firmach, które mają nadzieję uzyskać większą wydajność pracy nad konkretnymi aplikacjami. Oczywiście jest to dopiero pierwszy krok w stronę CASE, ale dowodzący głęboko praktycznych korzeni systemów tej klasy.

5. Literature

[1] Alavi M.: 'An Assessment of the Prototyping Approach to Information Systems Developement', Communications of the ACM, Vol. 27, No. 6, pp.556-563, 1984

[2] ANSI. Standards on Information Resource Dictionary Systems, 1988.

[3] Appleton D.S.: 'Data-Driven Prototyping', Datamation, pp. 259-268, 1983

[4] Boar B.H.: Application Prototyping: A Requirements Definition Strategy for the 80s, New York, John Wiley & Sons, 1984.

[5] Boehm B.: 'Improving Software Productivity', Computer No. 8 1987

[6] Chen P.P.: 'Entity-Relationship Model: Towards a Unified View of Data', ACM Transactions on Database Systems,t. 1 nr. 1.

[7] Connel J., Brice L.: 'Rapid Prototyping', Datamation, Vol. 30, No.4, 1984

[8] Connel J., Schafer L.: Rapid Prototyping, New York, Yourdon Press/Prentice-Hall, 1989.

[9] Davies C. G. S., Jajodia P. Ng., Yeh R.: Entity-Relationship-Approach to Software Engineering, New York, North-Holland, 1983.

[10] DaMarco T.: Structured Analysis and Systems Specification, Englewood Cliffs NJ., Prentice-Hall, 1979.

[11] Dolin K.: Business Computer Systems Design, Santa Cruz, CA, Mitchell Publ. 1984.

[12] Falk H.: 'CASE Tools Emerge to Handle Real-time Systems', Computer Design, January 1, 1988, pp. 53-74.

[13] Gane C.: Rapid System Developement, Englewood Cliffs NJ., Prentice-Hall, 1988.

[14] Gane C., Sarson T.: Structured system analysis: Tools and Techniques, Englewood Cliffs NJ., Prentice-Hall, 1979.

[15] Gibson M.L.: 'The CASE Philosophy', BYTE, pp. 209-218, No.4, 1989

[16] Hansen K.: Data Structured Program Design, 2nd ed., Engelwood Cliffs, NJ: Prentice-Hall, 1986.

[17] Higgins D. A.: Designing Structured Programs, Englewood Cliffs, NJ, Prentice-Hall, 1983.

[18] IEEE Guide to Software Requirements Specifications, ANSI/IEEE Std. 830-1984, Institute of Electrical and Electronical Engeneers Inc., 1984.

[19] Jackson M. A.: Priciples of Program Design, New York, Academic Press, 1975.

[20] King D.: Current Practices in Software Developement, Englewood Cliffs NJ, Yourdon Press, 1984.

[21] March S. (ed): Entity Relationship Approach, New York, North Holland, 1988.

[22] Martin C. F.: User Centered Requirements Analysis, Engelwood Cliffs, NJ: Prentice-Hall, 1988.

[23] Martin J., McClure C.: Diagramming Techniques for Analysts and Programmers, Engelwood Cliffs, NJ: Prentice-Hall, 1985.

[24] Martin J., McClure C.: Action Diagrams: Clearly Structured Specifications, Programs and Procedures, Engelwood Cliffs, NJ: Prentice-Hall, 1989, 2nd ed.

[25] Maczyński J.: Papers on Program Design, Prace IPPT 5/1983, Warszawa.

[26] McClure C.: 'The CASE Experience', BYTE, pp. 235-246, April 1989.

[27] Orr K.T., Gane Ch., Yourdon E., Chen P.P., Constantine L.L.: 'Methodology: The Experts Speaks', BYTE, pp. 221-233, April 1989.

[28] Orr K.T.: Structured Requirements Definition, Topeka, KS, Ken Orr&Ass., 1980.

[29] Page-Jones M.: The Practical Guide to Structured Systems Design, 2nd ed., New York, Yourdon Press/Prentice-Hall, 1988.

[30] Rook P. 'Controlling software projects', Software Engineering Journal No. 1 1986

[31] Smith M. C.: 'Database Design: Composing Fully Normalized Tables from a Rigorous Dependency Diagram', Communications of the ACM, vol.28, no.8, 1985.

[32] Teorey T., Fry J.: 'An extended Entity-Relationship Approach to Logical Database Design', ACM Survey, 1986.

[33] Wallace S.: 'Methodology: CASE's Critical Cornestone', Business Software Review, April 1988.

[34] Wang Yu: 'A Distributed Specification Model and Its Prototyping', IEEE Transactions on Software Engineering, No. 8, 1988

[35] Ward P., Mellor St.: Structured Techniques for Real-Time Systems, New York, Yourdon Press/Prentice Hall, 1985.

[36] Warnier J.D.: Logical Construction of Systems, New York, Van Nostrand Reinhold, 1981.

[37] Yourdon E.: Modern Structured Analysis, New York, Yourdon Press/Prentice-Hall, 1989.

[38] Yourdon E., Constantine L.L.: Structured Design, Englewood Cliffs NJ, Yourdon Press/Prentice Hall, 1987.

Formularz oceny narzędzia CASE

INFORMACJA OGÓLNA
Dostawca.....[_]
Nazwa pakietu.....[_]
Rok wprowadzenia.....[_]
Rozpowszechnianie.....[_]
Cena.....[_]

KOMPUTER DO IMPLEMENTACJI
Firma produkująca.....[_]

KATEGORIA
Pakiet narzędziowy.....[_]
Pakiet zintegrowany.....[_]

TYP NARZĘDZI
Planowanie.....[_]
Analiza.....[_]
Projektowanie.....[_]
Projektowanie bazy danych.....[_]
Projektowanie oprogramowania czasu rzeczywistego.[_]
Generator kodu.....[_]
Programowanie.....[_]
Utrzymanie oprogramowania.....[_]
Otoczenie programowe.....[_]
Zarządzanie pracami projektowymi.....[_]

GRAFIKA
Kolor.....[_]
Mysz.....[_]
Okna.....[_]

REPOZYTORIUM PAKIETU CASE
W centralnym komputerze.....[_]
W komputerze personalnym.....[_]
Architektura systemu sterowania bazą danych.....[_]
Sprawozdawczość.....[_]
Sterowanie zmianami.....[_]
Śledzenie przebiegu.....[_]
Sterowanie zmianami wersji.....[_]
Zabezpieczenie stanu aktualnego.....[_]
Logiczne rozdzielenie danych.....[_]
Konsolidacja.....[_]

OPROGRAMOWANIE PROTOTYPOWANIA
Projektowanie ekranów interakcji.....[_]
Projektowanie raportów.....[_]
Model funkcjonalny.....[_]
Symulacja działania.....[_]

GENEROWANIE KODU
Program szkieletowy.....[_]
Pełny program.....[_]
Język.....[_]
Program interakcyjny.....[_]
Program wsadowy.....[_]

OPROGRAMOWANIE PRZEBUDOWY SYSTEMÓW
Analizator statyczny.....[_]
Dokumentacja odnawiana.....[_]
Modyfikacja struktury.....[_]
Odtwarzanie.....[_]
Analizator dynamiczny.....[_]
Konwerter.....[_]

KONTROLA JAKOŚCI
Składnia.....[_]
Zwartość.....[_]
Kompletność.....[_]
Zgodność z wymogami.....[_]
Kontrola jakości.....[_]

OPROGRAMOWANIE PROCESU ROZWIJANEGO PAKIETU
Planowanie.....[_]
Analiza.....[_]
Projektowanie.....[_]
Generowanie kodu.....[_]
Utrzymanie w ruchu.....[_]
Zarządzanie wytwarzaniem pakietu.....[_]

SYSTEM BIBLIOTEKI
Interakcyjny.....[_]
Wsadowy.....[_]
Przetwarzanie transakcji.....[_]
Czasu rzeczywistego.....[_]
Do wbudowania.....[_]

DOSTĘPNE SCHEMATY
Przepływu danych.....[_]
Sterowania.....[_]
Tablice decyzyjne.....[_]
Hierarchiczne struktury drzewiaste.....[_]
Schemat strukturalny.....[_]
Działania.....[_]
Metoda Warniera-Orra.....[_]
Przejścia między stanami.....[_]
Pseudokod.....[_]
Generator makiet ekranowych.....[_]
Generator struktury interakcji.....[_]
Generator sprawozdań.....[_]
Struktura danych.....[_]
Bytowo-relacyjny.....[_]
Rekordy logiczne.....[_]
Booch.....[_]
Sieci Petriego.....[_]
Inne.....[_]

PODSTAWY METODOLOGICZNE
Yourdon.....[_]
DeMarco.....[_]
Gane/Sarson.....[_]
Buchman.....[_]
Chen.....[_]
Martin.....[_]
Merise.....[_]
Orr.....[_]
Jackson.....[_]
Ward/Meller.....[_]
Hatley.....[_]
Object-Oriented.....[_]
SADT.....[_]
Stradis.....[_]
Method-1.....[_]
LSDM.....[_]
Inne.....[_]

FACTOR:
SYSTEM WSPOMAGANIA PROJEKTOWANIA SYSTEMÓW INFORMACYJNYCH

Wojciech Olejniczak, Ireneusz Szydłowski
Uniwersytet Szczeciński, Instytut Cybernetyki
Ekonomicznej i Informatyki,
ul. Mickiewicza 66, 71-101 Szczecin

1. W S T Ę P

Dziedzina tworzenia systemów informacyjnych, wspomagana komputerowo, rozwija się ostatnio szczególnie intensywnie. Śledząc jej rozwój na świecie obserwujemy, że od 1988 r., co roku można mówić o pojawianiu się nowych generacji w inżynierii systemów. Przykładowo w 1989 na znaczącej konferencji w Sztokholmie mówiono o CASE - tools, w tym roku obok CASE - tools występują już CASE - shells.

W ten obiecujący nurt projektowania możemy oczywiście włączyć się jako bierni czytelnicy trudno dostępnej literatury i "świecić tym samym światłem odbitym" - publikując artykuły, referaty, wykłady; czyli możemy tylko opowiadać o tym co się na

świecie dzieje. Można w ten nurt włączyć się czynnie i do tego też zmierzamy.

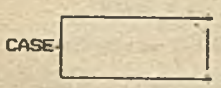
Dorobek inżynierii systemów informacyjnych jest dyskontowany w postaci produktów programowych, systemów wspomagających projektowanie. Systemy te, z natury także bardzo drogie, są produktami chronionymi, jak każda zaawansowana technologia. Uwzględniając te okoliczności, zespół nasz przed kilku laty postanowił zmierzyć się z tym tematem. Można też powiedzieć, że czynimy to z pewnymi sukcesami i to nie tylko teoretycznymi. Powstała i powstaje metodologia projektowania systemów informacyjnych FACTOR wraz z integralną częścią tej metodologii jaką jest narzędzie - system komputerowego wspomagania projektowania SI, porównywalny z CASE - shells. Celem naszym jest stworzenie łącznie s y s t e m u p r o j e k t u j ą c e g o, będącego kwintesencją dorobku teoretycznego i praktycznego zespołu. Obecnie jesteśmy zaangażowani w rozwój prototypu systemu wspomaganego projektowania systemów informacyjnych nowej generacji, w oparciu o pomysły i wynalezione algorytmy.

System umożliwia planowanie, realizację i kontrolę projektowanego systemu informacyjnego w jego strukturach, funkcjonowaniu i rozwoju.

Elementem niniejszego wprowadzenia jest też demonstracyjna wersja systemu FACTOR - wraz ze szczegółowym omówieniem realizowanym podczas prezentacji komputerowej i ew. ćwiczeniami z

systemem.

Wyjaśnijmy też, że sformułowanie CASE rozumiemy jak poniżej:



Celem CASE jest system. System jest też (w wąskim znaczeniu) zbiorem programów. Rezultatem stosowania CASE nie jest 'system z komputera' lecz 'system tworzony z komputerem'.

2. Z A Ł O Ż E N I A

Przypominamy (z poprzednich szkół PTI) nasze spojrzenie na system informacyjny (SI):

Patrząc globalnie na 'organizm' systemu informatycznego, można go rozwarstwić na cztery układy podstawowe:

- strukturę funkcjonalną (STF),
- strukturę informacyjną (STI),
- strukturę techniczną (STT),
- strukturę przestrzenną (STP),

i dodać układ w którym SI egzystuje (jego środowisko):

- strukturę organizacyjną (STO),

która jest światem ludzi, maszyn i pieniędzy. SI jest częścią tego świata, na który aktywnie oddziaływuje. Układy te (których definicje są proste i intuicyjnie zrozumiałe) są zawsze przedmiotem projektowania SI. W ich projektowaniu występują:

- różne obiekty i konfiguracje obiektów tworzących te układy,
- specyficzne dla tych układów problemy,
- odmienne 'spoiwo' układów w cały organizm SI.

* podejście E-R, czy ogólniej podejście obiektowo zorientowane, jest jeszcze bardzo teoretyczne, im bardziej teoretyczne, tym dalej mu w naszych układach do praktyki. Podejście to wystarczy stosować i wtedy nie trzeba specjalnie o nim mówić. Niech 'normalny' projektant SI, stosując podejście E-R znajdzie się w sytuacji owego mieszczanina, który mówił prozą nie wiedząc o tym. Jest to oczywiście kwestia posiadania odpowiednich metod i narzędzi.

* sens tworzenia SI istnieje, gdy istnieją:

- strategia,
- metodologia,
- narzędzia.

* Centralne miejsce ma metodologia, która profiluje i aktualizuje strategię, tworzy narzędzia - oczywiście decyduje o SI. Metodologia, z drugiej strony, realizuje określoną strategię,

wykorzystując odpowiednie narzędzia. Tak powstaje SI.

* CREDO:

W tworzeniu 'naszej' globalnej metodologii poszukiwaliśmy systemu (narzędzia), który będzie ją wspomagał. W tym systemie zaś, dla różnych struktur obiektów (w ramach STF, STI, STT, STP i STO) poszukiwaliśmy wspólnych mechanizmów 'manipulowania' tymi obiektami. Szukaliśmy też takiej metodologii - właściwej dla systemu - będącej narzędziem do tworzenia SI.

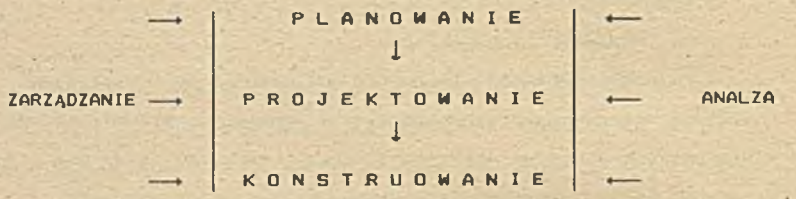
3. S T R A T E G I A

Przykładowo, przeglądając materiały prezentowane na CASE'89 w Sztokholmie, szczególnie przez osoby związane z James Martin Associates, można odeczytać strategię realizacji SI składającą się z następujących członów:

- zarządzanie,
- planowanie,
- projektowanie,
- konstruowanie,
- analiza,

wg zasad inżynierii informacji (por. referat K.Gajewskiego w niniejszym zbiorze)

i to akceptujemy, przyjmując następujący nasz układ powiązań:



I choć tak to się robi w praktyce, to potrzebna jest metodologia, która właściwie wpleciona w strategię i do niej dopasowana, jest elementem decydującym o jakości systemu. Stąd i geneza naszej propozycja.

4. METODOLOGIA

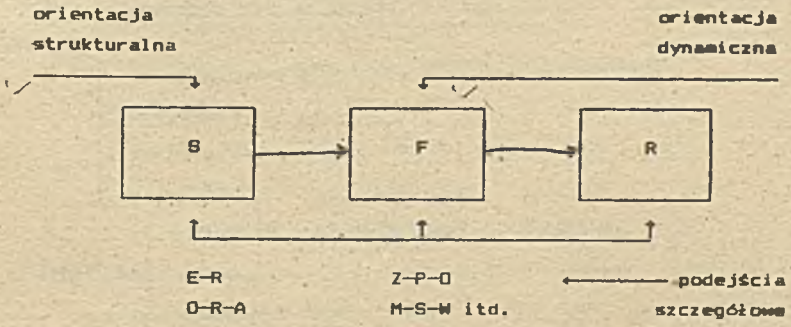
W pierwszej kolejności, zasygnalizowana na wstępie kwestia formuły, podejścia wyjściowego, które jest 'filozofią' rozumienia istoty procesów informacyjnych. Wiele podejść zdobyło pewną popularność, na czoło wybija się E-R, są jeszcze inne. Ich mankamentem jest też to, że mimo ich znacznych walorów i wpływu, jaki wywarły na konstrukcję systemów i narzędzi - rozstrzygają w sumie w dość wąskim zakresie kompleksu zagadnień występujących w

projektowaniu systemów. Brak im też jest charakteru operacyjnego, czy nawet receptualnego.

Dlatego proponujemy metaformułę:

S T R U K T U R A ↕ F U N C J O N O W A N I E ↕ R O Z W Ó J

z następującym powiązaniem jej członów:



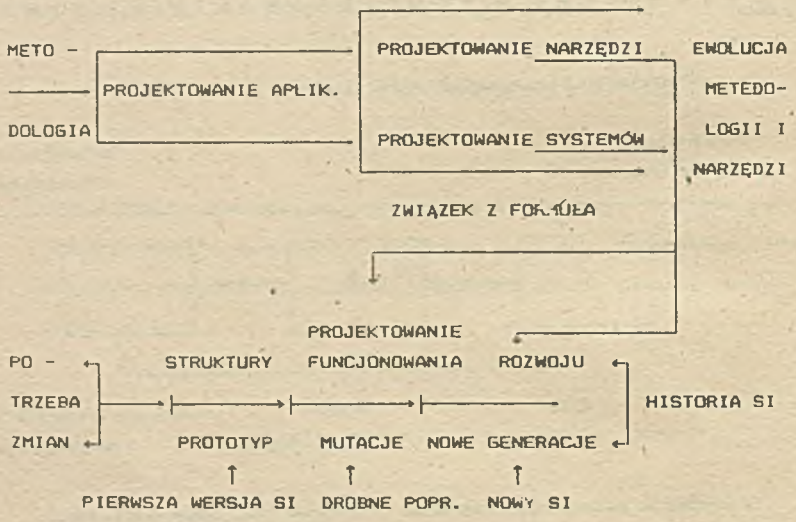
Orientacja metodologiczna daje projektantowi dwie możliwości wejścia w projekt systemu. A więc projektant musi się zdecydować czy:

- truktura wyznacza działanie SI,
- czy działanie strukturę ?

Decyzja jest tu określana w pewnym sensie przez skłonności psychotechniczne ludzi, tradycje konkretnych ośrodków itd.

Niezależnie od powyższego wyboru można określić:

- cykl życia systemu,
- formułę,
- projekt, ułożone w całość jak na rysunku:



Sama metodologia zakłada:

- akceptację (nawet wchłonięcie) formuł szczegółowych,
- konkretyzację obiektów SI w kategorie obiektów,
- przypisanie kategorii obiektów strukturom,

- konkretyzację kategorii obiektów w wystąpienia obiektów,
- wprowadzenie standardowych określeń, oznaczeń, nazw,
- stosowanie mechanizmów:
    - typowej obsługi projektanta,
    - rozpoznawanie systemu:
        - przez hierarchizację,
        - przez powiązania technologiczne,
        - przez powiązania funkcjonalne,
    - tworzenie kategorii obiektów niestandardowych,
    - odsyłaczy,
    - cechowania,
    - łączenia,
    - wyróżniania,
    - stawania się,
    - oznaczania,
    - eksperymentów projektowych (wariantowania),
    - obliczeń,
    - generowania aplikacji,
    - raportowania,
    - zapytań (konsultacji),
    - przewodnika,
    - kontroli.

Mechanizmy mają różne znaczenie dla projektanta i system ma własną hierarchię i zdeterminowany obszar działania. I to jest powód dlaczego muszą być zawarte w SYSTEMIE FACTOR (SF).

Mechanizmy są wspólne i mogą być użyte do projektowania każdej z pięciu struktur. Struktury te mają własną specyfikę w projektowaniu. Specyfika ta jest wyrażana po prostu przez kategorie obiektów - atrybuty struktur. Ale wspólną kategorią dla całego systemu jest kalendarz SI.

Metodologia widzi system wielopoziomowo, podzielony między struktury sieci połączonych objektów. Projektant wchodzi w tę sieć w dowolnym węźle. Projektant 'chodzi' po sieci - jak w grze planszowej - sterowany dostępnymi mechanizmami, dowolnie chociaż obowiązkowo sensownie. Przejście z jednej planszy na drugą (z jednej struktury do drugiej) jest dowolne - przez mechanizmy. Taka wizja metodologii czyni projektowanie działaniem swobodnym.

5. N A R Z Ę D Z I E

Koncepcja narzędzia zwanego SYSTEMEM FACTOR akceptuje prezentowaną powyżej strategię i metodologię. Narzędzie jest także częścią metodologii. Innymi słowy, metodologia realizuje strategię z pomocą SYSTEMU FACTOR. Pewna koncepcja architektoniczna SYSTEMU FACTOR została zarysowana. Koncepcja ta sprawia, że:

- SYSTEM FACTOR sam w sobie jest całością,
- poszczególne programy SYSTEMU FACTOR są autonomiczne i samoistnie mogą funkcjonować w obrębie swojego przeznaczenia,

- poszczególne grupy programów można łączyć w pewne moduły mające też użytkowe znaczenie,
- programy i moduły stanowią podsystemy SYSTEMU FACTOR,
- z SYSTEMU FACTOR można też 'wyciąć' inny system, o innym przeznaczeniu - np. system projektowania organizacji.

Innymi słowy SYSTEM FACTOR to jakby sklep samoobsługowy, z którego można wyjąć poszczególne towary, ba nawet pewne stoiska, albo utworzyć obok inny sklep. I to wszystko bez szkody dla FACTOR'a.

Zatem koncepcja architektoniczna SYSTEMU FACTOR nie jest opracowana na zasadzie 'wszystko albo nic', a wprost przeciwnie: 'wszystko albo część'. Tych pochodnych możliwości SYSTEMU FACTOR doliczono się już sporo.

np.:

FACTOR 1: System wspomagający projektowanie organizacji (jej struktury i funkcjonowania), w szczególności zawierający:
- bazę danych projektowych,
- bank metod projektowych,
- bazę wyników,
- przyjazny dialog z projektantem,
- metodologię projektowania organizacji.

FACTOR 2: System wspomagający analizę, projektowanie i rozwój systemów informacyjnych i informatycznych wg

struktura funkcjonalnej, informacyjnej, technicznej, przestrzennej i organizacyjnej.

FACTOR 2 to:

- metodologia wspomaganego projektowania systemów z zachowaniem zasad CASE i strukturalizacji,
- wspomagana komputerowo metodologia projektowania i rozwoju systemów informacyjnych REMORA (wg prof. Colette Rolland),
- baza danych projektowych umożliwiająca utworzenie, utrzymanie i analizę wielu opisów systemów przez wielu projektantów,
- bank metod projektowych dla przeprowadzania eksperymentów projektowych,
- bazę metodyczną i normatywną projektowania,
- bazę wyników eksperymentów projektowych,
- specjalne interfejsy (podstawowy, graficzny, inteligentny).

FACTOR 3: Bazę danych badawczych, gromadzącą dane o badanych obiektach, ich artybutach oraz relacjach między obiektami. Określenie nazwy obiektów, atrybutów i relacji są definiowane przez użytkownika prowadzącego badania i akceptowane przez bazę danych.

FACTOR 4: Programy wstępnej obróbki i selekcji danych badawczych (normalizacja danych, dyskryminacja cech, badanie podobieństwa).

FACTOR 5: Programy (typu cluster) grupowania, klasyfikacji, restrukturalizacji obiektów dla różnych sytuacji badawczych.

FACTOR 6: Programy komputerowego opracowania wyników dla danych badawczych.

FACTOR 7: Unikatowe programy opracowania wyników danych badawczych wg życzenia użytkownika.

FACTOR 8: Standardowa baza wyników badań i eksperymentów współpracująca z wszystkimi wyżej wymienionymi programami.

FACTOR 9: Programy przydziału i dobierania obiektów (np. pracownik - praca),

FACTOR 10: Programy planowania sieciowego przedsięwzięć,

oraz

FACTOR 11: Metoda heurystycznej dekompozycji schematów (np. logicznych).

FACTOR 12: Metoda iteracyjna podziału grafów.

FACTOR 13: Metoda rozmieszczenia obiektów w przestrzeni.

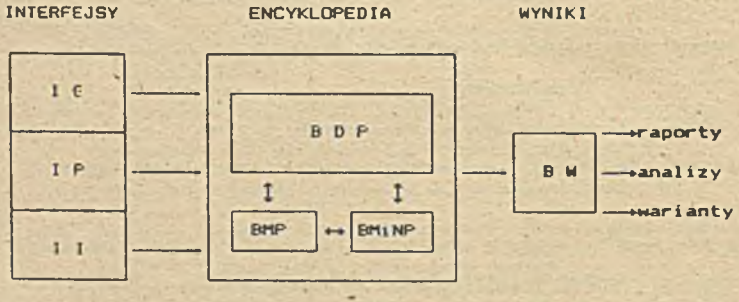
I nie jest to jeszcze wypełniony całkowicie zakres możliwości systemu FACTOR, zrealizowanego na zasadzie CASE - shells.

Sama architektura SYSTEMU FACTOR jest modułowa i wyróżnia

się:

- interfejsy:
  - . podstawowy (IP),
  - . graficzny (IG),
  - . inteligentny (II),
- encyklopedię:
  - . baza danych projektowych (BDP),
  - . baza metod projektowych (BMP),
  - . bazę metodyczną i normatywną projektowania (BMiNP),
- bazę wyników projektowych (BWP):
  - . raportowanie,
  - . analizy,
  - . warianty.

To cała lista komponentów SYSTEMU FACTOR tworzących architekturę, a powiązania tych komponentów prezentuje rysunek:

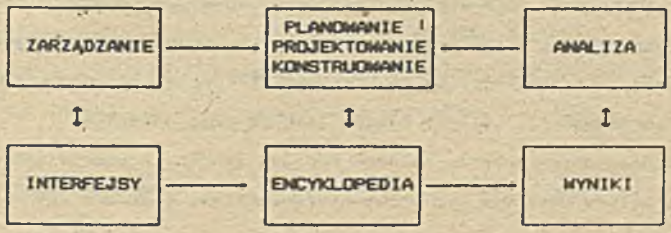


Taka architektura sprzyja rozwojowi SYSTEMU FACTOR przez rozwój

poszczególnych modułów, których treść merytoryczna zależy od pojętności projektantów, a zastosowalność systemu FACTOR od spolegliwości użytkowników.

6. I N T E G R A C J A

Nie bez znaczenia jest również fakt, że przyjęta architektura systemu FACTOR pozwala stosować zaprezentowaną na wstępie ogólną strategię realizacji systemu informacyjnego. Powiązania istniejące w ramach strategii odzwierciedlają jakby powiązania w ramach systemu FACTOR, co w efekcie pozwala na pogłębienie zależności i integrację strategii, metodologii i narzędzia. Jeden z wymiarów tej integracji został zaprezentowany na rysunku.



Oznaczało to, że niezależnie od faktu istnienia globalnych odwzorowań między strategią a systemem FACTOR, istnieją również powiązania szczegółowe określające rolę poszczególnych modułów systemu FACTOR w realizacji elementów strategii.

7. ROZWÓJ W KIERUNKU SYSTEMÓW NOWEJ GENERACJI

Zarysowana dotychczas metodologia na zastosowanie do projektowania zaawansowanych acz "klasycznych" zastosowań informatyki w zarządzaniu. Cechą charakterystyczną tych zastosowań jest proces opanowywania masywu danych, ich przetworzenie i dostarczenie użytkownikowi.

Obecnie można już mówić o dwóch wymiarach ("filozofiach") systemów informacyjnych, a w konsekwencji o dwóch podejściach do projektowania. Pierwszy wymiar, oczywiście pionierski, został ukształtowany przez klasyczne systemy przetwarzania masowych danych i transakcji. Do tej "filozofii" dopasowano projektowanie, programowanie i języki. System jest tu dość dowolnym zgromadzeniem zbiorów różnego rodzaju danych, w którym spójność struktury uzyskuje się przez hierarchiczne, sieciowe czy relacyjne powiązania między danymi. Dane są reprezentantami różnego rodzaju przedmiotów i zjawisk. Stąd ich mnogość - o wiele za dużo niż trzeba, ze stałą tendencją wzrostu, pod presją próśb

użytkowników.

Także metodologia, narzędzia i ludzie wikłają się w wielowspektowej analizie danych, tracąc czas i nerwy. Pomysły zastosowania do takich analiz systemów expert, choć oczywiście obiecujące i pomagają (także nam), należy uznać za radykalnie chybione, bo służą "złej sprawie".

Drugi wymiar związany jest z tworzeniem nowych generacji systemów informatycznych. Wymiar ten jest związany z badaniem relacji, a nie samych obiektów, którymi n i e są d a n e. Termin "relacja" nie jest użyty w ściśle matematycznym znaczeniu, lecz szeroko i tworzy całą klasę pojęć pokrewnych: powiązanie, zależność, oddziaływanie, zazębianie się, połączenie, podobieństwo itd. W tym drugim wymiarze można właściwie mówić o systemach sensu stricto i o postulacie redukcji danych do niezbędnego minimum.

Oba wzajemnie uzupełniają się wzajemnie i oba są istotne, gdy chodzi o wspomaganie procesów zarządzania, choć pierwszy oparty jest na ekspansji danych, a drugi na syntezie problemów i redukcji.

Istota podejścia

Istota podejścia wynika ze zmiany orientacji, której punktem wyjścia jest inne spojrzenie na system informatyczny, który jest s t e r o w n i k i e m w obiekcie zastosowań. System taki, to

już nie *piggyback* dość przypadkowo dobranych danych, lecz:

- zbiór zmiennych, właściwych obiektów SI,

- zbiór powiązań między tymi zmiennymi (związek, zależność, korelacja etc),

- założenie niezmienności powiązań na tle czasu, przestrzeni, populacji - są wiedzą o systemie,

- zmienne dają obraz jakiejś cechy świata realnego,

- ich "przebieg" powinien się odbywać w granicach stanów pożądanych, stąd mówimy o sterowaniu,

- obserwowalność (i sterowalność) zmiennych jest związana ze skończonym zbiorem wartości (stanów), który reprezentuje przejawianie się odpowiedniej cechy,

- to ostatnie, to nasze dane, ale jakże inaczej usytuowane - z sensem,

- stany mają pewną właściwość matematyczną (uporządkowanie, odległość), która oddaje podobną własność wykrytą dla odpowiadającej cechy,

- oczywiście są też zmienne jakościowe, ktore nie maja właściwości matematycznych,

- dynamiczne aspekty systemu są określone nie bezpośrednimi obrazami cech, lecz pośrednimi (zmienne opóźnione),

- sposoby powiązań między zmiennymi: matematyczne, probabilistyczne, rozmyte,

- pewne zmienne, przeważnie specjalnie tworzone są zmiennymi normami sterowania,

- system taki może podejmować decyzje, gdy określi się ich warunki, scenariusze,

- i jeszcze potrzebne są regulatory czułości, opóźniacze, przyspieszacze etc.

- zmienne wejściowe i wyjściowe są fragmentami (aspektami) zmiennych ogólnych, zmienne wejściowe podają wyniki obserwacji stanów,

- strukturę systemu tworzy zbiór zmiennych i relacji.

Tak nakreślony system staje się właściwym m o d e l e m obiektu.

Nowe problemy

Jeśli tak patrzeć będziemy na system jak na model obiektu, to można zdefiniować dwa problemy:

Problem A jest problemem identyfikacji: "z danego zbioru zmiennych celem jest wydobycie tak wielu informacji o obiekcie, jak to jest tylko możliwe".

Problem B jest, można powiedzieć, odwrotny. Należy określić taki zbiór zmiennych, który jest odpowiedni dla wnioskowania o obiekcie za pomocą wiedzy "zwartej" w zmiennych z możliwym do przyjęcia stopniem przybliżenia. Jest to problem rekonstrukcji obiektu w modelu (przy okazji zwracamy uwagę na pojawiające się słowo "wiedza", w znaczeniu istotnym dla konstruowania systemów expertowych).

Oba te problemy można też sprowadzić do problemu redukcji złożoności projektowanego systemu, a granice tej złożoności wyznaczone są możliwością rozumienia i k i e r o w a n i a obiektem.

Nowy paradygmat SI

W kategorii celu, obecnie celem systemu informacyjnego, nie jest zebranie dla kierowania wszelkiego, maxymalnego zestawu danych. Lecz celem jest samo kierowanie właśnie. Kierowanie oparte na wiedzy, nie na danych. Stąd na czoło wychodzić będzie algoryta, algoryta kierowania, który wyznaczy nieodłączny od niego zestaw danych. Takich i tylko takich (jak trzeba).

Jeśli to jest ten nowy paradygmat, to sprostać mu mogą tylko systemy expertowe.

Całkowita reorientacja

Systemy ukierunkowane na sterowanie, oparte o koncepcję "expert" wymagają całkowitej reorientacji w sferze projektowania. Obok "starej" problematyki dojdą nam problemy:

- dekompozycji i syntezy,
- oceny stosunków części i całości,
- nowych mediów informacji,
- nowych standardów projektowania,

- nowych kategorii, terminów,

- nowych języków porozumiewania się projektantów,

- nowych narzędzi,

- przeniesienie akcentów ze spraw organizowania informacji na organizowanie funkcji.

Szczególnie jednak należy docenić rolę integracji "starych" systemów z nowymi. Generalnie jednak brak jest jeszcze nowych propozycji metodologicznych. Brak jest nowych szkół projektowania. Wszystko jest w jakiejś mierze uwikłane w systemy z przewagą danych.

Jaka metodologia projektowania ?

Jeśli czegoś nie ma, pozostaje jedno - należy to stworzyć. Tak też będzie z nowymi metodologiami projektowania. Będziemy świadkami ich tworzenia. Jednak uprzednio musi nastąpić uświadomienie nowego kształtu SI. Obecny stan zastosowań informatyki i jego perspektywa prowadzi nas na manowce - ściślej na polskie manowce.

U PROGU KOLEJNEGO PRZEŁOMU W TECHNICE KOMPUTEROWEJ - ZESTAWY KOMPUTEROWE Z ŁĄCZNOŚCIĄ SZYNOWĄ JAKO PRZYSZŁOŚCIOWA POSTAĆ WIELODOSTĘPNYCH ZESTAWÓW KOMPUTEROWYCH

TOMASZ RAWIŃSKI

INSTYTUT CYBERNETYKI POLITECHNIKI GDAŃSKIEJ

SPIS TREŚCI

1. WSTĘP - CEL I ZAWARTOŚĆ REFERATU

Celem referatu jest przedstawienie nowych zjawisk i faktów w dziedzinie konstrukcji wielodostępnych zestawów komputerowych jakie zaistniały w Stanach Zjednoczonych na szerszą skalę w ciągu ..ch r... i wskazanie ich istotnego znaczenia dla dalszego rozwoju techniki komputerowej.

Ostatnia tendencja ocena znaczenia tych zjawisk z punktu widzenia dalszego rozwoju techniki komputerowej i na tej podstawie sformułowane będą przewidywania co do tendencji rozwojowych i zjawisk w technice komputerowej jakie winny nabrać dominującego znaczenia w niedalekim czasie.

Istota tych nowych zjawisk jest to, że na rynkach zachodnich pojawiły się wielodostępne zestawy komputerowe (wzk) o zupełnie nowej konstrukcji i posiadają one jakościową przewagę nad zestawami o dawniej znanych, "tradycyjnych" konstrukcjach. W szczególności wartość współczynnika ceny sprawność dla nowego rodzaju zestawów jest wielokrotnie mniejsza niż jego wartość dla zestawów o konstrukcjach klasycznych.

Opracowanie niniejszego referatu uznałem za konieczne ponieważ sytuacja ta winna moim zdaniem, doprowadzić do kolejnej rewolucji w światowej technice komputerowej, przy czym rewolucja ta będzie znacznie głębsza i bardziej istotna niż dotychczasowe rewolucje komputerów osobistych, gdyż będzie polegać na nagłym wystąpieniu zużycia moralnego wielodostępnych zestawów komputerowych o tradycyjnych konstrukcjach. Przyczyna tego będzie pojawienie się na rynku zestawów wielodostępnych, które będą miały znacznie większe możliwości funkcjonalne niż zestawy tradycyjne, a jednocześnie będą znacznie tańsze.
Komputery osobiste nigdy nie dorównywały możliwościami funkcjonalnymi komputerom tradycyjnym.

Posiadanie świadomości powstawania takiej sytuacji jest, moim zdaniem, sprawą ważną dla każdego użytkownika czy szerzej uczestnika procesów komputeryzacji organizacji, zaś sprawą kluczową dla osób mających wpływ na wybór rodzaju zestawów zakupywanych przez te organizacje.

W chwili obecnej na rynku oferowane jest i to w sposób bardzo agresywny, wiele rodzajów zestawów komputerowych o konstrukcji tradycyjnej, które z jednej strony mają bardzo wysoką wartość współczynnika cena/sprawność, a z drugiej strony znajdują się o krok od śmierci moralnej, od całkowitego zejścia ze sceny światowej techniki komputerowej. Zestawy takie można określić jako "żywe trupy".

Nie posiadając świadomości rzeczywistego stanu rzeczy, rzeczywistych tendencji rozwojowych w technice komputerowej decydent może łatwo fałszywie ocenić wartość moralną oferowanego mu zestawu komputerowego i podjąć decyzję o zakupie takiego "żywego trupa", a tym samym narazić na nadmierne koszty przedsiębiorstwo czy instytucję podlegającą komputeryzacji, a także całą gospodarkę narodową. Niebezpieczeństwo takie jest naprawdę duże, gdyż "żywe trupy" produkowane są przede wszystkim przez wielkie i prestiżowe firmy komputerowe, jak: IBM, DEC, Hewlett Packard czy Honeywell i są dla nich źródłem bardzo znacznych zysków. Firmy te będą agresywnie bronić tych zysków i wmawiać wszystkim naiwnym, iż ich tradycyjne produkty są szczytem techniki, gdy w istocie są to już tylko "żywe trupy".

Pisząc ten referat, chciałem umożliwić decydentom zapoznanie się z rzeczywistym stanem techniki komputerowej i dać im w ten sposób szansę na trafną ocenę proponowanych zestawów komputerowych i uniknięcia błędnych i szkodliwych decyzji.

Przyczyną tej nowej, głębokiej i zasadniczej rewolucji w

technice komputerowej stana się wielodostępne zestawy, które określimy, ze względu na ich konstrukcję, jako "zestawy komputerowe łączone szynowo" - ich istotnym składnikiem jest system operacyjny Netware firmy Novell, który organizuje ich działanie. System ten powstał i jest zwykle określany jako "sieciowy system operacyjny", jego znaczenie i zakres możliwych zastosowań są jednak znacznie szersze. System Netware jest w istocie systemem operacyjnym dla wielodostępnych zestawów wielokomputerowych.

Łączone szynowo zestawy wielokomputerowe, w skrócie "zestawy wk/s", są w literaturze angielskiej określane, niezbyt szczęśliwie jako "the bus-based networks" lub "the clustered - CPU networks". Bardziej dokładny opis zestawów wk/s przedstawiony jest oczywiście w dalszym ciągu.

Zgodnie z powyższym, w referacie będziemy zmierzali do uzasadnienia tezy, iż w technice komputerowej przyszłości dominujące znaczenie będą posiadać zestawy wk/s.

W tym celu w pierwszej części referatu sformułujemy ogólne kryteria oceny przydatności czy też wartości użytkowej dowolnego zestawu komputerowego dla użytkownika.

Następnie omówimy bliżej wielodostępne zk o różnych konstrukcjach. Jako konkretny przykład zestawu wk/s omówione zostaną zestawy oparte o komputery płytkowe serii QL (QuickLink) firmy Cubix Corp. z Carson City, Nevada.

Na podstawie przedstawionych kryteriów i opisów dokonamy oceny przydatności czy też wartości użytkowej zestawów komputerowych różnej konstrukcji i w oparciu o te oceny sformułujemy następujące przewidywania:

- jakie będą główne kierunki rozwojowe techniki komputerowej w zakresie konstrukcji zestawów wielodostępnych;
- jakie rozwiązania i konstrukcje nabiorą dominującego znaczenia.

- jakie procesy o znaczeniu rewolucyjnym mogą się pojawić.

Na koniec postaramy się sformułować wnioski jakie z tych przewidywań winny wynikać dla kryteriów, sposobów i zasad doboru wielodostępnych zestawów komputerowych dla krajowych organizacji.

Na zakończenie wstępu wyjaśnimy nieco dokładniej co to jest zestaw wk/s.

Zestaw taki składa się z mikrokomputera głównego typu PC/AT oraz zestawu komputerów płytkowych, które są podłączone do szyny wewnętrznej tego mikrokomputera głównego.

Mikrokomputer główny pełni rolę składnicy zbiorów i działa pod kontrolą jakiegoś sieciowego systemu operacyjnego, najlepiej systemu Netware firmy Novell.

Komputer płytkowy jest to płytka dodatkowa przystosowana do szyny typu PC/AT, na której znajdują się kompletne komputery zgodne z PC/XT lub PC/AT o odpowiednio ograniczonym wyposażeniu. Na jednej takiej płytce może znajdować się jeden lub dwa, a nawet cztery mikrokomputery. Płytka taka będąca komputerem płytkowym umieszczona jest w gnieździe szyny wewnętrznej składnicy zbiorów i każdy komputer na płytce samodzielnie współpracuje ze składnicą - zasada jego współpracy jest taka sama jak zasada współpracy ze składnicą mikrokomputera podłączonego poprzez urządzenia tzw. sieci lokalnej czyli karty sieciowe i kabel.

Podłączenie mikrokomputera bezpośrednio do szyny, bez pośrednictwa urządzeń sieci lokalnej, znacznie zwiększa szybkość przesyłania danych między stanowiskiem roboczym a składnicą, co jest głównym źródłem bardzo wysokiej sprawności takich zestawów.

Jednym z głównych producentów takich komputerów płytkowych jest firma amerykańska Cubix Corp. z Carson City, Nevada.

Dokładniejszy opis budowy zestawów wk/s oraz analiza sposobu ich działania przedstawione zostaną w dalszym ciągu.

1. WIELODOSTĘPNE ZESTAWY KOMPUTEROWE I ICH ZNACZENIE DLA UŻYTKOWNIKÓW

Zestawy komputerowe znajdują zastosowanie w organizacjach jedynie dlatego, że mogą stanowić podstawę, a dokładniej tworzywo dla budowy urządzeń określanych jako "systemy informatyczne" czy też "systemy przetwarzania danych". Są to nazwy wyjątkowo nieodpowiednie - urządzenia te stanowią w istocie urządzenia czy też maszyny do przetwarzania dokumentów i tak winny być określane. Bliższą analizę tej sprawy można znaleźć w pracach (1) i (2).

Maszyny przetwarzania dokumentów, w skrócie maszyny pdk, wykorzystywane są w organizacjach przy realizacji, przez komórki tych organizacji, zadań przetwarzania dokumentów.
Warunkiem skutecznego wykonywania tych zadań przy użyciu maszyny pdk jest stwarzanie przez tą maszynę możliwości jednoczesnego przetwarzania jednego wspólnego zbioru dokumentów przez wiele różnych uczestników organizacji.
W tym celu maszyna pdk winna posiadać wiele stanowisk przetwarzania dokumentów zdolnych do jednoczesnej pracy i równocześnie zapewniać odpowiednią koordynację ich pracy, tak aby nie mogły wystąpić kolizje między poszczególnymi stanowiskami czy też naruszenia poprawności zawartości dokumentów.

Sprawy te mają charakter całkowicie niezależny od zagadnień techniki komputerowej, należą do dziedziny, którą możnaby określić jako "technika przetwarzania dokumentów". W ramach tej techniki należałoby określić w sposób abstrakcyjny konstrukcję i sposób działania maszyny przetwarzania dokumentów.

W praktyce jednak maszyny pdk realizowane czy budowane są przy wykorzystaniu sprzętu komputerowego. Maszyna pdk powstaje przez połączenie dwu składników, którymi są:
- zestaw komputerowy, który stanowi w istocie pewien zespół sprzę-

towo-programowy obejmujący zestaw urządzeń oraz system operacyjny; system operacyjny steruje współpracą urządzeń wchodzących w skład zestawu i powoduje, że działają one jako całość;
- oprogramowanie użytkowe działające na zestawie komputerowym.

Ogólnie biorąc właściwości maszyn pdk zależne są zarówno od właściwości zestawu komputerowego, jak i od właściwości oprogramowania użytkowego. Przy czym pewne właściwości zależne są głównie lub wyłącznie od właściciwości zestawu komputerowec, a inne głównie lub wyłącznie od właściwości oprogramowania użytkowego.

W szczególności maszyna pdk umożliwiająca jednoczesną pracę na wielu stanowiskach przetwarzania może być utworzona jedynie w oparciu o zestaw komputerowy o odpowiedniej budowie i właściwościach.

Zestaw taki będziemy określali jako "zestaw wielodostępny" lub "wielostanowiskowy zestaw komputerowy"

Zestaw taki zawiera zespół sterująco-przetwarzający, do którego podłączone są:
- urządzenia obsługi nośników danych składowanych,
- urządzenia wprowadzania/wyprowadzania danych,
- stanowiska działania użytkowników zestawu.

Zespół sterująco-przetwarzający może mieć dowolną konstrukcję - konieczne jest jedynie, aby zapewniał możliwość jednoczesnej pracy na stanowiskach. Wynika stąd, że możliwe są różne konstrukcje zespołu sterująco - przetwarzającego, a tym samym zestawy wielodostępne o różnych konstrukcjach, a tym samym o różnych właściwościach.

Różne właściwości zestawów będą powodować, że różne też będą wartości użytkowe czy też przydatność zestawów dla użytkowników.

Powstaje tym samym problem jaka konstrukcja zespołu sterująco- przetwarzającego zapewnia posiadanie przez zestaw wielodostęp-

czy właściwości najbardziej odpowiednich dla użytkowników, czyli zapewnia największą przydatność czy też wartość użytkową zestawu. By ten problem rozwiązać należy najpierw ustalić jakie właściwości zestawu wielodostępnego są najbardziej pożądane czy odpowiednie dla użytkowników, a następnie ustalić jaki rodzaj zestawów posiada te właściwości w największym stopniu.

2. WŁAŚCIWOŚCI WIELODOSTĘPNYCH ZESTAWÓW KOMPUTEROWYCH OKREŚLAJĄCE POZIOM ICH PRZYDATNOŚCI DLA UŻYTKOWNIKÓW

Poziom przydatności wielodostępnego zestawu komputerowego dla użytkownika czy też stopień w jakim zestaw taki odpowiada potrzebom i wymaganiom użtkowników określony jest przez stan następujących właściwości:

W1. Rodzaj i wielkość zdolności czynnościowych jakie może posiadać zestaw, co wynika z rodzaju i wielkości zasobów jakie mogą zawierać się w zestawie,

W2. Łatwość zwiększania zdolności przerobowych zestawu w miarę wzrostu zapotrzebowania na nie - wraz ze wzrostem zakresu zastosowań,

W3. Jakość i warunki pracy użytkowników zestawu,

W4. Poziom niezawodności i ciągłości działania zestawu.

W dalszym ciągu określimy bliżej poszczególne właściwości.

2.1. RODZAJ I WIELKOŚĆ ZDOLNOŚCI CZYNNOŚCIOWYCH I ZASOBÓW ZESTAWU

Z punktu widzenia każdego użytkownika zestaw wielodostępny musi spełniać następujące wymagania:

- umożliwiać przechowywanie dostatecznie dużych ilości danych i zapewniać natychmiastowy dostęp do każdej z nich,
- posiadać dostatecznie dużą ilość stanowisk pracy pozwalających na równoczesny dostęp do danych i zapewniać szybką, w zasadzie natychmiastową obsługę tych stanowisk,
- umożliwiać skuteczną koordynację dostępu do danych przy jednoczesnym ich użytkowaniu przez różne stanowiska pracy i umożliwiać w ten sposób tylko jednorazowy zapis każdej danej elementarnej,
- umożliwiać przechowywanie danych przez odpowiednio długi okres czasu, w szczególności umożliwiać tworzenie "wieczystych", nie-

zniszczalnych zapisów danych,

- umożliwiać skuteczne regulowanie zakresu dostępu do danych przez różnych użytkowników polegające na udostępnianiu każdemu użytkownikowi tylko tych danych, które są mu niezbędne do pracy.

Spełnianie tych wymagań zależne jest od poziomu odpowiednich zdolności czynnościowych zestawu, zaś poziom ten określony jest przez wielkość odpowiednich zasobów zawierających się w zestawie oraz przez konstrukcję zespołu sterująco - przetwarzającego. I tak:

- ilość danych, które można przechowywać w zestawie określona jest przez pojemność stałych nośników danych, są nimi przede wszystkim stałe dyski magnetyczne, choć ostatnio pojawiła się możliwość stosowania dysków optycznych niewymazywalnych (tzw. WORM - write one read many - jeden zapis - wiele czytań) oraz wymazywalnych,
- szybkość dostępu do danych zależna jest od jakości stałych nośników danych, czyli głównie dysków magnetycznych oraz od konstrukcji i sposobu działania urządzeń biorących udział w obsłudze tych nośników; są to m.in. tzw. sterowniki, ale nie tylko;
- ilość stanowisk zależna jest od konstrukcji zespołu sterująco - przetwarzającego,
- szybkość obsługi stanowisk pracy zależna jest od szybkości dostępu do danych oraz od szybkości ich przetwarzania; szybkość przetwarzania zależna jest od konstrkcji zespołu sterująco - przetwarzającego, a przede wszystkim od mocy i ilości procesorów w tym zespole i sposobu ich współdziałania oraz od wielkości pamięci operacyjnej i sposobu jej wykorzystania przez procesory,
- dla długotrwałego przechowywania danych stosowane są nośniki zewnętrzne czy archiwacyjne; dotychczas były to głównie taśmy magnetyczne, mają one jednak poważne wady; ostatnio pojawiła się

możliwość stosowania dysków optycznych, które są znacznie lepsze od taśm magnetycznych. Dyski optyczne niewymazywalne (WORM) pozwalają na tworzenie zapisów niezniszczalnych, "wieczystych".

- skuteczność koordynacji dostępu do danych i regulowanie zakresu tego dostępu określona jest przez właściwości programu organizującego pracę zespołu sterowniczo - przetwarzającego czyli tzw. systemu operacyjnego.

2.2. ŁATWOŚĆ ZWIĘKSZANIA ZDOLNOŚCI PRZEROBOWYCH ZESTAWU

W procesie użytkowania zestawu komputerowego w pewnej chwili powstaje zwykle taka sytuacja, że jego zdolność przerobowa jest w pełni wykorzystana, a równocześnie pojawia się wymaganie rozszerzenia zakresu stosowania zestawu. Spełnienie tego wymagania może nastąpić jedynie przez zwiększenie zdolności przerobowej zestawu.

Możliwe są wtedy dwie sytuacje:

- zwiększenie zdolności przerobowej zestawu można uzyskać przez zwiększenie zasobów zestawu przy zachowaniu bez zmian zasobów istniejących,
- zwiększenie zdolności przerobowej zestawu może nastąpić jedynie przez wymianę zestawu na inny lub też przez wymianę istotnych elementów zestawu, tych które określają poziom jego zdolności przerobowej.

Z punktu widzenia użytkownika najbardziej korzystna jest sytuacja pierwsza, czyli istnienie możliwości rozszerzenia zestawu istniejącego. Czyli dla użytkowników najbardziej odpowiednie są zestawy, których konstrukcja pozwala na zwiększenie ich zdolności przerobowej przez wprowadzanie nowych zasobów.

Wszystkie rodzaje zestawów posiadają w pewnym stopniu takie możliwości lecz większość z nich tylko do pewnego poziomu zasobów, który można określić jako zasoby maksymalne zestawu.

Po osiągnięciu zasobów maksymalnych rozbudowa zestawu staje się niemożliwa - zwiększanie zdolności przerobowych można wtedy uzyskać tylko przez wymianę całego zestawu lub jego elementów.

Dla użytkowników najbardziej odpowiednie byłyby zestawy nie posiadające w praktyce ograniczenia w postaci zasobów maksymalnych - ograniczenia takie oczywiście zawsze istnieje, lecz traci praktyczne znaczenie gdy zasoby maksymalne są bardzo wielkie, istotnie przekraczające mogące wystąpić w praktyce potrzeby. Wymiana zestawu lub jego elementów jest zawsze zabiegiem bardzo niepożądanym, gdyż powoduje ona przerwanie ruchu zestawu - oznacza to poważne zakłócenie pracy użytkowników oraz jest bardzo kosztowne.

2.3. JAKOŚĆ OBSŁUGI I WARUNKI PRACY UŻYTKOWNIKÓW ZESTAWU

Jakość obsługi i warunki pracy użytkowników zestawu komputerowego są określone przez wiele czynników, lecz najważniejsze z nich to:

1. Średni czas odpowiedzi zestawu na żądanie użytkownika i stabilność tego czasu;
2. Wpływ właściwości zestawu na stosunki między jego użytkownikami.

Najbardziej odpowiedni dla użytkowników jest zestaw zapewniający czas odpowiedzi, który:

- jest bardzo krótki, tak że odpowiedź zestawu jest subiektywnie oceniana jako natychmiastowa,
- jest stabilny, zawsze taki sam, niezależnie od tego, ilu użytkowników współpracuje z zestawem i jakie są ich działania.

Jak dotychczas w praktyce spotyka się głównie zestawy, których sposób zachowania się, przede wszystkim czas odpowiedzi, jest silnie zmienny w czasie - silnie zależy od ilości i sposobu działania użytkowników. Jest to bardzo dla nich uciążliwe i może nawet

dezorganizować ich pracę.

Wspólne użytkowanie zestawu komputerowego przez grupę użytkowników wywiera zwykle określony wpływ na stosunki między nimi. Wynika to stąd, że przy takim wspólnym użytkowaniu warunki i skuteczność pracy każdego z użytkowników są w jakimś stopniu zależne od sposobu działania i zachowania innych użytkowników. Istnieją zestawy komputerowe o takich właściwościach, że działania jednych użytkowników mogą poważnie utrudniać pracę innych z nich. Prowadzi to zwykle do konfliktów między użytkownikami.

Najbardziej odpowiedni dla użytkownika jest zestaw, który nie stwarza przesłanek dla konfliktów między nimi, czyli zestaw, w którym wpływ sposobu działania jednych użytkowników na warunki pracy innych będzie możliwie mały. Zestaw taki można określić jako zestaw zapewniający wysoki poziom niezależności użytkowników.

Silna zależność między użytkownikami pojawia się zawsze w związku ze wspólnym użytkowaniem jakiegoś zasobu zestawu, w szczególności procesora. Każdy użytkownik podejmujący pracę na zestawie angażuje część mocy takiego zasobu, a tym samym ogranicza jego dostępność dla innych użytkowników i utrudnia ich pracę. Musi to prowadzić do konfliktów między nimi.

Konflikty między użytkownikami mogą również pojawić się w związku ze zwiększaniem ilości stanowisk roboczych podłączonych do zestawu.

Mogą tu zaistnieć dwie sytuacje:

- dołączenie nowych stanowisk roboczych do istniejącego zestawu jest możliwe,
- dołączenie nowych stanowisk roboczych do istniejącego zestawu nie jest możliwe i zwiększenie ilości stanowisk roboczych wymaga wymiany lub przebudowy zestawu.

Dołączenie nowych stanowisk może powodować jeden z dwu efektów:

- nie wpływać na warunki pracy dotychczasowych użtkowników,
- utrudniać pracę dotychczasowych użytkowników.

Najbardziej odpowiedni jest zestaw do którego można podłączać nowe stanowiska robocze bez utrudniania pracy dotychczasowym użytkownikom.

Wymiana lub przebudowa zestawu w celu dołączenia nowych stanowisk roboczych wprowadza bardzo powane zakłócenia w pracy dotychczasowych użytkowników i będzie zwykle przez nich źle widziana.

Najbardziej odpowiedni jest zestaw, do którego można wciąż dołączać nowe stanowiska robocze bez konieczności jego całkowitej wymiany lub zasadniczej przebudowy.

2.4. POZIOM NIEZAWODNOŚCI I CIĄGŁOŚCI RUCHU ZESTAWU

Poziom ciągłości ruchu zestawu jest wysoki, jeśli niezawodność zestawu jest wysoka, czyli gdy rzadko występują zacięcia zestawu.

Zacięcie zestawu jest to zjawisko pełnej lub częściowej utraty przez zestaw zdolności do działania, można więc wyróżnić zacięcia pełne i częściowe.

Zacięcie pełne jest zawsze skutkiem zacięcia czy uszkodzenia jednego z kluczowych podzespołów zestawu - zacięcie częściowe jest skutkiem zacięcia podzespołu niekluczowego.

Poziom ciągłości ruchu zestawu jest wysoki, jeśli rzadko występują zacięcia jego podzespołów, czyli jeśli niezawodność tych podzespołów jest wysoka.

Zacięcie podzespołu może mieć dwie przyczyny:
- ujawnienie wad wewnętrznych podzespołu,
- uszkodzenie podzespołu przez czynniki zewnętrzne.

Poziom niezawodności podzespołów określony jest przez dwa czyn-

niki:

- wadliwość wewnętrzna podzespołu,
- podatność podzespołu na uszkodzenie przez czynniki zewnętrzne mogące takie uszkodzenie wywołać, czyli podatność na zagrożenia zewnętrzne.

Zestaw zawierający podzespoły czy elementy o dużej podatności na zagrożenia zewnętrzne nie będzie posiadał wysokiej niezawodności nawet przy wysokiej jakości tych podzespołów. Zestawy posiadające takie podatne na zagrożenia elementy mogą okazać się bardzo kłopotliwe dla użytkownika.

Najbardziej odpowiednie dla użytkowników są zestawy nie posiadające takich podatnych podzespołów.

3. MOŻLIWE SPOSOBY REALIZACJI WIELODOSTĘPNYCH ZESTAWÓW KOMPUTEROWYCH - ZESTAWY JEDNOKOMPUTEROWE I ZESTAWY WIELOKOMPUTEROWE

3.1. WIELODOSTĘPNE ZESTAWY JEDNOKOMPUTEROWE

W początkowej fazie rozwoju techniki komputerowej znany był tylko jeden sposób realizacji wielodostępnych zestawów komputerowych i dlatego sposób ten będziemy w dalszym ciągu określali jako "tradycyjny sposób realizacji" lub "tradycyjna konstrukcję" wzk.

Zestaw komputerowy o tradycyjnej konstrukcji zawiera następujące elementy:

- jednostka centralna składająca się z procesora i pamięci operacyjnej czyli komputer właściwy,
- urządzenia wprowadzania i wyprowadzania danych, jak odpowiednio różne czytniki i drukarki,
- urządzenia składowania danych w postaci przede wszystkim dysków magnetycznych i ich sterowników,
- urządzenia komunikacji z użytkownikiem.

Każde z takich urządzeń zawiera procesor komunikacyjny i podłączone do niego terminale ekranowe.

Praca takiego zestawu organizowana jest przez specjalny program określany jako system operacyjny. Zestaw taki zawiera tylko jedną jednostkę centralną, jeden komputer i dlatego będzie on określany jako "zestaw jednokomputerowy" (w skrócie "zestaw jk").

Wielodostępne zestawy jednokomputerowe powstały w latach 60-tych przez rozwinięcie pierwotnych zestawów komputerowych, które mogły być użytkowane jedynie w trybie jednodostępnym, czyli przez operatora. Rozwinięcie to polegało na dołączeniu procesora komunikacyjnego oraz związanych z nim urządzeń kontaktu z użytkownikiem, którym początkowo były dalekopisy.

Wielodostępność zestawu jk, czyli możliwość jednoczesnej pracy wielu użytkowników powstaje w ten sposób, że komputer właściwy kierowany przez system operacyjny kontaktuje się, za pośrednictwem procesora komunikacyjnego, kolejno z poszczególnymi terminalami, czyli czas pracy komputera dzielony jest między użytkowników. Dlatego też wielodostępne zestawy jk określane są jako zestawy z podziałem czasu.

Od swojego powstania zestawy jk podlegały ciągłemu rozwojowi i doskonaleniu. Rozwijała się technologia elementów elektronicznych i wraz z nią powstawały nowe generacje zestawów komputerowych posiadające jednostki centralne, czyli właściwe komputery, o coraz większej mocy; coraz większe pamięci operacyjne i dyskowe, jak również coraz bardziej sprawne procesory komunikacyjne i coraz bardziej dostosowane do potrzeb użytkownika terminale.

Nie ulegała jednak żadnym zmianom zasada podziału czasu stanowiąca istotę sposobu realizacji wielodostępności w zestawach jk.

Przez dłuższy czas miało to swoje poważne przyczyny ekonomiczne. Początkowo procesory były realizowane jako zespoły wielkiej ilości elementów elektronicznych, a następnie układów scalonych, procesory takie będziemy określać jako "procesory wieloukładowe". Procesory takie były niezwykle kosztowne i trudne w produkcji - ekonomia nakazywała możliwie ograniczać ich ilość, a każdy istniejący dzielić między możliwie wielu użytkowników.
Stosowanie w zestawie komputerowym więcej niż jednego komputera właściwego stwarzało bardzo krótko poważne problemy ekonomiczne. W czasach procesorów wieloukładowych zestawy jednokomputerowe wydawały się jedynym naturalnym i sensownym rozwiązaniem.

Gdy około 1985 roku pojawiły się mikroprocesory dostatecznie dużej mocy, zostały one również użyte do konstruowania zestawów jk. Nie jest to już jednak działanie racjonalne - cena 1 MIPS-a

mikroprocesorze jest kilkanaście razy mniejsza niż w procesorze wieloukładowym i znosi to ekonomiczne przesłanki do trzymania się konstrukcji jednokomputerowych. Niecelowość ich stosowania dokładniej uzasadnimy w dalszym ciągu. Oczywiście na skutek działania kilku czynników, z których głównym jest trudność wyzwolenia się przez różnych działaczy komputerowych od zakorzenionych wyobrażeń o komputerze, jeszcze przez pewien czas zestawy jk będą produkowane i stosowane, lecz ich dni są już policzone.

3.2. PRZESŁANKI POWSTAWANIA ZESTAWÓW WIELOKOMPUTEROWYCH

Przyszłość nieuchronnie należy do konstrukcji pozwalających na wykorzystanie możliwości stwarzanych przez niski koszt i wielką moc mikroprocesorów i pamięci półprzewodnikowych. Są nimi konstrukcje pozwalające na włączenie do zestawu wielodostępnego i współpracę wielu komputerów. Zestawy takie określimy jako "wielodostępne zestawy wielokomputerowe", w skrócie "zestawy wk".

Zestawy takie pojawiły się w światowej technice komputerowej w połowie lat osiemdziesiątych i ciągle się rozwijają, choć niewątpliwie są to dopiero początki tego rozwoju.

Powstanie zestawów wk było skutkiem rozwoju komputerów osobistych co znowu było skutkiem rozwoju mikroprocesorów i pamięci półprzewodnikowych.

Jak powszechnie wiadomo pojawienie się na rynku bardzo tanich mikroprocesorów i pamięci spowodowało powstanie i szybki rozwój bardzo tanich jednodostępnych zestawów jednokomputerowych określanych jako "komputery osobiste". Początkowo sądzono, że komputery osobiste będą stanowić jedynie indywidualne narzędzia pracy pojedyńczych osób i zawsze nimi pozostaną. Szybko jednak okazało się, że komputery osobiste pozwalają na łatwe i tanie wykonanie wielu zadań występujących w przedsiębiorstwach i instytucjach, których

wcześniej nie dawało się praktycznie rozwiązać przy pomocy tzw. dużych komputerów, czyli zestawów jednokomputerowych z procesorami wielokładowymi. W przedsiębiorstwach powstał znaczny nacisk na zwiększenie zakresu stosowania komputerów osobistych. Oczywiście okazało się wtedy, że dalszy rozwój stosowania komputerów osobistych będzie możliwy jeśli użytkownicy będą mogli korzystać ze wspólnych zbiorów danych czy dokumentów.

Powstała potrzeba zestawu komputerowego, w którym każdy użytkownik pracuje na swoim komputerze osobistym, a równocześnie może korzystać ze wspólnego zbioru danych czy dokumentów. Zestawy takie powstały i są to właśnie wielodostępne zestawy wielokomputerowe, zawierające wiele współpracujących komputerów osobistych, czy ogólnie mikrokomputerów.

3.3. WIELODOSTĘPNE ZESTAWY WIELOKOMPUTEROWE Z ŁĄCZNOŚCIĄ KABLOWĄ

W zestawie takim komputery osobiste występują w dwu rolach:

- jako urządzenia przechowujące ogólnodostępne zbiory danych, komputery te określane są jako "składnice danych/zbiorów", po angielsku "fileserver",
- jako urządzenia wykonujące przetwarzanie na rzecz użytkowników te komputery określane są jako "stanowiska robocze".

Aby zestaw taki mógł działać, wszystkie stanowiska robocze muszą być podłączone do składnicy danych - w początkowej fazie rozwoju takich zestawów dla realizacji tych połączeń wykorzystano technikę sieci lokalnych.

Technika ta rozwinęła się niezależnie od rozwoju komputerów osobistych i równolegle z nim. Jej rozwój wywołany został potrzebą przesyłania danych między różnorodnymi urządzeniami komputerowymi zainstalowanymi w dużych ilościach w amerykańskich przedsiębiorstwach. Powstało szereg standardów określających sposób przesyłania

danych między bliskimi urządzeniami komputerowymi przy wykorzystaniu łączy kablowych i przy zastosowaniu dość wysokich szybkości przesyłania w granicach 2,5 do 10 Mbitów/sek.

Posługując się tymi standardami w krótkim czasie skonstruowano tanie urządzenia pozwalające na przesyłanie danych między komputerami osobistymi i uruchomiono ich masową produkcję.
Urządzeniami tymi są oczywiście tzw. karty sieciowe typu Aronet, Ethernet itd.
Powstała możliwość dostatecznie szybkiego przesyłania danych między komputerami osobistymi.
Połączenie zbioru komputerów nie powoduje jeszcze powstania zestawu wielodostępnego, aby zestaw taki powstał konieczne jest jeszcze odpowiednie zorganizowanie pracy zestawu.
W tym celu powstały tzw. sieciowe systemy operacyjne. System taki działa na komputerze osobistym, będącym składnicą danych i organizuje przesyłanie danych między składnicą a stanowiskami roboczymi.
W każdym stanowisku roboczym działa stanowiskowy system operacyjny, który organizuje wykonywanie programów na stanowisku roboczym oraz prowadzi wymianę danych ze składnicą.
Na rynku światowym występuje obecnie kilka sieciowych systemów operacyjnych, lecz dominującą pozycję zajmuje na nim system Netware firmy Novel. Wynika to stąd, że system ten zapewnia składnicy największe możliwości funkcjonalne oraz najwyższą sprawność działania, przede wszystkim sprawność wymiany danych.
Opis systemu Netware zawiera opracowanie (3).

Stanowiskowy system operacyjny składa się z systemu MS-DOS oraz współpracującego z nim dodatkowego modułu programowego organizującego wymianę danych ze składnicą. Moduł ten określany jest jako "network shell".

Wielodostępne zestawy wielokomputerowe oparte o sieć lokalną,

czyli o łączność kablową okazały się bardzo dogodne dla użytkowników i znalazły szerokie zastosowanie. W roku 1989 istniało na świecie ok. 300.000 zestawów tego typu kierowanych przez system Netware, co oznacza, że łączna liczba tego typu zestawów wynosi ok. 0,5 mln.

Okazało się, że zestawy sieciowe pod wieloma względami przewyższają tradycyjne jednokomputerowe zestawy wielodostępne i obserwuje się, jak to pokazują publikacje (5), (9) procesy zastępowania tradycyjnych zestawów typu duże komputery (mainframes, czy minikomputery przez zestawy oparte o sieci lokalne. Nie jest to jak na razie zjawisko masowe, choć w komentarzu (9) wyraźnie stwierdza się, że upadek tradycyjnych zestawów jest nieuchronny.

Praktyczne doświadczenia pokazały, że wąskim gardłem zestawów sieciowych jest oparty o kabel układ łączności między komputerami osobistymi. Ograniczona przepustowość tego układu, która nie przekracza 1 Mbita/sekundę, niezależnie od rodzaju tzw. kart sieciowych, powoduje że przy ilości stanowisk roboczych większej niż 12 - 15 sprawność systemu, mierzona głównie szybkością jego odpowiedzi, silnie spada. Skutkiem tego średnia ilość stanowisk roboczych w zestawie sieciowym z jedną składnicą wynosi 7 - 8.
Fakt ten stanowi wyraźną barierę dalszego rozwoju zestawów sieciowych.

3.4. WIELODOSTĘPNE ZESTAWY WIELOKOMPUTEROWE Z ŁĄCZNOŚCIĄ SZYNOWĄ

Poszukując wyjścia z tej sytuacji zauważono, że sieć lokalna stanowi jedynie środek łączności między komputerami osobistymi i jest ona dla użytkowników całkowicie niewidoczna. Właściwości użytkowe i funkcjonalne takiego "zestawu" nie ulegną żadnej zmianie jeśli kablowy układ łączności zapewniający co najmniej te same możliwości przesyłania danych między komputerami. I takie układy zostały wynalezione, powstała nowa konstrukcja wielodostępnego zestawu wielokomputerowego. Przedstawiona w (4) konstrukcja opiera się o dwa rozwiązania :

- mikrokomputery stanowiące stanowiska robocze mają postać płytek tak dodatkowych dla PC/AT. Mikrokomputery te będziemy określać jako "płytkowe stanowiska robocze" lub "komputery płytkowe". Komputer na płytce może być typu PC/XT lub PC/AT i każdy z tych komputerów posiada maksymalnie ograniczone wyposażenie: mikroprocesor, pamięć RAM i sterownik do łączności z ekranem. Jedna płytka może zawierać 1,2 lub nawet cztery oddzielne komputery;
- takie płytki z komputerami wstawiane są do gniazd szyny głównej komputera będącego składnicą danych i przekazywanie danych między płytkowymi stanowiskami roboczymi a składnicą odbywa się poprzez szynę wewnętrzną składnicy.

Zestaw o takiej konstrukcji będziemy określać jako "zestaw wielokokomputerowy z łącznością szynową" lub też "szynowy zestaw wielokomputerowy".

Szynowy zestaw wk jest to więc zestaw komputerowy, którego składnikami są :

- składnica danych w postaci typowego komputera typu PC/AT,
- płytkowe stanowiska robocze włączone do gniazd szyny wewnętrznej składnicy.

Przekazywanie danych między składnicą a stanowiskami roboczymi od-

bywa się poprzez szynę wewnętrzną z szybkością 16 Mbitów/ sekundę czyli 16 razy szybciej niż w zestawie opartym o kablową sieć lokalną.

W składnicy działa system Netware przystosowany do obsługi przekazywania danych przez szynę wewnętrzną zamiast przez sieć lokalną - z tym jednak, że normalne połączenia sieciowe nadal mogą być stosowane.

Możliwe jest tworzenie zestawów mieszanych, z łącznością szynową i kablową w ramach tego samego zestawu.

Stosowanie systemu Netware w zestawach z łącznością szynową wskazuje, że określenie tego systemu jako "sieciowego systemu operacyjnego" nie jest trafne - jak widać jest on stosowany również wtedy gdy żadna sieć nie występuje.

Istotą działania systemu Netware jest organizowanie działania wielodostępnego zestawu wielokomputerowego i w związku z tym system Netware należy określić jako system operacyjny wielodostępnego zestawu wielokomputerowego.

Znane mi są w tej chwili trzy firmy produkujące płytkowe stanowiska robocze przystosowane do współpracy z systemem Netware, są to:

- Cubix Corp., Carson City, Nevada
- Advanced Digital Corp., Huntington Beach, California
- Integrated Workstations, Inc.

Jednak tylko firma Cubix Corp. produkuje stanowiska płytkowe pozwalające na współpracę z terminalami poprzez łącza telefoniczne RS 232/422. Tylko takie stanowiska płytkowe pozwalają na pracę użytkowników w dowolnej odległości od składnicy do której są one podłączone. W związku z tym jedynie stanowiska płytkowe firmy Cubix są dla nas praktycznie interesujące. Produkty tej firmy będą nieco dokładniej omówione w dalszym ciągu.

W następnym rozdziale przedstawimy porównanie właściwości użytkowych poszczególnych rodzajów zestawów wielodostępnych posługując się kryteriami określonymi w rozdziale 2.

4. PORÓWNANIE WŁAŚCIWOŚCI UŻYTKOWYCH I POZIOMU PRZYDATNOŚCI DLA UŻYTKOWNIKÓW RÓŻNYCH RODZAJÓW ZESTAWÓW WIELODOSTĘPNYCH

Właściwości użytkowe i poziom przydatności dla użytkowników wielodostępnych zestawów komputrowych (wzk) określone są, jak to wcześniej stwierdziliśmy przez następujące cechy zestawów:

1. rodzaje zdolności czynnościowych występujących w zestawie i górne granice ich powiększania,
2. łatwość zwiększania zdolności czynnościowych zestawu w miarę wzrostu potrzeb,
3. jakość obsługi i warunki pracy użytkowników,
4. poziom niezawodności i ciągłości działania zestawu.

W dalszym ciągu określimy poziom i stan tych cech dla opisanych wcześniej trzech rodzajów zestawów wielodostępnych, którymi są:

- zestawy jednokomputerowe,
- zestawy wielokomputerowe z łącznością kablową,
- zestawy wielokomputerowe z łącznością szynową.

4.1. RODZAJE ZDOLNOŚCI CZYNNOŚCIOWYCH ZESTAWÓW I GÓRNE GRANICE ICH POWIĘKSZANIA

Każdy rodzaj wzk posiada wszystkie niezbędne zdolności czynnościowe czyli:

- zdolność równoczesnej obsługi wielu stanowisk roboczych z koordynacją dostępu do danych,
- zdolność składowania danych,
- zdolność archiwacji danych.

Miarą zdolności równoczesnej obsługi jest ilość stanowisk roboczych obsługiwanych w sposób subiektywnie natychmiastowy tzn. w czasie niedoczuwalnym dla użytkownika jako oczekiwanie przy wyko-

rywaniu przez użytkowników ich typowych zadań.

Miara zdolności składowania jest mierzona w MB ilość danych która może być umieszczona na nośnikach o bezpośrednim dostępie.

Miara zdolności archiwacji jest ilość danych umieszczonych na nośnikach archiwacyjnych do których dostęp może być uzyskany w dostatecznie krótkim czasie, np. w ciągu 3 min.

Istotną właściwością każdego rodzaju wzk jest górna granica wielkości każdej zdolności czynnościowej jaka może być osiągnięta przez konkretny zestaw danego rodzaju jak też przez przodujący czyli najsilniejszy zestaw danego rodzaju.

Granice te są różne dla zestawów różnych rodzajów. Praktyczne znaczenie posiada przede wszystkim górna granica zdolności równoczesnej obsługi tzn. w wielu wypadkach potrzebna jest ilość stanowisk znacznie większa niż osiągalna przez rozpatrywany wzk.

Zdolność równoczesnej obsługi stanowisk roboczych.

Zdolność obsługowa zestawu określona jest przez łączną moc efektywną komputerów czy też jednostek centralnych działających w zestawie. Określimy ją jako moc przerobową zestawu. Moc efektywna może być różna od mocy nominalnej tych komputerów, gdyż w pewnych konstrukcjach zestawów pełna nominalna moc procesorów wchodzących w skład komputerów nie może być wykorzystana. Zjawisko to występuje głównie w tzw. komputerach wieloprocesorowych.

Zestawy jednokomputerowe.

Łączna moc przerobowa każdego konkretnego jak również przodującego zestawu jednokomputerowego posiada ściśle określoną górną granicę.

W zestawie konkretnym moc ta jest równa mocy jedynego komputera jednostki przetwarzającej występującej w zestawie. Zdolność obsługowa zestawu posiada tym samym ściśle określoną górną granicę wynikającą z mocy tego jedynego komputera.

Moc przerobowa zestawów przodujących jest zmienna w czasie i w każdej chwili określona jest przez poziom technologii elektronicznej czy może dokładniej poziom technologii budowy komputerów/ jednostek przetwarzających na co składa się technologia budowy procesorów, pamięci operacyjnych oraz szyn je łączących. Wraz z rozwojem technologii następuje wzrost mocy przerobowej zestawów przodujących. W roku 1988 moc ta wynosiła ok. 30 MIPSów, zaś w 1989 roku pojawiły się nowe konstrukcje firmy DEC VAX 9000 mogące osiągać prawie 100 MIPSów.

Zestawy wielkomputerowe.

Górna granica mocy przerobowej konkretnego jak również przodującego zestawu wk nie jest określona przez górną granicę mocy pojedynczego komputera. Zestaw wk zawiera wiele komputerów górna granica jego mocy zależy głównie od tego jak wiele komputerów można połączyć w jednym zestawie, a to zależy od możliwości czy dokładniej przepustowości układu łączącego komputery.

Przepustowość kablowego układu łączności jest znacznie niższa niż szynowego układu łączności. Przepustowość ta w układach szeroko stosowanych w roku 1989 odpowiednio wynosiła:

- układy łączności kablowej 1 Mbit/sek,
- układy łączności szynowej 16 Mbitów/sek.

W zestawie z łącznością kablową jedna składnica może efektywnie współpracować z ok. 12 - 16 stanowiskami roboczymi. Ilość stanowisk roboczych mogących efektywnie współpracować ze składnicą w zestawie z łącznością szynową nie jest dokładnie zbadana, lecz realne jest stworzenie efektywnego zestawu zawierającego 40-50 stanowisk.

Moc przerobowa takiego zestawu nie jest jednoznacznie określona, gdyż stanowiska robocze mogą być różne i mieć różną moc. Przy zastosowaniu stanowisk z komputerem typu PC/AT łączna moc

przerobowa zestawu ze składnicą typu PC/AT/386 wyniesie:

4 + 2 x 40 = 84 MIPSy.

Biorąc pod uwagę powyższe dane można ustalić górną granicę mocy przerobowej zestawu wk z jedną składnicą - wartość jej dla zestawów z łącznością szynową będzie znacznie wyższa niż dla zestawów z łącznością kablową.

Składnice można jednak łączyć ze sobą i zestaw wk może posiadać wiele składnic - system Netware nie stwarza jak się wydaje żadnych ograniczeń niż ich ilość. W każdym razie podczas Wystawy Sieciowej w USA w r. 1987 działał zestaw zawierający ok. 100 składnic oraz 1300 stanowisk roboczych. W firmie Novell działa zestaw wk zawierający 209 składnic rozrzuconych po całych Stanach Zjednoczonych, lecz stanowiących scalony zestaw.

Takie możliwości łączenia składnic powodują, że dla zestawów wk nie można określić górnej granicy mocy przerobowej i zdolności obsługowej - można je zwiększać nieograniczenie.

Zdolność składowania danych

Zdolność składowania danych zestawu określona jest przez ilość jednostek wchodzących w jego skład oraz pojemność tych jednostek.

Górna granica zdolności składowania określona jest przez maksymalną ilość jednostek dyskowych jaka może być włączona do zestawu oraz maksymalna pojemność tych jednostek. Zestawy jednokomputerowe posiadają zwykle taką konstrukcję, że maksymalna ilość jednostek dołączalnych do zestawu jest ściśle określona, określona jest też maksymalna pojemność jednostki dyskowej. Górna granica zdolności składowania zestawu jk jest więc wyraźnie określona.

Inaczej to wygląda w zestawach wk. Mikrokomputery typu PC pełniące role składnic w zestawach wk mają budowę modułową - brak jest ostrej granicy na ilość dołączalnych sterowników i związanych

z nimi jednostek dyskowych. Ponadto zestaw może zawierać wiele składnic, a na ich ilość nie ma tej wyraźnego ograniczenia.

Z tego punktu widzenia zdolność składowania danych zestawu wk nie ma wyraźnej górnej granicy. System Netware 386 jest w stanie obsługiwać na jednej składnicy dyski o pojemności 32000GB - z praktycznego punktu widzenia jest to wielkość nieskończona.

Zdolność archiwacji danych.

Stosowanie dysków optycznych stwarza nieograniczoną zdolność archiwacji danych. Każdy zestaw do którego można dołączyć dyski optyczne posiada taką nieograniczoną zdolność archiwacji.

4.2. ŁATWOŚĆ ZWIĘKSZANIA ZDOLNOŚCI CZYNNOŚCIOWYCH ZESTAWU

Każdy zestaw jk posiada wyraźnie określoną górną granicę zwiększania zdolności obsługowej oraz składowania danych. Zwiększanie ilości stanowisk roboczych oraz pojemności nośników danych jest dość łatwe do chwili osiągnięcia tej górnej granicy. Po jej osiągnięciu większa zdolność czynnościowa może być uzyskana tylko przez wymianę zestawu lub jego istotnych podzespołów. Jest to zabieg trudny, kosztowny i kłopotliwy. Zwiększanie zdolności czynnościowych zestawu jk w niektórych wypadkach może okazać się bardzo trudne.

Zestawy wk nie posiadają wyraźnie określonej górnej granicy zdolności czynnościowych. Granicę taką posiada natomiast zestaw wk z jedną składnicą danych. Zwiększanie zdolności czynnościowych takiego zestawu jest łatwe do chwili osiągnięcia kresu jej możliwości. Dalsze zwiększanie tych zdolności może nastąpić przez dołączenie dodatkowej składnicy. Jest to zabieg nieco trudniejszy lecz nie wymaga on naruszenia ciągłości pracy zestawu ani nie powoduje jego istotnej przebudowy. Zwiększanie zdolności czynnościowych zestawu wk jest więc znacznie łatwiejsze niż zestawu jk.

4.3. JAKOSC OBSŁUGI I WARUNKI PRACY UŻYTKOWNIKOW

Jakość obsługi i waruki pracy użytkownika zestawu określon sa przez dwa główne czynniki:

1. średni czas odpowiedzi zestawu i jego stabilność,
2. wpływ użytkowania zestawu na stosunki między jego użytkownikami.

Omówimy wpływ tych czynników dla zestawów różnych rodzajów.

Średni czas odpowiedzi i jego stabilność.

Średni czas odpowiedzi zestawu zależny jest od średniej mocy przerobowej przypadającej na jednego użytkownika zestawu.

W zestawach jednokomputerowych wskaźnik ten w roku 1986 mieścił się w granicach 0.25 MIPS/użytkownika do 1 MIPSa/użytkownika, a w r. 1989 prawdopodobnie przekroczył tę granicę.

W zestawach wielokomputerowych każdy użytkownik posiada oddzielny komputer oraz korzysta ze składnicy. Tym samym na każdego użytkownika przypada moc przerobowa jego komputera oraz pewien ułamek mocy przerobowej składnicy. Wielkość mocy przerobowej przypadającej na użytkownika zależy od rodzaju mikroprocesorów znajdujących się w komputerach użytkowników i będą to wielkości następujące:

Typ mikroprocesora	Moc przypadająca na użytkownika
8088	0.5 - 1 MIPS
80188	2 - 4 MIPS
80386	4 - 8 MIPS

Jak widać średnia moc przerobowa przypadająca na użytkownika w zestawach wk może być znacznie wyższa niż w zestawach jk, tym samym średni czas odpowiedzi zestawów wk będzie przeważnie znacznie krótszy niż zestawów jk.

Stabilność czasu odpowiedzi zestawu zależna jest od zmian w czasie mocy chwilowej dostępnej dla każdego użytkownika.

W zestawie jk moc chwilowa zależy bezpośrednio od aktualnej ilości czynnych użtkowników zestawu i jest tym samym silnie zmienna w czasie. Skutkiem tego również sposób zachowania się zestawu, w szczególności jego czas odpowiedzi są również silnie zmienne w czasie, nie są stabilne - silnie zależą od ilości czynnych użytkowników zestawu i sposobu ich działania.

W zestawach wk każdy użytkownik ma własny komputer i moc dostępna dla użytkownika jest stała w czasie. Zmiany czasu odpowiedzi mogą być powodowane jedynie zmiennym obciążeniem składnicy oraz układu łączności. Jednak poziom obciążenia tych urządzeń może mieć odczuwalny wpływ na czas odpowiedzi jedynie w niektórych, wyjątkowych sytuacjach. W typowych okolicznościach czas ten jest bardzo słabo zależny od ilości i sposobu działania użytkowników i tym samym jest stabilny, a jeśli już zmienia się to w wąskich granicach.

Podsumowując, można stwierdzić, że czas odpowiedzi zestawów jednokomputerowych jest raczej dłuższy i znacznie bardziej niestabilny niż czas odpowiedzi zestawów wielokomputerowych.

Wpływ użytkowania zestawu na stosunki między jego użytkownikami.

Wpływ ten jest ściśle związany z warunkami dostępności komputera/jednostki centralnej dla użytkowników zestawu. Warunki te są istotnie różne w zestawach jedno- i wielokomputerowych.

W zestawach jk występuje zjawisko wspólnego użytkowania komputera przez wielu użytkowników zestawu. Zjawisko to posiada charakter konfliktogenny - wspólne użytkowanie jednostki centralnej musi nieuchronnie prowadzić do konfliktów między użytkownikami. Jest tak gdyż każdy użytkownik podejmujący użytkowanie zestawu odbiera część czasu pracy komputera pozostałym użytkownikom, i tym samym utrudnia ich pracę.

Konflikty pojawiają się również, gdy zainstalowana ilość końcówek/stanowisk roboczych staje się zbyt mała i pojawia się konie-

czność zwiększenia ich liczby. Czasami jest to możliwe w ramach istniejącego zestawu czasami konieczna jest wymiana procesora lub też całego zestawu na nowy, o większych możliwościach.
Konfliktogenne są obie sytuacje. Dołączenie nowych końcówek do istniejącego zestawu powoduje widoczne obniżenie sprawności zestawu i pogorszenie warunków pracy z punktu widzenia dotychczasowych, "starych" użytkowników. Wymiana procesora czy zestawu jest kosztowna i kłopotliwa, powoduje ona poważne zakłócenia w pracy użytkowników, a także całej organizacji.

W zestawach wk nie występuje zjawisko wspólnego użytkowania jednego komputera - każdy użytkownik posiada w nim swój własny, oddzielny komputer o mocy dobranej do jego potrzeb. Nowy użytkownik podejmujący użytkowanie zestawu włącza się wraz ze swoim komputerem i tym samym nie odbiera nic starym użytkownikom.

Wspólnie użytkowane są jedynie składnica danych oraz układ łączności i tylko w związku z tym działalność jednego użytkownika może wywierać wpływ na jakość obsługi i warunki pracy innych użytkowników. Wpływ ten pojawia się, gdy obciążenie składnicy i układu łączności jest bliskie jego przepustawości co następuje na skutek podłączenia nadmiernej ilości stanowisk roboczych.

W zestawach wk pogorszenie jakości obsługi i warunków pracy użytkowników może nastąpić jedynie na skutek podłączenia nadmiernej ilości stanowisk roboczych, a nie może wystąpić nagle, jak to zdarza się w zestawach jk na skutek uruchamiania przez użytkowników o dużym zapotrzebowaniu na czas pracy komputera. Czas odpowiedzi zestawu wk w normalnych warunkach prawie zupełnie nie zależy od sposobu działania innych użytkowników, a jedynie od mocy komputera własnego danego użytkownika.

Przewaga zestawów wk ujawnia się głównie w sytuacji, gdy pojawia się konieczność zwiększenia liczby stanowisk roboczych - no-

we stanowisko pojawia się łącznie z własnym komputerem i jeśli nie pojawia się przeciążenie układu łączności, to jego wprowadzenie jest bardzo słabo odczuwalne przez "starych" użytkowników.
W zestawach jk "starzy" użytkownicy zawsze tracą część czasu komputera i w każdej sytuacji wyraźnie, a czasem mocno odczuwają wejście nowego użytkownika.

Przewaga zestawów wk jest najwyraźniejsza w sytuacji, gdy istniejąca ilość stanowisk roboczych wyczerpuje moc zestawu. W zestawach wk można dołączyć dodatkową składnicę nie naruszając istniejących elementów zestawu - może jedynie powstać potrzeba zmiany rozmieszczenia danych na nośnikach dyskowych. Możliwa jest też wymiana składnicy na nową o większej przepustowości i wtedy znaczna część zestawu tj. stanowiska robocze z komputerami nie ulegają żadnej zmianie. Dołączenie nwej składnicy słabo zakłóca działalność "starych" użytkowników, a "nowi" uzyskują natychmiastowy dostęp do "żywego" działającego zestawu.

4.4. POZIOM NIEZAWODNOŚCI I CIĄGŁOŚCI RUCHU ZESTAWU

Poziom niezawodności zestawów jk i zestawów wk/s jest podobny. W zestawach wk obu rodzajów bardzo łatwo jest co prawda zastosować zdwojone dyski stałe, gdyż standardowo oferowana jest odpowiednia wersja systemu Xetware i pozwala to na łatwe uzyskanie zestawu wk o podniesionej niezawodności. Lecz nie jest to skutek istotnych różnic między zestawami jedno- i wielokomputerowymi, lecz skutek różnic w polityce i inwencji producentów różnych rodzajów zestawów.
Nie ma żadnych przeszkód, aby zdwojone dyski oferować również standardowo w zestawach jk.

Istotne różnice pod względem niezawodności występują natomiast między zestawami wk/s i wk/k. Kluczowym podzespołem każdego

zestawu wk jest układ łączności między komputerami zestawu - zacięcie tego układu powoduje unieruchomienie całego zestawu.

W zestawie wk/k układ łączności składa się z kabla oraz znacznej ilości tzw. kart sieciowych - jest on więc rozległy przestrzennie i zawiera wiele elementów.
Ochrona takiego układu przed działaniem szkodliwych czynników zewnętrznych jest trudna i tym samym jest on (przede wszystkim kabel) narażony na zagrożenia zewnętrzne. Sygnały elektryczne przesyłane przez kabel mogą łatwo ulec zakłóceniu, a sam kabel ulec uszkodzeniu.

Układ łączności zestawu wk/k jest więc podzespołem silnie narażonym na zagrożenia zewnętrzne, jest najsłabszym podzespołem całego zestawu.

Zupełnie inne właściwości posiada układ łączności zestawu wk/s. Podstawą tego układu jest szyna wewnętrzna składnicy - znajduje się ona wewnątrz komputera i zajmuje mało miejsca, tym samym jest dobrze chroniona przed działaniem zagrożeń. W normalnych warunkach nie jest możliwe zakłócenie sygnałów przesyłanych po tej szynie ani jej uszkodzenie. Poziom niezawodności układu łączności zestawu wk/s jest podobny jak poziom niezawodności innych układów elektronicznych zestawu.

W zestawach jk występuje również układ łączności między podzespołami zestawów - ich podstawą jest również szyna wewnętrzna komputerów, tym samym ich poziom niezawodności jest podobny jak układu łączności w zestawie wk/s.

Można więc wyróżnić dwie klasy niezawodnościowe zestawów komputerowych:

- zestawy z normalnym poziomem niezawodności: zestawy jk i zestawy wk/s
- zestawy z obniżonym poziomem niezawodności: zestawy wk/k.

5. PŁYTKOWE STANOWISKA ROBOCZE WSPÓŁPRACUJĄCE Z SYSTEMEM NETWARE OFEROWANE NA RYNKU ŚWIATOWYM - FIRMA CUBIX CORPORATION JAKO GŁÓWNY PRODUCENT

Według posiadanych przez nas wiadomości w chwili obecnej płytkowe stanowiska robocze współpracujące z systemem Netware produkowane są przez trzy firmy ze Stanów Zjednoczonych.
Najbardziej interesujące są produkty firmy Cubix Corp, gdyż pozwalają one stworzyć zestaw wk współpracujący z terminalami przez łącza RS232.

5.1. PŁYTKOWE STANOWISKA ROBOCZE FIRMY CUBIX CORPORATION

Siedziba firmy Cubix Corp. znajduje się w Carson City, Nevada. Jej przedstawicielstwo działa pod Londynem. Firma Cubix Corp. produkuje serię płytkowych stanowisk roboczych o nazwie QuickLink (w skrócie QL)
W skład serii wchodzą między innymi:

- płytka QL 1004 zawierająca cztery oddzielne komputery typu PC/AT z pamięcią 768 kB każdy. Każdy z tych komputerów wyposażony jest w gniazdo typu RJ45 będące miniaturową wersją gniazda typu RS232 do którego podłączany jest terminal pełniący taką rolę jaką w typowym komputerze PC/XT pełni monitor i klawiatura;
- płytka QL 1002 zawierająca dwa oddzielne komputery typu PC/XT z pamięcią do 768 kB każdy. Każdy z nich wyposażony jest w gniazdo typu RS232 służące do podłączania terminala głównego podobnie jak w QL 1004. Oprócz tego każdy z komputerów wyposażony jest w gniazdo COM1 oraz LPT1;
- płytka QL 2256 zawierająca dwa oddzielne komputery typu PC/AT z pamięcią podstawową do 1 MB rozszerzalną do 2 MB za pomocą płytki dodatkowej. Każdy z komputerów przystosowany jest do współ-

pracy z monitorem i klawiatura w oparciu o sterownik typu EGA. Monitor i klawiatura nie są podłączane bezpośrednio do płytki łącz do specjalnego bloku pośredniczącego, który jest podłączony do płytki za pomocą kabla o długości do 100 stóp (60 m), jest to kabel 36 przewodowy. Tak więc z komputerami na tej płytce nie mogą współpracować terminale. Każdy z komputerów wyposażony jest w gniazdo COM1 i LPT.

- zapewniana jest płytka CL EF88T zawierająca dwa oddzielne komputery typu PC/AT pracująca z terminalami poprzez łącza RS232

Zamiast łącza RS232 może być także stosowane łącze RS422, które umożliwia używanie kabla o długości do 4000 stóp, jak również większych szybkości przesyłania danych.

Płytki CL 1003/1004 mogą współpracować z dwoma rodzajami terminali:

- terminale typu ANSI jak VT 100, WYSE 50,
- terminale znakowe typu PC (PC terminals), jak WYSE 60, WYSE 150,
- terminale graficzne typu PC (PC graphic terminals), jak WYSE PPGT

Terminal znakowy typu PC posiada typową klawiaturę PC/AT oraz obsługę ekranu identyczną jak obsługa ekranu przez kartę Hercules - tryb znakowym czyli ekran zawiera 25 linii po 80 znaków.

Terminal graficzny typu PC zapewnia dodatkowo obsługę ekranu w trybie graficznym zgodnym z kartą Hercules, jak również z kartą CGA

Stosowanie terminali typu PC zapewnia pełną wykonalność na płytkach CL 1003, 1004 wszystkie programów przystosowanych do współpracy z kartą Hercules

Przeciętny komputer typu PC AT posiada szynę wewnętrzną wyposażoną w nie więcej niż 10 gniazd, z których zwykle tylko 3 lub 4

są wolne. Do takiego komputera możnaby więc włączyć nie więcej niż 4 płytkowe stanowiska robocze i uzyskać zestaw z ilością terminali nie większą niż 12 - 16, tymczasem możliwości łączności poprzez szynę są znacznie większe.

W związku z tym firma Cubix produkuje także bloki rozszerzania szyny:

- GL 101 zawierający 8 gniazd podłączenia do szyny,
- GL 102 zawierający 12 gniazd podłączenia do szyny.

Do jednej składnicy można jednocześnie podłączyć dwa bloki rozszerzenia co pozwala uzyskać 22-24 wolne gniazda łączeniowe. Instalując w tych gniazdach 22-24 płytki GL 1004 można więc uzyskać zestaw zawierający 88-96 terminali, instalując 22-24 płytki GL 1002/2286 można uzyskać 44-48 terminali/stanowisk roboczych.

Łączna moc procesorów w zestawach zawierających 22-24 płytki serii GL wyniesie więc:

- GL 1004 - 45 - 50 MIPS,
- GL 1002 - 22 - 25 MIPS,
- GL 2286 - 66 - 98 MIPS.

Oczywiście można tworzyć dowolne zestawy różnych typów płytek GL i kształtować zestaw dokładnie według potrzeb.

Takie zestawy elementarne można łączyć ze sobą i w ten sposób tworzyć zestawy wielodostępne o praktycznie nieograniczonej ilości terminali/stanowisk roboczych.

Należy to liczyć się z tym, że wkrótce powstaną specjalne wersje komputerów typu PC/AT pozwalające na jeszcze dalsze zwiększanie ilości gniazd łączeniowych szyny i tym samym łatwe tworzenie zestawów o jeszcze większej ilości terminali/stanowisk roboczych.

Widzimy więc, że stosując płytki typu GL można uzyskać zestaw wielodostępny, który posiada wszelkie możliwości stwarzane przez zestawy wielodostępne realizowane w oparciu o klasyczne zes-

tawy jednokomputerowe, a równocześnie znacznie większą sprawność i przydatność dla użytkownika. Większa sprawność wynika z relatywnie znacznie większej mocy zestawu, większa przydatność zestawu wynika z tego, że użytkownik pracuje w istocie na komputerze osobistym, które są znacznie bardziej dostosowane do potrzeb użytkowników niż stanowiska robocze terminale zestawów klasycznych.

Firma Cubix dostarcza również program CB 500, który pozwala na wykorzystanie w charakterze terminala dowolnego komputera typu PC XT AT. Program CB 500 powoduje, że komputer ten zachowuje się jak terminal graficzny typu PC.

Firma Cubix dostarcza także pakiet CB 500, który umożliwia współpracę z płytkowym stanowiskiem roboczym, a tym samym z całym zestawem zewnętrznego komputera PC/XT/AT. Pakiet zawiera dwa programy: AMODEM i GCOMMECT.

Program AMODEM działa na płytce GL i obsługuje przesyłanie zbiorów między komputerem zewnętrznym a płytką.

Program GCOMMECT działa na komputerze zewnętrznym i posiada dwa tryby pracy: tryb emulacji i tryb przesyłania zbiorów. W trybie emulacji program GCOMMECT powoduje, że komputer zewnętrzny zachowuje się jak terminal typu PC - program jest niewidoczny dla płytki.

W trybie przesyłania zbiorów program GCOMMECT współpracuje z programem AMODEM działającym na płytce i programy te organizują przesyłanie zbiorów między komputerami.

Pakiet CB 500 znajduje zastosowanie w dwu sytuacjach:

- gdy użytkownik posiada komputery PC/XT/AT i chce je wykorzystać jako terminale
- gdy istnieje potrzeba współpracy z zestawem głównym zewnętrznego komputera wykonującego też samodzielne przetwarzanie.

Komputer zewnętrzny może być do płytki GL podłączony przez

modem, co umożliwia efektywną współpracę z zestawem komputerów oddalonych na znaczną odległość. Efektywna współpraca takiego oddalonego komputera z zestawem opartym o kablową sieć lokalną jest możliwa tylko za pośrednictwem drugiego komputera podłączonego bezpośrednio do składnicy.

Obsługa łączności zestawów opartych o sieć lokalną z innymi oddalonymi zestawami komputerowymi wszelkich rodzajów stanowi w firmy Cubix istotny zakres zastosowań płytek QL.

5.2. PŁYTKOWE STANOWISKA ROBOCZE FIRM ADC I INTEGRATED WORKSTATIONS

Firma ADC (Advanced Digital Corporation) produkuje trzy rodzaje płytkowych stanowisk roboczych określanych przez nią jako PNS (Personal Network Stations):

- PNS 286EGA - zawierające dwa komputery typu PC/AT,
- PNS 2HCG - zawierające dwa komputery typu PC/XT,
- PNS 2MG - zawierające dwa komputery zgodne z PC/XT, lecz wyposażone w mikroprocesor 80186.

Istotną cechą każdej z tych płytek PNS jest to, że komputery na nich zawarte mogą jedynie współpracować z monitorem i klawiaturą poprzez specjalny wielożyłowy kabel o długości do 200 stóp (60 m).

Firma Integrated Workstations produkuje płytkowe stanowisko robocze o nazwie IWAK/286 Networkstation zawierające komputer PC/AT ze sterownikiem monitora obsługującym wszystkie typowe tryby graficzne aż do Super VGA. Współpraca tego komputera z monitorem i klawiaturą odbywa się też poprzez specjalny wielożyłowy kabel o długości do 310 stóp (90 m).

Jak widzimy stanowiska firm ADC i Integrated Workstations nie dają możliwości współpracy z terminalami poprzez łącza telefoniczne

na typu RS232, co w istotny sposób ogranicza zakres ich zastoso-
wań. W większości wypadków efektywny zestaw wielodostępny może być
stworzony jedynie przez wykorzystanie terminali współpracujących z
zestawem poprzez łącza RS232.

6. KOSZTY JEDNOSTKOWE W RÓŻNYCH RODZAJACH WIELOOSTKOWYCH ZESTAWÓW KOMPUTEROWYCH

Obok właściwości użytkowych zestawów komputerowych, które omówiliśmy wcześniej ważna jest również efektywność wydatków na zakup takiego zestawu. Efektywność ta określona jest przez iloraz koszt zakupu/wartość użytkowa, czyli przez koszt jednostki wartości użytkowej.

Obliczenie tego kosztu wymaga przede wszystkim znalezienia miary wartości użytkowej. Problem ten nie znalazł na razie ostatecznego i pełnego rozwiązania, znalazł jednak rozwiązanie przybliżone. W praktyce za miarę potencjalnej czy też maksymalnej osiągalnej wartości użytkowej zestawu uważa się łączną moc komputerów zawartych w zestawie, której miarą jest ilość działań na sekundę wykonywanych przez te komputery. Przyjętą jednostką miary tej mocy jest milion działań na sekundę określany po angielsku jako MIPS : million instructions per second.

Moc przerobowa jest dobrą miarą potencjalnej wartości użytkowej, czyli osiąganej przy największym możliwym wyposażeniu w inne zasoby, gdyż zestaw nie może nic więcej zrobić na rzecz użytkownika niż pozwala na to moc komputerów w zestawie, wszystkie inne zasoby muszą działać pod kontrolą tych komputerów.

W dalszym ciągu porównamy koszt mocy i MIPSa w zestawach wielodostępnych różnych rodzajów.

Dane na temat kosztu mocy i MIPSa w zestawach jednokomputerowych przedstawione są w raporcie specjalnym (3 pt. "Rewolucja mastodonta" "Desktop revolution" zamieszczonym w tygodniku Business Week z grudnia 1988 r., jak również w artykule (8) w tygodniku "Computerworld Focus" z sierpnia 1988 r., w którym rozważa się wyzwania dla pozycji minikomputerów jaki stanowią sieci lokalne czyli zestawy wk z łącznością kablową. Najnowsze informacje pozwa-

lające ocenić aktualny koszt 1 MIPSa w nowych modelach zestawów komputerowych znaleźć można w doniesieniu (10) z października 1989 roku zapowiadającym wprowadzenie na rynek nowej serii VAX 9000 przez firmę DEC.

W raporcie tygodnika "Business Week" wprowadza się podział zestawów jednokomputerowych na 5 rodzajów różniących się ilością [illegible] tzw. mainframes mogących posiadać ponad 150 końcówek do małych minikomputerów mogących posiadać od 11 do 20 końcówek. Koszty 1 MIPSa zmieniają się wg raportu od 180 tys. dolarów w tzw. mainframes do 20 tys. dolarów dla małych minikomputerów. W raporcie podkreśla się także, że w komputerach osobistych koszt 1 MIPSa jest znacznie niższy niż w klasycznych zestawach jednokomputerowych i wynosi ok. 2 tys. dolarów.

Autorom raportu nie udaje się jednak dokonać analizy konsekwencji tego faktu dla rozwoju techniki komputerowej. Dane zawarte w raporcie [illegible] zbliżone są do danych przedstawionych w artykule [illegible], gdzie sprawa kosztu 1 MIPSa ujęta jest w sposób następujący: "Dostawcy dużych komputerów zawsze próbują i jeśli tylko mogą to sprzedają użytkownikowi duży komputer. IBM i DEC liczą sobie wtedy ok. [illegible] tys. dolarów za 1 MIPSa. Jeśli nie mogą tego dokonać, to wtedy sprzedają użytkownikowi minikomputer. Pod koniec w minikomputerze wypadało 60 tys. dolarów za 1 MIPSa. Teraz minikomputer "Olympus" VAX 6200 od DECa zbliża cenę do poziomu 40 tys. dolarów za [illegible] tys. dolarów za 1 MIPSa."

W artykule [illegible] sygnalizuje się, że "PC LAN" czyli zestawy w [illegible] łączności kablowej pozwalają na uzyskanie 1 MIPSa po znacznie niższej cenie, lecz brak jest dokładniejszej analizy tej sprawy. Powołuje się natomiast wypowiedź jednego z analityków rynku komputerowego, który stwierdza: "Ludzie będą pytać, czy na jakiś sens nie [illegible] minikomputer z takim kosztem MIPSa?"

W informacji (16) podaje się, że cena różnych modeli serii VAX 9000 będzie wynosiła od 1,24 mln. dolarów do 4,4 mln. dolarów i że stanowi to około połowy ceny porównywalnych pod względem mocy komputerów serii IBM 3090. Moc najmniejszego modelu VAX 9000 można ocenić na około 30-40 MIPSów, daje to cenę od 30 do 40 tys. dolarów za 1 MIPSa, a dla komputerów IBM 3090 cenę od 60 do 80 tys. dolarów za 1 MIPSa.

Dla porównania w dalszym ciągu obliczymy nową cenę 1 MIPSa w zestawach wielokomputerowych z łącznością szynową. Zestawy z łącznością kablową pominiemy jako typowe rozwiązania przejściowe, które zaniknie wraz z rozwojem zestawów z łącznością szynową.

Koszt zestawu wk/s zawierającego:

- składnicę typu PC/AT z dyskiem 300 MB,
- 2 stanowiska typu PC/AT na płytce GL 2286,
- 8 stanowisk typu PC/XT na 2 płytkach GL 1004,

można, na podstawie cennika firmy Cubix dla rynku amerykańskiego, ocenić na około 35 tys. dolarów. Łączna moc komputerów w powyższym zestawie wynosi:

- składnica PC/AT		2,5 MIPS
- 2 stanowiska PC/AT	2 x 2,5 =	5 MIPS
- 8 stanowisk PC/XT	8 x 0,5 =	4 MIPS
		13,5 MIPSa

Koszt mocy 1 MIPSa w takim zestawie wynosi więc:

35000/13,5 = 2500 dolarów/MIPS.

Zastosowany wyżej sposób obliczenia kosztu 1 MIPSa w zestawie wk/s wskazuje, że koszt ten będzie prawie taki sam przy każdej ilości końcówek. W zestawach wk/s nie występuje więc charakterystyczne dla zestawów jak zjawisko wzrostu kosztu 1 MIPSa wraz ze wzrostem ilości końcówek. Kosztowa przewaga zestawów wk/s nad zestawami jk będzie tym większa, im większy będzie zestaw - a możliwości rozbu-

dowy zestawów wk/s są praktycznie nieograniczone.

Stosunek kosztu mocy 1 MIPSa w zestawach jk do kosztu mocy 1 MIPSa w zestawach wk/s będzie dla poszczególnych rodzajów zestawów jk wymienionych w raporcie specjalnym (5) następujący:

- duży komputer (mainframe - 60 razy,
- duży minikomputer - 20 razy,
- mały minikomputer - 5 razy.

Według danych z artykułu (6) koszt 1 MIPSa w minikomputerze VAX 8600 będzie 15 do 30 razy większy niż w zestawie wk/s, zaś według danych dodatkowych (10) koszt 1 MIPSą w dużym komputerze VAX 9000 i IBM 3090 jest odpowiednio co najmniej około 20 i około 40 razy większy niż w zestawi wk/s.

Skutkiem takich relacji kosztowych musi być wystąpienie, prędzej czy później, zjawiska wypierania zestawów jk przez zestawy wk.

Zjawiska takie już pojawiły się i nabierają wyraźnemu nasileniu. Sprawę tę omówimy bliżej w dalszym ciągu.

7. ZESTAWY WIELOKOMPUTEROWE Z ŁĄCZNOŚCIĄ SZYNOWĄ JAKO PRZYSZŁOŚCIOWA POSTAĆ ZESTAWÓW WIELODOSTĘPNYCH

Celem niniejszego referatu jest, jak to już stwierdziliśmy we wstępie, sformułowanie przewidywań określających kierunki rozwojowe i dominujące w przyszłości rozwiązania w zakresie wielodostępnych zestawów komputerowych wzk. Przedstawiona wyżej analiza właściwości użytkowych oraz kosztów jednostkowych różnych postaci wzk pozwala sformułować oceny określające w sposób syntetyczny przydatność dla użytkowników wzk różnych postaci.

Oceny te są następujące:

1. górne granice powiększania zdolności czynnościowych, a tym samym możliwości rozbudowy zestawów wk są znacznie większe niż te granice i możliwości zestawów jk, w szczególności w zestawach wk osiągane są bardzo duże łączne moce komputerów przewyższające moce tzw. dużych komputerów;
2. zestawy wk pozwalają na uzyskanie stabilnych, a jednocześnie krótszych niż w zestawach jk, średnich czasów odpowiedzi na żądania użytkownika;
3. zestawy wk stwarzają użytkownikom lepsze, niż zestawy innych rodzajów, warunki pracy, gdyż znacznie osłabiają oddziaływania na siebie użytkowników, a tym samym znacznie osłabiają działanie czynników mogących prowadzić do konfliktów między nimi,
4. zestawy wk z łącznością szynową posiadają niezawodność działania na tym samym poziomie jak niezawodność zestawów jk.
   Zestawy wk z łącznością kablową posiadają niższą niezawodność działania,
5. koszt mocy 1 MIPSa jest w zestawach wk wielokrotnie niższy niż w zestawach jk, szczególnie [illegible] jest koszt tej mocy w zestawach wk z łącznością szynową
   Koszt mocy 1 MIPSa w zestawie [illegible] posiadającym taką samą ilość

końcówek jak zestaw typu "mainframe" jest 40 do 80 razy niższy, a w zestawie wk/s posiadającym taką samą ilość końcówek jak mały minikomputer jest 5 do 6 razy niższy.

Wynika stąd następująca ocena końcowa: zestawy wk istotnie przewyższają zestawy jk w zakresie właściwości użytkowych oraz kosztów jednostkowych, zaś wśród nich najkorzystniejsze właściwości posiadają zestawy wk z łącznością szynową.

Oceny te dają podstawy do sformułowania przewidywań: zestawy wk z łącznością szynową będą miały zasadnicze znaczenie w rozwoju techniki komputerowej. Ich zasadnicza przewaga nad zestawami jk wkrótce spowoduje, że zaczną one wypierać te zestawy, aż do ich całkowitego wyparcia. Można przyjąć iż w przyszłości zestawy wk z łącznością szynową będą stanowiły dominującą w praktyce postać wielodostępnych zestawów komputerowych.

Już w chwili obecnej stosowanie zestawów jk z technicznego punktu widzenia nie ma żadnego uzasadnienia, zestawy te można określić jako "żywe trupy" lub "dinozaury" techniki komputerowej.

Oceny te, uzyskane w drodze czysto teoretycznej, znajdują już swoje praktyczne potwierdzenie, gdyż procesy wypierania zestawów jk, w szczególności tzw. mainframes, już się rozpoczęły w Stanach Zjednoczonych. W artykule '6' zamieszczonym w październikowym numerze miesięcznika "The Network Report" jest to powiedziane bardzo wyraźnie: "Ludzie, których teraz spotykam, widzą możliwość pozbycia się posiadanych systemów opartych o duże komputery (mainframe-based systems) i zastąpienia ich przez sieci lokalne z komputerami osobistymi wyposażone w komputery obsługi baz danych (database servers). Zamierzają oni zastąpić obecne systemy użytkowe przez systemy działające na komputerach osobistych i zastosować maszyny z mikroprocesorami Intel 386 i 486 do obsługi baz danych. Są oni tak przekonani, iż jest to lepsze i mniej kosztowne rozwiązanie.

że już teraz zaplanowali oni usunięcie ich "mainframes" po 15 miesiącach od teraz. Nie ma już powrotu."

W komentarzu (9) zamieszczonym w biuletynie nowości "EDP Weekly" z października 1989 roku sprawa ta ujęta jest następująco:

"Wytwórcy minikomputerów i mainframes będą musieli schować swoją dumę i sieć maszyny naśladujące komputery osobiste i stanowiska robocze" mówi analityk firmy Duff & Phelps Inc. z Chicago, zajmującej się badaniami inwestycji i doradztwem finansowym. "Jest to przypadek niewiarygodnie malejącego komptera" powiedział Martin P. Reasinger, który śledzi przemysł komputerowy dla firmy Duff & Phelps. "Komputery i stanowiska robocze (workstations) odpowiadają na potrzeby, które dawniej wymagały bardziej kosztownych minikomputerów i "mainframes". Większe maszyny muszą walczyć o byt, tak że dostawcy wprowadzają modele budowane w oparciu o tą samą technologię co komputery osobiste..."

"Maszyny są coraz częściej sprzedawane w konfiguracjach standardowych niż robione na zamówienie i co ważniejsze, punkt ciężkości dostaw przesunął się dramatycznie do mniejszych maszyn, które często są sprzedawane z półki" - powiedział analityk. Walka konkurencyjna nie toczy się już więcej między indywidualnymi dostawcami a między różnymi sektorami. Na przykład sieci komputerów osobistych i "workstations" stanowią konkurencję dla minikomputerów z podziałem czasu w zakresie przetwarzania interakcyjnego w przedsiębiorstwach.

"Maszyny nastolne (desktop machines) wygrywają większość bitew." - powiedział Reasinger. "...Prostsze i szybsze procesory stwarzają zagrożenie, że tradycyjne rozwiązania staną się przestarzałe."

W prasie amerykańskiej pojawiają się też artykuły opisujące pionierskie firmy, które już zdecydowały się usunąć komputery typu IBM 4300 i zastąpić je zestawami sieciowymi opartymi o komputery

osobiste. Pionierami tymi są:

- firma Schlin z Brandford, Connecticut, opisana w raporcie specjalnym (5) z grudnia 1988 r. oraz w (6),
- firma American Sterilizer z Erie, Pensylwania, opisana w artykule w numerze czasopisma "The Network World" z kwietnia 1988 r.

Firma Schlin produkuje części samochodowe i osiąga roczną sprzedaż w wysokości ok. 1 mld dol. Firma ta zdecydowała się uzupełnić komputer IBM 4300 i zastąpić go zestawem sieciowym opartym o system Netware zawierającym ok. 80 stanowisk roboczych. Według dyrektora wdrożenia systemów w tej firmie, cytowanego w (5), dało to od razu oszczędności ok. 1 mln. dol. i powinno przynieść kolejny milion w ciągu roku od wymiany zestawów.

Firma American Sterilizer (AS) produkuje sprzęt medyczny i w 1987 r. osiągała sprzedaż około 250 mln. dol. Jack Preboda, dyrektor systemów informacyjnych w firmie AS w następujący sposób (7) przedstawia zdarzenia w firmie: "Aż do roku 1985 całe przetwarzanie danych w firmie AS wykonywane było przez "IBM mainframes", powiedział Preboda. Działały na nich zarówno systemy wsadowe, jak też interaktywne, podłączonych do nich było ok. 750 terminali znajdujących się w różnych miejscach firmy.

Czas odpowiedzi zmieniał się w szerokich granicach, w zależności od obciążenia komputera, powiedział Preboda. Nie byliśmy zbyt radzi klienci z czasu odpowiedzi i niezawodności systemów "on-line".

Wtedy został podjęty decyzję o zmianie tych IBM mainframes na zestawy sieciowe a skutki tych decyzji przedstawione są w (7) w sposób następujący:

"Firma AS przewiduje, iż przez odesłanie jej "mainframes" i przeniesienie systemów użytkowych na sieci komputerów osobistych, zyska poprawę niezawodności i czasu odpowiedzi dla użytkowników końcowych. Uzyska także roczną oszczędność 1 mln. dol.

To co się zaczęło od pilotowej instalacji sieciowej w 1986 r. rozrosło się w obejmujące firmę przedsięwzięcie sieciowe, które obaliło królowanie w AS "IBM mainframes". Firma odesłała już jeden z komputerów "mainframe" tj. 4341, z powrotem do firmy dzierżawiącej i zainstalowała sieci lokalne w budynkach centrali, dwu oddalonych zakładach i w wielu biurach sprzedaży.

...firma zauważyła już dramatyczny wzrost sprawności od czasu przejścia na sieci lokalne. Tylko w ostatnim roku firma oszczędziła 800 tys. dol. w porównaniu do tego co było wydane w 1989 r. na wynajmowanie i utrzymywanie ruchu sprzętu i oprogramowania typu "mainframe" - ocenia Prahoda. Powiedział też, że oczekuje oszczędności rzędu 1 mln. dol. rocznie, gdy nastąpi przeniesienie wszystkich systemów użytkowych. Prahoda nie powiedział natomiast ile zostało wydane na sieci.

Dodatkowo, użytkownicy końcowi zestawu sieciowego są zadowoleni z lepszego, w porównaniu do zestawu z "mainframe", czasu odpowiedzi..."

Zgodnie z [7] zestaw sieciowy w firmie AS zawiera 13 składnic file servers w postaci "klonów AT" działających pod kontrolą systemu Netware firmy Novell oraz 350 stanowisk roboczych połączonych siecią opartą o karty typu Arcnet. Po zakończeniu rozbudowy zestawu sieciowego ma on liczyć 700 stanowisk roboczych.

Sprawność działania oceniana jest następująco:

"Prahoda mówi, że sprawność performance zestawu sieciowego jest bardziej stała i konsystentna niż zestawu z "mainframe". Podczas gdy sprawność zestawu z "mainframe" ulegała na skutek zmienności obciążenia, to użytkownicy zestawu sieciowego mają stale dobrą sprawność, ponieważ ich programy działają na ich indywidualnych komputerach."

Sumaryczne oceny wyników wprowadzenia zestawu sieciowego są

następujące:

"Wszystko, co zwykłem robić z "mainframe" jest obecnie robione n
sieci komputerów osobistych. IBM 4341 jest przeszłością" - powie
dział Prehoda.

Jest on całkiem zadowolony z możliwości zestawu sieciowego. "Może
my mieć wspólne dane, mamy możliwość restartu i odtworzenia danych
jest odporność na awarie - wszystkich tych rzeczy nie mieliśmy n
zestawie z "mainframe"."

"Wiemy, że mamy obecnie lepszą odporność na awarie niż kiedyko
lwiek mieliśmy przy zestawie z "mainframe"."

Praktyczne doświadczenia firmy American Sterilizer całkowicie
potwierdzają, jak widać, teoretyczne wyniki przedstawione w roz
dziale 5.

Przedstawione doświadczenia firm Echkin i American Sterilize
pokazują, że już zestawy sieciowe, czyli zgodnie z wprowadzoną
wcześniej terminologią, zestawy wielokomputerowe z łącznością kab
lową znacznie bardziej odpowiadają potrzebom użytkowników niż zes
tawy jednokomputerowe typu "mainframe". Tymczasem, jak to wcześ
niej pokazaliśmy, zestawy wk k są znacznie mniej sprawne niż zesta
wy wielokomputerowe z łącznością szynową. Jednak wiedza o zaletach
i pełnej sprawności zestawów wk/k nie jest jeszcze wystarczająco
rozpowszechniona.

Prędzej czy później, wiedza ta stanie się szerzej dostępna i wtedy
należy oczekiwać dużej gwałtownej zmiany w technice komputerowej.
Należy sądzić, że znaczna grupa użytkowników zaczną wtedy pozbywać
się systemów "mainframes" czy "superminikomputerów" i w ich miejsce
instalować zestawy z płytkowymi stanowiskami roboczymi i systemem
Netware.

Obecnie poważnym hamulcem wzrostu zakresu stosowania zestawów
łącznością kablową jest właśnie występowanie w zestawie tego kab

lowego układu łączności. Układ ten przez wielu użytkowników uważany jest za element niepewny i niebezpieczny, co wstrzymuje ich przed stosowaniem zestawu sieciowego.

Będzie to jednak bez wątpienia bardzo niekorzystny dla "wielkich" świata komputerów, takich jak firmy IBM czy DEC, gdyż pozbawi ich wielkich i łatwych zysków płynących ze sprzedaży "małych komputerów" - jest paradoksalna zasadnia, jakie będą ich poczynania wobec zagrożenia. Wydaje się jednak rzeczą pewną, że czasy wielkich firm komputerowych, jak właśnie IBM, DEC, Hewlett-Packard nieuchronnie zbliżają się do końca. Ich istnienie, było ściśle związane z istnieniem zestawów typu "mainframe", których projektowanie i budowa wymagały wielkich inwestycji i wielkich zespołów, na które mogły sobie pozwolić jedynie wielkie firmy. Zestaw wielokomputerowy z łącznością szynową o mocy przewyższającej "mainframe" może zaprojektować i produkować każda średnia firma o przyzwoitym poziomie technicznym.

8. ZASADY POPRAWNEGO DOBORU WIELODOSTĘPNYCH ZESTAWÓW KOMPUTEROWYCH DLA KRAJOWYCH ORGANIZACJI

Dla organizacji krajowej najbardziej odpowiedni jest wielodostępny zestaw komputerowy, który spełnia następujące wymagania:

1. posiada najwyższy poziom przydatności i najniższy poziom kosztów jednostkowych spośród zestawów, które:
   - spełniają główne, zasadnicze wymagania przyszłych użytkowników zestawu w organizacji,
   - mieszczą się w możliwościach finansowych organizacji;
2. daje możliwość uruchomienia użytkowania zestawu w ciągu czasu wymaganego przez organizację; praktycznie oznacza to, że daje możliwość uzyskania w tym czasie odpowiedniego oprogramowania użytkowego, czyli oprogramoania zapewniającego obsługę użytkownika odpowiednią pod względem rodzajowym i jakościowym.

Cała przedstawiona wyżej analiza wykazuje, że wymagania pierwsze w każdym wypadku spełniają tylko zestawy wielokomputerowe z łącznością szynową, gdyż zestawy te, choć brzmi to niewiarygodnie, są jednocześnie znacznie lepsze, mocniejsze i tańsze, niż wcześniej znane rodzaje zestawów wielodostępnych.

W niektórych wypadkach dla spełnienia niektórych wymagań użytkowników konieczne będzie zastosowanie mieszanych zestawów wk, w których część stanowisk roboczych podłączona jest przez układ kablowy.

Jeśli więc jednocześnie możliwe jest, przy wyborze zestawu wk/s lub wk mieszanego, spełnienie wymagania drugiego czyli możliwe jest dostatecznie szybkie uzyskanie niezbędnego oprogramowania, to sytuacja jest jasna: należy wybrać zestaw wk z łącznością szynową lub mieszany.

Może jednak wystąpić sytuacja taka, że nie jest możliwe uzyskanie w wymaganym czasie oprogramowania użytkowego dla zestawu wk,

a równocześnie oprogramowanie takie już istnieje i jest dostępne dla zestawu, który nie spełnia wymagania pierwszego, co praktycznie oznacza tradycyjny zestaw jednokomputerowy typu "mainframe" czy też superminikomputer. Przez pewien czas sytuacje takie mogą pojawiać się dość często, gdyż w ciągu wielu lat użytkowania zestawów tradycyjnych powstał znaczny dorobek w dziedzinie oprogramowania użytkowego, dla zestawów wk dorobek taki dopiero powstaje.

W szczególności sytuacja taka może zdarzyć się w firmie, która zamierza wymienić posiadany i już zużyty zestaw komputerowy na nowy. Firma taka posiada oprogramowanie działające na starym zestawie i zwykle uważa, iż oprogramowanie to należy zachować i nadal użytkować na nowym zestawie. Praktycznie oznacza to, że konieczny jest zakup nowego modelu zestawu przeznaczonego do wymiany.
Na pozór jest to rozwiązanie bardzo racjonalne, gdyż zapewnia wykorzystanie istniejącego dorobku, a tym samym pozwala na uniknięcie kosztów na wytworzenie nowego oprogramowania. Jest to jednak tylko jedna strona medalu - zakup nowego modelu starego zestawu oznacza też utrzymanie się starych, czasami wyraźnie przestarzałych rozwiązań, a także pozostawanie w zależności od monopolisty, jedynego producenta pewnego rodzaju zestawów, po bardzo wygórowanej, monopolistycznej cenie. Tradycyjni dostawcy komputerów, w szczególności tacy jak: IBM, DEC czy Wang, dobrze zdają sobie sprawę z dążenia użytkowników do uniknięcia kosztów na nowe oprogramowanie i bezwzględnie sytuację tą wykorzystują żądając bardzo wysokich, wygórowanych cen. W ten sposób sprzedaż nowych modeli zestawów komputerowych o przestarzałej konstrukcji jest dla tradycyjnych producentów komputerów, głównie firmy IBM, źródłem bardzo wysokich i łatwych zysków. Należy sądzić, że firmy te będą przeciwdziałać rozwojowi nowych rozwiązań, czyli zestawów wk.

W podsumowaniu należy stwierdzić, iż dokonanie przez organi-

zacje poprawnego doboru rodzaju zestawu komputerowego w sytuacji gdy może być łatwo uzyskane lub jest gotowe oprogramowanie użytkowe dla zestawu tradycyjnego i w istocie przestarzałego jest zadaniem trudnym. W każdym wypadku decyzja musi być podjęta na podstawie bardzo konkretnej i szczegółowej analizy obejmującej przede wszystkim dokładny bilans zestawiający:

- korzyści wynikające z przejścia na nowe, znacznie tańsze i bardziej efektywne zestawy wk,
- straty wynikające z rezygnacji z istniejącego oprogramowania i konieczności tworzenia go na nowo.

W bilansie tym trzeba też brać pod uwagę fakt, że losy tradycyjnych zestawów jednokomputerowych są już przesądzone, nie ma wątpliwości, że w niedługim czasie przejdą one do historii techniki, nie wiadomo jedynie kiedy to nastąpi.

W związku z tym w pewnej chwili każdy użytkownik stanie wobec konieczności rezygnacji ze swojego dorobku związanego z zestawami jk i poniesienia kosztów na utworzenie nowego oprogramowania dla nowego zestawu wk. Tym samym decyzja o zakupie zestawu tradycyjnego nie chroni na zawsze przed kosztami nowego oprogramowania, powoduje jedynie przeniesienie w przyszłość chwili poniesienia tych kosztów.

Gdy się jeszcze uwzględni fakt, że zakup zestawu tradycyjnego jest znacznie bardziej kosztowny niż zakup zestawu wk oraz to że użytkowanie starego oprogramowania też pociąga, czasami bardzo znaczne, koszty, to w wielu wypadkach może się okazać, że próba uzyskania oszczędności przez zachowanie oprogramowania doprowadzi do dodatkowych kosztów.

LITERATURA

1. Braniecki A., Rawiński T.: Koncepcja maszyny przetwarzania dokumentów, jako środka komputerowej obróbki informacji w strukturze organizacyjno - informacyjnej stoczni remontowej. Instytut Okrętowy PG, Prace Badawcze nr 1870, Gdańsk 1988 r.
2. Rawiński T.: Teoria maszyn przetwarzania dokumentów - cybernetyczne ujęcie systemów informatycznych. (w:) Materiały Krajowego Symp. CYBERNETYKA 83, 21-22.06.1983 r., t. III, PTC Warszawa 1983 r.
3. Rawiński T.: Podręcznik użytkownika systemu Netware. Instytut Okrętowy PG, Prace Badawcze Nr , Gdańsk 1989 r.
4. Marra F.: Chatred - CPU Networks. The Network Report, Vol. 4, No. 10, October 1989 r.
5. Lewis C., Field A.R., Hafner K.M.: Special Report (Dezktop Revolution) - Computers. Zoom. Here Come The New Micros. These Mitrafast Machines will challenge minis - and even mainframes. Business Week No. 2978, December 1, 1986r.
6. Kolodziej S.: Challenging the mid-range champ. Computerworld Focus Vol. 22 No. 31A, August 3, 1988 r.
7. Petrosky M.: User sends mainframes packing. The Network World, April ., 1988 r.
8. Vinzant D.: The Multi-Vendor Software Solution. The Network Report, Vol. 4, No. 10, October 1989 r.
9. Redakcja: Makers of minicomputers and mainframes will have to swallow their pride and have their machines imitate PCs and workstations, EDP Weekly Vol. 30, No.43, October 23, 1989 r.
10. Redakcja: DEC, CDC and Tandem launch new high-end mainframes. EDP Weekly, Vol. 30, No. 43, October 23, 1989 r.

AKTUALNE TRENDY
KOMPUTEROWO WSPOMAGANEGO
TWORZENIA SYSTEMÓW INFORMATYCZNYCH

Stanisław Wrycza
Uniwersytet Gdański
Katedra OPD
81-824 Sopot
ul. Czerwonej 119/121

1. WPROWADZENIE.

Idea wspomaganego komputerem tworzenia systemów informatycznych (TSI) nie jest nowa. Już w latach siedemdziesiątych przeprowadzono w tej dziedzinie pewne badania, e k s p e r y m e n t y i wdrożenia. Chodzi tu przede wszystkim o pakiet PSL/PSA, opracowany na Uniwersytecie Michigan. Prawdziwy jednak boom w tej dziedzinie nastąpił w latach osiemdziesiątych, wraz z masowym użytkowaniem mikrokomputerów. Stworzyły one możliwość powiązania dotychczasowego bogatego dorobku w sferze m e t o d y k TSI z cechą p r z y j a z n o ś c i (ang. user-friendly) oprogramowania. Powstają w ten sposób liczne pakiety wspomagające proces TSI nazywane powszechnie w świecie CASE:

C - computer

A - aided, assisted

S - software, system

E - engineering

Używa się również zamiennie skrótu IPSE (integrated project supporting environment). Niektórzy autorzy podkreślają, że chodzi w tym przypadku o pakiety wspomagające przede wszystkim

fazę programowania.

Błyskawiczna e k s p a n s j a rynkowa narzędzi CASE w ostatnich dwu latach zaskoczyła wielu profesjonalistów z dziedziny zastosowań informatyki. Obserwowali oni losy już niejednej koncepcji, pojęcia, słowa-klucza jak system zintegrowany, SIK, baza danych czy system ekspertowy. Wszystkie one miały być panaceum na kłopoty związane z projektowaniem i użytkowaniem systemów informatycznych, definiowaniem potrzeb informacyjnych przez użytkownika, dopasowaniem struktury systemu do tych potrzeb wreszcie. Czy więc w przypadku pakietów CASE nie znajdujemy się w punkcie przesadnej adoracji osiągnięć kolejnych guru nowej wiedzy i nadreklamy nowych narzędzi, w których stworzenie zainwestowano poważne środki finansowe? Inaczej - czy mamy do czynienia ze zjawiskiem trwałym? Czy pakiety CASE praktycznie udowodniły swoją użyteczność? Czy są efektywne? Jak oddziałują na pracę zespołów projektujących?

Nie na wszystkie te i inne pytania można już dziś odpowiedzieć jednoznacznie. Na pewno zdecydowaną, choć również nie ostateczną, jest odpowiedź w kategoriach finansowych. S p r z e d a ż tych pakietów w 1987 roku kształtowała się na poziomie 180 mln $. Tymczasem prognoza na 1990 rok mówi już o wielkości 3-5 mld $. Również symptomatyczna jest opinia J. Martina, iż z i n t e g r o w a n e pakiety CASE stanowią największą zmianę w dziedzinie profesjonalnego przetwarzania informacji od 30 lat. Głównej przyczyny popularności tych

narzędzi należy dopatrywać się jednakw ich podstawowym założeniu- idei zautomatyzowanej realizacji całego c y k l u życia systemu - począwszy od definiowania założeń do zakładania bazy danych i generowania kodu. Stopień realizacji tego założenia w konkretnych implementacjach jest różny, ograniczony, szczególnie w narzędziach wycinkowych. Jednrk kilka firm jest bliskich wykonaniu tego celu.

Powyższe przesłanki powinny skłonić do baczniejszego zainteresowania sie polskich informatyków sferą CASE. Krajowe doświadczenia w tej dziedzinie w dalszym ciągu są nad wyraz skromne. Należy jednak zauważyć drobne, choć konkretne zmiany, wyrażające się w kilku autentycznych wdrożeniach pakietów CASE (QUICKBUILDER, PAGEFIT, MASCOT). U źródeł tych zmian leżą raczej nowe zjawiska g o s p o d a r c z e. Należą do nich:

- czynniki wewnętrzne jak konieczność restrukturyzacji wielu organizacji gospodarczych, ich funkcjonowanie w warunkach rynkowej konkurencji,
- podejmowanie wspólnych przedsięwzięć w kooperacji z partnerami zagranicznymi - realizacja tych planów jest uwarunkowana zastosowaniem pakietów CASE, wspomagających te przedsięwzięcia.

Tak więc ogólne warunki gospodarcze będą wpływać na tempo wprowadzania nowoczesnych metodyk tworzenia systemów informatycznych i narzędzi je wspomagających.

C e l e m niniejszego opracowania jest charakterystyka i

o c e n a aktualnych podstawowych tendencji w dziedzinie narzędzi komputerowo wspomaganego TSI, zwanych CASE. Dokonano tego w wymiarze oceny zarówno merytorycznej jak i handlowej, finansowej. P o wp r owadzeniu, w drugiej części pracy przedstawiono koncepcję warsztatu twórcy SI. Jest to baza, szersze otoczenie dla treści prezentowanych w kolejnych częściach. Ścisłą tematykę CASE rozpoczęto od obszernej klasyfikacji, w trzeciej części, narzędzi CASE oraz charakterystyki poszczególnych ich rodzajów. Proces TSI przebiega na zasadzie wykorzystania adekwatnych metod i technik w kolejnych fazach cyklu życia systemu, toteż w czwartej części opracowania określono zasady doboru tych metod i technik do kolejnych faz procesu. Znajomość tych zasad jest podstawą umiejętności użytkowania zintegrowanych pakietów CASE. Ogólną prezentację światowego rynku tych narzędzi zawarto w części piątej. W załączniku przedstawiono treść programu kursu z zakresu narzędzi CASE prowadzonego przez amerykańską firmę doradczą Digital Consulting.

2. WARSZTAT TWÓRCY SYSTEMÓW INFORMATYCZNYCH.

Podstawowymi przesłankami określenia istoty i funkcji takich kategorii jak warsztat czy stanowisko pracy twórcy systemów informatycznych (w tym przypadku głównie analityka i projektanata) bądź środowisko metodyczne TSI, jest zdefiniowanie pojęcia metodyki TSI oraz jej składników. Otóż m e t o d y k ę TSI można określić jako spójny, logicznie uporządkowany zbiór metod i procedur o charakterze technicznym i organizatorskim, pozwalających realizować cykl życia systemu

przez zespół wykonawczy. Definicja nie precyzuje składników metodyki TSI. Przy aktualnym stanie wiedzy i praktyki są to:

a. f o r m a l i z m y, modele opisu wycinka rzeczywistości tj. dziedziny przedmiotowej, jej statyki i dynamiki,

b. p r o c e s TSI, zwany cyklem życia systemu, ustrukturyzowany w postaci odpowiedniej sekwencji etapów, podetapów i kolejnych zadań,

c. poszczególne m e t o d y i t e c h n i k i dokumentowania rezultatów procesu TSI, wraz z odpowiednią notacją graficzną,

d. n a r z ę d z i a wspomaganego komputerem TSI,

e. specyfikacja wymagań merytorycznych wobec poszczególnych twórców interdyscyplinarnego z e s p o ł u wykonawczego oraz odpowiednich materiałów szkoleniowych,

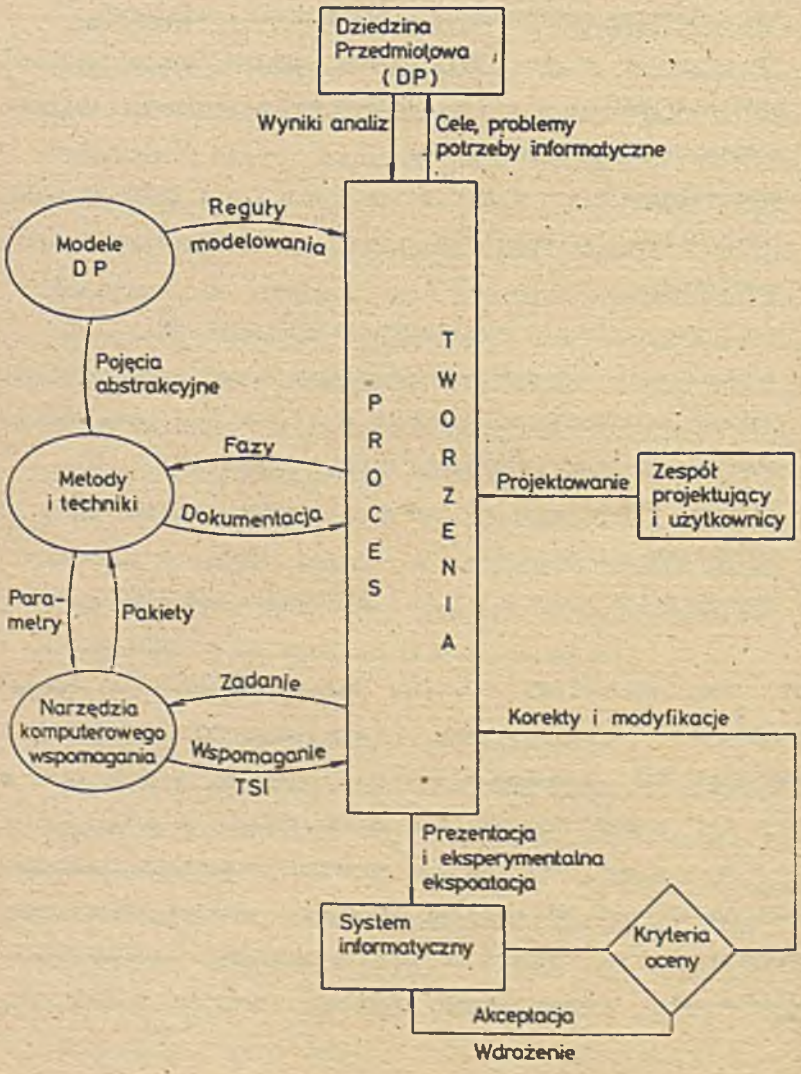
f. kryteria o c e n y jakości projektu i systemu oraz mechanizmy jej kontroli,

g. zasady p l a n o w a n i a i sterowania rozwojem systemu.

Ogół tych składników metodyk, wspomagających analityków, projektantów i użytkowników w procesie TSI stanowi ich środowisko TSI, podnoszące szeroko rozumianą e f e k t y w n o ś ć całego procesu oraz jakość funkcji k o m u n i k o w a n i a pomiędzy twórcami systemu. Powiązania pomiędzy wyżej wymienionymi składnikami metodyki w postaci uogólnionej przedstawiono na rysunku 1 [12].Potrzeby informatyczne, założone cele, zaobserwowane problemy inicjują proces TSI

Rysunek 1

Powiązania między składnikami metodyki TSI



realizowany i sterowany przez zespół projektujący.

Obiektem analizy w procesie TSI jest dziedzina przedmiotowa czyli badany wycinek rzeczywistości. Może nią być działalność organizacji gospodarczej lub jej wybranej agendy. Podstawą opisu dziedziny przedmiotowej są odpowiednie modele DP zwane modelami konceptualnymi. Pozwalają one na odzwierciedlenie zarówno s t a t y k i jak i d y n a m i k i DP. Poszczególne elementy modeli są dokumentowane odpowiednimi metodami i technikami. Analiza struktury ich parametrów i zasad użytkowania pozwala na konstruowanie narzędzi wspomagających proces projektowania. Ich użytkowanie w zasadniczy sposób oddziałuje na efektywność konstruowania systemu informatycznego.

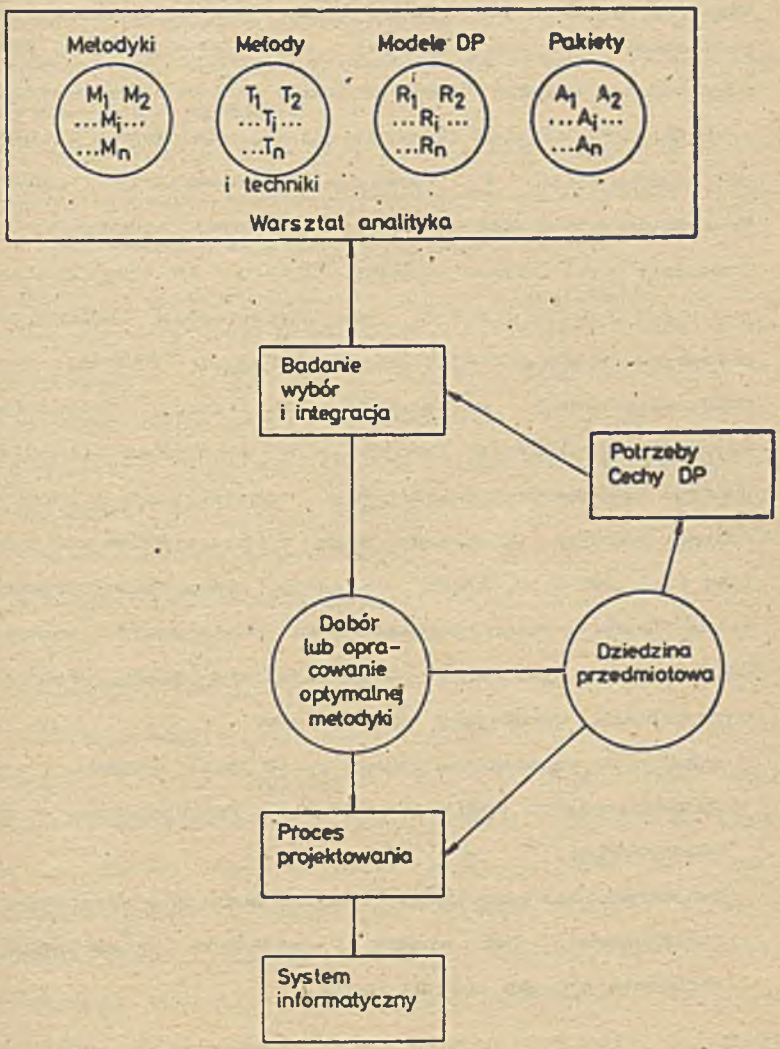
Rozwój, udoskonalenia i zróżnicowanie poszczególnych składników metodyk spowodowały powstanie w sposób naturalny warsztatu lub środowiska analityka i projektanta systemu. Przez w a r s z t a t twórcy SI należy tu rozumieć ogół dostępnych metodyk, modeli, metod, technik i pakietów CASE. Twórcy systemów, zespoły projektujące decydują o wyborze właściwych narzędzi dla realizacji procesu TSI. Oczywiście wybór ten nie jest całkowicie swobodny lecz uwarunkowany takimi przesłankami jak ograniczenia o f e r t y rynkowej w tym zwłaszcza dostępność i cena pakietów CASE oraz metodyk, profesjonalne przygotowanie oraz wiedza metodyczna twórców systemu, wreszcie specyfika dziedziny przedmiotowej. W związku zwłaszcza z pierwszą przesłanką, warsztat twórcy SI jest oczywiście bogatszy, pełniejszy i bardziej nowoczesny w krajach o

wysokiej technologii informatycznej. Należy zaznaczyć, że w warunkach krajowych bariery finansowe nie są jedyną czy nawet główną przyczyną ubóstwa tego warsztatu. Powszechnie nikła wśród analityków i projektantów znajomość na przykład analizy i projektowania strukturalnego, brak jednoznacznych przepisów i sankcji w dziedzinie ochrony oprogramowania czy brak zespołów tworzących oryginalne narzędzia CASE w kraju mają bardzo znaczący, negatywny wpływ na stan rzeczywisty. Koncepcję warsztatu twórcy SI i jego miejsce w procesie TSI przedstawia rysunek 2. Obrazuje on jak złożonym zagadnieniem staje się p r a c a współczesnego twórcy systemów a jednocześnie sygnalizuje ogromny zakres w i e d z y jaka wiąże się z dziedziną TSI.

Obok pojęcia warsztatu, używane jest określenie s t a n o- o w i s k a pracy (ang.: workstation) twórcy SI[ ]. Oznacza ono dostępne twórcy SI narzędzia, obejmujące sprzęt (często specjalizowany), materiały metodyczne i oprogramowanie, których funkcją jest zapewnienie komputerowego wspomagania TSI. Tak więc poszczególne stanowisko pracy twórcy SI czerpie z szerszego otoczenia - warsztatu. Pakiety CASE są zatem jednym ze składników wyposażenia takiego stanowiska. Zrozumienie przez twórców SI możliwości komputerowego wspomagania a w przyszłości automatyzacji procesu TSI ma bardzo znaczący wpływ na z m i a n y w praktyce projektowania systemów.

Rysunek 2

Warsztat twórcy systemów informatycznych w procesie TSI



3. KLASYFIKACJE PAKIETÓW CASE.

3.1 KRYTERIA KLASYFIKACJI.

Mimo, iż historia technologii CASE jest stosunkowo krótka, powszechnie mówi się o dwu jej g e n e r a c j a c h. Pierwszą zapowiedział artykuł "Computer-Aided Software Engineering" dra J.H. Manley'a (Nastec Corporation) zaprezentowany w 1984 roku na konferencji i wystawie przetwarzania danych w Waszyngtonie. Pierwsze pakiety wycinkowo wspomagały różne fragmenty cyklu życia systemu. Zadania te były realizowane a u t o n o m i c z n i e, bez zapewnienia automatycznego przepływu opisów pomiędzy kolejnymi fazami. Pierwsze produkty CASE eliminowały konieczność realizacji rudymentarnych czynności analityków, projektantów i programistów zwłaszcza poprzez komputeryzację kreślenia, aktualizacji i generowania różnego rodzaju diagramów. Wraz z pojawieniem się w 1985 i 1986 [ ] pakietów, które w sposób zintegrowany wspomagały kilka etapów procesu projektowania można mówić o technologii CASE drugiej generacji. Obserwuje się stały proces ich doskonalenia, wyrażający się m. in. w:

- dążeniu do wspomagania całego cyklu życia systemu,
- uzupełnieniu o możliwości technik programowania w czasie rzeczywistym,
- bezpośredniego generowania kodu na podstawie diagramów,
- zastosowania baz wiedzy i systemów ekspertowych dla skrócenia procesu projektowania.

Z generacjami pakietów CASE ściśle wiąże się ich podział na wycinkowe i zintegrowane. Równie spokrewniona jest

klasyfikacja uwzględniająca kryterium funkcjonalności [12]:

- pakiety wykonujące jedynie funkcje graficzne, dostosowane do przyjętej notacji graficznej i z pominięciem innych reguł metodycznych,
- z wbudowanymi regułami metodycznymi, umożliwiającymi kontrolę poprawności i spójności procesu projektowania oraz generowanej dokumentacji.

Obszerną a jednocześnie precyzyjną klasyfikację opracowała firma doradcza CASE Research[8]. Uwzględnia się w niej następujące klasy pakietów CASE:

- przechowywania danych (ang.: repository),
- modyfikacji, adaptacji systemów (ang.: re-engineering),
- wspomagania cyklu życia systemu,
- sterowania realizacją projektu,
- bieżącego doskonalenia jakości systemu.

3.2 PAKIETY PRZECHOWYWANIA DANYCH.

Pierwszą kategorię pakietów CASE można określić jako zbiornicy definicji wszystkich obiektów występujących w danym systemie oraz zależności między nimi. Obiekty te to różnego rodzaju specyfikacje systemowe jak diagramy przepływu danych, diagramy obiekty/związki, grafiki struktury, schematy baz danych, formatki ekranów czy zestawień wynikowych, definicje dialogu i menu oraz inne. W ten sposób narzędzia przechowywania są sercem, istotnym składnikiem integrującym całego środowiska czy stanowiska CASE. Wykraczają one poza zakres słowników/skorowidzów danych ponieważ wiążą

się z wszystkimi fazami cyklu życia systemu a poza tym z czynnościami sterowania projektem oraz modyfikacją systemu. Omawiane pakiety CASE zawierają nie tylko definicje obiektów ale i moduł z a r z ą d z a n i a obiektami. Aktualnie pakiety przechowywania są składnikami zinegrowanych środowisk CASE.

3.3 PAKIETY MODYFIKACJI I ADAPTACJI.

W tradycyjnym rozumieniu, modyfikacja i adaptacja oznacza zarówno k o r e k t ę błędów i braków jak i stałe p o - s z e r z a n i e istniejącago oprogramowania dla zaspokojenia nowych potrzeb informatycznych. Jednym z podstawowych problemów w rozbudowie systemów jest trudność w zrozumieniu ich logicznego i fizycznego projektu. Zmiana jednego, pozornie izolowanego aspektu systemu, bez analizy i sprawdzenia potencjalnego w p ł y w u na inne elementy systemu staje się źródłem poważnych problemów w fazie jego eksploatacji. Złożoność współczesnych zastosowań informatyki wymaga narzędzi wspomagających planowanie i analizę tych wpływów. Ważniejsza jest tu możliwość realizacji zmian na poziomie definiowania założeń aniżeli programowania.

Modyfikację i adaptację można przeprowadzać w dwu k i e r u n k a c h: zstępującym (forawrd engineering) i wstępującym (reverse engineering). W pierwszym przypadku zmiany wprowadzane we wstępnych fazach cyklu życia systemu transformowane są na odpowiednie specyfikacje np w fazie programowania. Odwrotnie dzieje się w przypadku modyfikacji

wstępującej, gdzie zmiany inicjowane są a specyfikacje przekształcane, począwszy od faz wdrożenia czy programowania. Modyfikację przeprowadza się zazwyczaj na zasadzie kombinacji tych dwu procedur.

3.4. WSPOMAGANIE CYKLU ŻYCIA SYSTEMU.

Aktualnie funkcjonuje wiele modeli cyklu życia systemu. Na przykład w omawianym podejściu [8] wyodrębniono następujące e t a p y tego procesu:

- planowanie strategiczne,
- analiza,
- projektowanie logiczne,
- projektowanie fizyczne,
- konstruowanie, czyli programowanie lub generowanie kodu.

W tym układzie wyróżnia się następujące rodzaje pakietów CASE (świadomie zrezygnowano tu z tłumaczenia na język polski, bowiem są to pojęcia powszechnie przyjęte w świecie):

- front-end CASE, wspomagające fazy planowania, analizy i projektowania logicznego,
- back-end CASE, które wspomagają fazy projektowania fizycznego oraz konstruowania.

Pakiety wiążące te dwa rodzaje nazywane są zintegrowanymi.

W nawiązaniu do klasyfikacji pakietów CASE Davis[3] zaproponował następujące fazy cyklu życia systemu:

a. planowanie strategiczne,
b. analiza potrzeb,
c. specyfikacje,

d. projektowanie,

e. kodowanie, testowanie i prototypowanie,

f. kompilacja i wdrożenie,

g. użytkowanie,

h. modyfikacja i archiwowanie.

Odpowiednio do tych faz autor wyodrębnia następujące rodzaje narzędzi CASE:

- pre-CASE: a,
- upper-CASE: b,c,d,
- lower-CASE: e,f,
- post-CASE: g,h.

Przyjęty jest też podział na [6]:

- upper-CASE,
- middle-CASE,
- lower-CASE.

3.5. STEROWANIE REALIZACJĄ PROJEKTU.

Większość oferowanych narzędzi CASE z założenia ukierunkowanych jest na użytkowanie przez indywidualnych twórców systemów. Jednocześnie większość systemów informatycznych projektowana jest przez z e p o ł y, grupy robocze. Stąd ogromne znaczenie posiada możliwość korzystania ze wspólnego środowiska TSI, dostępu do zasobów i modułów narzędzia oraz dokumentacji systemu. Realizacja tego założenia polega na utworzeniu i eksploatacji, w ramach funkcji przechowywania, pakietu CASE -bazy danych projektu, obejmującej wszystkie o b i e k t y związane z procesem projektowania oraz powiązania pomiędzy tymi obiektami.

Wspomaganie kierowania projektem obejmuje zarówno moduły komunikowania się w zespole, np. przesyłanie komunikatów, poczta elektroniczna jak i narzędzia osobiste np. harmonogramowania czy arkusze kalkulacyjne.

Moduły dokumentowania wspomagają wytwarzanie d o k u m e n t a c j i związanej z różnymi fazami cyklu życia systemu jak diagramy, teksty itd. Wspomaganie wytwarzania tej dokumentacji może być proste i polegać na zastosowaniu edytora tekstu albo zaawansowane jak pełne przetwarzanie dokumentów, elektroniczny skład, grafika prezentacyjna i automatyczne generowanie dokumentów.

Tak więc ogólnie pakiety kierowania projektami obejmują szacowanie zasobów, planowanie i kontrolę. Zintegrowane z nimi cechy to sterowanie bezpieczeństwem dostępu do danych, ochrona narzędzia i projektu.

3.6 BIEŻĄCE DOSKONALENIE JAKOŚCI SYSTEMU.

Narzędzia CASE są bardziej związane z jakością systemów informatycznych aniżeli efektywnością procesu TSI. Wprowadzając tu uporządkowane, systematyczne podejście można znacznie zredukować b ł ę d y i b r a k i we wczesnych fazach cyklu życia systemu. Oznacza to bardziej n i e z a w o d n e i poprawne zastosowania. W ten sposób efektywność procesu TSI niewątpliwie również wzrasta.

Aby pakiet CASE był mógł być pomyślnie użytkowany musi on

posiadać sprzężenie zwrotne, umożliwiające bieżącą kontrolę i k o r e k t ę procesów TSI. Jest to więc procedura stałego doskonalenia systemu. Winna ona objąć każdego uczestnika procesu TSI, użytkującego pakiet CASE. W ten sposób usuwanie braków, błędów i niedociągnięć nie jest jedną, wyodrębnioną fazą procesu projektowania lecz immanentnym składnikiem całego procesu.

4. DOBÓR NARZĘDZI CASE DO FAZ CYKLU ŻYCIA SYSTEMU.

Aktualnie istnieje wielka r ó ż n o r o d n o ś ć metod i technik wspomagających proces projektowania. Ich gama bieżąco poszerza się. Ich użyteczność należy rozpatrywać w aspekcie dopasowania do poszczególnych faz cyklu życia systemu. Istnieją metody i techniki adekwatne do kilku etapów. Zagadnienie to będzie rozpatrywane w nawiązaniu do zaprezentowanego w [ ] procesu.

Celem fazy planowania strategicznego jest a n a l i z a informacji gospodarczej i p o t r z e b informatycznych na ogólnym poziomie jak również ustalenie priorytetów działań w ramach procesu TSI. Traktując kwestię bardziej szczegółowo chodzi o wspomaganie analizy istotnych czynników powodzenia, analizy celów działania, modelowanie funkcjonowania przedsiębiorstwa, tworzenie architektury systemu, planowanie realizacji projektu. Typowymi technikami stosowanymi w tej fazie są modele obiekt-atrybut-związek (OAZ), powiązane z nimi macierze oraz różnego rodzaju diagramy przepływu danych. Techniki macierzowe pozwalają na analizę

zagnieżdżeń i pokrewieństw. Umożliwia to opracowanie wstępnej propozycji struktury bazy danych i grupowania funkcji i procesów. Znaczenie tej fazy jak również powiązanych technik jest istotniejsze przy przyjęciu z s t ę p u j ą c e j (ang. top-down) strategii TSI. Istnieją niezależne autonomiczne narzędzia wspomagania strategicznego jednak winny one być przede wszystkim składnikiem większego środowiska bowiem wyniki tu osiągnięte rzutują na dalszy przebieg cyklu życia systemu.

Faza analizy powinna umożliwić identyfikację c e l ó w pszczególnych systemów w ramach przedsiębiorstwa i zebranie odpowiednich informacji. Dominuje stosowanie w tej fazie metod analizy s t r u k t u r a l n e j akcentującej graficzne techniki modelowania d a n y c h i p r o c e s ó w, strukturyzacji systemu oraz opracowania słownika/skorowidza danych. Metody te uzupełniane są różnego rodzaju tekstem jak wyniki wywiadów, listy problemów, propozycji i specyfikacji, analizy kosztów i efektów. W związku z koniecznością modelowania danych i procesów, również i w tej fazie wykorzystuje się techniki diagramów OAZ oraz przepływów danych. Dodatkowo faza ta winna być wspomagana pakietami przetwarzania tekstu, grafiki prezentacyjnej i analizy finansowej.

W fazie p r o j e k t o w a n i a logicznego zostaje opracowany wstępny, ogólny projekt systemu. Projekt logiczny określa raczej "co system winien realizować aniżeli "jak", w

jaki sposób winien funkcjonować. Pozwala to określić niezależny od technologii wdrażania o p i s proponowanego systemu. Następuje tu ponownie modelowanie danych i procesów, tym razem jednak znacznie uszczegółowione. Pakiety lub moduły związane z tą fazą winny umożliwiać strukturyzację systemów i programów poprzez takie techniki jak diagramy struktury, diagramy działań, diagramy Warniera-Orra, pseudokod lub tablice decyzyjne.

W wyniku projektowania f i z y c z n e g o opracowany zostaje projekt, ukierunkowany na konkretną implementację sprzętu i oprogramowania. W związku z tym opracowuje się formatki ekranów i zestawień. Niektóre pakiety wykorzystują opracowane wcześniej modele danych dla wygenerowania schematu bazy danych a modele procesów dla automatycznego generowania całości lub części kodu. Moduły prototypowania umożliwiają symulację i dobór najbardziej adekwatnego rozwiązania spośród różnych alternatyw.

Wynikiem fazy konstruowania jest przetestowany, funkcjonujący system informatyczny, bazujący na projekcie fizycznym. Pojęcie konstruowania obejmuje zarówno konwencjonalne p r o g r a m o w a n i e jak i g e n e r o w a n i e k o d u na podstawie specyfikacji programowych. Pakiety CASE obejmują również narzędzia utrzymywania jakości i korekty programów, jak np debuggery. W fazie tej dominują aktualnie języki trzeciej generacji (3GL), np COBOL, choć inicjowane są zastosowania języków czwartej generacji(4GL) jak FOCUS czy

IDSAL. COBOL jest w dalszym ciągu dominującym językiem szczególnie w zastosowaniach gospodarczych informatyki z wykorzystaniem instalacji wielkokomputerowych. Pakiety CASE niewiele zmieniły w tym względzie.

Ogromna ilość metod i technik, głównie diagramowych, stosowanych w różnych fazach cyklu życia systemu skłania do ich porównań i k l a s y f i k a c j i. W najpełniejszy sposób dokonano tego w opracowaniu [7]. Techniki te wspomagają :

a. modelowanie przedsiębiorstwa i jego działań, opis hierarchicznie zdekomponowanych funkcji i procesów przedsiębiorstwa, ogólnego przepływu danych pomiędzy zdarzeniami a procesami, składników systemu i związków pomiędzy nimi; wymienione techniki związane są z fazami planowania, analizy i projektowania systemów,
b. architekturę programów - ogólną architekturę bądź zbioru programów ze wskazaniem oddzielnych modułów,
c. szczegółową logikę programów w ramach jednego modułu programowego,
d. tworzenie struktur zbiorów,
f. tworzenie struktur baz danych.

Z dwoma ostatnimi typami technik wiążą się modele danych.

Wymienione wyżej kategorie technik wspomagających realizację cyklu życia systemu cechują się wielką różnorodnością rozwiązań. Istnieją techniki możliwe do wykorzystania w kilku

fazach tego cyklu, wykonujące różne zadania. Zależności te zaprezentowano w tablicy 1[7]. Zawiera ona najbardziej popularne rozwiązania. W kolumnie rodzaje technik poszczególne litery nawiązują do klasyfikacji, przedstawionej wyżej.

Tablica 1

Porównanie technik tworzenia systemów informatycznych

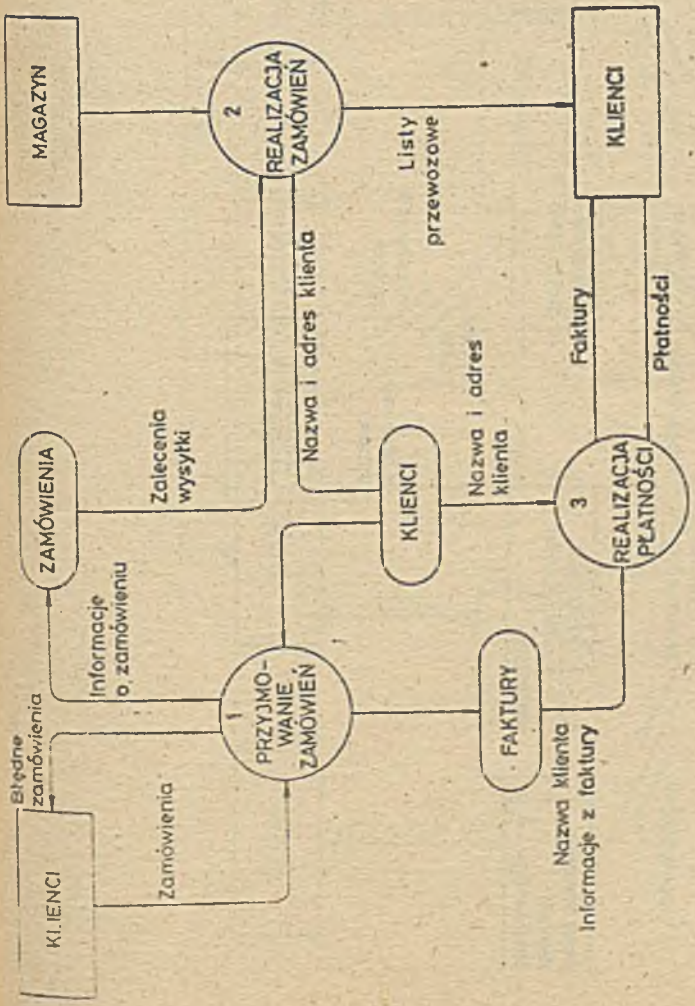
LP	Nazwa techniki	Rodzaj techniki
1.	diagramy przepływu danych (Yourdona/ DeMarco lub Gane'a/Sarsona	a
2.	diagramy dekompozycji funkcjonalnej	a
3.	schematy struktury	a b
4.	diagramy HIPO	a b c
5.	diagramy Warniera/Orra	a b c d
6.	diagramy Jacksona	b d
7.	schematy blokowe	c
8.	język strukturalny i pseudokod	c
9.	diagramy Nassi/Schneidermana	c
10.	diagramy działań	a b c d
11.	drzewa decyzyjne	c
12.	tablice decyzyjne	c
13.	diagramy struktur danych	e
14.	diagramy obiekt/atrybut/związek	e
15.	diagramy nawigacji danych	b e
16.	diagramy HOS	a b c d

Celem - niniejszym opracowanie przedstawić wszystkie techniki diagramowe wyspecyfikowane w tablicy 1. W związku z tym na rysunkach 3. 4 i 5 zaprezentowano najbardziej uznane t e c h n

i k i analizy danych oraz analizy funkcji tj. diagram przepływu danych, schemat struktury oraz diagram obiekt/atrybut/związek.

Rysunek 3

Diagram przepływu danych



Schemat struktury

Rysunek 1

Rysunek 3

Diagram obiekt/atrybut/kardynalność

5. RYNKOWA OCENA NARZĘDZI CASE.

R y n e k pakietów CASE jest niezwykle różnorodny i, jak wspomniano o tym już we wstępie, pojemny finansowo. Dostępne są wszystkie rodzaje pakietów wyodrębnionych w części 3 niniejszego opracowania. Wspomagają one nawet poszerzony w stosunku do tablicy 1 zestaw metod i technik. Analizę porównawczą tych narzędzi przeprowadzono w [4]. Ocenie poddano 30, głównie zintegrowanych pakietów. Brano przy tym pod uwagę m. in. następujące k r y t e r i a: funkcjonalność metodyczną, jego integralność, sprzęt na którym jest eksploatowany, możliwość realizacji prototypowania, stosowane metody, techniki i standardy, szkolenia i materiały metodyczne oferowane przez producentów, charakterystykę rynkową pakietu oraz cenę. Waha się ona od kilku do kilkudziesięciu tysięcy dolarów. Oferuje się również narzędzia wycinkowe, za kilkaset dolarów. Ich wartość użytkowa jest jednak znikoma. Najbardziej pogłębione, prowadzone na bieżąco badania narzędzi CASE prowadzi amerykańska firma CASE Research. Jej k a l k u l a c j a zakupu przeciętnego, zintegrowanego pakietu CASE przedstawia się następująco:

- stacja robocza	7 000 dol.
- pakiet CASE	8 000 dol.
- szkolenie	2 000 dol.
- eksperymentowanie i uczenie się	+
Razem	17 000 dol. +

Należy jednak zdawać sobie sprawę, iż ten niepozorny + może kilkakrotnie powiększyć ogólną sumę zakupu. W fazie wdrażania pakietu niezbędny bowiem okazuje się udział jego producentów

bądź wyspecjalizowanych firm doradczych.

Interesującą, choć płytką analizę porównawczą pakietów CASE zaprezentowano w [8]. Wydaje się ona jednak użyteczna jako swoiste wprowadzenie do aktualnej o f e r t y rynkowej w tej dziedzinie. Bierze się tu pod uwagę następujące rodzaje narzędzi:

a. pakiety front-end,

b. pakiety back-end,

c. przechowywania danych,

. modyfikacji i adaptacji,

e. kierowania projektami.

Odpowiednie porównanie zawiera tablica 2. Wymieniono w niej jedynie nazwy pakietów , pomijając nazwy producentów i sprzęt na którym są eksploatowane. Bliższe dane zawiera opracowanie [4]. Wskazując funkcję, której dany pakiet służy wprowadzono ocenę jego zakresu jako:

1. pełne wspomaganie,

2. częściowe wspomaganie.

Tak więc symbol c1 oznacza, iż dany pakiet w sposób pełny wspomaga przechowywanie danych.

8. ZAKONCZENIE.

W niniejszym opracowaniu nakreślono ogólne trendy rozwoju narzędzi CASE. Ta żywiołowo rozwijająca się dziedzina

Tablica 2

Możliwości wybranych pakietów CASE

LP	Nazwa produktu	Funkcje
1	Ze[illegible]	a1 b1 c1 d1
2	[illegible]romod	a1 b1 c1 d1
3	JEW/GAMMA	a1 b1 c1 d2
4	PAGEPLI	a1 b1 c1 d2
5	Foundation	a1 b1 c1 b1
6	IEF	a1 b1 c1
7.	CASE 2000	a1 b2 c1 e1
8.	System Developer	a1 b2 c2
9.	Auto-Mate	a1 b2
10.	TEAMWORK	a1 c2 e2
11	Excelerate	a1 c2 e2
...	Pro[illegible]tWorkbench	a1 c2
13	Systems Analyst	a1 c2
14	Maestro	a1 e1
15.	Analyst Toolkit	a1
16.	APS	b1 c2
17	Cor[illegible]ation	b1 e2
18.	Recorder	d1
19	Mc[illegible]	e1

informatyki wymaga stałej obserwacji i analiz przede wszystkim rynkowych. Decydującymi czynnikami wprowadzania pakietów CASE na rynek krajowy będzie e d u k a c j a w zakresie metodycznym i narzędziowym a przede wszystkim możliwości ich użytkowania dla tworzenia systemów informatycznych w organizacjach gospodarczych i administracyjnych.

BIBLIOGRAFIA.

1. Analyst Workbench, Infotech. State of the Art Report, Maidenhead 1987.
2. Current Trends in Information Systems Development Methodologies, Preprints of the Polish-Scandinavian Seminar Paraszyno, June 1988.
3. Davis,J.M.,The Evolution of CASE, The SQL Database Journal,ORACLE, vol.III, no.2, pp.88-90.
4. Elek A. Rawiński T. Wrycza S.,Charakterystyka wybranych narzędzi komputerowo wspomaganego tworzenia systemów informatycznych, Prace Badawcze Politechniki Gdańskiej. nr 162, 1989.
5. Gane,C.P., Sarson, T., Structured Systems Analysis. Tools and Techniques, Prentice hall, New York 1979.
6. Gibson M.L.,The CASE Philosophy, BYTE, April 1989, pp. 208-218.
7. Martin, J.,McClure C., Structured Techniques. A Basis for CASE, Prentice Hall, New York 1988 .
8. Merlyn V., Boone G., CASE Product Classification Model, CASE Bulletin, March 1989.
9. Soave Comments and Constituents of Contemporary Information Systems Development Methodologies. Proceedings of the IMACS, IFAC, IIASA Third International Symposium on Systems

Analysis and Simulation, Berlin, September 1988, band II, pp. 102-106.

10. The Extension Possibilities of Computerized Tool's Scope Supporting ISAC Methodology. Proceedings of the Polish-Scandinavian Seminar on Current Trends in Information Systems Development Methodologies, Paraszyno, June 1988. Republished in: Proceedings of International IFIP WG 8.1. Conference on Information Systems Work and Organization Design, Berlin, July 1989, Working Group 2, pp. 100-106.

11. The Content and Use of Analyst Workbench in Information Systems Development. Proceedings of Xth International Conference on Systems Science, Wrocław, September 1989.

12. Roles and Connections among Constituents of Information Systems Development Methodologies. Proceedings of ISICS IV: The Fourth International Symposium on Computer and Information Sciences, Cesme /Turkey/, October-November 1989, pp. 1175-1182.

13. The Impact of CASE Workbenches on Teamwork of Information Systems Developers. Proceedings of the First European Conference on Computer-Supported Co-operative Work, London, September 1989. To appear also in: Finkelstein A., Tauber M., Traunmueller R. /eds/, Proceedings of IFIP TC8 WG 8.1 Working Conference on Human Factors in Information Systems Analysis and Design, Schaerding, Austria, June 1990.

14. The ISAC-Driven Transition between Requirements Analysis and Conceptual Modelling. Discussion Paper of Department of Information Systems of University of Gdańsk, Sopot, 1989, pp. 1-30.

ZAŁĄCZNIK

PROGRAM KURSU
Z ZAKRESU NARZĘDZI CASE

Interesującą zawartość merytoryczną ma kurs z zakresu narzędzi CASE, prowadzony przez amerykańską firmę Digital ConsultingProgram ten jest jak gdyby specyfikacją podstawowych, aktualnych problemów, obszarów zainteresowań i trendów w tej dziedzinie Oto treść programu:

1. Przegląd ogólny komputerowo wspomaganej tworzenia systemów informatycznych CASE.
2. Technologia CASE
   a. Czym jest technologia CASE,
   b. Automatyzacja automatyzacji,
   c. Automatyzacja cyklu życia systemu,
   d. Powiązanie automatyzacji projektowania z automatyzacją programowania,
   e. Pełne rozwiązania kryzysu oprogramowania,
   f. Ewolucja technologii CASE,
   g. Bieżące problemy CASE.
3. Charakterystyka systemów CASE
   a. Nowe środowisko tworzenia oprogramowania,
   b. Graficzne możliwości projektowania,
   c. Centralna baza danych,
   d. Zintegrowane zestawy narzędzi,
   e. Powiązanie z pełnym cyklem życia,
   f. Wspomaganie prototypowania,
   g. Automatyczne generowanie kodu,
   h. Automatyczna kontrola poprawności i spójności,

i. Kluczowe pojęcia systemu CASE.

j. Kryteria porównawcze systemów CASE.

4. Przegląd dostępnych produktów CASE

a. Systemy CASE.

b. Warsztaty CASE.

c. Metodyki powiązane z pakietami CASE.

d. Przegląd rynku CASE. główni sprzedawcy i kierunki rozwoju.

5. Podstawy sprzętowe użytkowania narzędzi CASE

a. Stacje robocze.

b. Komputery głównego szeregu.

c. Sieci komputerowe.

d. Architektury jedno-, dwu i trójwarstwowe.

6. Modyfikacje cyklu życia systemu.

a. Inicjowanie systemu.

b. Projektowanie programów ukierunkowanych obiektowo.

c. Minimalizacja wdrażania ręcznego.

d. Szybkie iteracyjne prototypowanie.

e. Minimalizacja potrzeb testowania.

f. Powiązanie strategii tworzenia i eksploatacji.

7. Korzystanie z efektywności CASE.

a. Wzrost efektywności o współczynnik równy 20

b. Wzrost skuteczności twórców systemów.

c. Zmniejszenie wysiłku na zmiany i próbną eksploatację.

d. Analiza przypadków zastosowań CASE.

e. Korzyści krótko i długofalowe

f. Sposoby wdrażania CASE.

8. Technologie pokrewne
   a. Dominująca rola CASE.
   b. Adaptacja oprogramowania jako metodyka TSI.
   c. Rola COBOLu.
   d. Języki czwartej generacji.
   e. Technologia piątej generacji.
   f. Nowe problemy.
9. Przyszłe problemy
   a. Przyjazne środowisko.
   b. Inteligentne systemy szkolenia.
   c. Inteligentne metodyki.
   d. Ponowne użytkowanie oprogramowania.

Komputerowe wspomaganie tworzenia systemów metodą Jacksona, doświadczenia w wykorzystaniu pakietów PDF i SPEEDBUILDER

Zbigniew Benedyk
Befama
Zakładowy Ośrodek Informatyki i Organizacji
43-300 Bielsko-Biała
ul. Piotrowskiego 13

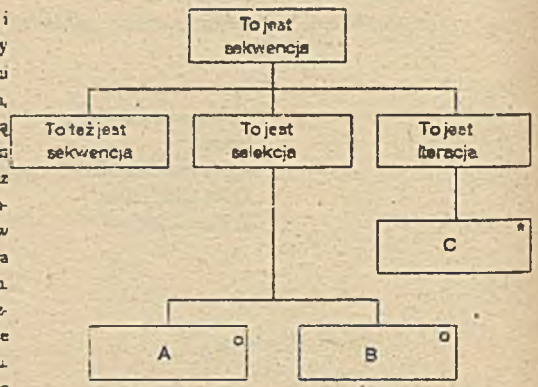
1 Wstęp

Powszechnie w kręgach polskich informatyków znane jest nazwisko Michaela Jacksona (nie tego od młodzieżowych przebojów lecz tego od diagramów), wiele spotyka się odwołań do jego metody budowania programów a notacja diagramów Jacksona weszła już niemal do folkloru informatycznego, jednak w praktyce wiedza ta jest fragmentaryczna i mało popularna (przynajmniej ja odnoszę takie wrażenie). Niemal co roku jest wznawiane wydanie jego pięknej książki 'Principles of Program Design', niestety w Polsce jest ona rarytasem. Jeszcze mniej wiadomo na temat dalszych prac Jacksona dotyczących metod projektowania systemów. Wyniki tych prac przedstawione zostały w książce 'System Development' wydanej w 1983 roku również autorstwa M. A. Jacksona. Metoda tam przedstawiona nazwana została JSD (Jackson System Development) i wraz z JSP (Jackson Structure Principles) stanowi przedmiot niniejszego opracowania. Oprócz opracowań teoretycznych istnieją również pakiety typu CASE wspomagające JSD i JSP wyprodukowane przez firmę M. Jackson. Pakiety te to odpowiednio SPEEDBUILDER i PDF. Oba zostały zakupione przez Befamę i są używane w Zakładowym Ośrodku Informatyki. Chciałbym tutaj podzielić się spostrzeżeniami z wdrażania CASE do codziennej praktyki informatyka, nie mogę jednak tego uczynić nie przedstawiając kolejno metody i narzędzi. Zatem dwa rozdziały poświęcam na krótkie opisanie podstawowych idei JSP i JSD, żeby w następnych mieć bazę do przedstawienia odpowiednich pakietów oprogramowania. Na prezentację spostrzeżeń, zachwytów i zastrzeżeń pozostawiam rozdział ostatni.

2 JSP

Jackson structure principle JSP jest nazwą metody, która prowadzi od specyfikacji programu do wygenerowania jego kodu. Podstawowym założeniem jej jest, że struktura programu musi ściśle odpowiadać strukturom wszystkich danych związanych z programem. Jeżeli struktury danych różnią się, to należy znaleźć taką strukturę, która jest nadstrukturą dla wszystkich struktur. Nie zawsze taki postulat można zrealizować, wtedy można wprowadzić struktury pośrednie tak aby uzyskać odpowiednie zgodności pomiędzy strukturami danych wejściowych i pośrednich oraz pośrednich i wyjściowych, program zaś podzielić na moduły, których wykonanie może być odpowiednio synchronizowane. JSP uczy jak zapisywać struktury danych, w jaki sposób przechodzić ze struktury danych do struktury programu, jak 'ubrać' tą strukturę w instrukcje konkretnego języka programowania, jak radzić sobie z niezgodnymi i sprzecznymi strukturami danych, jak zabezpieczać program przed skutkami danych wejściowych niezgodnych ze specyfikacją oraz przedstawia sposoby rozwiązania wielu technicznych szczegółów programowania. Programowanie przestaje być sztuką a staje się zwykłym chociaż pięknym rzemiosłem. Poniżej postaram się w zarysie przedstawić poszczególne aspekty JSP.

Zarówno struktury danych jak i programów przedstawiane są w JSP przy pomocy diagramów zwanych diagramami Jacksona. Diagram Jacksona jest drzewem, którego każdy węzeł jest sekwencją, selekcją lub iteracją. Na rysunku obok przedstawiamy sposób rysowania tych diagramów oraz stosowane oznaczenia. Liście A i B są elementami selekcji (wskazują to kółeczka w prawym górnym rogu klatki), której nazwa jest wpisywana w węźle nadrzędnym. Selekcja jest alternatywą wykluczającą, z węzłami A i B związane są warunki, które decydują o wyborze konkretnego elementu. Jeśli wybrany zostanie węzeł A, to jego opuszczenie determinuje opuszczenie całego poddrzewa narysowanego na rysunku 'To jest selekcja'. Oczywiście selekcja może składać się z większej niż dwa liczby elementów. Liść C jest elementem iteracji 'To jest iteracja', mówi o tym gwiazdka w prawym górnym rogu. Poddrzewo C będzie iterowane dopóty, dopóki spełniony będzie warunek związany z C. Ostatnim typem węzła jest sekwencja. Sekwencja nie jest oznaczana żadnym specjalnym znaczkiem, mówi ona, że jej elementy występują w kolejności jeden za drugim. Z sekwencją nie związany jest żaden warunek. Liście drzewa traktowane są jako sekwencje.



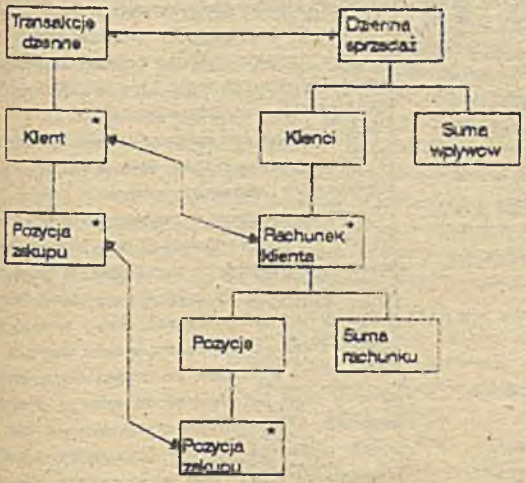
rys. 2.1

Oprócz notacji graficznej JSP posiada notację tekstową. Diagram z rys. 2.1 zapisałem obok w tej notacji. Wydaje mi się, że zapis ten jest tak oczywisty, że komentarz do niego jest zbyteczny. Ponad to co zostało przedstawione na diagramie dopisałem hipotetyczne warunki, aby przedstawić sposób ich zapisu. Słowa kluczowe wytłuściłem.

```
To jest sekwencja seq
To tez jest sekwencja seq
        |
To tez jest sekwencja end
To jest selekcja select kod = 'A'
            A seq
              |
            A end
          or kod = 'B'
            B seq
              |
            B end
To jest selekcja end
To jest iteracja iter while i < 100
            C seq
              |
            C end
To jest iteracja end
To jest sekwencja end
```

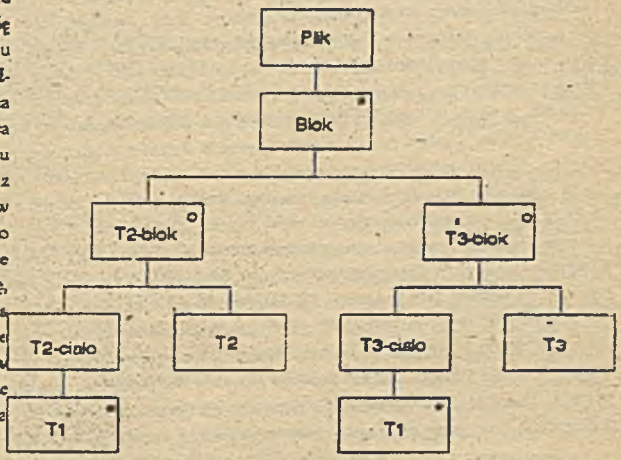
Strukturę programu otrzymujemy ze struktury danych poprzez wyspecyfikowanie czynności, które należy w programie wykonać. W tym celu uzupełniamy diagram danych o dodatkowe liście, które zawierają listy czynności (podobnie jak elementy selekcji i iteracji zawierają warunki). Jako nazw dla tych list używamy liczb (nie koniecznie kolejnych). W sytuacji tutaj przedstawionej dopisalibyśmy kolejne poziomy drzewa np.: do A dołączylibyśmy 1, do B - 2 i do C - 3. Listy czynności zapisalibyśmy w miejscu pionowych kresek widocznych obok. Dalsza praca nad programem jest już sprawą formalną. Należy warunki i czynności zapisać w języku w którym ma powstać tekst programu oraz dopisać odpowiednie sekcje danych i identyfikacji programu. Niestety nie zawsze program jest budowany na podstawie tylko jednej struktury danych. Przeważnie, częściej mamy do czynienia z sytuacją kiedy wejściu programu jest kilka struktur danych. Zazwyczaj mamy jakąś strukturę danych wejściowych i jakąś strukturę danych wyjściowych. Rozpatrzmy następujący problem: należy sporządzić raport dzienny ze sprzedaży w pewnym sklepie. Na wejściu mamy zbiór wszystkich rachunków wystawionych klientom, na wyjściu chcemy uzyskać raport, w którym zestawione wszystkie pozycje rachunków.

chcemy mieć podsumowanie dla każdego klienta i chcemy mieć podsumowany dzienny utarg. Na rys. 2.2 przedstawiłem diagramy danych wejściowych (lewy) i raportu (prawy), łatwo znaleźć odpowiadające sobie węzły na obu diagramach i łatwo również zauważyć, że program powinien mieć strukturę analogiczną do drzewa 'Dzienna sprzedaż'. Nie trudno dołączyć dalsze linie czynności, należy jednak zastanowić się nad tym jakie warunki powinny być wpisane dla iteracji i jak zabezpieczyć się przed skutkami błędnych danych np. na wejściu pojawi się zapis, który nie przedstawia sobą pozycji zakupu. Problem pierwszy rozwiązuje się często poprzez wyróżnienie pierwszego rekordu. W ten sposób uzyskujemy możliwość odniesienia się do danych pamiętanych. Pozadto chwyt ten pomaga często przy rozwiązywaniu niektórych kolizji strukturalnych chociaż czasami zachodzi potrzeba wyróżnienia więcej niż tylko jednego rekordu. Jeśli idzie o przeciwdziałanie złym skutkom nieprzewidzianych danych, to wstawienie odpowiednich selekcji pozwala na wyeliminowanie takich danych z dalszego przetwarzania i sporządzenie raportu o nieprzewidzianych wejścia. Szczegóły na ten temat można znaleźć w książce Jacksona. Osobiście uważam, że chwila zastanowienia (no może nie całkiem krótka) pozwoli na uzupełnienie niedopowiedzianych reguł. Szczególnie cennym wydaje mi się rozwiązanie problemu kojarzenia rekordów dwóch plików sekwencyjnych. Problem ten stanowi podstawę wszystkich programów aktualizacji wsadowych. W książce 'Principles of Program Design' jest on szczegółowo omówiony.



rys. 2.2

Na rysunku obok przedstawiłem sytuację, kiedy wielokrotne czytanie wstępne nie pomoże zapisać warunki definiujące T2-blok i T3-blok. O tym, który to jest typ bloku danych możemy dowiedzieć się dopiero po przeczytaniu pierwszego w bloku zapisu różnego od T1. Do rozwiązywania tego rodzaju problemów używa się w JSP mechanizmu nawrotów. Selekcje złożone z więcej niż dwóch elementów dzieli się na selekcje złożone tylko z dwóch elementów, następnie dla każdej z nich przyjmuje się, że pierwsza z dróg jest słuszna więc poruszamy się w drzewie przez nią wyznaczonym, w pewnym momencie może okazać się, że droga jest zła zatem druga możliwość musi zachodzić.



rys. 2.3

Formalnie powrót wygląda następująco: element select zmienia się na element typu posit , ex zmienia się na admit, pojawia się nowy typ elementu - quit, który faktycznie jest skokiem warunkowym. Zobaczmy to na następującym przykładzie: czytamy kartę na której powinny być zapisane dwie wartości p1 i p2 , kolejno, każda z tych wartości podlega kontroli, nie można skontrolować wartości p2 nie upewniwszy się uprzednio, że p1 jest poprawne, należy wykonać procedurę P jeśli karta jest poprawna. Załóżmy ponadto, że procedury kontroli są skomplikowane i nie można zapisać ich jako koniunkcji warunków w selekcji.

```
P:Karta        seq
                | (*  czytanie karty  *)
  cialo proc.   posit (* karta poprawna*)
    karta poprawna   seq
      p1 poprawne     seq
                       | (*  sprawdzanie pola p1 *)
    cialo proc.   quit if p1 bledne
      p1 poprawne     end
      p2 poprawne     seq
                       | (*  sprawdzanie pole p2 *)
    cialo proc.   quit if p2 bledne
      p2 poprawne     end
                       | (*   procedura P   *)
    karta poprawna   end
    cialo proc.   admit (*karta bledna *)
    karta bledna     seq
                       | (* procedura karty blednej*)
    karta bledna     end
  cialo proc.     end
                  | (*  procedury konca programu *)
P:Karta           end
```

Nawroty mogą być związane również z iteracjami. W tym przypadku zamiast słowa kluczowego posit użyjemy słowa kluczowego positer.

Przyjmując założenie o poprawności drzewa posit (positer) musimy wykonać w programie wszystkie czynności przynależne do tego drzewa. W przypadku kiedy okazuje się, że założenie nie jest spełnione, pozostają nam efekty uboczne. Efekty te mogą być niedopuszczalne, wtedy należy wycofać wszystkie ich skutki przed wyskokiem do drzewa admit. Efekty uboczne mogą być obojętne dla dalszego przebiegu programu, wtedy jedynym problemem jest fakt, że zostały wykonane operacje niepotrzebne - zatem niepotrzebnie zużyty czas pracy maszyny. Ostatnia kategoria to czynności, które występują zarówno w gałęzi posit jak i admit, wykonanie ich w posit powoduje, że przechodząc do gałęzi admit mamy za część pracy wykonaną, efekty dobroczynne. Z powyższego widać, że chociaż istnieje mechanizm pozwalający rozwiązywać problemy związane z trudnościami w definiowaniu wyrażeń warunkowych dla selekcji i iteracji, należy stosować go z rozsądkiem. Zazwyczaj lepiej zadać sobie trochę trudu i poszukać odpowiedniejszego przedstawienia struktur danych, niż później korzystać z wyskoków z wnętrza bloku (to rozwiązanie łatwiej znaleźć i jest często stosowane przez niedbałych programistów).

Ostatnim problemem konstruowania programów, który chciałbym tutaj poruszyć jest całkowita niezgodność struktur danych. Załóżmy, że mamy napisać program, który czyta macierz wierszami - elementy po elemencie i ma za zadanie wyprowadzić ją kolumnami. W tym przypadku struktura pliku wejściowego jest tak dokładnie niezgodna ze struktura pliku wyjściowego, że nie warto nawet próbować szukać jakichś analogii. Jeśli tablica jest na tyle mała, że w całości możemy zmieścić ją w pamięci operacyjnej wraz z tekstem programu, to możemy najpierw wczytać ją całą a następnie wydrukować w innym porządku. Jeśli tablica jest zbyt duża, to napiszemy program, który zapisze tablicę wraz z współrzędnymi każdego elementu w zbiorze pośrednim, zbiór ten posortujemy i następny program wydrukuje tablicę kolumnami. Raczej rzadko w praktyce występują sytuacje kiedy struktury są tak w całości niezgodne jak w przykładzie z tablicą. Częściej niezgodności dotyczą fragmentów pliku, zatem bufor pośredni możemy zmieścić w programie jako zmienną odpowiedniego typu. Zamiast programów piszącego i czytającego plik pośredni możemy użyć procedur. Pozostaje do rozwiązania tylko zagadnienie synchronizacji tych procedur. JSP proponuje aby nie tworzyć nadrzędnego modułu synchronizującego lecz potraktować jeden z modułów (obojętne czy będzie to moduł piszący czy czytający) jako nadrzędny a drugi wywoływać w nim jako procedurę. Technika ta nazwana została inwersją programową. Jackson w swojej książce dyskutuje również złożone inwersje programowe. Występują one wtedy gdy nie wystarcza jeden bufor pośredni. Ramy tego opracowania nie pozwalają na pełniejsze rozważania zainteresowanych, jednak mam nadzieję, że przynajmniej w zarysie przedstawiłem naukę pana Jacksona dotyczące produkcji programów.

3 JSD

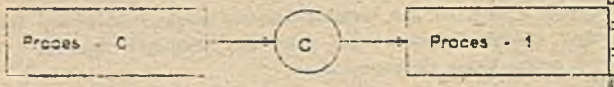
JSD (Jackson system development) jest dzieckiem tego samego ojca co JSP i nosi te same znamiona co JSP tzn. najpierw ściśle opisuje świat zewnętrzny, następnie przy pomocy dokładnie zdefiniowanych transformacji odwzorowuje w model i dalej uzupełnia ten model o listy operacji które przekształcają dane wejściowe w wyjściowe. W przypadku JSP celem było wyprodukowanie programu. Cele JSD są bardziej ogólne, światem zewnętrznym jest obszar zainteresowań zamawiającego system informatyczny, celem zaś właśnie ten system. JSD obejmuje całokształt działań od specyfikacji systemu poprzez jego projektowanie, implementację, integrację do zakończenia jego życia. W JSP program, który ma spełnić specyfikację rozbijamy na sekwencyjne procesy, JSP klasyfikuje niezgodności struktur i mówi jak skomponować kompletny program. W JSD sytuacja jest odwrotna, zaczynamy od specyfikowania rzeczywistości w postaci procesów sekwencyjnych. Dalsza działalność jest raczej syntezą tych drobnych procesów w jeden system. Metody tej syntezy czerpiemy wprost z JSP, głównie wykorzystujemy metody bazujące na inwersji programowej. Dotychczas byliśmy przyzwyczajeni do ostrego podziału prac związanych z budowaniem systemu na fazy analizy, projektowania i programowania. W JSD podziały zacierają się, wręcz nie można oddzielić projektowania od programowania (linty takie nie istnieją). W JSD wyróżnia się sześć kroków produkcji systemu:

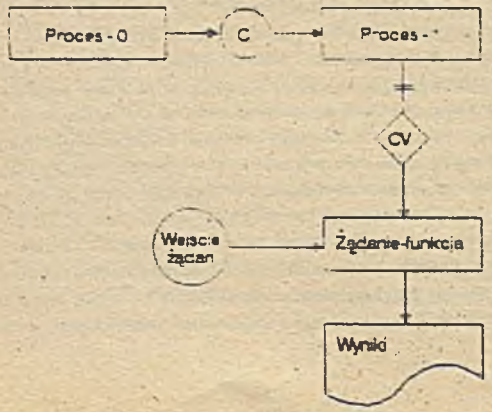
1. Krok encji i akcji; na tym etapie definiujemy obszar zainteresowań światem rzeczywistym poprzez listowanie encji (obiektów istniejących fizycznie) i akcji, na których system będzie się koncentrował.
2. Krok opisu historii życia encji; na tym etapie porządkujemy akcje wykonywane przez encje lub którym encje podlegają, zależności czasowe przedstawiamy w postaci diagramów Jacksona.
3. Krok inicjacji modelu; na tym etapie opis rzeczywistości w terminach encji i akcji jest przekształcany w model procesu i ustalane są powiązania modelu ze światem rzeczywistym.
4. Krok funkcji; - specyfikujemy funkcje systemu służące do wyprowadzania danych z systemu, mogą również zostać dołączone dodatkowe procesy jeśli są potrzebne.
5. Krok szeregowania procesów; - rozważamy zapekty związane z zależnościami czasowymi pomiędzy procesami, podejmujemy decyzję czy system będzie systemem wsadowym czy działającym w czasie rzeczywistym lub częściowo rzeczywistym.
6. Krok implementacji; na tym etapie podejmujemy decyzje dotyczące użycia konkretnego rodzaju sprzętu i oprogramowania niezbędnego do pracy systemu, decydujemy o mechanizmach synchronizacji i pielęgnacji systemu, użycia modelu opisu danych (bazy danych, niezależne pliki, jakiego rodzaju) itp..

Punktem startowym w budowaniu systemu metodą JSD jest jedynie ogólna idea dotycząca zakresu systemu. Pierwsze kroki koncentrujemy na bardziej ścisłym opisaniu zakresu systemu. Czynimy to poprzez specyfikowanie tego co jest najbardziej przedmioty (niekoniecznie fizyczne), które istnieją niezależnie od przyszłego systemu (nazywam je encjami ponieważ określenie obiekt zarezerwowane jest dla określania przedmiotów systemów - akcja, encja, proces, wektor stanu itd - a określenie encja jako tłumaczenie angielskiego entity zostało już wprowadzone do polskiej literatury informatycznej) oraz akcje z tymi przedmiotami związane. Stwierdzenie, że encja musi istnieć niezależnie od systemu eliminuje z listy encji takie rzeczy jak np. raporty błędów. Warto również zwrócić tu uwagę, że w JSD encją nie jest coś co nie jest związane z rzeczywistymi zdarzeniami. Ponadto encja musi być rozpatrywana jako byt unikalny, jeśli istnieje więcej niż jedna inkarnacja encji, to należy wprowadzić odpowiednie identyfikatory. Akcje w JSD są obiektami atomowymi, nie mogą być podzielone na drobniejsze elementy. Akcje zajmują punkt czasowy i nie są rozciągnięte w czasie, muszą istnieć niezależnie od systemu (analogicznie - wyprowadzenie raportu błędów nie może być kwalifikowane jakoakcja).

Do listy dopisujemy atrybuty wszystkich akcji oraz opisujemy zdarzenia zachodzące w świecie rzeczywistym, które są przyczynami akcji. Lista akcji może być w przyszłości poszerzona o kolejne akcje. Szczególnie często zachodzi taka potrzeba wtedy gdy specyfikujemy funkcje.

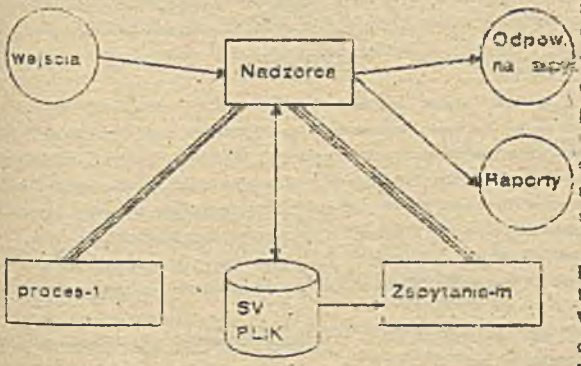
Ponieważ akcje encji (encja może być podmiotem lub przedmiotem akcji) są w swiecie zewnętrznym uporządkowane, musimy wyrazić ten porządek w naszym modelu. Czynimy to przy pomocy diagramów przedstawionych w rozdziale drugim. Dla każdej encji rysujemy diagram wyrażający kolejność akcji z nią związanych, diagram ten opisuje historie życia encji. Z informatycznego punktu widzenia mamy teraz do czynienia z kolekcją procesów sekwencyjnych. Procesy te zachodzą w świecie rzeczywistym, teraz naszym zadaniem będzie odwzorowanie ich w model systemu informatycznego. W terminologii JSD rozpoczynamy krok implementacji modelu. Idea transformacji modelu świata rzeczywistego w model systemu jest bardzo prosta: każdy proces świata zewnętrznego jest odbiciem procesu sekwencyjnego w systemie. Pozostaje tylko ustalić połączenia pomiędzy procesami rzeczywistymi i procesami systemu. Standardowym połączeniem jest strumień danych. Każda akcja wchodząca w skład procesu rzeczywistego wysyła komunikat. Komunikaty te tworzą dane wejściowe do procesów systemu. Graficznie przedstawiamy to w postaci diagramu zwanego SSD, (System Specification Diagram, rysunek obok). Standardowo oznacza się proces rzeczywisty wskaźnikiem 0 a proces modelu 1. Kółko łączące procesy oznacza strumień danych. Taka komunikacja tylko na pozór wygląda prosto, musimy zdać sobie sprawę z tego, że komunikaty ze świata rzeczywistego mogą dochodzić niekompletne, sfałszowane a nawet niektóre mogą nie dojść wcale. Czasami może być niezbędny cały podsystem przyjmowania i wstępnej obróbki tych komunikatów. Również może okazać się, że wyspecyfikowane wcześniej akcje mogą być trudne lub wręcz niemożliwe do identyfikacji (np. w systemie bankowym wyspecyfikowaliśmy akcję - klient wydaje swoje pieniądze, jak mamy teraz uzyskać informacje o takiej akcji ?). Musimy podjąć 'męską' decyzję i zrezygnować z niektórych potencjalnych możliwości systemu. Istnieją sytuacje kiedy komunikacja ze światem zewnętrznym poprzez strumień danych jest niepraktyczna, w JSD dopuszcza się jeszcze jeden sposób komunikacji. Jest to powiązanie poprzez wektor stanu. Ponieważ jest ono charakterystyczne dla funkcji, wrócę do niego później. Kiedy uporamy się już z wszystkimi tymi problemami przechodzimy do specyfikowania funkcji systemu. W JSD funkcje specyfikuje się formalnie: jeśli w świecie rzeczywistym zajdzie taka kombinacja zdarzeń, to system powinien wyprodukować takie wyniki'. Ponieważ nasz system jest modelem świata rzeczywistego, takie sformułowanie zadania jest najbardziej naturalne. Należy tylko sprawdzić czy na każde pytanie postawione przez zamawiającego system możemy zagwarantować odpowiedź, jeśli nie znaczy to, że nie wyspecyfikowaliśmy jakiejś akcji lub jakiejś encji. Jeśli nie zrezygnowaliśmy z nich w sposób świadomy (ograniczając zakres systemu) , to musimy wrócić do początku i braki uzupełnić. Pytania do systemu mogą dotyczyć konkretnej inkarnacji encji, wtedy wystarczy sięgnąć do wektora stanu odpowiedniego procesu (proces dotyczy jednej konkretnej inkarnacji encji, związany jest z nim zespół zmiennych lokalnych określających stan encji zwany wektorem stanu) i wysłać wyniki, lub mogą dotyczyć wielu inkarnacji. W każdym przypadku niezbędny jest nowy proces produkujący dane wyjściowe w odpowiedzi na dane żądań. Proces ten sięga do wektora stanu procesu modelującego świat rzeczywisty. Taki rodzaj komunikacji nazywamy powiązaniem poprzez wektor stanu. Na SSD oznaczmy je rombem. Podwójna kreska na diagramie oznacza, że proces 'Żądanie-funkcja' sięga do wektorów stanu wielu procesów 'Proces - 1' chociaż w danej chwili sięga tylko do jednego.





Podczas specyfikowania funkcji nie rozważamy bezpośrednio problemów związanych z zależnością czasową jako czynnikami wpływającymi na otrzymywane dane wyjściowe. Działanie modelu jest opóźnione stosunku do zdarzeń świata rzeczywistego. Spowodowane to jest czasem potrzebnym na transmisję komunikatów rozwiązaniami implementacyjnymi lub poprostu decyzjami administracyjnymi (decydujemy się na partiowe przetwarzanie danych źródłowych). Różne fragmenty systemu mogą w różny sposób wpływać na wielkość opóźnienia otrzymywanych wyników. Celem działań na etapie szeregowania procesów jest ustalenie jakie opóźnienia wyników są dopuszczalne z punktu widzenia użytkownika i podjęcie decyzji o różnicach czasowych pomiędzy różnych procesów systemu. Możemy zdecydować, że komunikacja procesu ze światem zewnętrznym odbywać się będzie poprzez dokumenty, które będziemy gromadzić i raz w miesiącu przetwarzać - otrzymamy wtedy system wsadowy. Możemy żądać bezpośredniej i natychmiastowej transmisji o zaistniałych zdarzeniach i natychmiast przetwarzać te dane - otrzymamy wtedy system pracujący w czasie rzeczywistym. Możliwe są rozwiązania pośrednie (system składa się z wielu procesów), zapewne koszt systemu będzie tutaj głównym czynnikiem decydującym o wyborze konkretnych rozwiązań. Wraz z zakończeniem tych prac zakończyliśmy specyfikację systemu. Zebrane materiały stanowią dokumentację systemu i powinny być zaakceptowane przez zamawiającego system.

Do zrealizowania pozostał nam już tylko jeden krok - krok implementacji systemu. Projektant musi decydować o wielu różnych rzeczach na tym etapie budowy systemu. JSD pozostawia do decyzji projektanta takie problemy jak wybór języków programowania, systemów zarządzania bazami danych, systemów zapytań czy telekomunikacyjnych. Narzędzi tych możemy używać w dowolny sposób. JSD koncentruje się na tych aspektach implementacji, które prowadzą od specyfikacji systemu do produktu, który możemy nazwać specyfikacją implementacji. Przyjrzyjmy się jeszcze raz temu co otrzymaliśmy w specyfikacji systemu. Jest to wielka mnogość drobnych procesów sekwencyjnych, każda inkarnacja encji posiada swój własny proces. Gdybyśmy chcieli w sposób bezpośredni implementować taką specyfikację, to musielibyśmy dla każdego z tych procesów przydzielić oddzielny processor i napisać oddzieloną procedurę z własnymi zmiennymi lokalnymi (wektor stanu). Zarządzalibyśmy setki tysięcy procesorów których głównym zadaniem byłoby oczekiwanie na strumień danych wejściowych. Będąc przy zdrowych zmysłach nikomu takiego rozwiązania nie zaproponujemy. Zatem oddzielmy teksty procesów od wektorów stanu, spróbujemy zmieścić je w najmniejszej liczbie procesorów (w naszych warunkach zazwyczaj w jednym). W tym momencie okazuje się, że potrzebne nam są następne procesy, których zadaniem jest administrowanie procesami systemu. Proces nadzorca odpowiedzialny będzie za zawieszanie i aktywowanie procesów nadzorowanych i za kontrolę komunikacji pomiędzy danymi. Dla wspomożenia i dalej udokumentowania tych prac służy nam SID (System Implementation Diagram, rysunek poniżej). Wektory stanów procesów jednostkowych utworzą nam wektory stanów sieci procesów i w konsekwencji systemu. Jednym z ważniejszych pomysłów w tej części prac jest inwersja programowa. Przy pomocy inwersji programowej dokonujemy syntezy systemu z drobnych procesów w całkowity produkt. Wielokrotne linie łączące procesy na SID oznaczają inwersje ze względu na odpowiednią ilość strumieni danych (w przypadku na rysunku mamy inwersje ze względu na dwa strumienie danych).



Powstaje pytanie : co ze zbiorami programami? Otóż procesy zapisane zostaną w postaci procedur lub programów. Wektory danych stanowić będą bazę danych systemu. Jak do tego dojść? Mamy do dyspozycji diagramy Jacksona procesów, które przez cały czas tworzyliśmy - wyznaczają one nam struktury danych. Z drugiej strony mniej więcej tekstowy, formalny zapis procesów (patrz rozdział JSP). Wygenerowanie zeń tekstów programów jest sprawą formalną i może być wykonane automatycznie.

4 PDF

PDF jest programem wspomagającym JSP. Zasadniczą część tego programu stanowi edytor diagramów Jacksona. Oprócz funkcji edytora wbudowano weń generator programów źródłowych, generator tekstów strukturalnych oraz wydruków graficznych przedstawień diagramów. Istnieją wersje PDF dla różnych języków (Befana posiada wersje dla Pascala i Cobolu). Program jest sparametryzowany: można łączyć go z różnymi edytorami tekstowymi, zadawać rozmiar strony wydruku, ustalać różne zestawy znaków służących do rysowania ramek, wymienić mapy klawiatury (ma to duże znaczenie ponieważ większość poleceń przekazywanych jest poprzez klawisze funkcyjne), można żądać aby tekst programu był komentowany przez nazwy poszczególnych węzłów, dla programów pascalowskich można żądać wcięć lub z nich rezygnować itp.. Edytor tekstowy służy do wprowadzania i edycji listy czynności oraz do edycji wyrażeń warunkowych. Program posiada bardzo bogate opcje HELP. Naciśnięcie klawisza F1 powoduje wyświetlenie obrazu klawiatury z nazwami funkcji przypisanymi poszczególnym klawiszom, wybranie funcji w tym trybie powoduje wyświetlenie opisu funkcji.

PDF możemy wywołać bez jakichkolwiek parametrów, wtedy program sam zapyta o nazwę zbioru i jeśli zbiór nie istnieje to otworzy nowy zbiór i wyświetli kratkę obrazującą korzeń drzewa nowej struktury, jeśli zbiór istnieje, to wyświetlony zostanie diagram. Pracując w PDFie cały czas (wyjątek stanowi edycja listy czynności i warunków) posługujemy się graficznym przedstawieniem diagramów. Kluczowymi pojęciami są pojęcia elementu bieżącego i kursora. Element bieżący jest to ustalony dla poszczególnych operacji element drzewa. Co to jest kursor każdy wie. Przy pomocy elementu bieżącego możemy poruszać się po drzewie, przy pomocy kursora możemy wskazać dowolne miejsce ekranu (znak, ponieważ program pracuje w trybie tekstowym). Dopuszczalne operacje to : utworzenie syna dla elementu bieżącego w miejscu w którym znajduje się kursor, utworzenie węzła pośredniego pomiędzy elementem bieżącym i poddrzewem, którego korzeniem jest ten element, kasowanie poddrzewa wyznaczonego przez element bieżący, kasowanie węzła pośredniego i liścia drzewa, kopiowanie poddrzew, nadawanie nazw poszczególnym elementom struktury, wpisywanie warunków, list czynności i prologu programu (lub procedury), oznaczanie selekcji, iteracji, nawrotów itd. Z technicznych udogodnień programu na uwagę zasługuje bardzo wygodna nawigacja w drzewie oraz bardzo sprytnie rozwiązana możliwość aranżacji wydruków struktur. Możemy podzielić całe drzewo na strony i zobaczyć jak poszczególne strony wypełniają się w ramach arkusza możemy polecić przesunięcie diagramu tak jak nam to odpowiada. Graficzna postać diagramów jest taka diagramy prezentowane w rozdziale JSP mają dokładnie taką postać jak wygenerowane przez PDF (za program sam ustala położenie poszczególnych kratek i odstępy między nimi, my wskazujemy tylko położenie poszczególnych elementów w stosunku do otoczenia. Wszystkie te operacje wykonujemy wybierając opcje klawiszami funkcyjnymi, których rozkład i znaczenie możemy w każdej chwili sprawdzić wybierając HELP (klawisz F1). Istnieje tylko niewielka ilość komend wydawanych poprzez nazwę. Należą do nich takie polecenia jak: generuj tekst programu (code), generuj tekst struktury (text), twórz wydruk diagramu (print), zapamiętaj (save), zakończ pracę zachowując zbiór (exit), zaniechaj pracę (quit), oraz polecenia dotyczące wyboru strony i rozmieszczenia diagramu na stronie. Muszę stwierdzić, że elegancja i ergonomia są wysoką stroną PDFa.

PDF ma możliwość obsługi nawrotów. Pomi : pointer oznaczane są pytajnikiem w lewym górnym rogu ramki a nawrót oznaczany jest wykrzyknikiem, nazwa elementu oznaczonego wykrzyknikiem musi być zgodna z nazwą elementu pointer (pointer) znaczonych elementów oznaczonych pytajnikiem). Generalnie program nie rozwiązuje problemów związanych z efektami niedopuszczalnymi i korzystnymi, można tylko oznaczać odpowiednie fragmenty. Również inwencja programowa nie posiada specjalnych mechanizmów obsługi. Ponieważ przy użyciu PDFa możemy generować zarówno programy jak i procedury problem ten nie ma większego znaczenia.

Innym ograniczeniem jest wielkość diagramu - dwadzieścia jeden poziomów okazuje się czasami niewystarczające dla programów cobolowych, w programach pascalowskich nie ma to znaczenia z uwagi na łatwość posługiwania się procedurami.

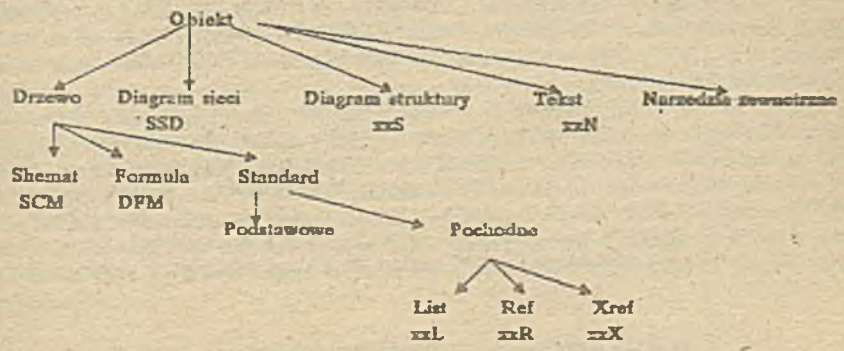
Generalnie stwierdzam, że mimo wysokiej ceny warto w program zainwestować. Procentuje to jakością programów oraz szybkością ich tworzenia.

5 SPEEDBUILDER

SPEEDBUILDER to pakiet oprogramowania typu CASE wspomagający metodę JSD wyprodukowany przez MICHAEL JACKSON SYSTEM Ltd. Wersja 3.0 zakupiona przez Befamę jest zrealizowana jako aplikacja GEM dla mikrokomputerów IBM. Głównym programem pakietu jest program SB3 (aplikacja GEM) pracujący w charakterystyczny dla GEMa sposób (tzn. okna obsługiwane jak w desktop, rozwijalne menu). Spełnia on następujące funkcje: wertowanie bazy danych systemu, włączanie różnych edytorów obiektów, kontrola zupełności i poprawności powiązań pomiędzy obiektami, przygotowanie raportów. Ponadto w skład pakietu wchodzą edytory obiektów zintegrowane z SB3, edytor SBF służący do przygotowania formatów wydruków oraz pakiet programów realizujących wydruki. SBF i programy realizujące wydruki są niezależne od SB3, są to programy DOSowskie. Pakiet wyposażony jest w dwa szkielety baz danych (wyjaśnię to później) jeden dla wersji podstawowej, drugi dla wersji rozszerzonej. Zbiory konfiguracyjne pozwalają na zmianę niektórych parametrów tak jak ma to miejsce w PDP. Wbudowane opcje HELP w istotny sposób wspomagają posługiwanie się pakietem. Do pakietu dołączony jest przykład specyfikacji prostego systemu obsługi biblioteki w dwóch wersjach podstawowej i rozszerzonej.

Specyfikacja budowanego przez nas systemu jest zapisywana w sieciowym modelu bazy danych. Można stwierdzić, że SB3 jest swojego rodzaju administratorem bazy danych. Jednak model konceptualny tej bazy został ustalony przez producenta i nie możemy go zmieniać (chyba, że kupimy dodatkowe oprogramowanie pozwalające na takie operacje, oprogramowanie to jest oferowane przez producenta). W ramach dostawy otrzymuje się przygotowane dwie bazy (dwa różne modele konceptualne). Pierwsza wersja uboższa odpowiada temu co prezentowałem w rozdziale trzecim. Druga wersja zawiera rozszerzenia, najważniejsze z nich to: wprowadzenia pojęcia roli dla encji, specyfikacji błędów oraz połączenia strumieni danych. W dalszym ciągu ograniczę się tylko do wersji podstawowej.

Wszystkie obiekty bazy podzielone są na klasy, podział ten wygląda następująco:



Obiekty SPEEDBUILERa są drzewami, z wieloma z nich związane związane są dodatkowe przedstawienia lub opisy. Tak więc obiekt podstawowy może posiadać diagram (znany nam już dobrze diagram Jacksona) lub słowny opis, który może być tak długi jak pozwala nam na to wybrany przez nas edytor tekstowy (parametr w zbiorze konfiguracji). Obiekty są ze sobą powiązane np. dla każdej encji istnieje lista akcji z nią związanych, możemy kontrolować kompletność połączeń, obiekty opisujące te zależności nazywane są obiektami pochodnymi. Poszczególne klasy obiektów rozróżniane są po rozszerzeniu. Na diagramie w dolnym wierszu wypisałem rozszerzenia odnośnych obiektów. Litery xx oznaczają, że dwie pierwsze litery rozszerzenia są zgodne z obiektem podstawowym i tak ENN oznaczać będzie, że obiekt jest opisem słownym dla encji a ENS jest diagramem Jacksona encji.

Kompletna lista klas obiektów wygląda następująco:

Obiekt	Rozszerzenie					
	Podstawowy	Opis	Struktura	List	Ref	Xref
Action	AC	ACN		ACL	ACR	ACX
Entity	EN	ENN	ENS	ENL		
Network	NE	NEN		NEL		
Process	PR	PRN	PRS	PRL	PRR	PRX
Datastream	DS			DSL	DSR	DSX
SVConnection	SV			SVL	SVR	SVX
Type	VA			VAL	VAR	VAX
Document	DO	DON		DOL		
Network Diagram	SSD					
Report	RPT					

Z każdym z tych obiektów związana jest lista jego atrybutów. Niektóre z nich występują we wszystkich klasach obiektów, do takich należą: klasyfikator służący do grupowania obiektów w zadany przez nas sposób (możemy użyć go np. do oznaczenia autora poszczególnych opisów), odniesienie - pełniący rolę skróconej nazwy w ... powiązań, krótki opis. Inne atrybuty są specyficzne dla danej klasy obiektów np. atrybutami encji są akcje oraz nieformalny opis dotyczący akcji, atrybutami akcji są zmienne wewnętrzne oraz typy tych zmiennych. Wszystkie ważne dla metody obiekty zostały omówione w rozdziale JSD myślę, że czytelnik bez trudu dostrzeże sposób ich używania. Document jest obiektem czysto opisowym i może być używany do sporządzania notatek. Type jest obiektem opisującym typ zmiennej, jest tam miejsce na opis słowny i jeśli kto chce na opis typu wyrażony w języku programowania. Przy pomocy obiektów klasy report opisujemy wydawnictwa, podajemy nazwę bazy zawierającej formuły raportów (obiekty DFM przechowywane są w oddzielnej bazie), opis strony na drukarce ... wiersz, szerokość, marginesy) oraz podajemy listę obiektów typu DFM z wyspecyfikowanej bazy. Raporty sporządzamy wywołując program nazwany SBOUTPUT z trzema parametrami : nazwą bazy formuł, nazwą obiektu report, i nazwą bazy zawierającej interesujące nas dane. SPEEDBUILDER nie posiada opcji wspomagających generację dokumentacji.

Interesującym nas sposobem definiowania formuł. Ustalamy klasę obiektów wiodących dla danej formuły, następnie możemy ustalić postać i zakres wyprowadzanych danych, możemy rozwijać informacje związane z obiektem ... obiektem bazowym jest encja, to ponieważ atrybutami encji są akcje możemy wydrukować potrzebne informacje dotyczące akcji oraz informacjami dotyczącymi encji itd. Możemy żądać wyprowadzenia informacji tylko ... dla wybranego obiektu klasy lub dla wszystkich obiektów klasy. Raporty w swej szacie graficznej mogą ... być urozmaicane, można włączać własne stałe, sterować wysuwem strony, umieszczać informacje w ram... Producent włącza standardową bibliotekę formuł P300, w której zawarte zostały definicje najpotrzeb... ... formuł.

6 BEFAMOWSKIE DOŚWIADCZENIA Z CASE

Oprogramowanie metod tworzenia aplikacji znalazło się w Befamie nieprzypadkowo. Od wielu lat poszukiwano narzędzi wspomagających prace projektowe, programowe oraz dokumentacyjne. Najbardziej drażniącym był problem dokumentacji. Wielokrotnie zmieniał się zespół pracowników Ośrodka Informatyki, pozostawały efekty pracy w postaci systemów i programów. Ponieważ systemy podlegają ciągłym modyfikacjom problem dokładnej i pełnej dokumentacji był jednym z większych problemów aktualnie pracującego zespołu. Znane z literatury metody były zbyt pracochłonne i nie zapewniały stałej aktualności dokumentacji. Dość szybko zorientowaliśmy się, że żadna metoda dokumentowania nie będzie zaspakajać naszych wymagań jeśli dokumentacja nie będzie tylko produktem ubocznym procesu tworzenia aplikacji.

Od tego czasu przestaliśmy traktować problem dokumentacji jako samoistny a zaczeliśmy interesować się metodami inżynierii oprogramowania. W toku tych poszukiwań natknęliśmy się na informacje o pracach Jacksona (JSP). Tym sposobem rozwiązaliśmy problem precyzyjnego opisu struktur danych, formułowania założeń programowych oraz unormowaliśmy sposób pisania programów. Programy przestały nosić piętno autora, stały się łatwo czytelne dla innych programistów i projektantów. Mimo, że przestaliśmy mówić o optymalizacji programy stały się znacznie szybsze i mniejsze. Ten etap łatwo było przejść ponieważ dotyczył wąskiej grupy ludzi oraz zakresu blisko związanego z językiem programowania. W dalszym ciągu brakowało dobrej metody wspomagającej fazę analizy systemu oraz wczesnych etapów projektowania. W 1987 roku na Jesiennej Szkole Informatyki dowiedziałem się od profesora Turskiego, że Jackson jest właścicielem firmy tworzącej narzędzia wspomagające metodę JSP, dowiedziałem się również o JSD. Z wypożyczonych mi przez prof. Turskiego książek zapoznałem się z prezentowanymi tutaj metodami. Prawie dwa lata trwały starania o zakup produktów Jacksona dopiero w jesieni ubiegłego roku otrzymaliśmy odnośne oprogramowanie. Dostawa ta niemal zbiegła się w czasie z dostawą dużego komputera. W firmie ICL kupiliśmy komputer serii 39 wraz z potężnym oprogramowaniem w tym oprogramowaniem CASE wspomagającym metodę ISDM. "Nadmiar szczęścia" sprawił, że JSD nie jest wdrażany w stopniu na jaki zasługuje. Obecność dwóch metod pozwala wyciągnąć wnioski bardziej ogólne oraz dokonać porównania metod. Porównaniem nie będę się tutaj zajmował, chciałbym natomiast przedstawić zalety stosowania oraz trudności okresu niemowlęcego (mam nadzieję, że tylko niemowlęcego) sformalizowanego sposobu prowadzenia prac informatycznych.

Na początek parę zdań o PDF. Jak już wspomniałem w Befamie trafił on na przygotowany grunt. Łatwość posługiwania się nim zachęca do jego używania, więc z tym nie ma kłopotu. Nowym efektem jego stosowania jest możliwość lepszego wykorzystania kadr. Ponieważ pracę nad programem można w każdej chwili przerwać i przekazać następnemu pracownikowi, w każdym programie pozostaje zakres czynności, który może wykonać mniej wykwalifikowana osoba. Dotychczas doświadczeni programiści byli nadmiernie obciążani, ludzie potrafiący mniej nie byli "dociążani". Teraz można przekazać im takie czynności jak zapisywanie list czynności bez obawy, że zepsują poważny program. Idea inwersji programowej jest naturalnym źródłem modułów, które mogą być pisane równolegle, powoduje to, że nawet bardzo duży program może być zaprojektowany i napisany w ciągu kilku dni. Bardzo niewiele czasu zużywa się na testowanie, programy są raczej dobre, ponadto stosując diagram struktury danych nietrudno przygotować dane do testowania. Po wstępnych doświadczeniach uważam, że PDF nie ma wad. Kłopoty jakie mamy wywodzą się z faktu, że program jest zabezpieczony i każdorazowe jego wczytanie wymaga dyskietki kluczowej. Na razie mamy tylko po jednej dyskietce dla Pascala i Cobolu.

Trudniejsza jest sprawa z metodami wspomagającymi analizę i projektowanie systemu. Informatycy na ogół wciąż nie wierzą w uniwersalność tych metod (łatwiej jest z młodymi projektantami), trudno im się pogodzić z myślą, że ich twórcze procesy myślowe mogą być ujęte w karby formalizacji. Ponadto wszechwładnie panuje przekonanie o konieczności rozpoczynania analizy systemu od specyfikacji funkcji. Metody bazujące na modelowaniu świata rzeczywistego stanowią inaczej. Stąd rodzą się kłopoty i trudności w przyjmowaniu się tych metod. Szczególnie sprawa wydaje mi się beznadziejna dopóki metoda nie jest wspomagana narzędziami. Chęć posiadania dobrej dokumentacji bez potrzeby spędzania wielu godzin nad jej tworzeniem i uaktualnianiem jest pokusą tak dużą, że ona stanie się głównym motorem napędowym stosowania CASE. Uważam, że SPEEDBUILER dobrze będzie służyć budowaniu oprogramowania aplikacyjnego chociaż posiadana przez nas wersja ma sporo usterek wynikłych głównie z faktu, że jest to wersja po gruntownych przeróbkach (pierwsza wersja pracującaw OEMie).

Bibilografia:

M. A. Jackson; Principles of Program Design; Academik Press , London, New York , San Francisco 1975;
M. A. Jackson; System Development; Prentice-Hall International, Inc, 1983.
PDF dokumentacja systemu.
SPEEDBUILDER for IBM PC/Compatibles, USER GUIDE, Michael Jackson Systems Ltd.

PRZECHOWYWANIE I PRZETWARZANIE DANYCH O ZŁOŻONYCH, NIEREGULARNYCH STRUKTURACH.

Krzysztof Dworakowski, Henryk Klimek, Tadeusz Korniak,
Elżbieta Przepióra, Adam Zgagacz

1. Wprowadzenie

Niniejsze opracowanie opisuje podstawowe cechy oprogramowania systemu zarządzania hierarchiczną bazą danych GRBASE, przeznaczonego do obsługi danych o nieregularnych i nie dających się z góry określić strukturach.

W opracowaniu przedstawiono zarys struktury danych bazy GRBASE, możliwości funkcjonalne systemu zarządzania, a także informacje o zrealizowanej przykładowej aplikacji.

Tradycyjne bazy danych nie uwzględniają możliwości łatwego tworzenia dynamicznie zmieniających się struktur danych. Aktualnie podejmowane są próby zaadaptowania istniejących baz danych do zarządzania tego typu obiektami. Wiąże się to z niską efektywnością ze względu na specyficzne własności przechowywania danych.

SZBD GRBASE zapewnia, w odpowiedzi na istniejące zapotrzebowanie, efektywność zarządzania danymi o dynamicznych strukturach drzewiastych.

Przechodzimy teraz do ogólnej charakterystyki funkcjonalnej SZBD GRBASE omawiając także krótko strukturę oprogramowania.

Baza DANYCH GRBASE służy do przechowywania i przetwarzania danych o nieregularnej strukturze i powiązaniach hierarchicznych między nimi. System zarządzania bazą danych GRBASE umożliwia obsługę danych użytkownika w zakresie kreowania, modyfikacji, wyszukiwania i przetwarzania obiektów Bazy. Przewidziano również kontrolę dostępu do obiektów.

Oprogramowanie jest przeznaczone dla komputerów zgodnych z IBM-PC/XT oraz IBM-PC/AT i systemu operacyjnego MS-DOS (dla wiersji wielodostępnej również dla systemu XENIX).

Wykorzystując oprogramowanie SZBD GRBASE, użytkownik (programista aplikacyjny) będzie mógł łączyć dane w określone przez siebie grupy (segmentacja danych). Pomiędzy tak utworzonymi grupami danych (segmentami) może być dowolnie ustalana hierarchiczna struktura drzewiasta. Każdy zespół segmentów, połączonych dowolnie w drzewo, stanowi obiekt Bazy Danych.

Korzystając z oferowanych środków programowych programista aplikacyjny może realizować, między innymi, następujące operacje:

- definiowanie i modyfikacja obiektów, zarówno w zakresie struktury, jak i wchodzących w ich skład segmentów;
- definiowanie zestawów atrybutów dla obiektów oraz ich modyfikacja (dopisywanie, usuwanie, aktualizacja);
- wyszukiwanie obiektów lub ich grup, spełniających określone przez siebie warunki.

Oprogramowanie GRBASE stanowi więc zestaw narzędzi pozwalających na tworzenie systemów aplikacyjnych, w których istotną rolę odgrywają złożone i zmieniające się w szerokim zakresie, powiązania między danymi.

2. Dane i ich struktury

Baza danych GRBASE służy do przechowywania i przetwarzania obiektów o strukturze drzewa. Należy rozumieć, iż informacja użytkownika (dane użytkowe) jest skupiona w węzłach drzewa, a powiązania między węzłami ustalają hierarchię między danymi. Obiekty bazy danych GRBASE mają specyficzną budowę, którą teraz, rozpoczynając od określenia podstawowych pojęć, przedstawimy.

Dana elementarna

Dana elementarna jest najmniejszą jednostką danych, udostępnianą użytkownikowi środkami systemowymi. W systemie zaimplementowano między innymi dane typu **INTEGER, REAL, TEXT, DATE, CHAR.**

Pole danych

Polem nazywamy dowolną sekwencję (być może o krotności 0) danych elementarnych tego samego typu. Poszczególne dane elementarnewchodzące w skład pola nazywamy jego wystąpieniami.

Segment

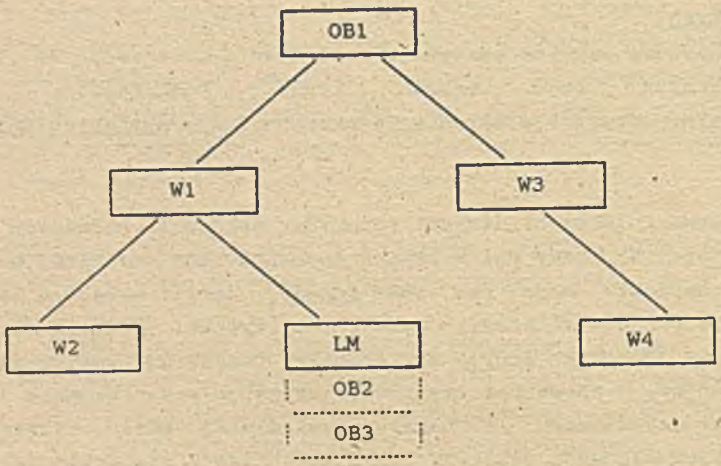
Przez segment (segment danych) rozumiemy sekwencję nazwanych pól. Zakłada się, że nazwy pól w ramach segmentu będą unikalne. Mówimy, że dwa segmenty mają ten sam format, jeśli składają się z sekwencji pól o tych samych nazwach i typach. Segment stanowi podstawową jednostkę danych użytkownika. Dostęp do poszczególnych wystąpień pól w segmencie użytkownik uzyskuje przez zadanie nazwy pola (lub jego numeru) i numeru wystąpienia pola. Szczególnym przypadkiem segmentu jest tzw. prymityw będący segmentem danych o wyróżnionym formacie: Postać segmentu danych ilustruje rys. 1.

POLE1	POLE2	POLE3	POLE4	POLE5
IDENT		122	122.5	TX1
		-2374	3.2E-5	TX2
			0.0	TX3
				TX4

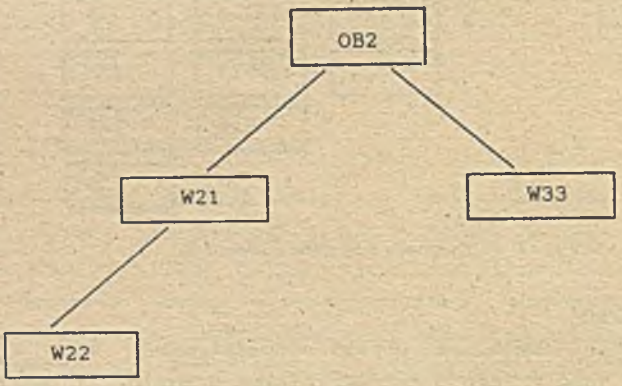
Rys. 1 Przykład segmentu danych.

Obiekt Bazy Danych

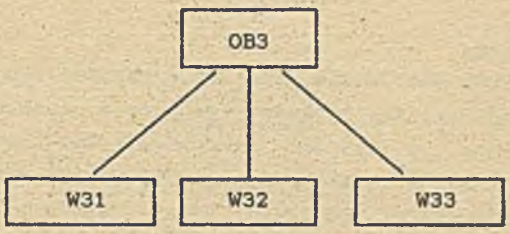
Z formalnego punktu widzenia obiekt projektowanej bazy danych należy rozumieć jako drzewo z rozróżnialnymi przez nazwy węzłami. Należy w tym miejscu podkreślić, że poszczególne obiekty bazy, traktowane jako drzewa, nie muszą być izomorficzne (w sensie izomorfizmu grafów). Przykłady obiektów bazy danych są podane na rysunkach 2, 3 i 4 - występujące tam pojęcie makroprymitywu jest opisane w dalszej części tekstu..



Rys. 2 Przykład obiektu z liściem makroprymitywów



Rys. 3 Przykład obiektu bazy danych



Rys. 4 Przykład obiektu bazy danych

Obiekt rozważanej bazy danych został powyżej określony jako drzewiasta struktura danych użytkownika. Z logicznego (użytkowego) punktu widzenia wśród węzłów obiektu wyróżniamy następujące:

- korzeń;
- węzeł prymitywów;
- liść makroprymitywów;
- gałązka;

Dalej podajemy interpretację użytkową poszczególnych rodzajów węzłów.

Korzeń

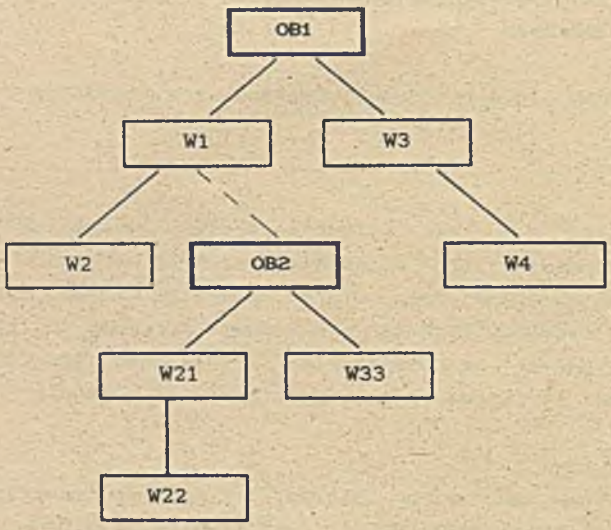
Jest rozumiany w sensie korzenia grafu. Informacją użytkową skupioną w korzeniu jest pojedyńczy segment danych. Korzeniom różnych obiektów odpowiadają różne segmenty danych, przy czym wszystkie te segmenty mają ten sam format. Korzenie (a tym samym obiekty Bazy Danych) są rozróżnialne - unikalnym identyfikatorem jest zawartość pierwszego pola segmentu danych korzenia.

Węzeł prymitywów

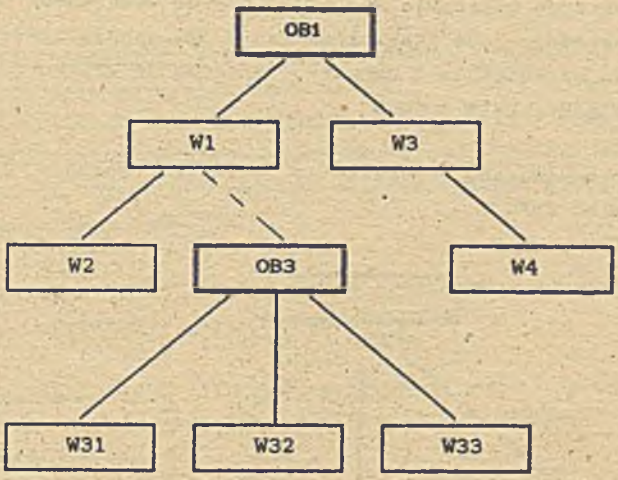
Węzeł prymitywów jest węzłem obiektu, z którym związana jest lista prymitywów (kolekcja prymitywów). Poszczególne prymitywy kolekcji mogą znajdować się w różnych Plikach Prymitywów. Ten sam prymityw może występować w różnych kolekcjach obiektu, jak również w kolekcjach róznych obiektów. W danej chwili tylko jeden prymityw z kolekcji (wybrany przez użytkownika) może być dostępny do przetwarzania.

Liść makroprymitywów

Liść makroprymitywów jest węzłem obiektu o szczególnym znaczeniu i nie jest z nim bezpośrednio związana informacja użytkowa (segment lub prymityw danych). Z liściem makroprymitywów związana jest lista obiektów (kolekcja makroprymitywów). Użytkownik wskazuje, który z makroprymitywów kolekcji jest aktualnie widoczny (dołączony do obiektu). W obiektach wykorzystywanych jako makroprymitywy również mogą występować liście makroprymitywów, co pozwala na "wielopiętrową" konstrukcję obiektów bazy. Liść makroprymitywów jest zawsze węzłem wiszącym obiektu. Rys. 5 ilustruje obiekt OB1 z rys 2, w sytuacji, gdy aktualnie widocznym makroprymitywem jest obiekt OB2 z rys.3. Rys 6 pokazuje ten sam obiekt z aktualnie widocznym makroprymitywem OB3 z rys. 4.



Rys. 5 Obiekt OB1 z aktywnym makroprymitywem OB2



Rys. 6 Obiekt OB1 z aktywnym makroprymitywem OB3

Gałązka

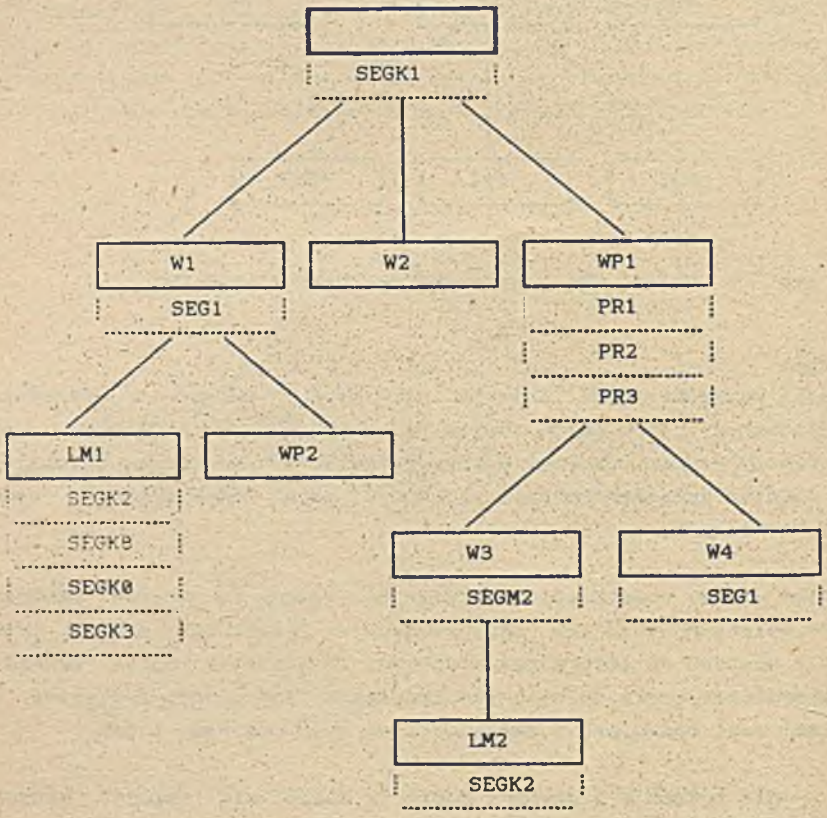
Gałązką nazywamy węzeł obiektu nie będący żadnym z uprzednio omówionych. Informacją użytkową skupioną w gałązce jest pojedyńczy segment danych. Różnym gałązkom w tym samym obiekcie, jak również gałązkom różnych obiektów, mogą odpowiadać te same segmenty danych.

Różnica między poszczególnymi rodzajami węzłów w grafie zasadza się w opisanym wyżej ich usytuowaniu w strukturze drzewa oraz trybie dostępu do informacji użytkowej związanej z danym węzłem. Udostępnienie węzła obiektu (udostępnienie informacji związanej z węzłem) jest rozumiane w zależności od jego rodzaju, i tak:

- dla korzenia i gałązki dostępny staje się segment danych przypisany do węzła;

- dla węzła prymitywów i liścia makroprymitywów dostępnymi stają się operacje umożliwiające przejście do wskazanego elementu kolekcji i operacje jego przetwarzania. W tym sensie powiemy, że - w odróżnieniu od korzenia i gałązki - informacja użytkowa nie jest bezpośrednio skupiona w liściu.

Przykład rozbudowanego obiektu, z różnego rodzaju węzłami, znajduje się na rys. 7.



Rys. 7 Przykład rozbudowanego obiektu bazy danych

W przykładzie obiektu z rys. 7 węzły (nazwy węzłów w pogrubionych ramkach) mają następujące znaczenie:

węzeł oznaczony pustą ramką jest korzeniem obiektu z przypisanym segmentem danych "SEGK1", ustalonego jednolicie dla wszystkich obiektów bazyformatu. Nazwa SEGK1 stanowi identyfikator obiektu. Pusta ramka na rysunku odzwierciedla fakt, iż korzeniowi każdego obiektu system przypisuje nazwę złożoną ze znaków NULL;

węzeł W1 jest gałązką z przypisanym segmentem "SEG1" przechowywanym w pliku segmentów ;

węzeł LM1 jest liściem makroprymitywów. Kolekcję tworzą obiekty o identyfikatorach "SEGK2", "SEGK8", "SEGK0" i "SEGK3";

węzeł WP2 jest węzłem prymitywów. Aktualnie kolekcja prymitywów jest pusta;

węzeł W2 jest "pustą" gałązką, tzn. gałązką do której aktualnie nie jest przypisany żaden segment danych;

węzeł WP1 jest węzłem prymitywów. W kolekcji znajdują się aktualnie prymitywy "PR1", "PR2" i "PR3";

węzeł W3 jest gałązką z dołączonym segmentem "SEGM2" przechowywanym w pliku danych;

węzeł LM2 jest liściem makroprymitywów. Kolekcja składa się z pojedyńczego elementu (obiekt o identyfikatorze SEGK2);

węzeł W4 jest gałązką z dołączonym segmentem "SEG1".

Zauważmy, że segment danych "SEG1" został dołączony do dwóch węzłów W1 i W4. Nie oznacza to oczywiście jego fizycznego zdublowania w bazie.

Przewiduje się, że część informacji związanej z obiektem będzie miała charakter dynamiczny, tzn. będzie zależna nie tylko od samego obiektu, lecz również od sytuacji w jakiej ten obiekt jest udostępniany użytkownikowi. Na informację, o której mowa powyżej, składa się zestaw danych elementarnych (parametrów), z których każda może być powtarzalna i które ładowane są do ustalonych pól korzenia w momencie gdy obiekt staje się widoczny jako makroprymityw. Każdemu wystąpieniu obiektu w kolekcji makroprymitywów jest przypisany indywidualny zestaw parametrów. Poszczególne zestawy parametrów mogą się oczywiście różnić między sobą.
Parametry pozwalają na zróżnicowane widzenie tego samego obiektu, co - mając zwłaszcza na uwadze konieczność pogodzenia potrzeb różnych użytkowników - uelastycznia możliwości przetwarzania bazy danych.

3. Usługi realizowane przez SZBD GRBASE

Dalej przedstawiono wykaz podstawowych usług oferowanych przez SZBD GRBASE, wraz z ich krótką charakterystyką, co pozwoli na zorientowanie się w możliwościach systemu.

System spełnia, za pośrednictwem oferowanych usług, następujące, podstawowe funkcje obsługi bazy:

- kreatora bazy danych;
- kreatora obiektów bazy;
- organizatora dostępu do obiektów bazy i ich części składowych;
- rejestratora wykonywanych operacji;
- organizatora przeszukiwania i selekcjonowania zbioru obiektów bazy.

System oferuje następujące grupy usług (operacji):

- operacje organizujące bazę danych.
  Do operacji tych należy m.in kreowanie bazy, definiowanie formatów segmentów oraz rejestracja użytkowników. Operacje tej grupy są realizowane przez samodzielne programy usługowe.

- operacje organizacyjne procesu przetwarzania.
  Są to głównie operacje rozpoczynające lub kończące przetwarzanie bazy. Operacje tej i pozostałych grup są realizowane przez procedury Biblioteki Procedur GRBASE.

- operacje na obiektach i węzłach.
  Operacje tej grupy pozwalają na kreowanie, modyfikację struktury oraz usuwanie obiektów bazy danych.

- operacje dotyczące liści i węzłów prymitywów.
  Jest to specjalna grupa operacji umożliwiająca dostęp do elementów kolekcji związanej z liściem makroprymitywów lub węzłem prymitywów i ich przetwarzanie.

- porządkowanie zbioru obiektów.
  Operacje tej grupy ustalają aktualnie obowiązujący porządek (przez wskazanie aktywnego klucza) w zbiorze obiektów, co umożliwia zorganizowane poruszanie się po tym zbiorze.

- wyszukiwanie obiektów.
  Są to operacje udostępniające użytkownikowi obiekt spełniający warunki zadane w postaci związków między polami korzenia lub segmentów węzła.

- przesłanianie obiektów.
  Operacje te pozwalają na logiczne wydzielenie z całej bazy podzbioru obiektów spełniających zadany przez użytkownika warunek. Pozostałe obiekty stają się wtedy "niewidoczne" i nie biorą udziału w przetwarzaniu.

- poruszanie się po obiekcie.
  Są to operacje pozwalające na poruszanie się (we wskazany przez użytkownika sposób) po węzłach aktualnie dostępnego obiektu. Węzeł docelowy zostaje udostępniony użytkownikowi

- operacje na danych węzła.
  Ta grupa operacji pozwala czytać, zapisywać i modyfikować pola segmentu związanego z aktualnie dostępnym węzłem. Możliwa jest również wymiana całego segmentu.

- operacje informacyjne.
  Funkcje tej grupy podają charakterystyki liczbowe związane ze zbiorem obiektów lub pojedynczym obiektem oraz realizują wybrane operacje arytmetyczne na polach wskazanych segmentów.

Zestaw operacji został dobrany tak, aby umożliwić użytkownikowi sprawną obsługę i przetwarzanie hierarhicznej bazy danych.
Niezależnie od podstawowego trybu wykorzystywania usług (operacji) SZBD GRBASE za jaki należy uważać stosowanie w programach aplikacyjnych procedur z Biblioteki Procedur GRBASE - możliwy jest równoległe, konwersacyjny tryb pracy z bazą.
Program Interpretatora Zleceń GRBASE pozwala użytkownikowi na interakcyjne wywoływanie poszczególnych operacji (w sposób zbliżony do stosowanego w relacyjnej bazie danych dBase III Plus). Opcja ta pozwoli użytkownikowi nie będącemu programistą na wykonywanie praktycznie wszystkich operacji z Biblioteki Procedur. Przy tym trybie wykorzystania realizowanego oprogramowania stosowany jest zestaw środków wspomagających pracę użytkownika (kontekstowe objaśnienia, pełnoekranowa wizualizacja danych itp.).

Ponadto, w rozszerzonej wersji systemu GRBASE, zrealizowano dwa następujące mechanizmy:

- mechanizm wielodostępu;

- mechanizm ochrony danych

Mechanizm wielodostępu umożliwia równoległe przetwarzanie tej samej bazy danych przez wielu użytkowników. System zapewnia bezpieczny dostęp do obiektów bazy i danych (segmentów, prymitywów), unikając jednocześnie nadmiernego blokowania tych elementów.

Mechanizm ochrony danych pozwala traktować selektywnie uprawnienia różnych użytkowników do obiektów bazy danych. Dla każdego z użytkowników bazy jest określony podzbiór jej obiektów do którego ten użytkownik ma dostęp.

4. Przykładowa aplikacja systemu GRBASE

Jeżeli będziemy dokładniej analizować strukturę danych opisujących rzeczywiste obiekty, zobaczymy, że w gruncie rzeczy układają się one w nieregularne, drzewiaste struktury. Aby to zbadać stworzono bardzo uproszczony model systemu informatyzacji przedsiębiorstwa - hipotetycznego Wydziału Klocków Fabryki Zabawek. Wyróżniono typowe zakresy stosowania systemów informatycznych - w tym dwa główne, to:

- wspomaganie projektowanie produktu złożonego z szeregu elementów (przez analogię do TPP),

- analiza wyników finansowych (analogia systemów tworzonych dla działu planowania)

Wspomaganie projektowania

Produktem Wydziału są Zestawy Klocków. Każdy Zestaw składa się ze ściśle określonej liczby klocków.

Model Główny

Podstawową cechą Zestawu jest, że można z niego zbudować ściśle określony model tzw. Model Główny, charakterystyczny dla tego Zestawu. Rysunek tego modelu widnieje na opakowaniu i w katalogu załączanym do Zestawu. Jest rzeczą dopuszczalną, aby po złożeniu Modelu Głównego została pewna, niewielka liczba klocków luzem.

Zestawienie materiałowe

Ostatnia strona katalogu zawiera rysunek wszystkich klocków wchodzących w skład Zestawu (podana jest także ich liczba). Jest to więc swoiste zestawienie materiałowe zestawu.

Sposób projektowania nowego Zestawu

Projekt nowego Zestawu stworzony przez Projektanta obejmuje między innymi:

- zestawienie materiałowe tj. liczbę i rodzaj wszystkich klocków wchodzących w skład Zestawu (Model Główny może wymagać mniej klocków, niż podano w Zestawie, aby można z niego było budować atrakcyjne inne modele),

- wstępna kalkulacja kosztów Zestawu oparta na średnich cenach klocków wchodzących w jego skład.

Analiza wyników finansowych

Jedynymi wejściowymi do tej części systemu są faktury wystawiane klientom na zamawiane partie zestawów. Dla uproszczenia pominięto cały dział planowania produkcji w oparciu o zamówienia, jest on analogiczny do części rozliczeniowej.

Głównym celem tej części systemu jest:

- wystawianie Klientom faktur na podstawie informacji o wysłanych partiach dostarczanych przez Dział Kompletacji i Magazynów,
- analiza opłacalności sprzedaży poszczególnych Zestawów,
- zmiana kalkulacji cen poszczególnych Zestawów pod wpływem zmiany ceny jednego z jego elementów.

Dane opisowe zawarto w segmentach o następującej strukturze:

Faktura (Sprzedaż):

KOD_SPRZ	DATA_SPRZ	KOD_ZEST	LICZ_ZEST	CENA_UMOWN	ZYSK
C10	D8	C10	N5	N6.2	N6.2

Zestaw:

KOD_ZEST	NAZWA_ZEST	DATA_ZATW	KOSZT_ZEST
C10	C20	D8	N6.2

Klocek:

KOD_KLOC	NAZWA_KLOC	WYMIARY_KL	KOSZT_KLOC	DATA_ZATW
C10	C20	N3	N6.2	D8

Pracownik:

KOD_PRAC	NAZWISKO	IMIE	DATA_URODZ	DATA_ZATRU	PREMIA
C10	C20	C10	D8	D8	N6.2

Klient:

KOD_KLIE	NAZWA_KLIE	MIASTO	ADRES
C10	C40	C40	C40

Parametry:

KLUCZ	LICZBA
C10	i*N4

Obiekty danych

Obiekty bazy odzwierciedlają strukturę typowych zapytań systemu. Identyfikatorem obiektu . jest dziesięcioznakowy klucz rozpoczynający się dwuliterowym przedrostkiem ll_.

Pierwsza litera definiuje obiekt:

- Z - Zestaw
- E - Klocek
- K - Klient
- S - Faktura (Sprzedaż)
- P - Pracownik

Druga litera określa rodzaj węzła:

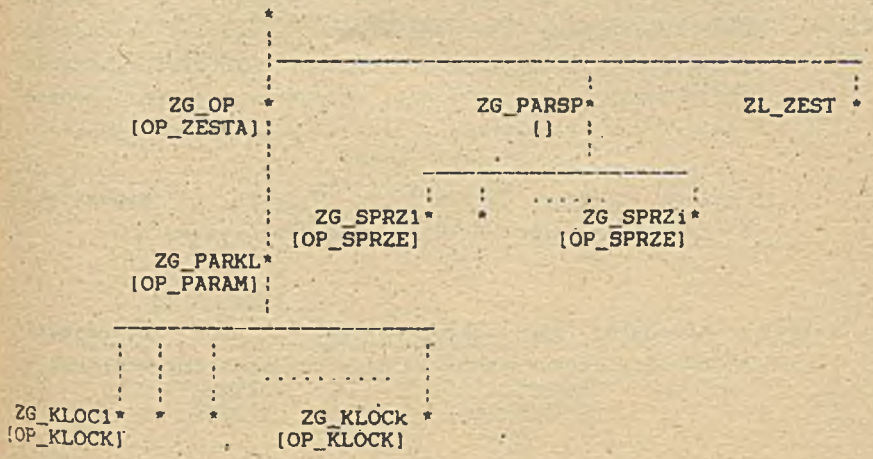
- G - gałązka.
- L - liść makroprymitywów

Druga część nazwy zazwyczaj związana jest z typem segmentu związanego z węzłem (jeśli jest), lub funkcją węzła, np. ZG_KLOC1. Każdy z segmentów (za wyjątkiem segmentu parametrów, niejako sztucznym) rozpoczyna się polem KOD_... . Pole to zawiera powtórzony unikalny klucz obiektu nadawany kolejno sekwencyjnie przez system.

Celem tego rozwiązania jest, aby dołączone do obiektów-nie-właścicieli segmenty mogły być łatwo wstępnie uporządkowane wg. narastających kodów. Pozwoli to na szybkie przeszukiwanie kolekcji liści lub szeregu węzłów-dzieci, nie tworząc skomlikowanych zbyt algorytmów.

Konsekwencją tej metody jest, że kod raz zarejestrowanego w systemie obiektu jest nieusuwalny, tzn. żaden inny obiekt go nie dostanie. Dziewięć znaków kodu wystarczy całkowicie do takich celów. Pierwsza litera kodu jest literą definiującą obiekt.

Obiekt: Zestaw



Obiekt składa się z trzech gałęzi:

gałęzi opisu technicznego, na którą składa się:

- węzeł-gałązka ZG_OP (segment OP_ZESTA), opisujący dane stałe zestawu,

węzeł-gałązka ZG_PARKL (segment OP_PARAM) zastępujący niezaimplementowane w tej wersji GRBASE parametry w ten sposób, że i-te wystąpienie pola LICZBA segmentu OP_PARAM określających liczbę klocków opisywanych przez i-te dziecko,

n-węzłów (liści) ZG_KLOCK1 - ZG_KLOCKn z segmentami OP_KLOCK opisującymi klocki, z których składa się Zestaw.

gałęzi opisu sprzedaży, na którą składa się:

- węzeł-gałązka ZG_PARSP, nie związany z żadnym segmentem. Służy on wydzieleniu części opisującej sprzedaż, dla urody oraz aby np. aby wykorzystać komendę PODAJ_LICZBE_DZIECI. Nie jest on potrzebny, jeśli będzie przeszkadzał, lub opóźniał działanie, można go usunąć,

  n-węzłów (liści) ZG_SPRZ1 - ZG_SPRZn z segmentami OP_SPRZE opisującymi kolejne sprzedaże Zestaw.

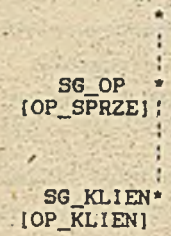
gałęzi zawierającej liść makroprymitywu ZL_ZEST zwierającej kolekcję makroprymitywów-zestawów będących modelami wtórnymi.

Obiekt: Klocki

EG_OP
[OP_KLICK]

Obiekt składa się z węzła-gałązki EG_OP (segment OP_SPRZE) opisujący dane stałe klocka. Obiekt wprowadzony aby w łatwy sposób obsługiwać zmian w segmentach opisujących klocek.

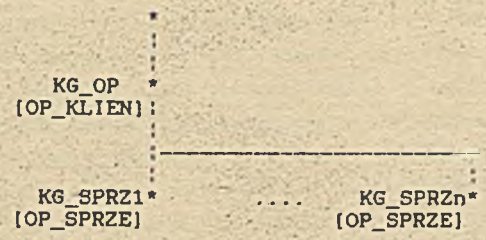
Obiekt: Faktura (Sprzedaż)



Obiekt składa się z węzłów:

- gałązki SG_OP (segment OP_SPRZE) opisujący dane stałe zestawu,
- węzeł-liść SG_KLIEN opisujący Klienta, którego dotyczy sprzedaż

Obiekt: Klient



Obiekt składa się z węzłów:

- gałązki KG_OP (segment OP_SPRZE) opisujący dane stałe zestawu,
- liści KG_SPRZi opisujących kolejne zakupy Klienta,

Przykładowa aplikacja, aczkolwiek jest bardzo dużym uproszczeniem pozwala domniemywać, że w rzeczywistych systemach przetwarzania wydają się występować struktury drzewiaste, rzeczywiste dane z dużym trudem mieszczą się bowiem w relacyjnym modelu danych.

Krzysztof Gajewski
Uniwersytet Szczeciński.
Instytut cybernetyki ekonomicznej i informatyki

INŻYNIERIA INFORMACJI JAMESA MARTINA

1. Czym jest inżynieria informacji ?

James Martin Association (JMA) jest organizacją, która projektuje i rozwija przodujące technologie na polu metodologii tworzenia systemów . Technnologia ta jest ukierunkowana na rozwiązywanie tradycyjnych problemów związanych z rozwijaniem systemów informacyjnych, którymi są :

- nadmierny czas projektowania,
- nieefektywne wspomaganie czynności dotyczących działalności instytucji,
- mała podatność na zmiany,
- duży poziom zręczności wymagany od osób zajmujących się projektowaniem,
- trudna konserwacja.

Metodologia JMA została zbudowana przez rozpatrzenie analogicznych światów dyscyplin inżynierskich, w których główne projekty są podejmowane i rozwiązywane przez małe grupy specjalistów i użytkowników i zapewnia się obrazową prezentację, tak aby każdy zainteresowany, z podstawowymi wiadomościami mógł wziąść udział w technikach konstrukcyjnych. Aby być adekwatnym tych celów metodologia ta została nazwana "inżynierią informacji".

Podstawowym zyskiem dla przedsiębiorstwa korzystającego z inżynierii informacji jest to, że ukazuje główny problem jako zaopatrzenie w informację użytkowników, a nie konstruowanie oprogramowania. Dla techników oznacza to podjęcie metod konstruowania systemów informatycznych w konwencji czwartej generacji. Dla użytkownika oznacza to natomiast, że ludzie przetwarzający dane będą formułować problemy w sposób, który jest zrozumiały i może być sprawdzony z perspektywy użytkownika.

2. Czwarta generacja.

Olbrzymie możliwości komputerów, zarówno w mocy przetwarzania jak i w pamięci zmieniają koszty i styl w

środowisku systemów informacyjnych. Koszty technologii maleją bardzo szybko we wszystkich kluczowych dziedzinach związanych z produkcją sprzętu komputerowego. Z drugiej strony ludzie i papier stają się relatywnie coraz droższi (u nas niestety tylko papier, ale w dalszym ciągu pozostaje cierpliwy). Kluczem do utrzymania równowagi pomiędzy wymienionymi wyżej grupami kosztów jest eliminować używanie papieru w instytucjach i czynić ludzi bardziej efektywnymi. Jest to możliwe dzięki czwartej generacji konstruowania systemów informacyjnych. Stylem czwartej generacji jest organizować dane w ten sposób, aby informacja była dostępna i wspólna w całym przedsiębiorstwie. Kładzie się nacisk na potrzebę istnienia systemów, które są modelem przedsiębiorstwa i wspomagają wszystkie jego komórki. Koncentrują się one najpierw na zrozumieniu działalności instytucji, przez analizę nienaruszalnych procesów i typów informacji, przy pomocy których te procesy powstają. Rozwijając się, użytkownicy mogą budować swoje własne systemy, co da z pewnością rezultaty w postaci znacznego wzrostu wykorzystania komputerów w przedsiębiorstwach. Ogromna produkcja mikroprocesorów ,może być sprzedana jedynie wtedy jeżeli mogą być one uruchomione bez profesjonalnych programistów.

Podstawowe cechy systemów czwartej generacji to:

* wspólnota danych (data sharing) - możliwość zespołowego wykorzystywania wspólnych danych, modelowanych w architekturze informacji.
* system użytkownika - systemy, modelują działalność instytucji w sposób prawdziwy i nie krępują użytkownika.
* dystrybucja - rozmieszczenie danych i możliwości ich przetwarzania tam gdzie są one potrzebne.
* interakcyjność - możliwość swobodnego komunikowania się ludzi i maszyn w całym przedsiębiorstwie.
* automatyzacja - automatyzacja projektowania i konstruowania systemów poprzez prototypowanie i generowanie kodu.
* konstrukcje użytkownika - użytkownicy sami budują swoje systemy.
* zbieżność technologi - systemy są zintegrowane pod kątem struktury technicznej; wszystkie technologie oparte na elektronice i digitalizacji są zbieżne.

Aby "odkorkować" potencjał czwartej generacji trzeba odpowiedzieć na następujące pytania

* jak planować efektywną strategię systemów informacyjnych, która jest połączona z cyklem planowania przedsiębiorstwa,
* jak upewnić się, że implementacja strategii zapewni spodziewane korzyści,
* jak przezwyciężać trudności aktualnego przyzwyczajenia przy adaptacji egzystującego systemu do zmieniającego się środowiska i pokonać uszkodzenia z powodu zaległości w konserwacji,
* jak oszacować technologie, produkty i narzędzia, które doprowadzą do pomyślnego wprowadzenia zintegrowanych systemów nowej struktury (traktowanej jako całość).

Niepowodzenie otrzymania maksymalnych zysków poprzez ujarzmienie nowej technnologi, poważnie naruszy możliwości utrzymania aktualnej pozycji w warunkach dzisiejszej rosnącej konkurencji na rynku.

3. Information Enginering Methodology - IEM
(Metodologia inżynierii informacji)

Próbą odpowiedzi na postawione powyżej pytania jest metodologia IEM stworzona przez JMA. Metodologia inżynierii informacji jest terminem używanym do określenia zestawu powiązanych ze sobą dyscyplin, które są potrzebne do zbudowania efektywnego systemu w konwencji czwartej generacji. Podstawową przesłanką IEM jest to, że dane leżą w centrum przetwarzania informacji. Fakt, że podstawowa struktura danych w jakimkolwiek przedsiębiorstwie jest względnie stabilna, podczas gdy ich procedury mają tendencję do zmian, był często demonstrowany. Z tego powodu, przez odpowiednio zastosowane techniki projektowania, zorientowane na dane, systemy informacyjne mogą być budowane w ten sposób aby bardziej efektywnie wyrażać potrzeby użytkownika. IEM idzie w kierunku zaspokajania potrzeb informacyjnych w warunkach zmian w zarządzaniu przedsiębiorstwem.

IEM odnosi sukcesy z czterech głównych powodów :

* systemy wywodzą się ze strategii i planów przedsiębiorstwa,

* systemy tworzone przez IEM są silnie zorientowane na użytkownika, początkowo włączając "główno-dowodzących", a następnie pozostałych użytkowników szczebla wykonawczego w bardziej skonkretyzowane etapy w procesie projektowania.

* proces tworzenia systemu składa się z dużej (wyczerpującej) liczby etapów, z których każdy służy jasno zdefiniowanemu celowi i każdy służy potrzebom dalszych etapów. Etapy mają dobrze określone zadania pozwalające na projektowanie systemów tak aby mogło być ono z łatwością zarządzane i kontrolowane.

* IEM używa najbardziej zaawansowanych i wypróbowanych technik dla analizy, projektowania i wspomagania środkami softwarowymi aby zapewnić utworzenie systemu w sposób szybki i odpowiedni.

Główne cechy IEM :

* planowanie - aby upewnić się, że przedsiębiorstwa otrzymują systemy, których potrzebują aby zabezpieczyć swe priorytety, oraz, że wszystkie systemy mają wspólną architekturę.

* projektowanie z centralnymi danymi - dla wyeliminowania duplikacji wysiłków przy zbieraniu i zarządzaniu danymi.

* techniki - aby dostarczyć dziedzinowych środków do przeprowadzenia analizy i zadań projektowych.

* wprowadzenie użytkownika w proces tworzenia systemu.

* automatyka metodologi - aby polepszyć produktywność i jakość tworzonych systemów.

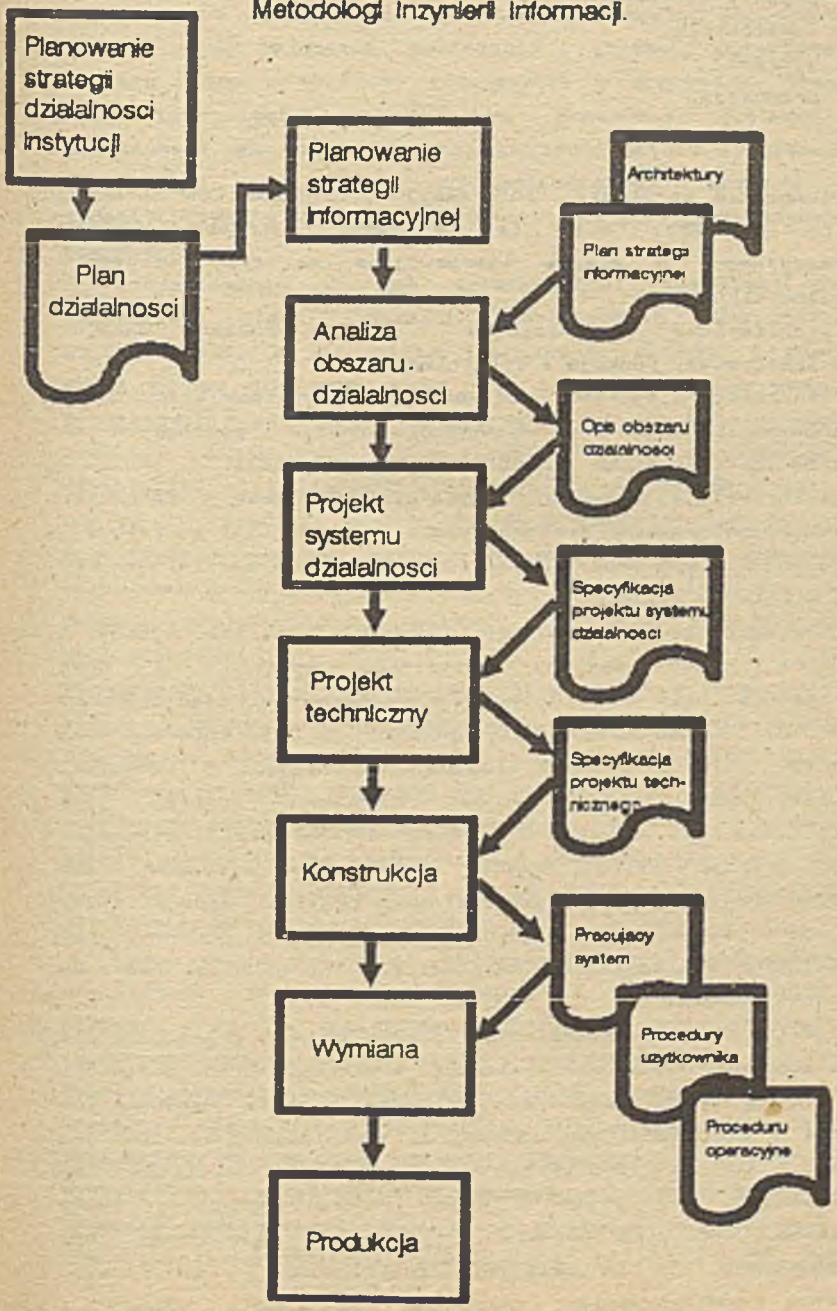
Jak widać cechy te są zgodne z opisanymi powyżej cechami systemów czwartej generacji.

4. Ramy tworzenia systemów informacyjnych za pomocą IEM.

IEM jest złożona z etapów, każdy z jasno zdefiniowanymi zadaniami i technikami. Przy końcu każdego etapu określa się alternatywne drogi dla dalszego cyklu projektowania. Ta elastyczność może w istotny sposób zredukować ramy czasowe potrzebne do przejścia od ustalania strategii informacji do pracującego systemu. Na rysunku przedstawiono wszystkie siedem etapów IEM i to czego one dostarczają. Etapy te to :

- planowanie strategii informacyjnej,

Etapy tworzenia systemow informacyjnych przy pomocy Metodologi Inzynierii Informacji.



- analiza obszaru działalności instytucji.
- projektowanie systemu działalności instytucji.
- projekt techniczny.
- konstrukcja.
- wymiana.
- produkcja.

Planowanie strategii informacyjnej (Information Strategy Planing - ISP), etap ten sprawdza potrzeby systemu informacyjnego. W etapie tymrozwijane są trzy wzajemnie powiązane struktury :

* architektura informacji - opisuje obszary danych przedsiębiorstwa, funkcje i ich interakcje.
* architektura systemu - określa główne systemy i potrzeby przechowywania danych przy wspomaganiu funkcji i podległych im obszarów zdefiniowanych w architekturze informacji.
* architektura techniczna dostarcza kierunków w zakresie sprzętu komputerowego, oprogramowania i komunikacji potrzebnej do wspomagania systemów i przechowywania danych zdefiniowanych w architekturze systemu.

Strategia zawiera również podstawową analizę zysków i konkretne plany wymiany aktualnego systemu na nową architekturę. Plan rozwoju jest przygotowany włączając programy pracy dla najbardziej priorytetowych i krótkoterminowych projektów zajmujących się analizą, technologią i kwestiami organizacyjnymi. Każdy projekt dotyczący analizy jest zdefiniowany tak, aby obsłużyć obszar działalności instytucji, który grupuje kilka ściśle powiązanych systemów, aby zabezpieczyć potencjał przyszłościowego podziału danych. Kiedy plan jest zakończony, pozostałe kwestie nie powinny mieć decydującego znaczenia dla strategicznie określonego kierunku tworzenia systemu.

Analiza obszaru działalności instytucji jest prowadzona dla zdefiniowanego obszaru działalności, w zakresie planu strategii (ISP), - są przeprowadzadzane dokładne badania danych, ich funkcji i informacji potrzebnych do spełnienia tych funkcji. Prowadzi to do identyfikacji specyficznych procesów tej działalności, jej wyjść i wejść informacyjnnych i typów jednostek (np. partnerów), o których dane mogą być przechowywane. Te są dokładnie analizowane : ich nazwy, wzajemne oddziaływanie, znaczenie, wielkości, zasady i

algorytmy są dokumentowane.

Ważną cechą jest maksymalne włączenie bezpośrednich użytkowników do prac nad opisywaniem tego obszaru, wymaganych priorytetów i cech. Na koniec tego etapu zostaje przygotowany opis obszaru działalności instytucji, ukazując zaprojektowane procesy, typy jednostek, powiązania i atrybuty dotyczące tego obszaru razem z wzorcami ich użycia w procesach. Własności wszystkich tych obiektów są dokładnie dokumentowane. Dostarczają one więcej detali dla architektury informacyjnej. Korzystając z tego uszczegółowienia, możliwe jest zidentyfikowanie wyraźnej struktury odpowiedniego wspomagania komputerowego.

Projektowanie systemu działalności instytucji dotyczy wszystkich części systemu bezpośrednio związanych z użytkownikami, a mianowicie transakcje, dialogi i kontrole. Ważną sprawą dla tego etapu jest to, że powinien uzupełniać projektowanie systemu do rozmiarów możliwych bez przesądzania kwestii technicznych. Etap jest ściśle zorientowany na użytkownika i wymaga zgody użytkowników na sposób w jaki będą oni współdziałać z systemem. Końcowym produktem tego etapu jest specyfikacja systemu, zawierająca skonsolidowaną dokumentację przepływów informacyjnych i procedury użytkowników dla każdego procesu, projekt dialogu, ekrany i inne interfejsy dostosowane do użytkownika.

Wiodąca technika w tym etapie wymaga najpierw reprezentowania logiki i użycia danych do procedur w formie opisu, diagramów dostępu do danych, a następnie bardziej dokładnych diagramów działania procedur. Mogą być one bezpośrednio przekształcone w polecenia języka czwartej generacji, a również dostarczyć ścisłej podstawy do kwantyfikacji dla projektu bazy danych. Wszystkie aspekty systemu które współdziałają bezpośrednio z użytkownikiem powinny być zdefiniowane i trwałe.

Celem etapu projektu technicznego jest zaprojektowanie efektywnego systemu komputerowego dla wspomagania wybranych procesów oraz dla zaplanowania właściwego szacunku kosztów i czasu potrzebnego na konstrukcję i wymianę systemu. Projekt ten zawiera strukturę przechowywania danych, programy komputerowe, procedury operacyjne i połączenia. Poziom szczegółowości jest uzależniony od wybranego narzędzia implementacji np. generatory systemów mają wiele struktur najczęściej używanych programów, -stępnie zdefiniowane. Rezultatem projektu technicznego jest specyfikacja techniczna zawierająca projekt bazy danych.

projekty procedur w formie diagramów działania i technologio - zależne szczegóły projektu systemu. Zawierają one opisy przetwarzania wsadowego (batch - runs), zakończone przepływy konwersacyjne i definicje podziału pracy nad oprogramowaniem (units)., Specyfikacja zawiera również architekturę techniczną i standardy wykorzystywane przez system (wybrane środowisko hardwarowe i softwareowe, sposób ich użycia, określone standardy i proponowaną konwencję).

W etapie konstrukcji dla każdego obszaru projektowania analizowanego podczas projektu technicznego budowany jest system. Etap ten zawiera instalację wyposażenia, założenie zbiorów, zdefiniowanie procedur, kodowanie i testowanie programów. Celem jest stworzenie systemu zdefiniowanego w specyfikacji technicznej, który odpowiada celom, ramom czasowym, budżetowi i jest akceptowanej jakości. Ten etap jest traktowany jako zakończony kiedy zdefiniowane kryteria akceptacji dla zadania zostaną uznane przez użytkownika za satysfakcjonujące.

Wymiana jest stopniową zamianą aktualnie pracujących systemów, procedur i zbiorów na nowe systemy i struktury danych. Jest to określane przez plan wymiany razem z planem pracy. Wymiana może być traktowana za udaną, kiedy system pracuje w określonym czasie z założoną tolerancją obserwowanych osiągnięć, proporcji błędów i użyteczności i przejdzie przez komisję postimplementacyjną.

Produkcja jest odpowiednim działaniem systemu z wyregulowaniem i modyfikacjami, jeżeli były potrzebne. Etap ten kończy się tylko wtedy, kiedy następny system zamieni system wymieniany w tej iteracji. Główne zadania podczas produkcji to utrzymywanie serwisu i zadań funkcjonalnych podczas całej działalności systemu.

5. Konkluzja ogólna.

IEM zakłada, że nie wszystkie systemy wymagają aż tak rygorystycznego cyklu postępowania. Chociaż kluczowe aspekty wszystkich etapów IEM mają zastosowanie dla projektowania wszystkich systemów, to można wykorzystywać i krótsze sposoby, a ich etapy nie zawsze muszą być tak jasno naszkicowane.
Podstawowe zyski jakie są osiągane przy zastosowaniu opisanej wyżej metodologii to:

- wysoka produktywność tworzenia systemów,

- systemy odpowiadają priorytetom użytkownika,
- użytkownik jest odpowiedzialny za system,
- kontrola procesu tworzenia systemu,
- jakość tworzonych systemów.

W wielu organizacjach system informatyczny jest zorganizowany bardzo źle; istnieją wieloletnie zaległości. Zbudowanie systemów zabiera zbyt wiele czasu i koszt ich tworzenia jest horrendalny. Rozwijanie oprogramowania aplikacyjnego nie może sprostać dynamicznym i stale zmieniającym się potrzebom przedsiębiorstw. Bałagan i nieefektywność przetwarzania danych jest poważnym problemem organizacyjnym, który musi zostać rozwiązany. Przedstawiona powyżej koncepcja jest jedną z tych, które stawiają sobie za cel wprowadzanie w tych dziedzinach uporządkowania, nawet jeżeli opisane rozwiązania nie wydają się szczególnie rewolucyjne.

mgr inż. Mirosław Izdebski
Politechnika Gdańska

ANALIZA I PROJEKTOWANIE SYSTEMÓW INFORMACYJNYCH Z WYKORZYSTANIEM ANALIZY SYSTEMOWEJ

1. ORGANIZACJA JAKO SYSTEM

Organizacje najczęściej traktuje się jako uporządkowane w pewien sposób systemy społeczno-techniczne. W systemach tych istotną rolę odgrywają ludzie, obok których występują inne obiekty.

Tworzenie postępu w działaniu systemów wymaga świadomości ludzkiej, ponieważ tylko człowiek może być nośnikiem postępu. W tak rozumianych organizacjach zwraca się uwagę na ich potencjał jakim w szczególności są informacje, którymi są wszystkie czynniki jakie człowiek lub urządzenie automatyczne mogą wykorzystać do celowego działania. Umiejętne wykorzystanie tego potencjału jest, w dobie tak zwanego "społeczeństwa informacyjnego", zagadnieniem szczególnie ważnym. Konsekwencją jest coraz większe zainteresowanie metodykami projektowania komputerowego wspomagania procesów organizowania i zarządzania. Preferowanym obecnie nurtem w tej dziedzinie jest systemowe ujmowanie zagadnień modelowania i sterowania tymi procesami.

2. PODEJSCIE SYSTEMOWE

Swiat składa się z wielu systemów działania, a każdy z systemów realizuje inny cel działania. Granice poszczególnych systemów są często rozmyte, a ich elementy określone nieprecyzyjnie. Obiektem badań najczęściej jest istniejący, rzeczywisty system działania lub wybrany element tego systemu.

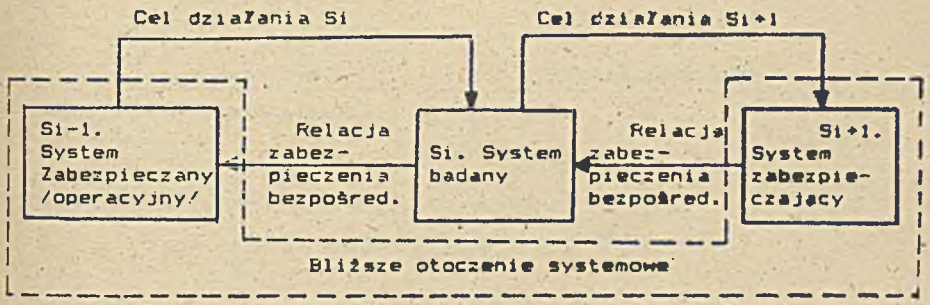
W widzeniu tak skomplikowanej rzeczywistości pomocne jest podejście systemowe ujęte w inżynierii systemów działania, której przedmiotem badań są całości złożone, zwane systemami.

Problem identyfikacji systemu badanego odbywa się trzystopniowo poprzez:

1) podejście prakseologiczne, w którym rzeczywistość systemową ujmuje się z punktu widzenia celów działania;
2) podejście cybernetyczne, w którym podkreśla się aspekt informacyjno-decyzyjny badanego systemu;
3) podejście matematyczne (ekonomiczne), które polega na zastosowaniu modeli ekonomiczno-matematycznych, gdzie podkreśla się aspekt ilościowy realizowanych procesów w badanym systemie.

3. PODEJSCIE PRAKSEOLOGICZNE

Podejście prakseologiczne można zaprezentować przy pomocy typowej sytuacji systemowej, w której przedział systemowy odzwierciedla rzeczywistość z punktu widzenia celów działania i relacji zabezpieczenia bezpośredniego między systemami, rys.1.



Rys.1. Typowa sytuacja systemowa wg podejścia prakseologicznego

W tak przedstawionej typowej sytuacji systemowej przyjmuje się założenie, że wszystkie procesy zachodzące w wyróżnionym systemie zależą bezpośrednio od jego bliższego otoczenia. Założenie to wyraźnie upraszcza proces badań i analizy systemów rzeczywistych, jest to istotne ułatwienie metodologiczne. W szczególnie złożonych systemach organizacyjnych w celu zrozumienia całości problemów tam występujących rozszerza się przedział systemowy o dalsze otoczenie systemowe z zachowaniem wszystkich zasad obowiązujących w typowej sytuacji systemowej.

Na podstawie typowej sytuacji systemowej można stwierdzić, że:

- system badany (Si):
  - realizuje cele działania wygenerowane przez system zabezpieczany (Si-1),
  - posiada konstrukcję, której działanie musi doprowadzić do zaspokojenia potrzeb systemu zabezpieczanego (Si-1),
  - generuje cel działania dla systemu zabezpieczającego (Si+1),
  - oczekuje na zaspokojenie potrzeb własnych przez system zabezpieczający (Si+1).

system zabezpieczany (Si-1):

- generuje cel działania dla systemu badanego (Si),
- oczekuje na zaspokojenie potrzeb własnych przez system badany (Si).

system zabezpieczający (Si+1):

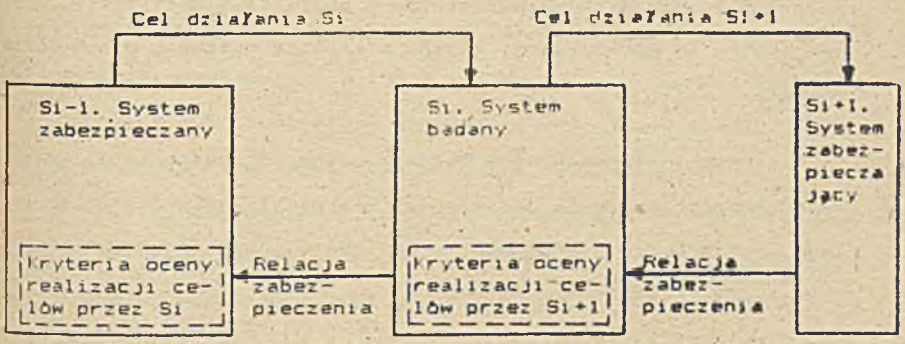
- realizuje cel działania wygenerowany przez system badany (Si),
- posiada konstrukcję, której działanie musi doprowadzić do zaspokojenia potrzeb systemu projektowanego (Si).

Własności przedziału systemowego.

W przedziale systemowym przedstawionym jako typowa sytuacja systemowa wyróżnia się własności dotyczące przedziału jako całości i te poszczególnych systemów składowych, a mianowicie:

- własności oceonowe,
- własności konfliktowe,
- własności czasowe,
- własności informacyjne,
- własności niezawodnościowe.

Własności oceonowe w typowej sytuacji systemowej dotyczą w szczególności systemu zabezpieczanego (Si-1), systemu badanego oraz całego przedziału. Ocenę systemów przedziału systemowego przedstawia rys.2.



Rys.2. Ocena systemów przedziału systemowego.

W tak przedstawionej sytuacji ocenie częstkowej podlegają:

- system badany Si, poprzez ustalone kryteria zaspokojenia potrzeb, ustalone przez jego system nadrzędny, którym w tym wypadku jest system zabezpieczany Si-1;

- system zabezpieczający Si+1, poprzez ustalone kryteria zaspokojenia potrzeb, ustalone przez jego system nadrzędny, którym w tym przypadku jest system badany Si.

Ocena całego przedziału systemowego jest bardzo skomplikowana ponieważ wymaga określania wskaźników ocenowych, które byłyby sprawiedliwe względem wszystkich systemów składowych. Z tego też względu można przyjąć, że dla danego przedziału systemowego oceny całości dokonuje system zabezpieczany Si-1.

Własności konfliktowe w typowej sytuacji systemowej dotyczą różnorodności celów działania systemów w przedziale systemowym. Fakt ten powinien być uwzględniony szczególnie w procesie projektowym, gdyż może doprowadzić do trudności we wdrażaniu.

Własności czasowe w typowej sytuacji systemowej dotyczą różnorodności czasów działania systemów w przedziale systemowym. Fakt ten zmusza do harmonizacji czasów działania systemów w ich przedziale.

Własności informacyjne w typowej sytuacji systemowej dotyczą różnorodności sposobów określania informacji (języków) przez systemy w przedziale systemowym. Fakt ten zmusza do ujednolicenia form zapisu informacji.

Własności niezawodnościowe w typowej sytuacji systemowej dotyczą stanów zdatności poszczególnych systemów i całego przedziału. Niezdatność któregokolwiek z systemów powoduje niezdatność całego przedziału. Tak więc warunkiem koniecznym, aby cały przedział był zdatny, muszą być zdatne wszystkie systemy występujące w tym przedziale.

Wnioski dotyczące podejścia prakseologicznego w ujęciu systemowym

1. Systemy zachowują się celowo, ponieważ celowo zachowują się podmioty w tych systemach.

2. Relacje zabezpieczenia ustalają porządek, dzięki któremu w działaniach złożonych wiadomo, który z systemów służy wyróżnionemu systemowi.

3. Systemy realizują cele systemów nadrzędnych.

4. Zastosowanie podejścia prakseologicznego w ujęciu systemowym umożliwia analizę systemów badanych poprzez określenie: celów, zadań, ról, pozycji i warstw każdego systemu składowego złożonego przedziału systemowego. To umożliwia dalsze pogłębione analizowanie cybernetyczne i matematyczne oraz ocenę i optymalizację wyróżnionego systemu działania.

4. PODEJSCIE CYBERNETYCZNE.

Podejście cybernetyczne można zaprezentować przy pomocy typowej sytuacji systemowej, w której przedział systemowy odzwierciedla rzeczywistość z punktu widzenia sterowania procesami zachodzącymi w systemach działania tego przedziału, rys.7.

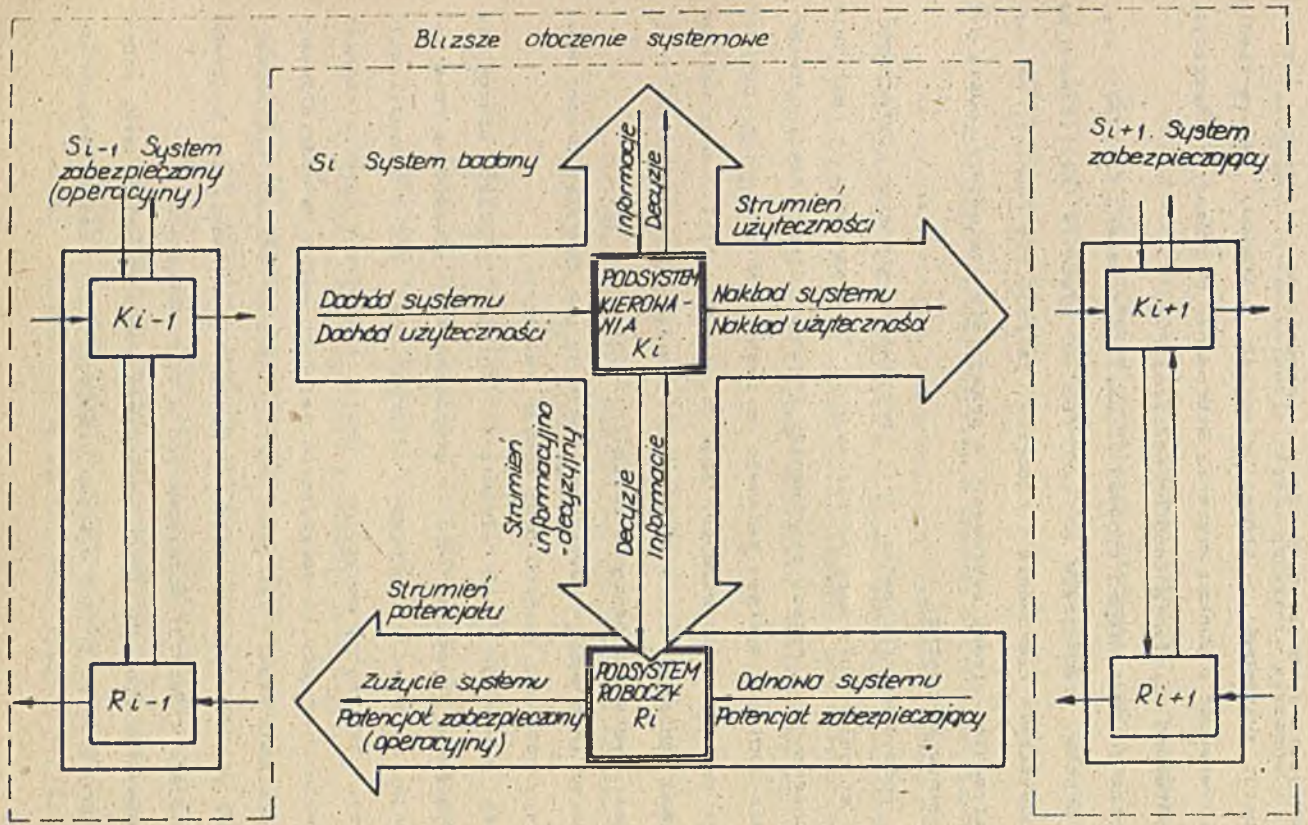
W tak przedstawionej typowej sytuacji systemowej przyjmuje się, że procesy zachodzące w wyróżnionym systemie realizowane są w podsystemie kierowania i roboczym. Zakłada się również, że oddziaływania między systemami w przedziale systemowym muszą być wzajemnie korzystne.

Podsystem kierowania (Ki) przetwarza strumień użyteczności oraz strumień informacyjno-decyzyjny. Przetwarzanie to realizuje w konsekwencji procesy sterowania. Strumień użyteczności odzwierciedla zdolność systemu badanego do zaspakajania swoich potrzeb przez system zabezpieczający jego działanie (Si+1) oraz wartość systemu badanego, która wyraża się różnicą między dochodami i nakładami systemu badanego w określonym czasie.

Strumień informacyjno-decyzyjny obejmuje dwa obszary: obszar wewnętrzny badanego systemu oraz obszar zewnętrzny.

Obszar wewnętrzny tego strumienia dotyczy procesów sterowania podsystemem roboczym (Ri). W procesach tych szczególną rolę odgrywają informacje i procesy informacyjne, zaś w optymalizacji decyzji - algorytmy decyzyjne. Obszar zewnętrzny dotyczy również procesów informacyjno-decyzyjnych mających wpływ na oddziaływania wzajemne między systemami wyróżnionego przedziału systemowego. Zwraca się tu uwagę na wzajemne reakcje i zachowanie się systemów wynikające z decyzji podejmowanych w tych systemach.

Podsystem roboczy (Ri) przetwarza strumień potencjału, który odzwierciedla potencjał systemu badanego, czyli zasoby bezpośred-



Rys. 3. Typowa sytuacja systemowa wg podejścia cybernetycznego.

nio użyteczne dla jego systemu zabezpieczanego (Si-1).
Potencjał ten możliwy do wykorzystania w określonym czasie wyraża możliwości systemu badanego.

Wnioski dotyczące podejścia cybernetycznego w ujęciu systemowym.

1. Warunkiem koniecznym poprawnego modelowania cybernetycznego jest uprzednie utworzenie modelu prakseologicznego sytuacji danego systemu.
2. W podejściu cybernetycznym uwzględnia się konkretne strumienie wielkości wpływających i wypływających z systemu.
3. W warunkach rzeczywistych każdy ze strumieni jest zakłócany. Organizacja systemu najczęściej umożliwia akumulowanie potencjału, użyteczności, decyzji i informacji. Mimo to pojawia się problem sterowania intensywnością strumieni, który należy rozwiązać poprzez ustalenie wspólnej jednostki przeliczeniowej oraz sposób pomiaru intensywności wyróżnionych strumieni.

Mariusz KLAPPER
Sekcja Metodyki i Dokumentowania
Projektów i Programów
ul. Spasowskiego 6/7
31-139 KRAKÓW

ORGANIZACJA I UŻYTKOWANIE NARZĘDZI PROGRAMOWANIA

MOTTO:
Historia Dawida i Goliata dowodzi,
że trafny dobór narzędzi
decyduje o skuteczności.

Jednym z podstawowych warunków skuteczności opracowywania i wdrażania systemów informatycznych jest dysponowanie sprawnymi i niezawodnymi narzędziami programowania. Programowanie jest bowiem podstawowym środkiem realizacji systemów informatycznych. Końcowa jakość i użyteczność systemu zależy w głównej mierze od dobrego i niezawodnego oprogramowania.

Określenie uniwersalnych kryteriów oceny narzędzi programowania nie jest sprawą łatwą, gdyż zależą one od uwarunkowań lokalnych takich, jak profil zastosowań, warunki pracy zespołu programistów, przyjęta metodyka projektowania systemów, zakres zastosowań, profil użytkowników, sprzęt i tp. Dlatego w referacie zostanie omówiony przykład organizacji i użytkowania wyspecjalizowanego pakietu narzędzi przeznaczonych dla programowania typowych zastosowań z zakresu przetwarzania danych w średniej i dużej skali, realizowanych dla różnych odbiorców i przeznaczonych dla sprzętu mikrokomputerowego o średniej i dużej mocy obliczeniowej (od prostych mikrokomputerów klasy IBM-PC do supermikrokomputerów i sieci mikrokomputerowych). Są to typowe warunki spotykane obecnie w kraju przy opracowywaniu systemów przetwarzania danych. W referacie zostaną omówione założenia funkcjonalne przykładowego pakietu narzędzi programowania używanego od pewnego czasu w praktyce z zadowalającymi rezultatami. Nie należy traktować tego referatu jako wskazówek o charakterze

uniwersalnym, ani szczególnie odkrywczym. Jest to jeden z wielu przykładów metod wypracowanych w praktyce, a jego omówienie może pomóc w uporządkowaniu problemu tym wszystkim którzy jeszcze nie poradzili sobie z nim do końca sami. Bardzo ograniczone ramy referatu pozwolą niestety jedynie na wypunktowanie najważniejszych idei, nie pozwolą natomiast na szersze omówienie wielu ciekawych zagadnień szczegółowych

Problemem wyjściowym dla organizacji narzędzi programowania jest wybór języka lub systemu programowania, który będzie wykorzystywany dla tworzenia narzędzi i pisania programów. Nie ma oczywiście uniwersalnego i jednoznacznego kryterium tego wyboru. Można jedynie określić kilka podstawowych wymagań stawianych wobec sprawnego i użytecznego systemu programowania:

- język programowania o czytelnej, precyzyjnej i łatwej w interpretacji składni umożliwiającej strukturalizację programów.
- sprawny system kompilacji umożliwiający szybkie pisanie, kompilowanie i uruchamianie programów oraz wyposażony w możliwości komilowania warunkowego.
- prosty w obsłudze i niezawodny edytor tekstów (najlepiej zintegrowany z kompilatorem).
- szybki i dokładny kompilator wyposażony w możliwości dołączania modułów i segmentów zewnętrznych, oraz tworzący oszczędny kod wynikowy programu.
- wszechstronny i wygodny w użyciu system wykrywania i diagnozowania błędów.
- możliwość komunikacji z najczęściej używanymi światowymi systemami kompilacji, przetwarzania danych, zarządzania sieciami i tp.
- niezawodna, łatwa w użyciu i możliwie wszechstronna biblioteka procedur pomocniczych.

Niestety niewiele spotykanych w praktyce krajowej systemów programowania spełnia w zadawalającym stopniu powyższe wymagania. Dobrym przykładem przemyślanego i wygodnego w użyciu systemu programowania są produkty firmy Borland.

Kolejnym ważnym problemem jest sposób organizowania i zapisywania programów. Czytelność kodu źródłowego programu połączona z ujednoliceniem stosowanego nazewnictwa i organizacji używanych procedur elementarnych bardzo ułatwia pisanie i weryfikacje niezawodnego oprogramowania. Aby programy były czytelne, łatwe do kontrolowania i niezawodne w działaniu wystarczy konsekwentnie przestrzegać kilku prostych zasad:

- prymat skuteczności i niezawodności programu nad jego optymalnością (jeśli te cechy kolidują ze sobą).
- odporność programu na błędy obsługi lub sytuacje awaryjne tzn. kontrolowanie i reagowanie z poziomu progamu na sytuacje awaryjne pojawiające się w

środowisku programu, oraz dokładne kontrolowanie i weryfikowanie współpracy operatora z programem (spełnienie tych wymagań na ogół komplikuje program, ale daje nieocenione korzyści przy jego uruchamianiu, wdrażaniu i eksploatacji).
- unikanie stosowania w programach niatypowych konstrukcji składniowych, i niestandardowych opcji funkcjonalnych (np. nietypowych możliwości systemu operacyjnego).
- ujednolicenie zasad organizowania programu (podział programu na procedury elementarne, sposób zapisywania programu, przestrzeganie reguł programowania strukturalnego i tp.),
- ujednolicenie zasad budowania i zapisywania nazw używanych w programach.
- ujednolicenie sposobów komunikowania się programu z otoczeniem (organizacja i sposoby dostępu do zbiorów danych, współpraca programu z użytkownikiem, kontrola i weryfikacja przesyłania danych pomiędzy programem i otoczeniem, parametryzacja strumieni danych wymienianych pomiędzy programem i otoczeniem i td.). Komplikuje to zazwyczaj strukturę programu i zmniejsza jego czytelność, ale za to znacznie ułatwia kontrolę nad programem, jego testowanie i obsługę.
- ujednolicenie pomocniczych narzędzi programowania (biblioteki procedur, systemy kompilacji, oprogramowanie pomocnicze i tp.).
- ujednolicenie zasad dokumentowania programów i ich składników.

Każda spośród wymienionych powyżej zasad powinna zostać omówiona szerzej, ale brak na to niestety miejsca, gdyż tematem referatu jest organizacja narzędzi programowania, a nie programów. Trzeba jednak zdawać sobie sprawę z faktu, że nawet najlepsze narzędzia programowania będą mało skuteczne, jeśli nie zostanie uporządkowany sposób ogranizowania, zapisu i użytkowania programów.

Podstawowym tematem referatu jest organizowanie i użytkowanie narzędzi programowania rozumianych przede wszystkim jako biblioteka procedur i funkcji pomocniczych, oraz pomocnicze pakiety programów i procedur (tablice decyzyjne, procedury arkuszy kalkulacyjnych i tp.).

Niewątpliwie najważniejszym narzędziem programowania jest biblioteka procedur pomocniczych (rozumianych od strony semantycznej jako procedury lub funkcje). Zarówno jej zawartość, jak też sposób zorganizowania mają istotny wpływ na jakość i sprawność programowania. Zawartość biblioteki procedur zależy od zawartości i jakości zestawu procedur podstawowych i pomocniczych w które jest wyposażony używany język i system kompilacji. Biblioteka

procedur stanowi uzupełnienie tego zestawu, oraz zawiera procedury wyspecjalizowane dla funkcji i zadań lokalnych (niestandardowych). Omawiana w referacie biblioteka została opracowana dla języka PASCAL wersji TURBO. Pomimo dużej łatwości i wygody użytkowania język ten jest stosunkowo słabo wyposażony w procedury biblioteczne przydatne w przetwarzaniu danych. Dlatego konieczne było włączenie do biblioteki procedur, które w innych systemach i językach bywają dostarczane jako procedury standardowe. Przy opracowywaniu omawianej biblioteki były uwzględniane wzorce stosowane w najczęściej używanych systemach i językach programowania. Dla uzyskania lepszej kontroli progamów nad stanem środowiska w którym one pracują zostały do biblioteki włączone procedury dublujące niektóre standardowe procedury języka, ale zapewniające lepszą, niż w standardzie, możliwość kontrolowania i weryfikowania ich działania i skutków.

Dobrze zorganizowana i wyposażona w użyteczne procedury biblioteka pozwala znacznie przyspieszyć i ułatwić pisanie sprawnych i niezawodnych programów. Organizacja i zawartość biblioteki procedur może też mieć istotny wpływ na usprawnienie projektowania systemów, ale problem ten nie mieści się w przyjętych ramach niniejszego referatu.

Organizując bibliotekę procedur należy przyjąć kilka podstawowych zasad:

- biblioteka procedur jest jedna i wspólna dla wszystkich programów.
- procedury biblioteczne są wspólne dla wszystkich programów. Niezbędną wariantowość działania procedur można uzyskać poprzez ich parametryzację. Kopie wariantowe procedur używane dla różnych zastosowań można stosować jedynie w wyjątkowych, uzasadnionych przypadkach.
- biblioteka procedur jest przechowywana łącznie tzn. wszystkie pakiety i procedury biblioteczne są przechowywane we wspólnym miejscu dostępnym dla kompilatora (najczęściej jest to ścieżka w katalogu i/lub wspólny zbiór). Kopie procedur specjalizowanych dla problemów lub użytkowników można uzyskiwać dzięki ujednoliconemu systemowi nazewnictwa kwalifikującemu pakiet lub procedurę do określonej kategorii.
- wszystkie znajdujące się w użytkowaniu egzemplarze biblioteki procedur są identyczne. Nie zezwala się na dokonywanie lokalnych zmian i modernizacji procedur, ani na użytkowanie biliotek procedur zmodernizowanych lokalnie.
- zmiany i modernizacje procedur powinny zapewniać ich zgodność z poprzednimi wersjami. Jeśli nie jest to możliwe, to równocześnie ze zmianą procedury

bibliotecznej należy zmodernizować całe oprogramowanie wykorzystujące tą procedurę.
- w zakresie dostępnych procedur wykorzystywanie biblioteki jest obligatoryjne tzn. w programach nie zezwala się na ponowne programowanie procedur, które znajdują się w bibliotece.
- biblioteka procedur jest podzielona na wyspecjalizowane moduły zawierające grupy procedur przeznaczone dla obsługi określonych dziedzin i problemów. Można to osiągnąć poprzez odpowiednie fizyczne grupowanie procedur w jednostki kompilacyjne (o ile pozwala na to stosowany system kompilacji), lub poprzez odpowiedni system nazywania procedur.
- procedury biblioteczne posiadają aktualną dokumentację opisującą ich możliwości, przeznaczenie i sposób używania.

Procedury przechowywane w bibliotece można podzielić na kilka podstawowych kategorii:

1. PROCEDURY UNIWERSALNE (OGÓLNEGO UŻYCIA).

Są to procedury podstawowe, używane przez wszystkie programy w identyczny sposób i w identycznych zastosowaniach. W tej kategorii powinny się znaleźć następujące grupy procedur:

a. Procedury obsługujące ekran monitora. Powinny one zawierać:
   - zestaw nazw symbolicznych predefiniujących kody znaków wyprowadznych na ekran monitora (ułatwia to przenoszenie oprogramowania na różne typy sprzętu).
   - procedury bezpośredniego przesyłania znaków i łańcuchów znaków na ekran monitora
   - procedury zapamiętywania i przywracania zawartości pamięci ekranu
   - procedury ustawiania i zapamiętywania atrybutów pól ekranu
   - procedury obsługujące wyświetlanie i kasowanie okienek na ekranie

b. Procedury obsługujące klawiaturę:
   - zestaw nazw symbolicznych predefiniujących kody znaków wprowadzane z klawiatury
   - procedury bezpośredniego wczytywania znaków i łańcuchów znaków z klawiatury
   - procedury badające stan klawiatury (o ile nie ma ich w zestawie procedur kompilatora)
   - procedury wczytujące z klawiatury określony kod klawisza ("T", "N", "Esc" i tp.)
   - procedury wczytujące z klawiatury pola elementarne w różnych wariantach (prostym, z możliwością redakcji pola, z predefinicją zawartości pola i tp.)

c. Procedury obróbki elementarnych pól danych:
- badanie numeryczności parametryzowanego fragmentu pola
- lewo-, prawo-, i obustronne kasowanie spacji w polu
- lewo- i prawostronna justyfikacja, oraz centrowanie tekstu w polu
- przekodowanie dużych liter w polu
- pakowanie i rozpakowywanie pola numerycznego na kod BCD
- funkcje kodowania i dekodowania binarnego zawartości pól dla różnych typów danych (funkcje te bywają przydatne również wtedy, gdy składnia języka programowania udostępnia możliwości formatowania i konwersji pól)
- procedury i funkcje obsługujące polski alfabet (identyfikacja znaków, przekodowywanie dla potrzeb sortowania i tp.)

d. Procedury obsługujące przyjęty lokalnie standard ogranizacji ekranu monitora i sposobów komunikacji programu z użytkownikiem i otoczeniem (standaryzacja tych procedur jest bardzo wygodna zarówno dla programisty, jak i dla użytkownika):
- wyswietlanie ogólnego formatu ekranu
- wyswietlanie określonych rodzajów komunikatów w określonych polach ekranu (komunikaty błędów i ostrzeżeń, objaśnienia, informacje o stanie programu i danych i tp.)
- obsługa komunikatów wymagających określonego działania operatora (komunikaty z określoną odpowiedzią, komunikaty z oczekiwaniem na określone działanie i tp.)
obsługa wprowadzania z klawiatury i weryfikacji pól danych parametryzujących przebieg programu
identyfikacja, pozytywanie i weryfikacja systemowych parametrów podawanych w momencie uruchamiania programu (bardzo wygodna jest standaryzacja zasad wprowadzania danych parametryzujących przebiegi, oraz opracowanie typowych procedur dla najczęściej używanych parametrów przebiegu: bieżącej daty, kodów urządzeń z danymi, kluczowych identyfikatorów i tp.)
- obsługa przekazywania parametrów pomiędzy procesami inicjowanymi przez program jako wątki systemowe. Możliwośc inicjowania wątków jest bardzo ciekawą i pożyteczną zaletą systemu operacyjnego, znacznie rozszerzającą możliwości użytkowe programów; przekazywanie parametrów pomiędzy podprocesami wątku ma dla tych możliwości kluczowe znaczenie; w systemie DOS specyficznym problemem jest przekazywanie parametrów powrotu z podprocesu do procesu wywołującego wątek)

e. Procedury obsługujące lokalny zbiór tekstowy (tworzenie, otwieranie, zamykanie, kasowanie, zapis rekordu, wczytanie rekordu, wczytanie ciągu rekordów i tp.). Standaryzacja tych procedur jest wygodna, gdyż w przetwarzaniu danych często są używane pomocnicze zbiory lokalne o organizacji tekstowej np. jako zestawy grup parametrów, pośrednie zbiory drukowania i tp.

f. Procedury obsługujące konwersację z użytkownikiem realizowaną przy pomocy "menu" okienkowych:
   - przechowywanie i odtwarzanie pamięci ekranu używanej przez wyświetlane okienko
   - wyświetlanie i gaszenie okienek na ekranie
   - wyświetlanie menu okienkowego
   - wybieranie opcji z menu okienkowego

   Ciekawymi problemami związanymi z tą grupą procedur jest możliwość zagnieżdżania okienek (menu "rozwijane"), oraz wykorzystywanie nieużywanej pamięci technicznej ekranu dla przechowywania zawartości ekranu zajmowanej przez okienka. Niestety brak miejsca na szersze ich omówienie.

g. Procedury obsługujące kalendarz:
   - wczytywanie i weryfikacja pol zawierających znormalizawaną datę
   - pakowanie i rozpakowywanie daty z postaci przechowywanej w zbiorach danych na postać używaną w konwersacji z otoczeniem
   - zamiana daty z postaci kalendarzowej na postać binarną i odwrotnie (należy zachować zgodność z formatem daty binarnej w systemie operacyjnym)
   - wyliczanie dat dla świąt stałych, świąt ruchomych, dni roboczych i wolnych i tp.
   - wyliczanie atrybutów dla określonej daty (dzień tygodnia, dzień świąteczny, dzień roboczy, najbliższy dzień roboczy i tp.)
   - wyliczanie relacji pomiędzy dwoma datami (ilość dni, ilość dni roboczych, ilość określonych dni tygodnia i tp.)
   - wczytywanie bieżącej daty (i jej składników) z systemu operacyjnego
   - wczytywanie tablicy dat parametrycznych (np. dla zmiennych dni wolnych i roboczych). Bardzo przydatne są tutaj procedury opisane powyżej w punkcie e

   Pakiet procedur obsługujących kalendarz powinien obejmować daty z okresu co najmniej kilkudziesięciu lat (najlepiej całe stulecie), w tym również zmianę daty związaną ze zmianą stulecia w roku 2000.

h. Procedury obsługujące systemową organizację danych w pamięci zewnętrznej:
   - kontrola przesyłania danych do- i z pamięci zewnętrznej
   - przeglądanie i tablicowanie identyfikatorów danych w systemowych katalogach pamięci zewnętrznych
   - przełączanie poziomów rozmieszczania danych w katalogach pamięci zewnętrznych

i. Procedury obsługujące drukowanie danych:
   - inicjowanie i kończenie procedur drukowania
   - wydruk zredagowanego wiersza danych
   - wydruk znaków sterujących pracą drukarki

   Pakiet procedur obsługujących drukowanie danych powinien zapewniać odporność programu na zakłócenia stanu technicznego procesu drukowania (np. próby drukowania na niesprawnej drukarce mogą powodować utratę dużej ilości danych), powinien dawać użytkownikowi możliwość przerwania wydruku w dowolnym momencie, oraz powinien umożliwiać wyprowadzanie wydruku na lokalny zbiór pośredni w pamięci zewnętrznej. Pakiet procedur obsługi drukowania powinien być uzupełniony oprogramowaniem pomocniczym obsługującym różne opcje związane z drukowaniem danych zapisanych do zbiorów pośrednich w pamięci zewnętrznej

k. Procedury organizowania i dostępu do danych w zbiorach. Na ogół procedury te są wbudowane w używany system kompilacji. Jeśli tak nie jest, lub jeśli chcemy używać lokalnego standardu organizacji danych, to dla jego obsługi należy przygotować pakiet procedur w formie makroinstrukcji realizujących wszystkie niezbędne operacje na danych i ich strukturze. Pakiet ten powinien zapewniać łatwą, niezawodną i efektywną współpracę programu z danymi. Struktury i organizacja danych powinny zostać ujednolicone dla wszystkich zbiorów i programów. Dużą uwagę należy poświęcić dobremu "dostrojeniu" parametrów systemu organizowania danych (rozmiar bloku i rekordu, organizacja indeksów, rozmiar, ilość i typy kluczy, parametry indeksów i tp.), gdyż ma to decydujący wpływ na efektywność systemów i programów.

l. Pakiet procedur pomocniczych dla obsługi standardów wymaganych w bankowości:
   - inicjowanie tabeli stopy procentowej
   - naliczanie odsetek bankowych dla zadanego przedziału dat, z uwzględnieniem zmiennej stopy procentowej i ograniczeń wynikłych z kalendarza
   - zamiana liczby na jej wartość słowną
   - obsługa specjalnych formatów pól używanych w bankowości (kredyt, debet, kwalifikatory, format zapisu kwot, wypełnianie pustych znaków i tp)

m. inne pakiety pomocnicze np. dla obsługi wyświetlania na ekranie i drukowania formularzy z rubrykami, nagłówków programów, objaśnień dla użytkownika programu i tp.

2. PROCEDURY SPECJALIZOWANE DLA KONKRETNYCH ZASTOSOWAŃ.

Do tej kategorii wchodzą przede wszystkim pakiety procedur obsługi podstawowych zbiorów danych używanych w systemach. Zbiory danych można podzielić na kilka grup funkcjonalnych:

- zbiory używane globalnie (np. indeks materiałowy, indeks dostawców-odbiorców, indeks wyrobów i tp). Są to zbiory o długich okresach przechowywania i niewielkiej zmienności danych.
- główne zbiory używane lokalnie (np. zbiory transakcji, zbiory rejestrów i tp). Są to zbiory o długim okresie przechowywania i dużej zmienności danych.
- pomocnicze zbiory używane lokalnie (np. wyciągi ze zbiorów transakcji, zbiory pośrednie i tp). Są to zbiory o krótkim okresie przechowywania i dużej zmienności danych.
- zbiory robocze i przejściowe.

Przy organizowaniu pakietu specjalizowanych procedur obsługi zbiorów danych należy przyjąć następujące zasady:

a. Dla zbiorów globalnych istnieje tylko jeden program ich zakładania i aktualizacji. Pozostałe programy mogą jedynie wyszukiwać i wczytywać dane z tych zbiorów. Dostęp do zbiorów może być realizowany przez programy jedynie za pośrednictwem odpowiedniego pakietu procedur bibliotecznych. Pakiet ten predefiniuje strukturę danych w zbiorze i obsługuje podstawowe operacje na zbiorze: otwarcie, zamknięcie, obsługę struktury danych, wyszukanie danych, czytanie danych, inicjowanie wątka systemowego do programu aktualizującego zbiór i tp.

b. Dla głównych zbiorów lokalnych należy przyjąć zasady organizacji pakietu procedur obsługi zbioru podobne jak w punkcie "a" powyżej, dodając możliwości aktualizowania danych w zbiorze bez aktualizowania struktury danych, lub w bardziej złożonych przypadkach umożliwiając aktualizowanie zarówno danych, jak i struktury.

c. Dla pomocniczych zbiorów lokalnych można przyjąć zasady obsługi podobne jak w punktach "a" i "b" powyżej, rozszerzając jednak możliwości aktualizowania danych przez programy użytkowe.

d. Zbiory robocze i przejściowe powinny być obsługiwane przez procedury ogólnego użycia opisane w punkcie 1. powyżej. Należy przy tym dążyć do zachowania zasady definiowania struktury danych tylko w jednym miejscu i ujednolicania procedur dostępu do zbioru.

Zaproponowany sposób organizowania dostępów do zbiorów danych znacznie ułatwia i przyspiesza pisanie programów, oraz zapewnia ich niezawodność. Dobrze zaprojektowany i przetestowany pakiet procedur dostępu gwarantuje bowiem niezawodną komunikację programów z danymi. Ujednolicenie sposobów dostępu do danych upraszcza strukturę programów i poprawia ich czytelność, ułatwia też restrukturyzację danych (wystarczy odpowiednio zmodernizować pakiet procedur dostępu), a nawet całkowitą wymianę strumienia danych (np. przejście z pracy w trybie lokalnym na pracę w sieci lub całkowitą wymianę zbiorów i struktur danych na inne) bez konieczności dokonywania zmian w programach. Ujednolicenie nazewnictwa zbiorów i procedur dostępu do nich ułatwia posługiwanie się tymi procedurami. Wszystkie te korzyści są osiągalne pod warunkiem zapewnienia niezbędnej elastyczności funkcjonalnej procedur dostępu (uzyskiwanej dzięki ich parametryzacji), ujednolicenia zasad ogranizacji i używania zbiorów danych w systemach, oraz ujednolicenia struktury i funkcji typowych programów przetwarzania danych. Pomimo występującej w systemach różnorodności struktur danych i sposobów ich wykorzystywania opisana unifikacja jest możliwa. Ewentualne przypadki nietypowe można rozwiązywać poprzez definiowanie nietypowych procedur pomocniczych, a w krańcowych przypadkach należy zweryfikować prawidłowość projektu struktur danych. Dotychczasowe doświadczenia praktycznego użytkowania omawianej koncepcji organizowania pakietu procedur dostępu do danych przyniosły bardzo zachęcające wyniki i wykazały jej dużą przydatność.

2. PROCEDURY WYSPECJALIZOWANE DLA KONKRETNEGO UŻYTKOWNIKA.

Do tej kategorii wchodzą przede wszystkim procedury specyficzne dla określonego użytkownika np. lokalne sposoby kodowania danych, algorytmy cyfr i symboli kontrolnych, lokalne algorytmy przeliczeniowe (np. sposoby wyliczania cen, algorytmy zaokrągleń, zakresy wartości określonych danych itp.) i tym podobne. Pakiety tej kategorii procedur są w zasadzie unikalne. Należy jednak dążyć do unifikacji nazewnictwa procedur w ramach tych pakietów, oraz (w miarę możliwości) do ujednolicenia ich parametrów i sposobów wywoływania. Grupy procedur przeznaczone dla określonego użytkownika powinny być wyróżniane w sposób jednoznaczny np. poprzez składnik nazwy.

Ograniczone ramy referatu nie pozwalają niestety na obszerniejsze omówienie prezentowanej tematyki. Oddzielnego omówienia wymaga też problem organizowania i użytkowania pomocniczego oprogramowania narzędziowego (np. preprocesorów tablic decyzyjnych, programów weryfikująco-korygujących i tp.). Dlatego zarówno bardziej szczegółowe omawianie tematyki organizowania i użytkowania pakietów narzędziowych programowania, jak też ich rozwój i doskonalenie oraz wszelką wymianę doświadczeń w tym zakresie należy uznać za temat ważny, aktualny i wart zainteresowania.

Kraków, marzec 1990r.

Wspomaganie procesu tworzenia aplikacji użytkowych - w praktyce

Tadeusz Korniak, Jacek Miler, Elżbieta Przepióra

1. Wstęp.

Przydatność tworu (obojętnie, materialnego, czy nie) wynikać musi ze zgodności jego cech z zapotrzebowaniem na takowe. To twierdzenie dotyczy także narzędzi programistycznych. Muszą one odpowiadać możliwościom technicznym użytych urządzeń komputerowych oraz stanowi wiedzy samych informatyków a dopiero za ich pośrednictwem możliwości zastosowania przez użytkowników.

Twory prekursorskie są zazwyczaj zabawką historyków, piszących potem, że już kiedyś, ktoś, gdzieś to wymyślił (np. język Simula). Jeżeli ludzie nie dojrzeli do przyjęcia pewnej wiedzy, to jej po prostu nie przyjmą.

Pośrednim dowodem tego twierdzenia jest fakt, że pojawienie się nowego typu (a właściwie podtypu) narzędzi odbywa się zazwyczaj niezależnie i prawie równolegle w kilku ośrodkach. Równoległość prac ma tę zaletę, że czego nie dojrzą jedni, zobaczą drudzy. Wyniki prac wielu ośrodków mogą być potem uogólnione i proces rozpoczyna się na nowo.

2. Rozwój narzędzi.

Chcielibyśmy na początku przypomnieć koleżankom i kolegom jak wyglądała nasza praca 15 lat temu. Pisaliśmy najczęściej w assemblerze, najwyżej w Fortranie lub Cobolu. Projektowaliśmy przy pomocy schematów blokowych, pisaliśmy kod ołówkiem (sławne twierdzenie: "Programist rabotajet karandaszom"), potem wprowadzaliśmy na dalekopisach. Aby uruchomić program wprowadzaliśmy ręcznie (były to skomplikowane manualne operacje) kilkanaście rozkazów tzw. bootstrap, potem wczytywaliśmy taśiemkę tzw. bootstrap-loader, który umożliwiał wczytanie loadera, który wczytywał kompilator i już po piętnastu minutach mogliśmy dokonać translacji. Po wykryciu błędu translacji przechodziliśmy na halę na której stały dalekopisy, poprawialiśmy taśiemkę i rozpoczynaliśmy zabawę na nowo. Nic dziwnego, że przy błędach wykonania myśl o ponownej kompilacji jeżyła nam włos na głowie. Poprawialiśmy więc kod wynikowy stosując skoki na koniec programu i umieszczając tam zmiany modyfikujące np ciało programu (tę technikę przejęli twórcy różnych dowcipaśnych wirusów, weszła też w skład techniki zabezpieczeń - czyli nihil novi). Problemem wtedy było utrzymanie zgodności projektu - schematów blokowych, tekstu wydruku (drukowanie na dalekopisach dłuższych programów miało czas doprawdy niezerowy), tekstu źródłowego oraz tekstu wynikowego programu. Powodem do dumy była umiejętność czytania taśiemki papierowej i poprawiania jej tzw. chińczykiem. Wspaniałe to były czasy.

Zmiana techniki zdezaktualizowała całkowicie naszą wiedzę. Natychmiastowo osiągana kompilacja pozwala zapomnieć w ogóle o problemie rozbiegania się tekstu źródłowego i wynikowego programu. Języki wysokiego poziomu i współczesny najmodniejszy

assembler tzn. język C pozwalają programować dużo szybciej. Nie zmieniła się jedynie istota naszej pracy - tyle, że problemy, które rozwiązujemy są bardziej skomplikowane niż wtedy. Zamiast rozbieżności między schematem blokowym, programem źródłowym (a właściwie projektem czy propozycją programu, jaką to funkcję pełnił wówczas kod źródłowy), a faktycznie działającym kodem wynikowym programu, mamy obecnie rozbieżność między problemem użytkownika, projektem przedstawionym użytkownikowi, projektem technicznym (którego techniki zapisu są, w dzisiejszej praktyce programowania w Polsce na poziomie wciąż jeszcze schematów blokowych) a wreszcie faktycznie wykonanym systemem. Pielęgnacja takiego tworu to najczęściej skomplikowana praca przypominająca archeologię.

Przyspieszeniem pracy jest budowa bibliotek programów realizujących często powtarzane funkcje. Ta technika stosowana jest prawie od początku, od chwili wynalezienia podprogramu. Ma ona jednak pewną wadę: aby ktoś inny używał cudzych programów musi przyswoić sobie sposób myślenia autorów takiej biblioteki wyrażający się między innymi w nazewnictwie, liczbie i kolejności parametrów czy zakresie realizowanej przez pojedyńczy program funkcji. Przy nieporządnej budowie narzędzia - występuje wiele różnych i niekoniecznie spójnych sposobów myślenia. Jeżeli zauważymy, że projektant-programista musi znać i głęboko rozumieć (opanować):

- kilkadziesiąt pojęć: komend języka, jego funkcji bibliotecznych, opcji kompilacji, różnic między kolejnymi wersjami kompilatorów i systemów operacyjnych, drobnych błędów i kaprysów sprzętu i oprogramowania, itp.

- specyficzne potrzeby i wymagania użytkownika, dla którego tworzy oprogramowanie,

- kilkadziesiąt zazwyczaj nowych pojęć związanych z naszą biblioteką

nie dziwmy się, że rezygnuje z tej ostatniej przyjemności, nawet, gdy dokumentacja jest poprawna, zupełna i dobrze napisana. Pisząc od nowa samemu, odkrywa Amerykę, wydłuża czas realizacji przekraczając wszystkie dopuszczalne terminy, tworzy w międzyczasie własną, bardzo podobną bibliotekę i, zajmując się walką z komputerem i kompilatorem, traci często z oczu główny cel jego pracy - czyli użytkownika jego systemu.

3. NanoX

W wyniku takich przemyśleń rozpoczęliśmy prace nad modelem systemu wspomagającego tworzenie aplikacji. Przyjęto że zakres tworzonych przez ten model aplikacji ogranicza się do dialogowych systemów gromadzenia i przetwarzania danych. Ponieważ najbardziej dostępnym i najsilniejszym narzędziem był wtedy (dwa lata temu) system bazy danych dBASE, a potem kolejne wersje Clippera, a próba dotyczyła metodyki tworzenia narzędzia, (a nie stworzeniu narzędzia do pisania wszelakich systemów), uważamy, że był to wybór słuszny. Stosunkowo proste struktury danych, których obsługę umożliwia Clipper występują na tyle często, że zakres

stosowania narzędzia jest wystarczający do pokrycia kosztów jego opracowania.

Wyodrębniono z funkcji systemów użytkowych te, które występują najczęściej w podobnej postaci:

- dialog z użytkownikiem,
- edycję i prezentację danych zapisanych w systemie,
- zabezpieczenia funkcji (i danych) przed utratą i niepowołanym dostępem,
- przetwarzanie danych.

Podział ten nie jest ani ścisły, ani zupełny. Jest tylko roboczym wydzieleniem etapów do opracowania.

Przetwarzanie uznano za najbardziej indywidualną część każdego z systemów i przyjęto, że nie będzie ono wchodzić w zakres wspomagania tworzonym narzędziem. Nie można przewidzieć jakie może być standardowe przetwarzanie w systemach, dla których przewidziano narzędzie. Opracowanie 80% przypadków spowoduje taką rozbudowę kodu narzędzia, że systemy nie zmieszczą się w pamięci operacyjnej.

3.1. Dialog z użytkownikiem

Funkcje dialogowe można podzielić na trzy grupy:

- określenie i uściślenie żądań użytkownika, dotyczących tego co system ma robić w danej chwili,
- informacje o wymaganych od użytkownika decyzjach i aktualnym stanie systemu,
- zestaw pomocniczych objaśnień (tzw. help)

3.1.1. Menu

Najpopularniejszą i uznaną za optymalną dla posiadanych środków technicznych (alfanumeryczna klawiatura) formą dialogu uściślającego funkcje jest wybór z menu.

Syntetycznie ujmuje to standard IBM [1] zalecający:

- wyraźne wydzielenie części ekranu prezentującej aktualnie dostępne menu (tzw. technika okien),
- podział okna menu na część nagłówkową, pozycje menu oraz obszar kluczy funkcyjnych,

- wybór pozycji równolegle dwiema metodami:

  - natychmiastowym wyborem przez wprowadzenie identyfikatora elementu (numeru, pierwszej litery),

  - ustawieniem kursora na wybranym elemencie i akceptacją wyboru.

Najczęściej dialog taki polega na ułożeniu menu w pewną hierarchię. Z wyborem funkcji wiążą się pewne czynności. Są to:

- badanie rodzaju funkcji (czy menu kolejnego poziomu, czy realizacja szczególnej funkcji),

- badanie dostępności elementu (istnienie, bądź nieistnienie procedury obsługi, uprawnienia użytkownika),

- konieczność wykonania specyficznych działań przed i po uruchomieniu obsługi elementu.

Charakterystyczną cechą tego dialogu, zwłaszcza w początkowym okresie wdrażania systemu, jest zmienność. Użytkownik zmienia swoje zdanie dotyczące położenia danego elementu w hierarchii, tekstów opisujących element, położenia elementu na ekranie, kształtu i kolorów menu, nagłówka, sposobu odzwierciedlenia hierarchii.

Budowa funkcji realizującej dialog musi umożliwiać łatwą zmianę wszystkich tych cech dialogu, pozwalając niejako stworzyć dla użytkownika jego własne, unikalne narzędzie "pasujące do ręki". Niebanalną sprawą jest tu możliwość konfekcjonowania funkcji systemu w zależności od komputera, na którym będzie on zainstalowany lub osoby, która system uruchamia, w sposób niezależny od oprogramowania.

Wnioskiem z tej analizy było stworzenie interpretatora dialogu. Dane opisujące dialog zawarto w bazie dBASE'owej. Zapis danych opisujących powyższe cechy w bazie umożliwia bowiem dostosowanie się do wymagań użytkownika bez konieczności kompilacji. Nie wyklucza to oczywiście uzupełnienia w późniejszym okresie programu o dialog wygenerowany automatycznie na podstawie danych zawartych w opisie.

3.1.2. Teksty informacyjne i żądania decyzji

Uwagi dotyczące zmienności dialogu dotyczą generalnie wszystkich tekstów wyświetlanych przez system. Użytkownik używa na codzień własnych skrótów określających np. tworzone zestawienia (np. FE, KFE, PK), i użytą w rozmaitych komunikatach oficjalna nazwa może go wręcz irytować. Teksty komunikatów, które dla nas są zrozumiałe i precyzyjnie oddają treść, nie muszą być takimi dla ludzi z innej branży, posiadających własny żargon zawodowy. Wreszcie różna jest indywidualna wrażliwość ludzi na słowa i zwroty użyte w dialogu, a nawet na kształt liter napisu (duże i małe litery). Dla jednego tekst "....poczekaj trochę, ja pracuję" jest nieznośnym spoufalaniem się i wymądrzaniem martwej maszyny, dla innego "PROSZĘ CZEKAĆ" jest bezduszne i zbyt

bezosobowe.

Tych cech systemu nie da się ustalić do końca i na zawsze na etapie projektu. Z tych to właśnie powodów wszystkie teksty wymagane w warstwie dostarczanego narzędzia także zapisano na zewnątrz oprogramowania. Po uruchomieniu część narzędziowa odpowiedzialna za dialog wczytuje teksty do odpowiednich tablic. Zaciemnia to co prawda przejrzystość oprogramowania, - inaczej bowiem wygląda fragment programu:

```
@10,15 say "Prosze czekac"
```

niż:

```
@10,15 say qva_tx[10]   && komunikat typu prosze czekac
```

ale korzyści osiągane dzięki temu przewyższają koszty porządnego komentowania programu.

3.1.3. Teksty objaśniające (help)

W NanoXie objaśnienia przechowywane są w specjalnej bazie. Funkcja wyświetlania tego tekstu jest zawsze dostępna (związana ze zdefiniowanym dla danego systemu klawiszem, najczęściej F1). Sam tekst objaśnienia może być związany:

- z nazwą procedury,
- z nazwą aktualnie wczytywanej zmiennej,
- z nazwą procedury i nazwą aktualnie wczytywanej zmiennej,
- ze zdefiniowanym słowem kluczowym.

Istotną cechą systemu podpowiedzi NanoXa jest pozostawienie w oprogramowaniu aplikacyjnym części mechanizmów generacyjnych, umożliwiających na bieżąco, w trakcie pracy systemu zmieniać, poprawiać i dopisywać teksty objaśniające.

3.2. Edycja i prezentacja danych.

Podstawowe funkcje edycji danych to wprowadzanie, poprawianie, usuwanie i przeglądanie pewnych konglomeratów danych, złożonych z ciągu danych elementarnych. W tym celu używa się tzw. formatek - opisanych na ekranie miejsc wprowadzenia. (Mogą istnieć także inne funkcje edycji np. wymiana podzbioru danych formatki w podzbiorze zgromadzonych danych).

Ponieważ termin "funkcja" jest w informatyce przeciążony - nazwijmy tak określone szczegółowe funkcje edycji - akcjami. Proponujemy także drugi termin - usługa na określenie zestawu (modułu) powiązanych akcji.

3.2.1. Moduł (usługa) nawigacji

Ukształtował się obecnie pewien standard realizacji funkcji edycji w systemach - na nasz użytek nazwaliśmy go usługa nawigacji. Jego cechami charakterystycznymi są:

1. Podział ekranu na obszary:

   - obszar akcji zawierający ich nazwy.

   - obszar przeglądu, w którym wyświetlane są jednolinijkowe ogólne informacje identyfikujące jednoznacznie konglomeraty danych (np. klucze rekordów) umożliwiające użytkownikowi wybór elementu do dalszej pracy. Linie te będą zwane dalej paskami przeglądu.

   - obszar klawiszy funkcyjnych, (pod-akcji, funkcji szczegółowych).

2. Dwa tryby obsługi obszaru przeglądu i obszaru akcji:

   A. Niejawna akcja przeglądu - obszar przeglądu i obszar akcji są dostępne równocześnie;

      - klawisze strzałek poziomych (oraz np. pierwsze litery nazw akcji) pozwalają na wybór akcji. Wybór dokonany strzałką zatwierdzany jest klawiszem Enter, wybór litery daje efekt natychmiastowego przejścia do akcji.

      - pozostałe klawisze kursorowe (strzałki pionowe, PgUp, PgDn, Home, End) umożliwiają poruszanie się po obszarze przeglądu

      - w obszarze klawiszy funkcyjnych dostępny jest co najmniej klawisz wyświetlania tekstów objaśniających (standardowo F1)

   B. Jawna akcja przeglądu - przegląd jest traktowany tak samo jak każda akcja, w jej trakcie dostępne są:

      - klawisze kursorowe (strzałki pionowe, PgUp, PgDn, Home, End) umożliwiające poruszanie się po obszarze przeglądu

      - klawisze strzałek poziomych uzyskują indywidualne dla danej funkcji znaczenie (np. jeżeli wyświetlany pasek jest szerszy niż okno przeglądu, realizują one przejście do niewidocznej części paska), bądź są nieczynne

      - w obszarze klawiszy funkcyjnych, oprócz klawisza tekstów objaśniających dostępny jest co najmniej jeden klawisz przejścia do wyboru akcji.

Drugi tryb obsługi obszaru przeglądu umożliwia dwa sposoby przejścia do obszaru akcji:

- po wyborze elementu z obszaru przeglądu,
- bez wyboru elementu

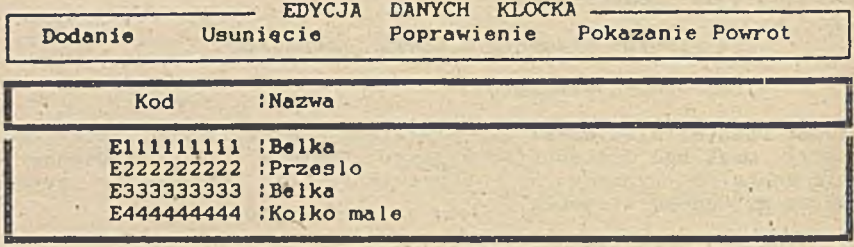
dwoma różnymi klawiszami np. odpowiednio Enter i Esc. Niektóre z akcji mogą być dostępne po wyborze elementu , a niedostępne gdy nie został on wybrany (np. poprawienie czy obejrzenie pełnego opisu wybranego elementu).

3. Typowe (standardowe) akcje:

- Wyświetlenie pełnej informacji o wskazanym elemencie,
- Dodanie elementu,
- Zmiana danych wskazanego elementu
- Usunięcie wskazanego elementu,
- Zaznaczanie elementu/grupy elementów,
- Szukanie elementu,
- Opcje - inne, często występujące akcje jak np.:
  - Filtr - zmiana sposobu wyboru podzbioru elementów wyświetlanych w obszarze przeglądu,
  - Indeks - zmiana sposobu uporządkowania,
  - Wypełnienie - wstępne grupy danych wskazanymi wartościami,
  - Raport - szybki wydruk informacji w uproszczonej formie.
- Koniec pracy modułu nawigacji, powrót do funkcji wywołującej,
- Przegląd - dla trybu jawnej akcji przeglądu.

Moduł nawigacji musi umożliwiać:

- zdefiniowanie układu graficznego ekranu i zawartości obszaru przeglądu,
- wyboru podzbioru typowych akcji i dołączenia akcji niestandardowych,
- zdefiniowania klawiszy funkcyjnych czynnych dla akcji przeglądu,
- określenia własnych tekstów opisujących akcje i klawisze funkcyjne.



Rysunek 1. Przykładowa realizacja funkcji nawigacji edycji z jawną akcją przeglądu

3.2.2. Wprowadzanie danych.

Podstawowe akcje edycji to akcje umożliwiające wprowadzenie do systemu poprawnych danych. Aby to umożliwić, należy w trakcie wprowadzenia badać każdą z wprowadzonych danych elementarnych, a także ich wzajemne związki oraz wspomagać operatora-użytkownika systemu w wyborze poprawnej danej.

Zakres badania poprawności określa zazwyczaj tylko sam obiekt, którego dotyczy system. Zakres wspomagania jest silnie zależny od typu użytkownika, dla którego przeznaczony jest system. W systemach, w których dane wprowadza użytkownik odpowiedzialny za nie (np. handlowiec tworzący fakturę) wspomaganie obejmować powinno dosyć szeroki zakres: od udostępnienia rozszerzalnego w trakcie edycji słownika i specjalizowanego kalkulatora, po komunikaty sygnalizujące odstępstwa od standardowych algorytmów. Tam, gdzie wprowadzania dokonuje operator (np. wielostanowiskowe systemy fakturowania, przeznaczone dla operatorów przepisujących dane z formularzy) wystarczającym wspomaganiem jest sygnalizowanie błędu komunikatem, bowiem podjęcie decyzji o poprawności danej odbywa się na innym szczeblu.

O ile drugi typ edycji nie przedstawia specjalnych problemów w Clipperze (dostarczane przez opcje komend say i get mechanizmy wzorców (picture) i funkcji sprawdzających (valid) są całkowicie wystarczające), o tyle pierwszy typ wymaga bardziej skomplikowanej procedury opracowania pojedyńczego pola pobrania.

Edycja pojedyńczego pola pobrania

Pełna edycja pojedyńczego pola pobrania wymaga:

- **funkcji pre-walidacji** - funkcji wykonywanej przed wejściem do pola pozwalającej określić warunek dostępności pola i/lub dokonać czynności wstępne np. ustawienia wartości początkowej (domyślnej) danej elementarnej,

- zdefiniowania wzorca wprowadzanej danej eliminującego błędne znaki.

- funkcji walidacji danej czyli możliwości badania poprawności danej jedną lub wybranymi z trzech metod:

  - słownikiem poprawnych danych dla danych należących do skończonego zbioru wartości.

  - kalkulatorem pozwalającym wyliczyć wartość danej i wpisującym wynik bezpośrednio do pola pobrania.

  - funkcją której argumentami mogą być także uprzednio wprowadzone dane elementarne.

funkcji post-walidacji - funkcji wykonywanej po poprawnym zakończeniu wprowadzania danej (np. zmiana danych opisujących, sumowanie, zapis śladu operacji itp.)

Strony logiczne i fizyczne formatki

Wszystkie dane elementarne mogą być jeszcze podzielone na grupy - strony logiczne na początku i na końcu edycji których należy także dokonywać procedur pre- i post-kontrolnych. Ponadto na podział ten nakładają się ograniczenia związane z wielkością ekranu - tzw. strony fizyczne.

Tak zdefiniowane potrzeby edycji wymagają zbudowania własnych struktur umożliwiających opis każdej z edytowanych danych elementarnych. W NanoXie problem został rozwiązany w oparciu o tablice opisujące wszystkie te atrybuty edycji dla wszystkich pól jednej strony logicznej. Założono przy tym, że strona logiczna mieści się całkowicie na stronie fizycznej, strona fizyczna zaś może składać się z jednej lub wielu stron logicznych.

Słownik

Zbudowano dwa mechanizmy słowników pozwalających na kontrolę wartości przy użyciu zbioru danych zapisanego w innej bazie i rozbudowę tej bazy słownikowej w trakcie edycji. Słowniki te różnią się techniką realizacji, co przejawia się w liczbie i rodzaju akcji dostępnych w trakcie użycia słownika:

- **Mały słownik** pozwala dopisać tylko dwie dane -kod i jego opis.

- **Duży słownik** jest realizowany przy użyciu pełnego modułu nawigacji.

Mały słownik przeznaczony jest do wspomagania wprowadzania prostych danych kodowanych np. jednostek miary. W jego budowie użyto standardowej opcji valid oferowanej przez Clipper.

Duży słownik przeznaczony jest do wspomagania i walidacji wprowadzania danych o bardziej złożonych strukturach opisu np.

kodu materiału. W trakcie dopisywania takiej danej (dopuszczalnego w chwili edycji danej elementarnej głównej formatki) należy wprowadzić pełny jej opis. W skład tego opisu mogą wchodzić dane kodowane prostymi (a także nie daj Boże i złożonymi) słownikami np. owej jednostki miary. Słownik ten został zbudowany jako klasa na tyle poprawnie, na ile na taką działalność pozwala prymitywna metoda zasłaniania zmiennych dostępna w użytym narzędziu.

Słownik jest dostępny w dwóch trybach:

- Po wprowadzeniu do pola pobrania nieistniejącej w słowniku wartości słownik jest uruchamiany automatycznie.

- Przed i w trakcie wpisywania danej dostępny jest klawisz funkcyjny (wskazany na etapie definiowania systemu), po użyciu którego uruchamiany jest słownik.

Wybór elementu ze słownika dokonywany jest przy pomocy klawiszy obsługi kursora. Wybrana wartość jest automatycznie' przenoszona do pola pobrania. Słownik, aż do końca obsługi pola jest ustawiony na tej wartości umożliwiając postprocedurom wypełnienie pól związanych z aktualnym danymi opisującymi np.: w przypadku kodu materiału - wpisanie do kolejnego pola formatki jednostki miary zapisanej przy tej pozycji słownika.

Możliwość dopisania danej do słownika małego, oraz funkcje słownika dużego są parametrami opisu walidacji.

Kalkulator zwykły i kalkulatory specjalizowane

Nieprawdą jest, jakoby użytkownikowi wystarczyło użycie kalkulatora zewnętrznego w stosunku do naszego systemu dla dokonania obliczeń edytowanych danych numerycznych. Oczywiście może on wyjąć z szuflady kalkulator (stać go wszak na komputer, to i na kalkulator się znajdzie), albo można go nauczyć użycia np. Side-Kicka jeżeli nie zajęliśmy całej dostępnej pamięci operacyjnej (co nastąpi szybciej niż myślimy). Jest jednakże powód wiele poważniejszy - jeżeli narzędzie ma być zupełne, wynik kalkulacji musi znaleźć się w polu edycji bez przenoszenia go tam przez człowieka, czyli po zakończeniu przez niego operacji arytmetycznych.

W NanoXie kalkulator jest dostępny, analogicznie jak słownik, po zdefiniowaniu, że edycja danego pola jest wspomagana kalkulatorem.

Kalkulator zwykły pozwala na dokonanie podstawowych operacji arytmetycznych. Kalkulator "specjalizowany" jest traktowany jako swoista procedura kontrolna (jeden parametr jest wprowadzany (i np. ma wartość domyślną), reszta jest pobierana ze zmiennych i pól bazy.

Kalkulator (patrząc z zewnątrz) może być uruchamiany dwiema metodami:

- Po wprowadzeniu do pola pobrania niezgodnej z uzgodnioną procedurą kalkulacji jest uruchamiany automatycznie.

- Przed i w trakcie wpisywania danej dostępny jest klawisz funkcyjny, po użyciu którego uruchamiany jest kalkulator.

Formatka interpretowana

Jeżeli zanalizujemy kilka formatek zauważymy, że wiele elementów opracowania pola edycji jest wspólnych. Ponadto samo tworzenie opisu edycji formatki nawet w postaci definiowania tablic zajmuje bezproduktywnie kod programu, zabierając cenną już w tym momencie pamięć operacyjną. I tak systemy używające narzędzi tego typu muszą już być nakładkowane, ale jeżeli nie zaczęlibyśmy oszczędzać, to zabrakłoby nam miejsca nawet na nakładki!

Z drugiej strony zmienność sposobu wprowadzania danych jest dwojakiego typu: zmiany wynikające z uściślenia żądań użytkownika oraz zmiany wynikające z warunków zewnętrznych np. zmiana obowiązujących instrukcji wypełniania dokumentów. Poważne zmiany muszą oczywiście wywołać zmianę i ponowną kompilację kodu, ale zmiany polegające np. na zaniechaniu wprowadzania pewnej danej elementarnej lub zmianie jej położenia na ekranie bez zmiany algorytmu wprowadzania nie powinny wymagać aż takich zabiegów.

Dlatego zdecydowano się na stworzenie interpretatora formatek (co nie jest specjalnie nowatorskim, ale za to wygodnym rozwiązaniem). Dane opisujące wszystkie formatki systemu umieszczono w bazie, z której opisy odpowiedniej formatki są w razie potrzeby przepisywane do dynamicznie tworzonych w pamięci tablic.

3.3. Zabezpieczenie danych

Zabezpieczenie danych obejmuje:

- ochronę przed zniszczeniem lub utratą (archiwację),

- ochronę przed niepowołanym dostępem do danych, a tym samym i do pewnych funkcji systemu.

3.3.1. Archiwacja danych

Problem zachowania kompletu danych ze wskazanego momentu występuje w każdym systemie przetwarzania. W NanoXie opracowana została usługa archiwacji. Polega ona na:

- opracowaniu procedury archiwacji (rozplanowane z góry, cykliczne przepisywanie danych na n-kompletów archiwacyjnych składających się z oznaczonych dyskietek; zapisywanie historii archiwacji, możliwość tzw. wiecznej archiwacji, tzn. przechowywania danych ze wskazanego okresu przez czas nieograniczony; komplet przeznaczony do wiecznej archiwacji jest zastępowany innym o tym samym numerze i wycofywany z obiegu).

- przygotowaniu zestawu funkcji (akcji) archiwacyjnych:

  . archiwacja danych,

  . odtwarzanie danych po awarii,

  . przygotowanie kompletu archiwacyjnego o wskazanym numerze,

  . wycofanie kompletu do "wiecznej archiwacji".

- wspomaganiu planowania rozłożenia zbiorów na dyskietkach kompletu archiwacyjnego.

3.3.2. Ochrona przed niepowołanym dostępem

Często wymagane jest badanie, czy użytkownik ma prawo użycia funkcji zarówno edycyjnych (wprowadzanie istotnych danych, zatwierdzanie wprowadzonych danych), jak i prezentacyjnych (np. raporty podające przekrojowe dane ekonomiczne przedsiębiorstwa). Narzędzie dostarcza standardowych mechanizmów ochrony takich jak:

- ochrona dostępu kodem ochrony; istnieje lista uprawnień użytkowników, każdy z użytkowników jest identyfikowany swoim nazwiskiem i hasłem, dostęp do funkcji chronionej kodem i-tym ma użytkownik posiadający na swojej liście i-te uprawnienie.

- ochrona przez szyfrowanie informacji w bazach danych.

Mechanizmy te są dostępne w postaci gotowych procedur bibliotecznych. Ze względu na użytą bazę danych (która i tak potrafi odczytać każdy informatyk), nie budowano bardziej wyrafinowanych środków ochrony. Zresztą i tak przed informatykiem jedynym zabezpieczeniem jest solidna kłódka na drzwiach komputerowni, a dla pozostałych użytkowników takie zabezpieczenie jest zwykle wystarczające.

Generacja systemu aplikacyjnego.

Już przy omawianiu rozwoju narzędzi wspomnieliśmy o niewygodzie związanej z użyciem narzędzi w postaci biblioteki. Dlatego też na bazie omówionej biblioteki zbudowano generator wspomagający tworzenie systemów w sposób dialogowy.

Nie doprowadzono do całkowitej generacji (nie przygotowano bowiem praktycznie żadnego wspomagania przetwarzania danych), wykorzystano jedynie fakt, że większość realizowanych przez NanoXa usług jest zaprojektowana jako interpretatory danych zapisanych w dBASE'owych strukturach baz danych, a generator systemu aplikacyjnego jest wszak sam w sobie systemem przetwarzania danych. Uzyskano przy tym pośredni dowód na czystość kodu NanoXa.

Dodatkową korzyścią jest możliwość szybkiego tworzenia makiet, na bazie których można uzgadniać szczegóły realizacji systemu z użytkownikiem. W tym wypadku używa się symulacji przetwarzania, oraz silne wspomaganie tworzenia końcowej dokumentacji technicznej systemu (dokumentacja struktur baz i ich wzajemnych związków, dokumentacja funkcji, podręcznik operatora zbudowany z tekstów podpowiedzi).

4. Inne narzędzia.

Równolegle z NanoXem przebiegały prace nad innymi narzędziami realizowanymi w postaci bibliotek. Biblioteki były pisane w języku C i mogą być wykorzystywane w programach napisanych w tym języku, lub w programach pisanych w PASCALu i FORTRANie. Są to:

- GRMETODS - (metoda dostępu) - indeksowa metoda dostępu do danych zapisanych w postaci rekordu o dowolnej krotności każdego z pól,

- GRBASE - (baza danych) - system bazy danych umożliwiający zapisanie rekordów typu GRMETODS w postaci drzewiastej, całkowicie nieregularnej, acz w pełni obsługiwanej struktury.

- SCRLIB - (biblioteka procedur ekranowych) - wzorowana na narzędziach typu "Witamina C" biblioteka ułatwiająca interakcję z użytkownikiem; wraz z tą biblioteką powstały dwa modele generatorów zbudowanych w oparciu jej procedury. Są to: generator formatek ekranowych FORMBLD i generator menu systemów aplikacyjnych MENUBLD. Oba generatory tworzą kod źródłowy w języku C.

5. Dalsze prace.

Doświadczenia wyniesione z budowy narzędzi posłużyły nam do tworzenia kolejnego tworu - Biblioteki Programisty Aplikacyjnego (BPA). Jest on próbą podsumowania funkcjonalnego podejścia do budowy systemów aplikacyjnych. Podczas jego projektowania przyjęto założenia opisane poniżej.

5.1. Jednorodność i spójność struktur danych wewnętrznych

Zasadą, którą starano się kierować się przy budowie struktur użytych w bibliotece, była ich jednorodność (budowa z takich samych części składowych i posiadanie tych samych właściwości) oraz spójność (ścisła łączność, zwartość), co w rezultacie pozwolić powinno na uzyskanie przejrzystości kodu. Zasada ta, oprócz względów estetycznych, ma przesłanki praktyczne: skomplikowane, niejednorodne struktury powodują, że oprogramowanie pracujące na nich tworzy gęstą sieć powiązań, trudną do opanowania, modyfikacji, testowania i pielęgnacji.

Doświadczenie wyniesione z realizacji podobnych rozwiązań podpowiada, że rozwiązanie w którym mieszane są funkcje narzędzi różnych poziomów (zbyt usłużne, tzn. albo podaj precyzyjnie swoje żądania, albo zostaną one automatycznie dostosowywane do zmieniających się warunków wg pewnego algorytmu) powodują, że nigdy nikt nie jest tak naprawdę zadowolony. Niech przykładem będzie automatyczne tworzenie menu, gdzie położenie poszczególnych pozycji może być podane jawnie, lub jeśli takich dyspozycji nie ma, określane automatycznie.

Kod programu obsługującego tak postawione żądanie jest wyraźnie nadmiarowy w stosunku do rezultatu - w trakcie obsługi aplikacji wykonujemy wszak stale pracę generatora, czyli narzędzia wyższego poziomu!

Nie ma ceny na przejrzystość kodu i struktur. Nawet jeżeli jednolitość budowy, zwiększa zajętą na dane pamięć i wydłuża czas obsługi, to korzyści związane ze zmniejszeniem kodu (a tym samym z przejrzystością programu) przewyższają straty, w gruncie rzeczy wyimaginowane, bo wynikające z "zaoszczędzenia pracy" na zupełnie innym etapie budowy aplikacji i niezauważalne dla użytkownika zewnętrznego.

Teoretycznie zasada taka jest zawsze podstawą projektowania struktur danych, kłopotem jest jednak określenie, co jest przejrzyste i jednorodne. Zawsze zresztą kusi, aby zrobić coś i jeszcze troszeczkę, jeżeli tylko dołożymy ten, chrakterystyczny i odmienny element w strukturze. Na ile pomysły takie zostały wytępione w strukturach BPA - trudno ocenić, w każdym razie bardzo się starano i zrezygnowano z paru interesujących pomysłów "optymalizujących".

5.2. Zupełność danych

Zupełność danych (całkowitość, kompletność, pełnia) jest raczej abstrakcją, niż możliwym do spełnienia postulatem. Można dążyć do zupełności w takim znaczeniu, że dane zapisane w strukturach winny umożliwiać :

- realizację wszystkich typowych potrzeb aplikacji, z którymi zetknęliśmy się dotychczas,

- realizację nietypowych żądań nie przez rozbudowę, czy zmianę struktur, a przez mechanizmy wymiany informacji zapisanej w tych strukturach.

- wymienność warstw oprogramowania (np. możliwość wymiany

systemu bazy danych).

5.3. Zasada rozdziału danych między struktury

Struktury danych i procedury na nich działające są pogrupowane w hierarchicznie podporządkowane moduły. Każdy z modułów jest odpowiedzialny za inną właściwość oferowanego mechanizmu np.

- związek z zastosowaną bazą danych,
- związek z akcją dotyczącą bazy (nawigacja, edycja),
- związek z elementem akcji (edycja strony logicznej, edycja ciągu pól)

W ramach modułu dla rozwiązania podobnych potrzeb będą stosowane takie same mechanizmy np. listy.

Mechanizmy te współdziałają na zasadzie analogicznej, jak mechanizm "zasłaniania zmiennych" w Pascalu - mechanizmy lokalne przesłaniają mechanizmy wyższego poziomu.

Wyjątkiem są tu nieprzedefiniowywalne mechanizmy globalne w rodzaju np. klawisza wywołującego objaśnienie (help).

Wszystkie dane opisywanych struktur można podzielić wg kryterium pochodzenia na trzy grupy:

- Indywidualne dane podawane przez aplikację w chwili zgłoszenia potrzeby wykorzystania danego mechanizmu tzw. parametry inicjacji.
- Dane, którym w chwili inicjacji nadawane są wartości domyślne, takie, które realizują najczęściej używany wariant mechanizmu. Wszystkie te dane są dostępne programiście aplikacji za pośrednictwem funkcji zmiany, pozwalając mu tym samym dostosować oferowany mechanizm do swoich potrzeb.
- Dane dostarczane w chwili aktywizacji mechanizmu (dane spustowe) - ograniczone do niezbędnego minimum.

5.4. Zasada budowy funkcji BPA

Funkcje BPA zbudowane są metodą klasy tzn. opierają się na własnych danych wewnętrznych oddzielając je od użytkownika. Wszystkie funkcje dzielą się na:

- zewnętrzne - widziane przez użytkownika i dostępne dla niego,
- wewnętrzne - wykonujące wyłącznie operacje na własnych strukturach danych (pełnią więc rolę analogiczną jak struktury danych).

Wyraźne odcięcie wnętrza warstwy od warstwy wyższego rzędu nie jest oczywiście ani złośliwością, ani wyrazem pychy twórców narzędzia. Celem tego zabiegu jest, aby w razie wykrycia

niesprawności użytych mechanizmów wymienić je na poprawne bądź lepsze bez zmian w warstwach wyższego poziomu.

Od strony użytkowej funkcje dzielą się na:

- funkcje inicjacji warstwy,
- funkcje podmiany danych domyślnych,
- funkcje aktywizacji mechanizmu

który to podział jest odbiciem trzech grup danych zapisywanych w strukturach.

5.5. Ograniczenia

Podstawowym ograniczeniem jest zdrowy rozsądek, który np. podpowiada, że nie jest rozsądnym żądanie, aby w pasku przeglądu literki były w kilku kolorach, natomiast jest już rozsądnym wydaje się być żądanie, aby każdy element paska przeglądu mógł być w innym kolorze (co w konsekwencji prowadzi umożliwia stworzenie paska, w którym każda literka jest w innym kolorze - trudno powiedzieć, czy to ma coś wspólnego ze zdrowym rozsądkiem).

Drugim ograniczeniem są klasy budowanych aplikacji - dialogowe systemy przetwarzania danych.

Trzecim ograniczeniem są struktury danych zapisywanych w bazie danych, a właściwie sposób patrzenia na te struktury przez projektantów aplikacji.

Najsilniejszym elementem jest tu indoktrynacja strukturami relacyjnymi, większość bowiem projektantów aplikacji wykorzystywała głównie dBASE i widzi nowe narzędzia przez filtr doświadczeń z niezupełnie relacyjną bazą danych.

Powoduje to, że dla ułatwienia pracy należy co najmniej powtórzyć mechanizmy oferowane przez dBASE rozszerzając je o te elementy, które są związane ze specyfiką GRMETODS i GRBASE takie jak np. pojedyńcze pola wielokrotne.

5.6. Przegląd struktur BPA

Przyjęto, że aplikacja jest zbiorem akcji (z których każda jest zbiorem pod-akcji itd.) dokonywanych na obiekcie odwzorowanym przez zapis w bazie danych. Tak więc opis obiektu jest w miarę statyczny, jest tym, co można uznać za stałe, a więc najważniejsze.

Opis obiektu jest w bazie, czyli pierwotny dla każdej akcji będzie opis bazy, na której akcja jest dokonywana (bądź z którą akcja jest związana pośrednio).
Samo uszeregowanie i hierarchia akcji jest raczej przypadkowa, związana ze zdroworozsądkowymi zasadami np. "wkładam_dane_do_bazy ->przeglądam_dane_z_bazy ->wyjmuję_dane_z_bazy". Nie ma żadnych

przeciwwskazań do innego rozdziału np. akcji na podakcje

5.7. Spodziewany efekt końcowy

Przyczyny niewygody związanej z użyciem biblioteki w postaci wywołań funkcji w programie opisaliśmy przy omawianiu NanoXa. Wykorzystując doświadczenia nabyte przy tworzeniu generatorów NanoXa, FORMBLD i MENUBLD tworzony jest generator pozwalający generować część oprogramowania automatycznie. Dane opisujące aplikację przechowywane są w bazie danych. Nie zamykamy jednak narzędzia w tym sensie, że efektem generacji jest w ostateczności zawsze kod źródłowy, zbudowany według precyzyjnie określonych reguł. Projektant i progamista może zawsze w końcowym etapie poprawić bądź zmienić ten kod zgodnie z indywidualnymi potrzebami. Elastyczność taka pozwala na rozbudowę narzędzia o elementy stworzone w kolejnych aplikacjach. Sądzimy, że ta rozbudowa narzędzia będzie trwała aż do chwili, gdy zauważymy, że przyjęte przez nas zasady budowy narzędzia są tylko podzbiorem tych, które nareszcie zrozumieliśmy. Wtedy zabierzemy się do tworzenia nowego, dużo lepszego, genialnego tworu, który zrobi za nas absolutnie wszystko.

6. Literatura

1. Systems Application Architecture. First Edition. December 1987

Jadwiga Kowalska
Uniwersytet Szczeciński
Instytut Cybernetyki Ekonomicznej i Informatyki

METODY WSPOMAGANIA PROJEKTOWANIA STRUKTUR WG KRYTERIUM WSPÓŁZALEŻNOŚCI

Na ubiegłorocznej szkole wiosennej PTI w Świnoujściu wygłoszono dwa referaty dotyczące wspomaganego komputerem projektowania systemów informatycznych [1] [2].
Pierwszy z nich dotyczył metodologii projektowania systemów informatycznych w warunkach stosowania oprogramowania wspierającego to projektowanie. Drugi natomiast omawiał strukturę systemu wspomagania projektowania SI , a więc narzędzia wykorzystywanego we wcześniej wspomnianej metodologii.

Niniejszy referat stanowi uszczegółowienie fragmentu metodologii, a także systemu wspomagania projektowania w części związanej ze stosowaniem metod podziału zbioru obiektów zorientowanych na relacje.

1.Podejście do projektowania struktur.

W metodologii przyjęto podejście do projektowania SI zorientowane na struktury przyjmując występowanie struktur: informacyjnej, funkcjonalnej, technicznej, przestrzennej i organizacyjnej. Zaprojektowanie ich opiera się o dane zawarte w bazie danych systemu wspomagającego z wykorzystaniem banku metod. Dane te to informacje wprowadzone przez projektanta systemu zgodnie z przyjętą systematyką. Z banku metod projektant dobiera do projektowanego zagadnienia odpowiednie metody.

We wspomnianym wcześniej referacie omówiono bliżej projektowanie struktury funkcjonalnej, wskazując na sposoby definiowania składowych oraz kroki prowadzące do uzyskania projektu. Jak stwierdził autor [1] "celem zabiegów jest optymalny podział sieci struktury funkcjonalnej na podsystemy - podział

optymalny ze względu na przyjętą strategię podziału i sytuację.

Przyjmuje się dwie strategie podziału :

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

Reguły decyzyjne podziałów odpowiednio brzmią:

1\ łącz w podsystemy funkcjonalne zadania (algorytmy,procesy) najbardziej do siebie podobne,

2\ łącz w podsystemy funkcjonalne zadania (algorytmy,procesy) najbardziej ze sobą powiązane - minimalizując powiązania między podsystemami.

Sytuacje podaje się przez stopień ustrukturalizowania:

1\ sytuacja dobrze ustrukturalizowana : znana jest ilość podsystemów funkcjonalnych i wymagana w nich ilość zadań (procesów,algorytmów),

2\ sytuacja średnio ustrukturalizowana : znana jest tylko ilość podsystemów,

3\ sytuacja słabo ustrukturalizowana : nic nie jest znane a priori."

Przytoczony wyżej fragment referatu, chociaż odwołuje się do projektowania struktury funkcjonalnej, może znaleźć zastosowanie do dekompozycji innych zbiorów obiektów. Obiektem może być zarówno zadanie jak i proces, algorytm, funkcja, stanowisko.

Niezależnie zatem, która ze struktur jest przedmiotem projektowania, można wykorzystać dla uzyskania podziału zbioru na podzbiory (systemu na podsystemy, struktury na podstruktury) jedną z metod przechowywanych w banku metod. Wybór zdeterminowany jest wystąpieniem jednej z wcześniej wymienionych sytuacji.

Należy zwrócić uwagę, że podział wg drugiej reguły decyzyjnej tzn. kryterium relacji między obiektami, przyjmuje za przedmiot analizy związki między obiektami, a nie same obiekty. Wynikiem natomiast są grupy obiektów.

2. Budowa modelu struktury obiektów.

Funkcjonowanie pewnego systemu odbywa się poprzez realizowanie funkcji cząstkowych przez obiekty systemu. Obiekty połączone są

ze sobą siecią połączeń, umożliwiającą przekazywanie informacji między elementami, a tym samym realizowanie zadań funkcjonalnych warunkujących pracę całego systemu.

Wydzielenie z całego zbioru obiektów podzbiorów funkcjonalnych przebiega zwykle pod pewnymi warunkami ograniczającymi sposób podziału. Każdy z podsystemów może pomieścić tylko pewną liczbę obiektów. Podobnie każdy podsystem dysponuje pewną, zadaną liczbą powiązań funkcjonalnych i informacyjnych między elementami podsystemu a pozostałymi obiektami.

Ponieważ, nawet niezbyt liczny zbiór obiektów, może być podzielony wg różnych kryteriów, każdy podział daje się scharakteryzować:

- liczbą podsystemów (podgrup obiektów), przy czym może być to określenie minimalnego podziału,
- zawartością podsystemów, tj.wskazaniem obiektów, które mają się znaleźć w określonych podsystemach (zalążki podsystemów)
- mocą podsystemów tj. max liczebnością podsystemu.

Parametry te charakteryzują się dużą zmiennością, pozwalając w zależności od kombinacji uzyskiwać różne podziały. Stanowią więc one element decyzyjny, zarówno w wymiarze ilościowym jak i jakościowym, podejmowany każdorazowo przez projektanta systemu. Tworząc dowolne kombinacje parametrów z uwzględnieniem stopnia ustrukturalizowania sytuacji , możliwe jest uzyskanie wielu różnych podziałów. Wybór jednego z nich jako formy realizacyjnej dokonuje sam projektant, kierując się intuicją i doświadczeniem.

Najczęściej jednak podział podporządkowany jest wymogom: minimalizacji liczby podsystemów oraz minimalizacji powiązań zewnętrznych między podsystemami. Dąży się zatem do umieszczenia w każdym podsystemie możliwie dużej liczby obiektów mocno ze sobą powiązanych.

Przedmiotem podziału zbioru obiektów jest zatem struktura zbioru, a dokładniej model tej struktury, który po zastosowaniu odpowiednich metod przekształcany jest w model wynikowy, optymalny z punktu widzenia kryterium podziału.

Konstruowanie modelu struktury powinno uwzględniać:

- występowanie wzajemnych zależności między dwoma lub więcej obiektami (1:n, 1:1, n:1, n:n),

- kierunek lub przebieg wzajemnych zależności,
- moc lub intensywność tych zależności.

Model zgodny z dwoma pierwszymi postulatami ma postać macierzy binarnej lub sieci logicznej, która dla potrzeb algorytmu przekształcana jest w macierz. Dołączenie trzeciego postulatu daje model w postaci macierzy różnowartościowej.

3. Metody podziału.

Znalezienie algorytmu optymalnego podziału systemu na podsystemy, uwzględniającego zadane kryteria i ograniczenia oraz efektywnego na czas trwania nie jest proste. Algorytm taki musi spełniać, w odniesieniu do zbioru obiektów, postulaty:

- rozłączności
  element może być przypisany tylko do jednego podsystemu
- pełności
  suma elementów w podsystemach daje zbiór wejściowy
- mocy
  liczba elementów w każdym podsystemie jest nie mniejsza niż 2 i nie większy niż n-2

Zgromadzone w banku metody podziału zorientowane na relacje tworzą cztery grupy:

- metody heurystyczne,
- metody iteracyjne,
- metody konstrukcyjne,
- metody generowania zespołów minimalnych.

Zarówno ograniczona objętość referatu jak też i czas przeznaczony na jego wygłoszenie nie pozwalają na omówienie każdej z grup, tym bardziej, że omówienie to wymagałoby wprowadzenia szeregu formuł matematycznych. Dlatego też ograniczę się do przedstawienia zasad metod iteracyjnych, traktując to jako ilustrację zagadnienia.

Metody iteracyjne mają zastosowanie w sytuacjach dobrze ustrukturalizowanych, kiedy dana jest struktura podziału i stopień segmentacji. Model struktury jest siecią logiczną traktowaną jako multigraf bez pętli, w którym wierzchołki

reprezentują obiekty (elementy sieci),krawędzie zaś odpowiadają połączeniom między elementami. Model ten przekształcany jest na postać macierzy binarnej lub różnowartościowej.

Algorytm zakłada rozłożenie macierzy wejściowej na bloki przekątniowe, których wymiary są określone przez zadaną strukturę podziału. Bloki przekątniowe reprezentują przy tym bloki zadaniowe, podczas gdy w blokach nieprzekątniowych ujęte są zależności międzyblokowe. Celem jest takie przekształcenie, żeby suma elementów wewnątrz bloków przekątniowych była możliwie duża, albo aby suma elementów bloków nieprzekątniowych była możliwie najmniejsza. Algorytm ten ulepsza pewien zadany wstępny podział grafu, zmieniając położenie pary wierzchołków znajdujących się w różnych podzbiorach, jeżeli prowadzi to do zwiększenia liczby krawędzi wewnątrz podzbioru.

Wykorzystanie którejkolwiek z metod banku metod wymaga od projektanta określenia:

- jednorodnej grupy obiektów i opisanie jej w banku danych,
- relacji między obiektami,
- warunków wstępnych dostosowujących metodę do stopnia ustrukturalizowania sytuacji.

4. Podsumowanie.

Niezależnie od tego czy przedmiotem odniesienia są struktury matematyczne, rozwiązania technologiczne, projektowanie obwodów elektronicznych cy też projektowanie organizacji, we wszystkich przypadkach, jako kryteria rozgraniczenia i tworzenia podsystemów wykorzystuje się wzajemne zależności pomiędzy obiektami. Tym samym, dla uzyskania podziału możliwe jest wykorzystanie dowolnej z metod banku.

Bibliografia :

[1] Olejniczak W.:Globalna metoda projektowania systemów informatycznych, Druga Wiosenna Szkoła PTI, Świnoujście 1989
[2] Szydłowski I.: Metoda wspomaganego projektowania systemów informatycznych, Druga Wiosenna Szkoła PTI, Świnoujście 1989

Marian Niedźwiedziński
Ktedra Informatyki-Uniwersytetu Łódzkiego

OCENA ZAMIERZEŃ INFORMATYZACYJNYCH PRZEDSIĘBIORSTWA

Podejmując problem oceny zamierzeń informatyzacyjnych przedsiębiorstwa, chciałbym przede wszystkim podać przesłanki zajęcia się tym zagadnieniem.

Po pierwsze sądzę, że ocena zamierzeń informatyzacyjnych powinna być jednym z podstawowych elementów w ramach strategicznego myślenia o komputeryzacji w przedsiębiorstwie. Przez strategiczne myślenie rozumiem, w tym wypadku, długofalową politykę z jasno i jednoznacznie określonymi celami, której efektem powinna być wzajemna koherentność, komplementarność i spójność działań w dziedzinie informatyzacji. Jedynie w warunkach istnienia takiej długofalowej polityki, działania odcinkowe będą charakteryzowały się wystarczającą dojrzałością i skutecznością - aby doprowadzić, z czasem, do jakościowych zmian w systemie informacyjnym przedsiębiorstwa, a w istocie o to chodzi inwestorom. W tym miejscu z dużym prawdopodobieństwem można stwierdzić, że powszechna mikrokompuretyzacja lat 80-tych była raczej żywiołem , stymulowanym modą komputerową , prowadzącym do przypadkowych rezultatów - niż działaniem w pełni przemyślanym, konsekwentnie prowadzącym do osiągnięcia jasno sprecyzowanych celów strategicznych. Stąd, jak sądzę, rozczarowanie i zniechęcenie części użytkowników, którzy po krótkim okresie fascynacji nową techniką zaczęli dystansować się od nieprzemyślanych, niedopracowanych systemów.

Drugą przesłanką, uzasadniającą potrzebę dokonywania oceny zamierzeń informatyzacyjnych w przedsiębiorstwie jest radykalna reforma gospodarcza." Awansowała" ona przedsiębiorstwo, będące poprzednio jednostką realizującą określone zadania rzeczowe, do rangi mikrosystemu ekonomicznego. Stwarza to z kolei konieczność innego spojrzenia na komputeryzację. Nie należałoby jej obecnie traktować wyłącznie jako czynnika usprawniającego proces przetwarzania danych, ale szerzej - jako czynnik wzrostu efektywności działania przedsiębiorstwa.

Nowa sytuacja stawia nowe wymagania w stosunku do informatyków. Muszą oni umieć posługiwać się rachunkiem ekonomicznym, który umożliwi im udowodnienie, że proponowane inwestycje informatyczne są celowe i efektywne. Celem przedsięwzięcia informatycznego w warunkach reformy gospodarczej jest bowiem wzrost efektywności działania przedsiębiorstwa, a zaspokojenie potrzeb informacyjnych decydentów jedynie środkiem umożliwiającym w określonych warunkach osiąganie tego celu. Stąd deklaracja określonych potrzeb informacyjnych nie może być obecnie wystarczającym powodem podjęcia inwestycji komputerowej. Nie ma bowiem równoważności między deklarowanymi potrzebami informacyjnymi a obiektywnie osiągalnymi efektami, które ,dzięki zaspokojeniu tych potrzeb mogą być uzyskane.

Punktem wyjścia w prowadzonym rozumowaniu będzie refleksja nad czynnikami w ogóle determinującymi efektywność działania

przedsiębiorstwa i miejscem, które wśród nich zajmuje system informatyczny zarządznia (SIZ). Sensowność przyjęcia tak szerokiej perspektywy w spojrzeniu na problem efektywności informatyzacji podyktowana jest tym, iż ostatecznym celem wdrażania SIZ w przedsiębiorstwie jest właśnie wzrost efektywności działania tegoż przedsiębiorstwa oraz tym , iż efektywność ta jest osiągana w sferze działań realnych a nie tylko w sferze przetwarzania danch.

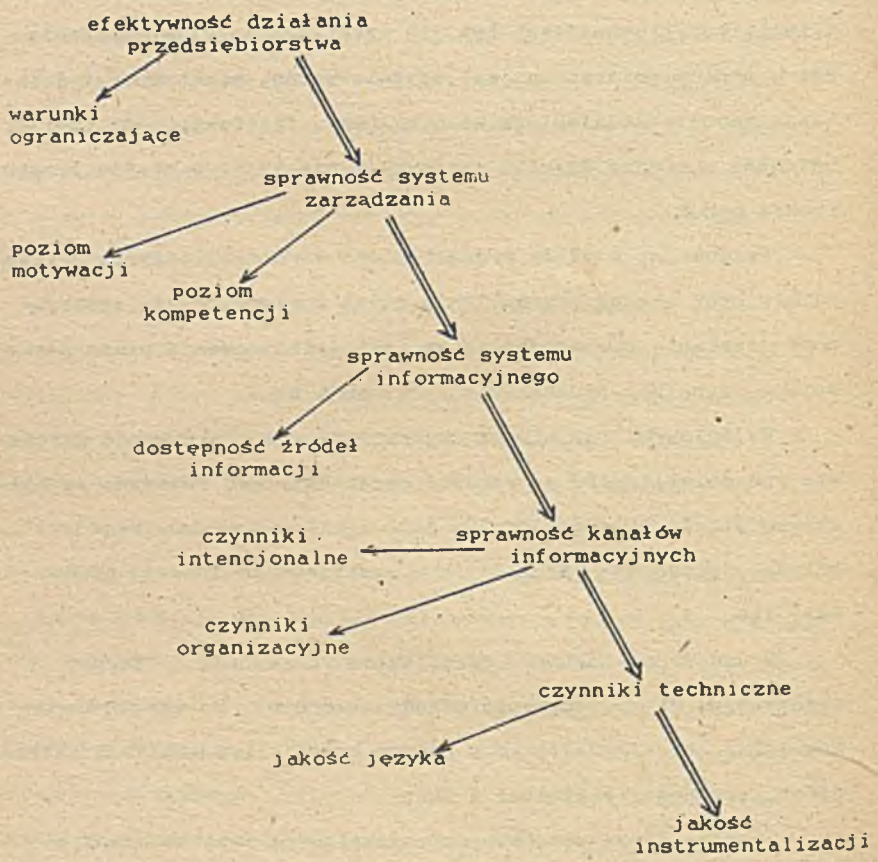
Reasumując poglądy przedstawione w różnych pozycjach literatury oraz własne przemyślenia można stwierdzić ,że efektywność działania przedsiębiorstwa jest deteminowania przez następujące czynniki. Przedstawia je schemat nr 1.

Wyjściowymi czynnikami determinującymi efektywność działania przedsiębiorstwa są warunki ograniczające, związane ze specyfiką danego przesiębiorstwa jako systemu produkcyjnego : branża , stosowana technologia , nowoczesność środków produkcji itp.

Druga grupa warunków ograniczających wynika z szeroko rozumianego stanu otoczenia przedsiębiorstwa. Do takich warunków można obecnie zaliczyć w naszym kraju : trudności zaopatrzeniowe, trudności płatnicze i inne .

Trzecia grupa warunków ograniczających jest związana z modelem gospodarowania realizowanym w państwie. Przy modelu parametrycznym , który jest obecnie wdrażany , należałoby do tej grupy warunków zaliczyć: ceny urzędowe, podatki , oprocentowanie kredytów itp.

GŁÓWNE DETERMINANTY EFEKTYWNOŚCI DZIAŁALNOŚCI PRZEDSIĘBIORSTWA



Źródło: opracowanie własne
/pogrubione strzałki wskazują główny kierunek diagnozy

Schemat nr 1

Wszystkie ww czynniki pozostają poza kontrolą decydentów w przedsiębiorstwie . Stąd określiłem je jako czynniki ograniczające tj. wyznaczające z góry osiągalny poziom efektywności działania przedsiębiorstwa.

Z kolei sprawność systemu zarządznia jest wyznaczana przez trzy następujące czynniki: pozim motywacji , poziom kompetencji oraz sprawność systemu informacyjnego .

Poziom motywacji determinuje sprawność systemu zarządzania w tym sensie ,iż pierwotnym warunkiem racjonalnych decyzji jest chęć ich podejmowania . Istotną sprawą jest zatem stworzenie systemu motywacyjnego w przedsiębiorstwie w najbardziej ogólnym sensie ,który powodowałby utożsamienie się celów indywidualnych pracowników z celami systemu ekonomicznego jakim jest przedsiębiorstwo . Reforma gospodarcza i wynikająca z niej ekonomizacjia działalności przedsiębiorstw stwarza przesłanki dla budowania takich koherentnych systemów motywacyjnych.

Poziom kompetencji determinuje sprawność systemów zarządzania w tym sensie , że niezbędnym warunkiem podejmowania racjonalnych decyzji jest posiadanie przez decydentów odpowiednich uzdolnień i umiejętności .

Sprawność systemu informacyjnego determinuje sprawność systemu zarządzania w tym sensie,że określa ona wielkość luki informacyjnej w procesie decyzyjnym.

Sprawność systemu informacyjnego jest funkcją dwóch czynników : dostepności źródeł informacji oraz sprawności kanałów informacyjnych.

Problem dostępności źródeł informacji wiąże się z niemożnością pozyskania pewnych danych spowodowaną bądź przyczynami metodologicznymi, bądź też świadomym utajnieniem pewnych informacji. Brak precyzyjnych i łatwych w użyciu metod pomiaru zjawisk socjo- i psychologicznych jest typowym przykładem uwarunkowanych metodologicznie ograniczeń w dostępności informacji. Zaś przykładem ograniczeń w dostępności informacji wynikłych ze świadomej polityki mogą być zjawiska utajnienia wielu informacji przed konkurentami m.in. cen w handlu zagranicznym. Bariera dostępności źródeł informacji stanowi obiektywne ograniczenie sprawności systemu informacyjnego w przedsiębiorstwie.

Z kolei, sprawność kanałów informacyjnych jest determinowana przez wymienione w schemacie nr 1 trzy grupy czynnikow.

Czynniki intencjonalne należy rozumieć jako ograniczenie sprawności kanałów informacyjnych, mające swe źródło w intencjach określonych pracowników .Na różnych szczeblach zarządzania mogą występować przypadki celowego wprowadzania w błąd decydentów poprzez świadome zniekształcanie przekazywanej informacji o pewnych zjawiskach. Przyczyny występowania intencjonalnych szumów w kanałach informacyjnych mogą być bardzo złożone. Należy do nich zaliczyć przede wszystkim : obowiązujący w przedsiębiorstwie system nagród i kar , system oceny i awansowania pracowników, konflikty interpersonalne , istnienie grup nieformalnych wzajemnie konkurujących ze sobą itp.

Czynniki organizacyjne determinują sprawność kanałów informacyjnych w tym sensie , że pewne rozwiązania organizacyjne

w przesiębiorstwie mogą wpływać na korzyść lub na niekorzyść tego systemu. Przykładowo , smukła, wieloszczeblowa struktura oranizacyjna może być przyczyną znacznych opóźnień i zniekształceń w przekazywaniu informacji zainteresowanym decydentom. Opóźniena mogą w tym wypadku , wynikać z nadmiernie wydłużonej drogi przepływu oraz zbyt dużej liczby punktów zatrzymania informacji - co związane jest z pewnym biurokratycznym rytuałem, nie zawsze uzasadnionym merytorycznie. np. opatrywanie dokumentów wieloma podpisami , stęplami, dekretacjami itp. Zniekształcenia treściowe natomiast, mogą mieć swoje źródło w niewłaściwej redukcji informacji / selekcji, agregacji itp./ dokonywanej w poszczególnych komórkach organizacyjnych przedsiębiorstwa. Wspomniana redukcja informacji jest koniecznością w warunkach rozbudowanych pionowo centralistycznych systemów zarządzania.

Trzecią determinantą drożności kanałow informacyjnych są , przedstawione w schemacie nr 1 , czynniki techniczne.

Jakość języka określa drożność kanałów informacyjnych orgazacji w tym sensie, że im bardziej sformalizowany język - tym mniejsza liczba symboli potrzebnych dla przekazania określonej treści. Zmniejszenie liczby symboli powoduje zaś zwiększenie pojemności kanałów informacyjnych. Potrzeba zawartej, jednoznacznej formy identefikacji i rejestracji zdarzeń gospodarczych doprowadziła, już przed wiekami, do wykształcenia się specyficznego języka rachunkowości.Jest to klasyczny przykład języka, który w ściśle sformalizowany sposób umożliwia komunikowanie się członków organizacji. Formalizacja języka jest jednak ograniczo-

na do przypadków , gdy mamy do czynienia z informacją kwantyfikowalną. W przypadkach kiedy informacja ma charakter niemierzalny potrzebne są nieformalne języki.Należy w tym miejscu dodać, że efektywne zastosowanie techniki komputerowej jest uwarunkowane uprzednim opracowaniem i wdrożeniem sformalizowanego języka . w organizacji . Najbardziej typowe elementy takiego języka stanowią - baza indeksowa i baza normatywna.

Instrumentalizacja ,albo inaczej wyposażenie / uzbrojenie/ techniczne, determinuje drożność kanałów informacyjnych w tym sensie, że zastosowanie nowoczesnych środków do zbierania,przetwarzania, przechowywania i przekazywania informacji umożliwia znaczne zwiększenie przepustowości kanałów informacyjnych oraz mniejsza pracochłonność przetwarzania danych. I odwrotnie - brak wydajnej technologii informacyjnej tworzy swoistą barierę instrumentalną , która ogranicza techniczną sprawność kanałów informacyjnch.

Wniosek, który nasuwa się w związku z przedstawionym wywodem jest następujący. Aby SIZ był ekonomicznie efektywny musi istnieć sytuacja, w której krytycznym czynnikiem efektywności działania przedsiębiorstwa jest niesprawność jego systemu zarządzania wynikająca z niedostatecznej bądź niewłaściwej instrumentalizacji kanałów informacyjnych.

W związku z tym, iż w wielu wypadkach podstawowymi czynnikami deteminującymi efektywność działania firmy są czynniki pozainformacyjne, należy mieć świadomość ograniczonej roli komputera w ekonomizacji funkcjonowania przedsiębiorstw. Można

stwierdzić, iż nawet najdoskonalsza instrumentalizacja nie rozwiązuje problemu luki informacyjnej w zarządzaniu . Błędem jest zatem "komputerocentryczne" widzenie problemów informacyjnych przedsiębiorstwa . Sprawność systemu informacyjnego zależy bowiem nie tylko od technologii informacyjnej , lecz również od wielu innych czynników m.in. ludzkich , organizacyjnych i in.

Mając świadomość roli czynników determinujących sprawność systemu zarządzania można stwierdzić , że zastosowanie technologii komputerowej może przyczynić się do wzrostu efektywności działania przedsiębiorstwa gdy :

1/ efektywność działania przedsiębiorstwa jest ograniczona w pierwszej kolejności przez sprawność systemu zarządzania.

2/ sprawność systemu zarządzania limitowana jest w pierwszym rzędzie przez sprawność systemu informacyjnego .

3/ sprawność systemu informacyjnego jest ograniczana przede wszystkim przez drożność kanałów informacyjnych ,

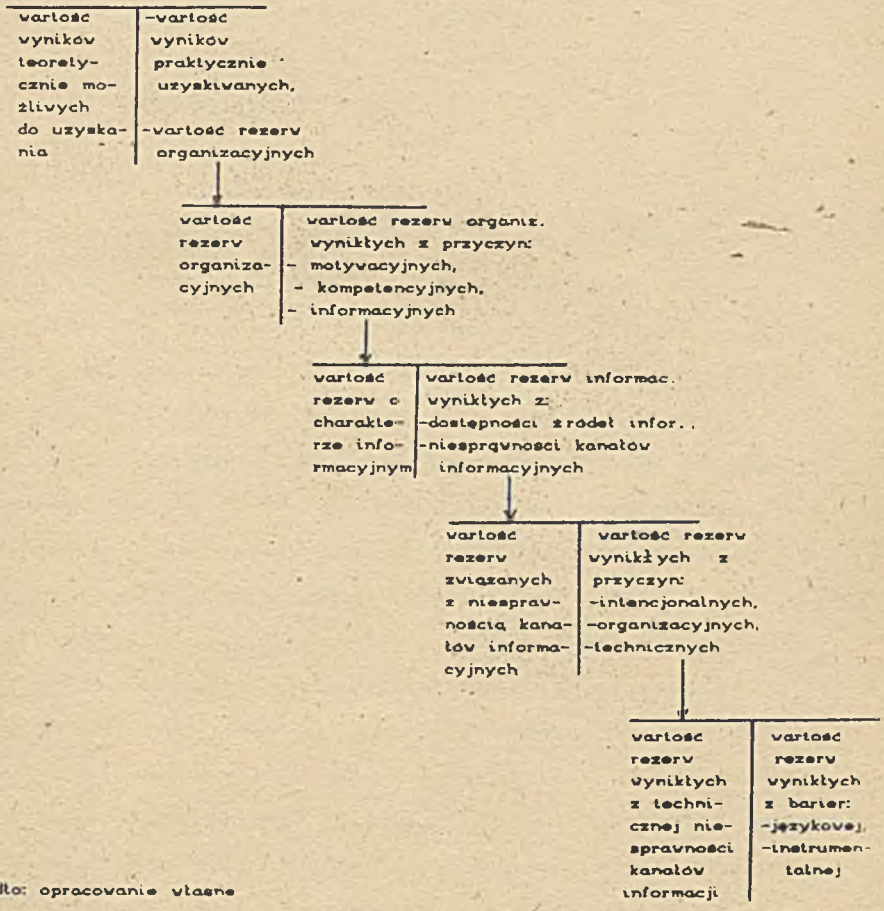
4/ niewystarczająca drożność kanałów informacyjnych wiąże się w pierwszym rzędzie z czynnikami technicznymi .

5/ czynnikiem technicznym ograniczającym w decydujący sposób drożność kanałów informacyjnych jest zbyt mała wydajność technologii informacyjnej ,czyli niewłaściwa bądź niedostateczna instrumentalizacja .

Obok rozważenia problemu czy w konkretnej sytuacji można liczyć na efekt ekonomiczny przedsięwzięcia informatycznego , istotne jest również , choćby przybliżone , oszacowanie wielkości

tego efektu. Należałoby w tym celu przeprowadzić badanie diagnostyczne , polegające na wyodrębnieniu tej części rezerw organizacyjnych przedsiębiorstwa , która związana jest z technologią informacyjną oraz wyrazić ją w formie wartościwej. Stosując technikę bilansową , można kolejne kroki tej diagnozy przedstawić jak na schemacie nr 2. Szczegółowe omówienie metodologicznych aspektów badania przekracza zakres niniejszego referatu. Zainteresowanym czytelnikom proponuję zapoznanie się z moją książką wydaną w br. przez PWE w ramach serii " Informatyka w praktyce " pt. "OCENA ZAMIERZEŃ INFORMATYZACYJNYCH PRZEDSIĘBIORSTWA". W tym miejscu pragnę jedynie zaznaczyć, iż prezentowane badania diagnostyczne zostały przeze mnie praktycznie przeprowadzone w kilku przedsiębiorstwach i stały się ważnym czynnikiem inspirującym przebieg prac projektowych i wdrożeniowych.

PROCEDURA DIAGNOZY INFORMACYJNEJ PRZEDSIĘBIORSTWA



źródło: opracowanie własne

Schemat nr 2

Zaawansowana inżynieria systemów na CASE'89 w Sztokholmie

Ireneusz Szydłowski
Instytut Cybernetyki Ekonomicznej i Informatyki
Uniwersytet Szczeciński
ul. Mickiewicza 66, 71-101 Szczecin

W dniach 9-11 maja 1989 roku odbyła się konferencja 'The first Nordic Conference on Advanced Systems Engineering' zorganizowana przez Szwedzki Instytut Projektowania Systemów (Swedish Institute for Systems Development) i Szwedzkie Towarzystwo Informatyczne (The Swedish Society for Information Processing). Miejscem konferencji była Kista (dzielnica Sztokholmu), gdzie znajduje się szwedzkie centrum przemysłu elektronicznego i informatycznego. Konferencja CASE 89 poświęcona była komputerowemu wspomaganiu tworzenia systemów informatycznych i odbywała się pod hasłem 'Mosty między nauką, użytkownikami i przemysłem'. Równolegle z konferencją odbywała się ekspozycja komputerowych narzędzi do tworzenia systemów informatycznych (wystawcami były czołowe firmy szwedzkie, zachodnioeuropejskie i amerykańskie). W konferencji uczestniczyło ok. 200 osób ze Szwecji, Europy Zachodniej, Japonii i Stanów Zjednoczonych. W ciągu 3 dni wygłoszono 43 referaty dotyczące projektowania systemów informatycznych, narzędzi

(systemów) wspomagających tworzenie systemów informatycznych, stanu obecnego i perspektyw komputerowego wspomagania tworzenia systemów informatycznych. Referaty prezentowane były równolegle w dwóch sekcjach o różnych profilach:

-niebieskiej - metodologie tworzenia systemów informatycznych, organizacja procesu tworzenia SI;

-żółtej - narzędzia wspomagające tworzenie SI.

Tematyka każdego dnia określana była przez wspólny referat programowy (sekcja różowa) wygłaszany przez wybitne osobistości świata informatyki :

1 dzień: 'Przyszłość modelowania - dlaczego obecne narzędzia wspomagające bazują głównie na 20 letnich metodach i technikach' - prof. Colette Rolland z Sorbony;

2 dzień: 'Przyszłość narzędzi wspomagających tworzenie systemów informatycznych' - Franz van Assche z James Martin Association;

3 dzień: 'Organizacyjne implikacje środowiska czwartej generacji' - Simon Holloway z DCE Information Management Consultancy.

W trzecim dniu po ostatnich referatach w obu sekcjach organizatorzy przeznaczyli ok. 1 godziny na dodatkową dyskusję i prezentację wypowiedzi spoza programu. Uroczyste zakończenie zostało poprzedzone ok. 20 minutową prezentacją slajdów wykonanych podczas trwania konferencji.

Wśród wielu interesujących prezentacji, obok referatów programowych, moim zdaniem na szczególną uwagę zasługują dwa wystąpienia referentów zza oceanu.

Referat P.D.Mc Daniela z Kwatery Głównej Armii Stanów Zjednoczonych dotyczył zastosowania pakietu PSL/PSA do planowania systemu informacyjnego. Tym co wywarło silne wrażenie była nie tyle zawartość merytoryczna, lecz sposób prezentacji. Warunki techniczne stworzone przez organizatorów umożliwiały rzutowanie na ekran o przekątnej ok. 7 metrów zawartości ekranu monitora. Zapewniło to doskonałą widoczność 'spektaklu' stworzonego przez referenta. Cały referat był wspomagany właściwie przez film animowany, który uwypuklał wszystkie ważne momenty i współgrał z wypowiedziami P.D.Mc Donalda. Zdaniem wielu uczestników konferencji była to modelowa prezentacja. Na marginesie warto podkreślić, że podczas gdy u nas pakiet PSL/PSA przemknął jak efemeryda na przełomie lat siedemdziesiątych i osiemdziesiątych i tylko niewielu o nim pamięta, to w USA jest on wciąż z powodzeniem stosowany.

Drugi referat amerykański, prezentowany przez M.Waltera z firmy IBM, nie był wydarzeniem artystycznym, natomiast przedstawione koncepcje poruszyły całą salę. W minionych latach powstało bardzo wiele narzędzi wspomagających wszystkie etapy tworzenia systemu informatycznego - od analizy potrzeb po wdrażanie. Bardzo często zdarza się, że potrzeby przedstawione przez użytkownika nie są ostatecznie realizowane przez system informatyczny, mimo że do jego tworzenia użyto wielu doskonałych narzędzi wspomagających. Zdaniem M.Waltera (IBM ?), główną tego przyczyną jest zmiana specyfikacji problemu w kolejnych fazach projektowania. Zaradzić temu ma nowa koncepcja IBM, będąca rozwinięciem SAA - Systems Application Architecture. Aby proces

projektowania miał właściwy przebieg spełnione muszą być trzy warunki:

- musi istnieć centralna składnica (central respository), która zapewni sterowanie i rozdział informacji we wszystkich fazach procesu projektowania,
- musi istnieć kompletny zestaw narzędzi dla wszystkich etapów procesu projektowania, umożliwiający uzyskiwanie informacji z centralnej składnicy,
- musi istnieć efektywny, łatwy w użyciu, konsekwentny (logiczny) interfejs do tego środowiska - ten wymóg jest najważniejszy.

Obecnie IBM kończy realizację swojej koncepcji. Wykonanych (adaptowanych) zostało kilkanaście narzędzi wspomagających. Centralna składnica realizowana jest w konwencji E-R-A (obiekt-relacja-cecha). Dostępne interfejsy umożliwiają wieloaspektową analizę projektu, którą przeprowadzać mogą zarówno użytkownicy jak i projektanci.

Wydaje się, że kierunek wyznaczany przez koncepcję BIG BLUE będzie realizowany w latach 90-tych również przez niezależnych producentów, gdyż chyba trudno przecenić efekty przedsięwzięcia, które ma zapewnić użytkownikowi realizację jego oczekiwań.

Inne referaty cieszyły się tym wiekszym zainteresowaniem, im bardziej były związane z praktyką. Prezentowano zarówno koncepcje wspomagania kompleksowego, jak i wycinkowego. Dominującym było przekonanie, że proces projektowania wymaga całego szeregu narzędzi wspomagających, które powinny być w jak największym stopniu zintegrowane, chociaż nie brakowało koncepcji luźnego

zestawu narzędzi wspomagających.

Świetnym uzupełnieniem konferencji były pokazy systemów wspomagających firm, które znane są na rynkach światowych od wielu lat. Standardem jest praca w trybie graficznym z wykorzystaniem myszki lub digitizera. Tylko w nielicznych systemach wspomagających użytkownik jest atakowany feerią barw - dominuje oszczędne operowanie kolorami. Produkty są oferowane do pracy pod kontrolą praktycznie dowolnego mikrokomputerowego systemu operacyjnego. Każdy zainteresowany traktowany był niezwykle poważnie, zapraszany do samodzielnego sprawdzenia narzędzia lub oglądnięcia bardzo konkretnych demonstracji. Największym powodzeniem cieszyły się produkty:

- Excelerator - zintegrowany pakiet analizy systemowej firmy Index Technology,
- pakiet narzędzi inżynierii systemów James Martin Association,
- zestaw narzędzi firmy KnowledgeWare wspomagający wszystkie fazy projektowania systemu,
- pakiety firmy EPOC System - GraphDoc, Modellator i RUTH - wspomagające analizę systemową.

Inni wystawcy byli jakby w cieniu 'tych wielkich', jednak również ich produkty odznaczły się profesjonalnym wykonaniem.

W sumie konferencja CASE'89 była świetnym przeglądem teorii i praktyki komputerowo wspomaganego tworzenia systemów informatycznych, pokazującym jak długą drogę mamy jeszcze przed sobą, jak wiele jest jeszcze do zrobienia. Tegoroczna konferencja CASE'90 stawia sobie za cel pogłębianie związków nauki z

przemysłem. Sześć spośród czternastu 1.5 godzinnych sesji poświęconych jest praktyce oraz komunikatom z doświadczeń w przemyśle. Tematyka CASE'90 obejmuje:

- narzędzia CASE,
- CASE 'shells' - metanarzędzia,
- modelowanie konceptualne i modelowanie narzędzi,
- zarządzanie zasobami informacji,
- wykorzystanie systemów bazujących na wiedzy w procesie projektowania,
- zaawansowane systemy (metody) analizy i projektowania,
- modele projektowania systemów (cyklu życia),
- prototypowanie.

Na zakończenie warto chyba poświęcić trochę uwagi polskiemu rynkowi CASE. O istnieniu zapotrzebowania na narzędzia wspomagające projektowanie systemów informatycznych nie trzeba przekonywać. Upowszechnianie się (mikro-) komputerów implikuje zapotrzebowanie na systemy informatyczne, które w wiekszości przypadków muszą być wykonywane 'od początku'. Działanie tradycyjnym sposobem czyni proces tworzenia systemu czaso- i pracochłonnym. Istotnym usprawnieniem mogą być narzędzia (systemy) wspomagające. Skąd je brać ? Zakup na Zachodzie to wydatek rzędu kilkunastu, kilkudziesięciu tysięcy dolarów na oprogramowanie i jeszcze trochę na odpowiedni sprzęt. Ponadto oprócz bariery językowej istnieje jeszcze bariera poznawcza i 'adekwatności' - profesjonalne, zachodnie pakiety wspomagające zwykle bazują na metodologiach, które jakby nie przystają do polskiej rzeczywistości, wymagają studiów teoretycznych przerastających

'zwykłego' informatyka. Jednak te trudności nie powinny zniechęcać i na pewno celowymi są działania zmierzające do przeniesienia rozwiązań zachodnich na rynek krajowy.

Inną metodą jest budowanie narzędzi wspomagających w kraju i bazowanie na metodologiach (podejściach) adekwatnych do warunków krajowych. Prezentowana na III Wiosennej Szkole PTI metodologia FACTOR i związany z nią system FACTOR są próbą wspomagania procesu projektowania systemów informatycznych (informacyjnych) w oparciu o podejścia, które dominują w narzędziach zachodnich wspomagających tę klasę zagadnień. Projektant nie jest krępowany żadną szczegółową metodologią (chociaż FACTOR zakłada podejście obiektowe, projektowanie poprzez struktury, charakterystyczne mechanizmy opisu) i może realizować własną sieć działań. Wspomagane są zarówno prace analityczne jak i typowo ewidencyjne, dokumentacyjne. Możliwe jest tworzenie różnych zestawień będących podstawą dokumentacji technicznej czy eksploatacyjnej. Ułatwione jest płynne przejście do projektu systemu nowej generacji. Wydaje się, że tego rodzaju systemy CASE mogą znaleźć swoje miejsce w polskiej praktyce informatycznej i zapełnić lukę, która zaczyna się niebezpiecznie powiększać.

Zdzisław Szyjewski
Uniwersytet Szczeciński

Komputerowe wspomaganie dokumentowania tworzenia systemów informatycznych

1. Wprowadzenie

Dokumentacja produktów informatycznych pełni dwie zasadnicze funkcje w cyklu życia systemu informatycznego. Pierwsza funkcja, to sposób komunikowania się zespołów wykonawczych realizujących poszczególne, kolejne etapy prac nad systemem. Druga, to opis wykonanych prac na potrzeby pielęgnacji systemu i jego prawidłowego użytkowania.

Sposób prowadzenia dokumentacji, jej czytelność, kompletność, przejrzystość stanowią w dużej mierze o jakości systemu. Dokumentacja jest elementem umożliwiającym zapoznanie się z systemem i jego oceną, która bardzo często decyduje o zakupie lub nie, o przyjęciu do stosowania lub rezygnacji z systemu. Dokumentacja jest jedynym zewnętrznym obrazem systemu, pozwalającym na wydanie opinii o nim. Wszystko to świadczy o dużej wadze, jaką należałoby przykładać do wykonania dokumentacji. A jak jest w praktyce?

2. Tradycyjne dokumentowanie systemow

W praktyce informatyka, dokumentowanie wykonanych czynności najczęściej stanowiło dodatkową mitręgę. W związku z tym dokumentacja jeśli w ogóle jest, to stanowi niekompletny zbiór, różnorodnych co do stylu i dokładności, nie zawsze dobrze skompletowanych instrukcji i opisów. Próbą formalizacji i wymuszenia niezbędnej zawartości informacyjnej dokumentacji były standardy. Standardy dokumentacyjne dostarczały informatykowi formularz, którego wypełnienie stanowiło dokumentację wykonanej pracy. Stosunkowo wygodna forma tworzenia dokumentacji ma kilka wad. Jedną z nich jest niemożność przystosowania zawartości formularza do konkretnych potrzeb dokumentowanej pracy. Często okazywało się, że rubryki na formularzu są zbędne, gdyż nie mają zastosowania w konkretnym przypadku. Brak natomiast wielu istotnych dla opisywanego projektu zagadnień zubażał dokumentację.

Cichym założeniem stosowania formularzy było wykorzystywanie ich po wykonaniu czynności dokumentowanej, co powodowało, że rozumiano to jako pracę dodatkową nie wpływającą na sam proces tworzenia systemu. Takie stanowisko powodowało, że dokumentacja na formularzu była wykonywana niedokładnie i bez należytej

staranności, gdyż nie stanowiła dokumentu roboczego a jedynie opis wykonanej już pracy.

Powodem takiego podejścia są trudności w operowaniu formularzem w trakcie wykonywania opisywanej czynności. Zmiany i modyfikacje wprowadzane w trakcie prac, wymuszają ciągłe zmiany dokumentacji, co wiąże się z uciążliwymi pracami technicznymi, polegającymi na przerysowywaniu schematów i przepisywaniu wcześniej wykonanych opisów, najczęściej w wielu egzemplarzach. Stąd naturalnym staje się wykonanie dokumentacji wówczas, gdy opisywana praca jest gotowa i przekazana do użytkowania.

3. Zawartość informacyjna dokumentacji

Zestaw formularzy składający się na dokumentację systemu jest uzależniony od przyjętej metodyki tworzenia systemu [2]. Formularze te wymagają naniesienia na nie odpowiednich informacji, które opisują poszczególne etapy prac a w konsekwencji cały system. Z uwagi na to, że wykorzystywane są różne metodyki, ponadto nie zawsze konsekwentnie, zawartość informacyjna dokumentacji systemu zależy od przyjętej metodyki i wspomagającego ją zestawu formularzy.

Sytuacja taka powoduje, że różnorodne opisy systemów, dokumentacje poszczególnych etapów prac i całego systemu, stają się nieczytelne dla osób nieobeznanych z przyjętą metodyką. Taki sposób dokumentowania, bazujący na różnych zestawach formularzy, powoduje, że systemy są nieporównywalne i bardzo trudne do szybkiej i łatwej oceny bez wgłębiania się w szczegóły rozwiązań projektowo-programowych.

Brak obowiązujących standardów dokumentacyjnych, pozwalający na prowadzenie "radosnej twórczości" dokumentacyjnej, prowadzi do sytuacji, gdzie dokumentacja zatraca swoje naturalne cechy informacyjne a staje się dziełem samym w sobie, często niestety nieczytelnym dla osób spoza kręgu wykonawców.

Brak standardu informacyjnego zawartości dokumentacji powoduje, że pod podobne sformułowania i hasła, różni autorzy podkładają różne treści. Bałagan dokumentacyjny powoduje sytuacje gdy dwa systemy o tych samych nazwach są oferowane w jednym katalogu obok siebie, z cenami różniącymi się o kilka rzędów, co powoduje całkowitą dezinformację potencjalnego nabywcy, który najczęściej rezygnuje z zakupu z powodu braku wiarygodnej informacji o systemie [por.1].

Żeby być dobrze zrozumianym, nie nawołuje do stosowania jednej, jedynie słusznej metodyki z odpowiednim zestawem

formularzy, ale sądzę, że należałoby określić niezbędne minimum informacji, które powinny w miarę precyzyjnie opisywać system, jego funkcje, zakres, wymogi i inne parametry, tak aby można było łatwo porównać kilka systemów i ocenić ich przydatność bez potrzeby wnikania w szczegóły projektowe

4. Propozycja standardu dokumentacyjnego

W trakcie Pierwszej Wiosennej Szkoły PTI była prezentowana metodyka tworzenia systemów informatycznych z wykorzystaniem punktów węzłowych [3]. Najogólniej metodyka ta dzieli cykl życia systemu na poszczególne punkty węzłowe, które stanowią konkretne prace wykonywane w procesie tworzenia systemu. Podział ten wykonany jest na kilku poziomach szczegółowości, przy czym dwa najwyższe, w sensie ogólności, są niezmienne, a pozostałe mogą być dowolnie modelowane w zależności od specyfiki tworzonego systemu informatycznego.

Taki sposób tworzenia systemu, pozwala z jednej strony na zachowanie niezbędnego minimum standaryzacji, a z drugiej strony nie krępuje wykonawców, pozwalając im na takie dopasowanie metodyki, jak wymaga tego specyfika tworzenia systemu. Dwa standardowe poziomy podziału procesu na punkty węzłowe, gwarantują unifikacje postępowania oraz porównywalną zawartość informacyjną dokumentacji wytworzonej w procesie tworzenia systemu.

Metodyka nie jest wspomagana żadnym standardowym zestawem formularzy dokumentacyjnych, pozostawiając wybór ewentualnego ich stosowania wykonawcom i ich przyzwyczajeniom. Natomiast metodyka koncentruje się na zawartości informacyjnej dokumentacji, nie rozstrzygając formy graficznej i technicznej umieszczania tych informacji. Mówiąc inaczej metodyka proponuje standardowy spis treści dokumentacji nie wnikając w sposób jego wypełnienia.

Konsekwentne stosowanie takiego podejścia powoduje, że dokumentacje różnych systemów są podobne w swej konstrukcji a różnią się treścią. Jednolitość konstrukcji dokumentacji pozwala na konfrontacje odpowiednich zapisów dokumentacyjnych celem wyrobienia sobie opinii o interesujących nas szczegółach bez potrzeby wgłębiania się w różne zapisy dokumentacyjne często niespójne.

Każdy punkt węzłowy w proponowanej metodyce to grupa czynności wykonanych w procesie tworzenia systemu informatycznego odpowiednio udokumentowana. "Odpowiednio" nie odnosi się do szaty graficznej czy wymogów technicznych, tak istotnych w technice

formularzowej, ale do zawartości informacyjnej danego punktu węzłowego. Punkty węzłowe układają się w sieć działań i zawartości informacyjne wynikają z potrzeb informacyjnych dla wykonania kolejnego w szeregu działań, punktu węzłowego. W takim rozumieniu dokumentacja pełni funkcje nośnika komunikacji, pomiędzy wykonawcami kolejnych czynności w procesie tworzenia systemu. Wymagany jest niezbędny standard zawartości informacyjnej dla każdego punktu węzłowego ale istnieje pełna dowolność dla przekazania dodatkowych, uzupełniających informacji, które są istotne na danym etapie prac.

5. Komputerowe wspomaganie

W technice formularzowej, układ graficzny formularza stanowi ograniczenie wykonawcy ale wspomaga jego działanie zwalniając go z zastanawiania się w jakiej kolejności i jak technicznie zamieścić odpowiednie informacje. Trudności z operowaniem formularzami przy częstych zmianach i modyfikacjach, tak naturalnych w twórczym procesie tworzenia systemu, całkowicie eliminują tą niewątpliwą zaletę.

Żadna metodyka tworzenia systemu informatycznego nie zdobędzie dostatecznej popularności jeśli nie będzie wspomagana innymi środkami i narzędziami. Można zaryzykować twierdzenie, że stopień popularności metodyki jest wprost proporcjonalny do stopnia wspomagania.

Zaproponowana metodyka punktów węzłowych dostarcza komputerowe wspomaganie przy pomocy specjalnego programu. Szczegóły związane ze wspomaganiem komputerowym procesu tworzenia systemu są opisane w pracy [3]. Obecnie skoncentrujemy się jedynie na funkcji komputerowego wspomagania dokumentacji systemu.

W metodyce tej dokumentacja ma standardowy, zdefiniowany spis treści dla całego cyklu życia systemu. Spis treści na poziomie części i rozdziałów jest standardowy i nie może być modyfikowany. Głębszy podział na podrozdziały, paragrafy i punkty z podpunktami można dowolnie modelować w trakcie wykonywania projektu, zgodnie ze specyfiką. Oprogramowanie wspomagające, proponuje podział uniwersalny na podrozdziały, paragrafy itd., ale dostarcza środki dla rezygnacji z takiego podziału oraz daje możliwość wprowadzenia innego, własnego podziału. Rozwiązanie takie gwarantuje, że zawartość dokumentacji może być dowolnie definiowana ale nie może pominąć pewnego niezbędnego minimum, które stanowią niezmienialne części spisu treści.

Dla prowadzenia dokumentacji wykorzystywany jest standardowy edytor tekstów, dostępny w oprogramowaniu mikrokomputera IBM PC, wmontowany w oprogramowanie wspomagające. Wszystkie zalety, powszechnie znane, komputerowych edytorów tekstów są dostępne, co powoduje, że trudności modyfikowania, zmieniania dokumentacji przestają być mitręgą a mogą być łatwo i sprawnie wprowadzane. Utrzymanie jednego zbioru dokumentacji do edycji, gwarantuje porządek i jednoznaczność dokumentowania. Mankament polegający na tym, że brak jednego zawsze aktualnego źródła dokumentacji jest poważnym niedociągnięciem innych metodyk. W przypadku utrzymywania komputerowego zbioru dokumentacji problem ten przestaje istnieć, a tylko rozwiązania organizacyjne przyjęte w zespole wykonawców decydują o zasadach utrzymania kopii, wersji poprzednich i innych zabezpieczeniach przed zniszczeniem, niepowołanym dostępem itp.

Przy pomocy specjalnego oprogramowania, można swobodnie poruszając się po spisie treści zaszytym w oprogramowaniu, wypełniać pisząc z klawiatury, poszczególne fragmenty dokumentacji. Wypełnianie to, może się odbywać zgodnie z zaakceptowanym standardem spisu treści lub z odpowiednio zmodyfikowanym. Nie ma obowiązku całkowitego zakończenia opisu punktu węzłowego w trakcie jednego seansu. Specjalny wyróżnik ustawiony standardowo na zero musi być zmodyfikowany na 1, wówczas gdy wypełnianie danego punktu węzłowego zostało całkowicie zakończone. Wyróżnik ten wykorzystywany jest przez oprogramowanie do informowania, które prace nie zostały jeszcze zakończone a na pewno nieudokumentowane.

Wszystkie opisy wykonywane na mikrokomputerze są zapamiętywane w odpowiednim zbiorze komputerowym. Dostęp do zbioru mogą mieć wszyscy wykonawcy projektu, zarówno w trybie "czytaj", gdy chcą sięgnąć do źródłowej dokumentacji po potrzebne do pracy dane, jak również w trybie "pisz", gdy chcą wynik swojej pracy udokumentować na potrzeby innych członków zespołu. Oprogramowanie edytora udostępnia wygodne mechanizmy redagowania dokumentacji, wprowadzania zmian w istniejących opisach, wymazywania opisów, które starciły aktualność.

Cała dokumentacja systemu stanowi jeden obszerny zbiór komputerowy zawierający opisy wszystkich prac wykonanych przy tworzeniu systemu w całym cyklu życia, od inicjacji prac nad systemem aż do eksploatacji użytkowej. Taki duży, jeden zbiór bazowy dokumentacji prawie nigdy nie jest wykorzystywany cały.

ale najczęściej zgodnie z potrzebami wybierane są z niego fragmenty. Oprogramowanie wspomagające umożliwia wybór potrzebnych fragmentów z bazowego zbioru dokumentacji w postaci podzbioru komputerowego. Wyboru dokonujemy oznaczając początek i koniec fragmentu wybieranego, przesuwając się po spisie treści całej dokumentacji. Obrazuje to kopia ekranu załączona na końcu referatu.

Wybrany podzbiór dokumentacji zapisywany jest w zewnętrznym zbiorze danych, który może być poddany dalszej obróbce polegającej na dodaniu stron tytułowych, włączeniu innych tekstów czy rysunków. Przygotowany, zredagowany zgodnie z potrzebami zbiór może być drukowany lub dystrybuowany w postaci zbioru komputerowego na dyskietce. Należy zwrócić uwagę, że podzbiór wycięty z dokumentacji bazowej, ma charakter zbioru czasowego, wygenerowanego na potrzeby konkretnej pracy lub sytuacji. Wszelkie zmiany i modyfikacje mogą być wykonane tylko na zbiorze bazowym.

6. Zakończenie

Zdaje sobie sprawę, że przedstawione rozwiązanie w połączeniu z niezbyt obecnie doskonałym narzędziem programowym nie rozwiązuje trudnego i ważnego problemu dokumentowania prac przy tworzeniu systemu informatycznego. Podjęta próba może być przydatna w wielu krajowych zastosowaniach, głównie z powodu łatwości dostępu do oprogramowania[1)] oraz dostępność sprzętu IBM PC, na którym oprogramowanie jest implementowane. Ponadto dostarczone narzędzie może być pomocne dla zespołów wykonawczych z niewielkim doświadczeniem w tworzeniu systemów w pełnym cyklu życia. Pokazuje ponadto drogę postępowania w kierunku częściowej standaryzacji, która będzie konieczna gdy wprowadzona zostanie prawna ochrona oprogramowania.

Literatura

[1] Katalog kart informacyjnych Ogólnopolskich Targów Oprogramowania Softarg'88, Ośr. Post. Tech. Katowice 1988

[2] M. Klapper W. Migas, Dokumentowanie projektów i programów w Metodyki Projektowania systemów informatycznych zarządzania. Pierwsza Wiosenna Szkoła PTI, Świnoujście 1988

[3] Z. Szyjewski, Projektowanie systemów informatycznych zarządzania w: Metodyki projektowania op cit str 87-95

---

1) System stanowi produkt handlowy rozprowadzany przez Spółkę "PROGRES" w Szczecinie.

>>>>>>>>> Tworzenie dokumentacji systemu informatycznego <<<<<<<<<<<<<<<

Uniwersytet Szczecinski

Punkty wezlowe projektu systemu informatycznego

S080 PROJEKT STRUKTURY INFORMACYJNEJ
S081 Projekt wyjsc
S082 Projekt systemu kodow
S083 Projekt wejsc
S084 Struktura logiczna danych
S085 Projekt zbiorow podstawowych
S090 PROJEKT TECHNOLOGII PRZETWARZANIA
S091 Wybor technologii
S092 Podzial systemu na jednostki technologiczne
S093 Projekt schematow przetwarzania

Poczatek dokumentacji

S080 PROJEKT STRUKTURY INFORMACYJNEJ

Koniec dokumentacji

S085 Projekt zbiorow podstawowych

Nazwa zbioru

>>>>>>>>>>>>>>>>>>>>>>>>>>>SYSTEM .DOK<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

WPROWADZ NAZWE ZBIORU DLA TWORZONEJ DOKUMENTACJI

Techniki wspomagające integrację cyklu życia systemu

mgr Grzegorz Wapiński

Uniwersytet Gdański

Katedra Organizacji Przetwarzania Danych

ul. A. Czerwonej 119/121

81-824 Sopot

Proces tworzenia systemów informatycznych można podzielić na mniejsze elementy stosując różne kryteria. Jednym z nich może być podział na kolejne fazy tworzenia systemu - cykl życia. Innym kryterium jest wyodrębnienie poszczególnych technik stosowanych w procesie tworzenia systemów informatycznych, przy czym wyodrębnienie to nie musi pokrywać się z fazami cyklu życia systemu. Stanem porządanym jest takie powiązanie elementów wyoderębnionych przy użyciu dwu wymienionych poprzednio kryteriów, ażeby można było stworzyć spójny warsztat twórcy sytemów informatycznych. Jedną z takich propozycji jest system PAGEFIT, który ze względu na swoje właściwości może służyć jako model takiego warsztatu.

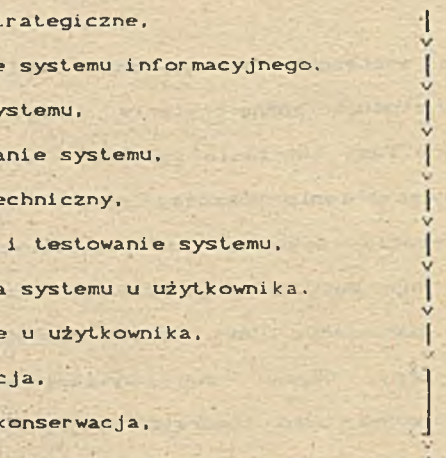
1. Modele cyklu życia systemu

Jednym z kluczowych zagadnień w tworzeniu systemów informatycznych jest wybór adekwatnego modelu cyklu życia systemu. Od tej decyzji bowiem zależą dalsze kroki postępowania

takie jak :

- wybór metodyki,
- wybór narzędzi,
- realizacja systemu.

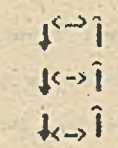
Obecnie najbardziej popularny jest tradycyjny, zwany kaskadowym model cyklu życia. Można przyjąć że dzieli się on na następujące etapy :



Model ten występuje w wielu wariantach jeśli chodzi o sformułowania werbalne, jednak zasadnicza jego treśc daje się sprowadzić do wspólnego mianownika. Jedyną pętlą sprzężenia zwrotnego jest sprzężenie występujące w fazie eksploatacja / rozwój i konserwacja.

Innym podejściem jest model oparty na prototypowaniu, inaczej zwany modelem operacyjnym. Można wyróżnić w nim następujące trzy fazy :

- wykonywalna specyfikacja - prototyp.
- przekształcenia.
- efektywne wdrożenie.



Model ten zdobywa sobie coraz większą popularność, począwszy od początku lat osiemdziesiątych. Związane jest to ze zdobywającą sobie coraz większe uznanie koncepcją tworzenia aplikacji bez programistów i powstaniem prostych w użyciu generatorów aplikacji oraz języków czwartej generacji.

Można przyjąć te dwa modele jako dominujące w chwili obecnej. Najszerzej stosowany jest oczywiście model kaskadowy, którego najnowszą inkarnacją jest specyfikacja Departamentu Obrony Stanów Zjednoczonych.

2. Systemy CASE jako forma integracji cyklu życia.

Współczesną formą warsztatu twórcy systemów informatycznych są systemy CASE (Computer Aided Software Engineering). Aczkolwiek są one zjawiskiem stosunkowo nowym - pierwsze znane użycie tej nazwy pochodzi z roku 1984, jednakże ich ewolucja następowała bardzo szybko. Przyjęto następujący podział systemów CASE, będący odzwierciedleniem ich miejsca w cyklu życia systemu, przyjmując za punkt odniesienia model kaskadowy :

- pre-CASE - etap planowania strategicznego,
- upper-CASE - od specyfikacji do projektu,
- lower-CASE - tworzenie i wdrażanie oprogramowania,
- post-CASE - eksploatacja i konserwacja systemu.

Systemy CASE wykorzystują szereg technik, takich jak :

- języki czwartej generacji,
- generatory aplikacji,
- narzędzia prototypowania,
- generatory kodu,
- języki wyszukiwania i manipulacji danymi,
- systemy zarządzania bazami danych,
- systemy tworzenia dokumentacji,
- grafikę komputerową.

Oczywiście nie każde narzędzie CASE korzysta ze wszystkich wyżej wymienionych technik, jednakże docelowym dążeniem twórców tych systemów jest stworzenie jak najbardziej spójnego i kompletnego warsztatu do tworzenia systemów informatycznych. Wybór technik wchodzących w skład systemu CASE zależy od przyjętej metodyki którą dany system ma wspierać.

3. PAGEFIT - przykład integracji technik

PAGEFIT jest zintegrowanym systemem CASE wspomagającym tworzenie systemów informatycznych. Narzędzie to jest elementem większej całości jaką jest własna metodyka firmy Inforem, wspierająca cały cykl życia systemu. Metodyka ta dzieli cykl życia systemu na następujące etapy

1. modelowanie przedsięwzięcia (business modelling)
   - modelowanie encji (entity modelling),
   - hierarchia funkcji,
   - modele dynamiki przedsięwzięć (business dynamic

modelowanie;

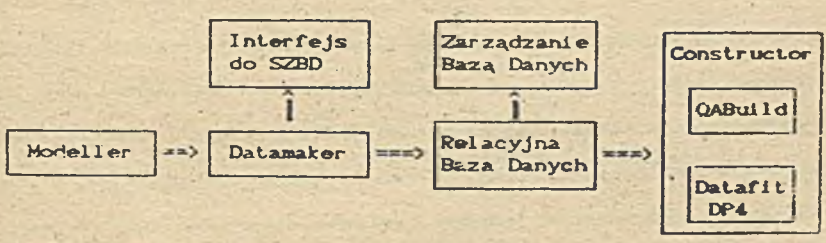
2. projektowanie systemu;

3. budowanie systemu;

4. testowanie systemu;

5. przekazanie systemu do użytkowania.

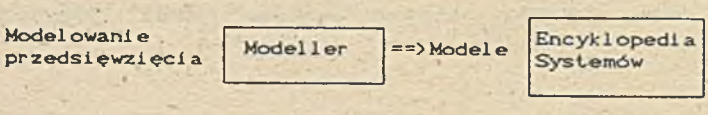
Jak widać jest to pewna odmiana modelu kaskadowego. Etap pierwszy (punkt 1.) jest realizowany wspólnie przez zleceniodawcę i firmę realizującą zlecenie. Wyniki pracy tego etapu są danymi wejściowymi do następnego etapu. System PAGEFIT obejmuje fazy od 2 do 4. Środowiskiem sprzętowym tego systemu są mikrokomputery klasy AT lub PS/2, wyposażone w dysk twardy i mysz, pracujące jako indywidualne stanowiska pracy bądź też w sieci. Środowiskiem programowym jest system operacyjny PC/MS-DOS oraz, dla części systemu, GEM. Elementy składowe systemu PAGEFIT przedstawia rysunek 1.



rysunek 1. Elementy składowe systemu PAGEFIT

Modeller

Modeller umożliwia definiowanie aplikacji poprzez użycie technik związanych z modelowaniem encji. Ponieważ Modeller funkcjonuje w środowisku GEM, praca z nim odbywa się, tak jak w ramach całego systemu PAGEFIT w trybie konwersacji z projektantem, jednak dzięki środowisku graficznemu tworzenie diagramów ER jest naturalnym, podstawowym trybem pracy. Poza tym, że projektant otrzymuje wynik swojej pracy w formie graficznej istotne jest także połączenie z wcześniejszym etapem pracy - modelowaniem przedsięwzięcia. Formułowane na tym etapie zasady (business rules) są odwzorowywane na model ER. Wprowadzony model przedsięwzięcia zostaje zapisany do *encyklopedii systemów.*

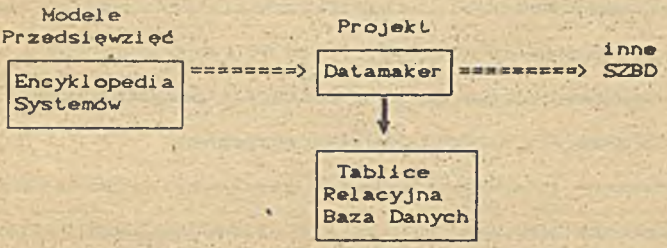


rysunek 2. Modeller - schemat działania

W trakcie budowania modelu ER powiązania i obiekty tworzone podczas tego procesu są automatycznie weryfikowane pod kątem zgodności z informacjami zawartymi w encyklopedii. Poza projektem w formie graficznej, Modeller tworzy także dokumentację w formie tekstowej. Pominięcie jednego z elementów wynikowych systemu nie pozwala na przejście do następnego etapu. Istotną cechą podsystemu Modeller jest możliwość wielokrotnego używania danych zawartych w Encyklopedii do tworzenia różnych systemów. Daje to możliwośc zwiększenia wydajności pracy twórcy systemów.

Datamaker

Datamaker automatycznie generuje schemat bazy danych korzystając z danych zawartych w encyklopedii. W tej fazie następuje konwersja modelu na schemat relacyjnej bazy danych. Pomimo tego, że PAGEFIT jest systemem który pracuje w środowisku mikrokomputerowym, docelowe systemy baz danych niekoniecznie muszą funkcjonować w tym środowisku. Obecnie dostępne są następujące formaty schematów baz danych - DB2, Oracle, Wang PACE. Oczywiście jest także dostępny system bazy danych działającej w środowisku PC - Datafit DP4. Możliwe jest także wykonywanie wszelkich niezbędnych operacj mających na celu optymalizację bazy danych.

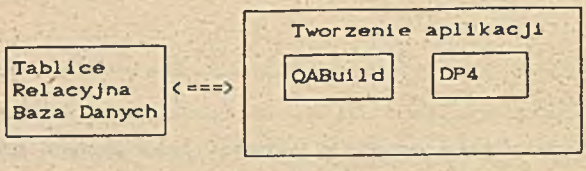


rysunek 3. Datamaker - schemat działania

Constructor

Ostatnim elementem systemu PAGEFIT jest Constructor - podsystem służący do tworzenia aplikacji. W jego skład wchodzą dwa moduły :

- QABuild - szybki generator aplikacji,
- Datafit DP4 - generator aplikacji "dużej mocy".



rysunek 4. Constructor - schemat działania

QABuild jest szybkim generatorem aplikacji w tym znaczeniu, że umożliwia szybkie tworzenie prototypów aplikacji oraz prostszych, pozbawionych bardziej skomplikowanych algorytmów zastosowań. Zasadnicze cechy tego podsystemu są następujące :

- aplikacje mogą być budowane bardzo szybko bez etapu kodowania, co poza tym że przyśpiesza ich tworzenie, lecz także umożliwia tworzenie aplikacji bezpośrednio przez użytkownika;
- proces tworzenia aplikacji jest kontrolowany automatycznie co poprawia ich jakość;
- wszystkie aplikacje są z definicji przystosowane do działania w środowisku wielodostępnym.

Proces budowy jest interakcyjny, kierowany przez system menu, przy czym po wybraniu danych na których dana aplikacja ma operować, większość decyzji autora aplikacji polega na udzielaniu systemowi odpowiedzi typu TAK/NIE.

DP4 jest zestawem narzędzi służących do budowy złożonych wielodostępnych systemów transakcyjnych, przy użyciu języków C, COBOL, Pascal i 4680 BASIC, poprzez generowanie kodu źródłowego w tych językach. W skład DP4 wchodzi także system zarządzania relacyjną bazą danych. Narzędziami są następujące elementy :

- Reporter - generator wydruków,

- Mapbuilder - narzędzie do projektowania ekranów.
- Menubuilder - generator menu.
- Interchanger - narzędzie do wymiany danych z innymi systemami.
- Browser - narzędzie do przeglądania i modyfikowania tablic.

4. Podsumowanie

System PAGEFIT wspiera oprócz modelu kaskadowego, także model oparty o prototypowanie. W tym wypadku etapy pracy z systemem są następujące:

- wstępne zaprojektowanie danych.
- prototypowanie przy użyciu QABuild.
- wykorzystanie sprzężenia zwrotnego pomiędzy Constructorem a encyklopedią do zmodyfikowania schematu bazy danych.
- na podstawie encyklopedii wygenerowanie aplikacji finalnej pry pomocy QABuild i DP4.

Ta opcja jest wynikiem historii rozwoju systemu PAGEFIT. Niezależnie od siebie powstawały - podsystem Modeller i system zarządzania bazą danych Datafit. PAGEFIT jest efektem współpracy dwu niezależnych firm. Możliwość wsparcia dwu całkiem różnych, opartych na diametralnie różnych założeniach modeli cyklu życia powoduje, że system PAGEFIT może być wykorzystywany przez twórców systemów o całkiem odmiennych zapatrywaniach.

W klasyfikacji systemów CASE PAGEFIT mieści się w kategoriach upper-CASE (Modeller,Datamaker), lower-CASE (Datamaker, Constructor) i post-CASE (Constructor).

Literatura

Davis, Jack M. , The Evolution of CASE, ORACLE, Vol III No 2,1989

DoD STD 2167A, Washington, 1989

Information Systems Methodologies, Addison-Wesley, 1988

Martin, James, Application Development Without Programmers, Prentice Hall, 1982

PAGEFIT - User Guide, Inforem Plc, 1989

Wegner, Peter, Capital - Intensive Software Technology, w Software Reusability Vol 1, Addison-Wesley, 1989,

Wrycza, Stanisław, Współczesne Metodyki Tworzenia Systemów Informatycznych Zarządzania, ORG-SERVICE, Gdańsk, 1988

Andrzej Maciej Wierzba
Instytut Informatyki
Uniwersytet Warszawski
PKiN p. 850, 00-950 Warszawa

System Organizacji Komunikacji z Użytkownikiem ViDE-2.

1. Wstęp.

Metody współpracy z użytkownikiem oparte na technikach graficznej interakcji używające piktogramów, menu czy okien, stają się nieodłącznym elementem współcześnie tworzonych programów. Coraz częściej twórcy podstawowego (systemowego) oprogramowania związani z producentami sprzętu komputerowego uznają je za standardowy element programów aplikacyjnych. Moduły komunikacji nie stanowią dzisiaj jakiegoś podrzędnego szczegółu programów. Organizacja dialogu z użytkownikiem staje się równie ważna jak sposób rozwiązania "zasadniczego" zadania danego programu. W związku z tym powstaje potrzeba stworzenia narzędzi ułatwiających tworzenie modułów komunikacji programów interakcyjnych klasy WIMP. W przedstawianej pracy prezentowany jest opracowywany przez pracowników i studentów Instytutu Informatyki System Organizacji Komunikacji z Użytkownikiem ViDE-2.

2. Klasa programów interakcyjnych WIMP.

Nazwa WIMP nie określa konkretnego systemu ani jego konkretnego kształtu graficznego. Skrót pochodzący od słów Windows, Icons, Mouse and Pull-down menus określa pewien styl pracy z systemem, styl prezentacji i prowadzenia dialogu. Do stylu tego należą nie tylko wymienione w nazwie elementy, ale tworzy go szereg postulatów dotyczących techniki prowadzenia dialogu:

- *intuicyjność i naturalność*, przejawiające się w dwóch płaszczyznach dialogu użytkownik - komputer:
  a) Maksymalne wykorzystanie naturalnych dla człowieka predyspozycji i umiejętności, zamiast zmuszania go do uczenia się nowych. Do tego służą *wizualizacja* (przedstawianie informacji przez program w postaci graficznej) i *preferencja gestu* (zastosowanie do wydawania poleceń przez użytkownika manipulatora typu mysz).
  b) Odpowiednie interpretowanie przez program akcji wykonywanych przez użytkownika, aby zachowane były:
  -*responsywność* (ang. responsivity)
  Każda akcja użytkownika powinna mieć natychmiastowy widoczny lub słyszalny skutek. Brak takiego widocznego skutku oznacza że akcja jest przez system ignorowana.

- *Spojnosc* (ang. consistency)
  Akcje powinny miec zawsze takie same znaczenie dla kazdego obiektu danego typu. Jesli odnoszą się do obiektow roznych typow znaczenie musi byc możliwie podobne, akcje nie mogą dawac zaskakujących rezultatow. Znaczenie akcji złożonych jest wtedy naturalnym rozszerzeniem znaczenia akcji elementarnych.
- *Bezpieczenstwo* (ang. safety)
  Akcje o poważnych skutkach globalnych powinny byc potwierdzane. Wskazane jest umożliwienie cofania skutkow akcji omyłkowych.

- *nieograniczanie użytkownika (permisywnosc)*, polegające na nie stawianiu użytkownikowi nienaturalnych i nieoczywistych barier w procesie interakcji (takich jak np. ograniczenie dostępu do widocznych na ekranie obiektow dialogowych).

- *efektywnosc informacyjna*, to jest możliwosc graficznego przekazania dużej ilosci informacji w prosty i czytelny sposob. (np. kształt kursora może informowac o dostępnych aktualnie operacjach)

- *efektywnosc operacyjna*, rozumiana jako sprawnosc (szybkosc) realizowania transakcji dialogowych, oraz łatwosc uruchamiania najczesciej używanych funkcji

3. Systemy wspomagające tworzenie programow klasy WIMP.

O ile systemy klasy WIMP są z punktu widzenia użytkownika narzędziami wygodnymi i prostymi, to tworzący je programista staje wobec niełatwego zadania. Tworząc dla swego programu moduł komunikacji realizujący "dialog w stylu WIMP", musi on częstokroc rozwiązywac poważne problemy nie związane bezposrednio z realizowanym przez program zadaniem. Jednym z tych problemow jest stworzenie odpowiedniego systemu okienek. Kolejnym - jest szybka realizacja operacji graficznych takich jak rysowanie podstawowych figur geometrycznych, krzywych, przesyłanie map bitowych i kopii fragmentow ekranu między pamięcią RAM programu a pamięcią obrazu itd.

W tej sytuacji koniecznoscią stało się powstanie systemow wspomagających tworzenie programow klasy WIMP, określanych skrotem **UIMS** (User Interface Management Systems). Ideą przewodnią przy rozwijaniu tego typu narzędzi dla programistow było początkowo jedynie wydzielenie kodu operacji powtarzających się w każdym programie, a więc stworzenie ogolnego systemu okienek, ogolnej biblioteki graficznej oraz pewnego standardowego zestawu obiektow dialogowych i funkcji do manipulowania tymi obiektami.

W ostatnich latach widac tendencję do tworzenia niezależnych od języka, maszyny i systemu operacyjnego, w pełni otwartych systemow UIMS. Otwartosc ta idzie w kierunku oddzielenia programisty od szczegołow obsługi obiektow dialogowych. Programista może traktowac proces interakcji w sposob abstrakcyjny, pozostawiając szczegoły realizacji tego procesu specjalistom od ergonomii czy psychologii pracy z komputerem.

Podobnie jak w wypadku opisu specyfiki programów interakcyjnych klasy WIMP, również dla systemów UIMS możemy wyodrębnić szereg charakteryzujących je typowych własności:

-*Dostarczenie narzędzi i mechanizmów pozwalających na tworzenie programów i środowisk klasy WIMP.*
-*Rozszerzalność*, rozumiana jako możliwość łatwego dołączania nowych elementów do zbioru standardowych obiektów dialogowych
-*Elastyczność*, czyli możliwość zmiany wyglądu i zachowania standardowych obiektów dialogowych.
-*Przenaszalność*, czyli niezależność od sprzętu i systemu operacyjnego.

UIMS powinien zawierać narzędzia ułatwiające korzystanie z niego, służące do projektowania wyświetlanych obiektów dialogowych takich jak menu, okienka i piktogramy, oraz składające z tych elementów całe programy.

4. Ogólna koncepcja systemu ViDE-2.

4.1. Podstawowe cechy.

System ViDE-2 pozwala na rzetelne, szybkie i wygodne tworzenie modułów komunikacji z użytkownikiem opartych na technikach graficznej interakcji. Nazwa "ViDE" jest z jednej strony skrótem angielskiego określenia Visual Dialog Environment, z drugiej zaś jej łacińskie znaczenie (łac. vide - patrz), podkreśla rolę wizualizacji w graficznych systemach kontaktu z użytkownikiem.

Zadaniem systemu ViDE-2 jest odciążenie programisty od żmudnych i rozpraszających go prac związanych z łączeniem poszczególnych modułów programu oraz komunikacją z użytkownikiem. Narzucony przez system, lecz równocześnie intuicyjny, podział programu na zadania wyjątkowo upraszcza pracę zespołową. Zgodnie z koncepcją UIMS, ostateczny kształt graficzny oraz strona funkcjonalna powstających modułów nie są sztywno określone i dają się w dużym stopniu modyfikować.

4.2. Atomy systemu ViDE-2 - obiekty dialogowe.

Podstawowymi elementami programów napisanych z użyciem ViDE-2 są obiekty dialogowe. Zadaniem każdego z nich jest połączenie części operacyjnej programu z częścią pozwalającą na komunikację z użytkownikiem. Obiekt dialogowy składa się ze:

- struktury danych;
- zbioru procedur i funkcji operujących na tej strukturze;
- funkcji wyświetlającej obiekt - pozwalającej na dokładne określenie jego stanu ;
- protokołu dialogu służącego do interpretacji poleceń użytkownika oraz komunikatów od innych obiektów dialogowych i podejmowania na ich podstawie stosownych operacji na strukturach danych;
- parametrów dla systemu ViDE-2 pozwalających na łatwe manipulowanie obiektami dialogowymi (np. aktualny rozmiar i położenie obiektu);
- funkcji inicjalizującej i kasującej obiekt dialogowy w Systemie;

Obiekt dialogowy stanowi dla użytkownika z zewnątrz zamkniętą całość, która w ścisle określony sposób, zgodnie z regułami dialogu, reaguje na jego działanie i komunikaty przesyłane z innych obiektów.

Rys 1. Obiekt dialogowy przetwarza komunikaty i polecenia Systemu. ViDE-2 korzysta z danych zawartych w standardowych strukturach obiektu dialogowego.

Program napisany z użyciem ViDE-2 składa się więc z niezależnych, jednocześnie dostępnych obiektów dialogowych o zróżnicowanym wyglądzie zewnętrznym i zachowaniu.

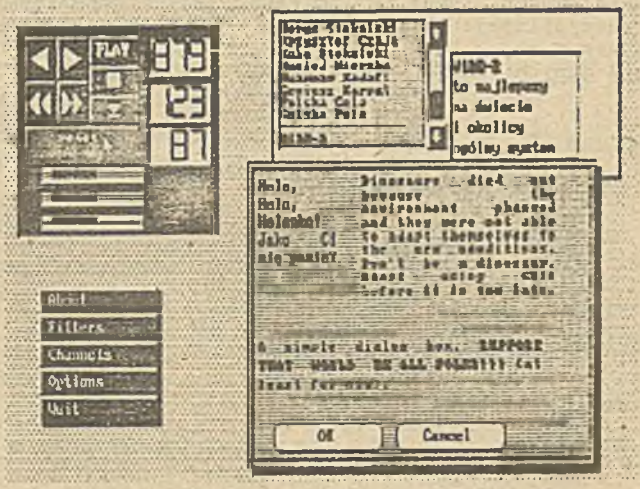
Każdy obiekt dialogowy można podzielić na dwie części. Jedna, zwana typem, jest związana z nim na stałe i zawiera informacje wspólne dla wszystkich realizacji danego obiektu. Druga część jest dostarczana podczas inicjalizacji obiektu w Systemie. Należą do niej, przykładowo, położenie na ekranie, nazwa wyświetlanego pola menu, funkcja uruchamiana receptorem.

Rozróżnia się 4 podstawowe rodzaje obiektów dialogowych:

-okno - prostokątny fragment ekranu wydzielający obszar, w którym są wyświetlane obiekty wykorzystywane do dialogu z użytkownikiem.
-obraz - graficzna reprezentacja struktury danych zadeklarowanej przez programistę. Obraz, jak każdy inny obiekt Systemu, może zostać uaktywniony tzn. przejść do stanu pozwalającego na przyjmowanie poleceń użytkownika. W stanie aktywności ruch kursora systemowego może być ograniczony do prostokąta zawierającego obraz (gdy dochodzi do brzegu okienka przesuwa się ono płynnie po obrazie).
-receptor - prostokątny obszar w oknie spełniający rolę dodatkowego urządzenia wejściowego. Receptor w systemach WIMP jest najbardziej wykorzystywanym elementem ekranu. Program wyświetla na ekranie zestawy dostępnych akcji w formie receptorów a użytkownik ruszając kursorem wskazuje czynność, która ma być wykonana. Receptor może również służyć do wyświetlania aktualnego stanu programu (np. receptor wyświetlający położenie kursora na obrazie).

-piktogram - rysunek wyswietlany na ekranie odpowiadający zadaniu bądz obiektowi dialogowemu zainicjowanemu w Systemie, ale czasowo zawieszonemu.

Każdy rodzaj obiektu dialogowego jest inaczej traktowany przez System. Okna mogą nawzajem się przesłaniać, receptory muszą leżeć na oknach nie zachodząc na siebie. Poprzez podanie rodzaju programista zobowiązuje System do odpowiedniego traktowania obiektu dialogowego.



Rys 2. Okno aplikacji ViDE-2 z typowymi obiektami dialogowymi.

Zamkniętosc obiektow dialogowych i ich standaryzacja (wydzielenie typu) pozwala na wielokrotne wykorzystywanie raz zdefiniowanych obiektów w rożnych programach. Najczęściej używane typy powinny być trzymane w bibliotece, aby można łatwo było z nich korzystac w innych aplikacjach.

4.3. Zdarzenia. Przykład prostego obiektu dialogowego.

Komunikacja w ViDE-2 odbywa się poprzez system zdarzeń. Ten sposób wymiany informacji, czasami niezgodny z przyzwyczajeniami tradycyjnych programistów, jest niezbędny ze względu na konieczność równoległego prowadzenia dialogu z wieloma obiektami. System oparty o wywoływanie funkcji wymusza poprzez stos powrót do wcześniej rozpoczętej czynności. System zdarzeń pozwala na swobodę równoczesnej pracy z wieloma obiektami.

Zdarzenia w Systemie dzielą się na polecenia przychodzące od użytkownika i komunikaty wysyłane przez obiekty dialogowe. Z każdym zdarzeniem są związane:

- identyfikator (nie musi być unikalny w Systemie)
- nadawca
- adresat zdarzenia (nie musi być określony)
- położenie kursora systemowego

- czas w chwili wysłania zdarzenia (np. moment wcisnięcia klawisza)
- parametry charakterystyczne dla poszczególnych zdarzeń.

Każdy obiekt dialogowy może znajdować się w kilku stanach. W każdym ze stanów obiekt ma inny wygląd zewnętrzny (np. kolor tła), kursor lokalny, a przede wszystkim, protokół dialogu.

Pojęcia stanu i protokołu dialogu najlepiej zilustruje poniższy przykład receptora zliczającego "kliknięcia" (naciśnięcie i puszczenie) na obiekcie lewego i prawego klawisza myszki. Receptor zmienia wartość liczby w swojej strukturze danych. Adres tej zmiennej jest przekazywany w fazie jego inicjalizacji. Typ tego receptora można opisać następująco:

struktura danych: napis, adres liczby typu integer,
wartość minimalna,
wartość maxymalna.

funkcja inicjalizująca nadaje obiektowi stan początkowy 0 i wypełnia danymi początkowymi strukturę danych.

3 stany obiektu dialogowego:
- -stan 0 - wyświetlanie napisu i aktualnej liczby
- -stan 1 - wyświetlanie napisu, aktualnej liczby i znaku '+'
- -stan 2 - wyświetlanie napisu, aktualnej liczby i znaku '-'

protokół dialogu:

stan początkowy	zdarzenie	akcja	stan końcowy
0	"naciśnięty lewy klawisz myszki na obszarze receptora"	- przejście do stanu 1; - wyświetlenie;	1
0	"naciśnięty prawy klawisz myszki na obszarze receptora"	- przejście do stanu 2; - wyświetlenie;	2
1	"puszczony lewy klawisz myszki na obszarze receptora"	- if liczba<wartoscmax liczba:=liczba+1; - przejście do stanu 0; - wyświetlenie;	0
1 2	"puszczony lewy klawisz myszki poza obszarem receptora" "puszczony prawy klawisz myszki poza obszarem receptora"	- przejście do stanu 0; - wyświetlenie;	0
2	"puszczony prawy klawisz myszki na obszarze receptora"	- if liczba>liczbamin liczba:=liczba-1; - przejście do stanu 0; - wyświetlenie;	0

Po otrzymaniu polecenia stworzenia receptora System wywołuje jego funkcję inicjalizującą i rysuje go. Powołany do życia obiekt będzie oczekiwał na wciśnięcie lewego, bądź prawego klawisza myszki na swoim obszarze i wtedy wykona określone regułami dialogu akcje. Wygląd obiektu w poszczególnych stanach został pokazany na rys. 3

Rys 3. Wygląd obiektu dialogowego w 3 stanach

4.4. Podział programu w systemie ViDE-2.

Program w systemie ViDE-2 jest podzielony na logicznie niezależne mniejsze części, stanowiące funkcjonalne całości - zadania. Taki podział umożliwia wykorzystywanie raz napisanych zadań w wielu programach. Typowym tego przykładem jest zadanie obsługi systemu plików pozwalające na wyszukiwanie, zapis i odczyt zbiorów. Nadanie programowi struktury zadaniowej ułatwia użytkownikowi posługiwanie się programem, a programiście jego pisanie.

W skład zadania wchodzi pewna ilość obiektów dialogowych, a więc okien wraz ze związanymi z nimi receptorami i obrazami, służących do prowadzenia konwersacji z użytkownikiem.

Każde zadanie w Systemie operuje na własnej strukturze danych zainicjowanej częściowo przez użytkownika (np. przez podanie pliku na którym będzie pracować), a częściowo przez funkcję inicjalizującą zadanie. W tej strukturze mieszczą się również parametry początkowe i dane dla wszystkich obiektów graficznych wchodzących w skład jego zadania. Z każdym zadaniem jest związany protokół dialogu pozwalający na komunikację między zadaniami.

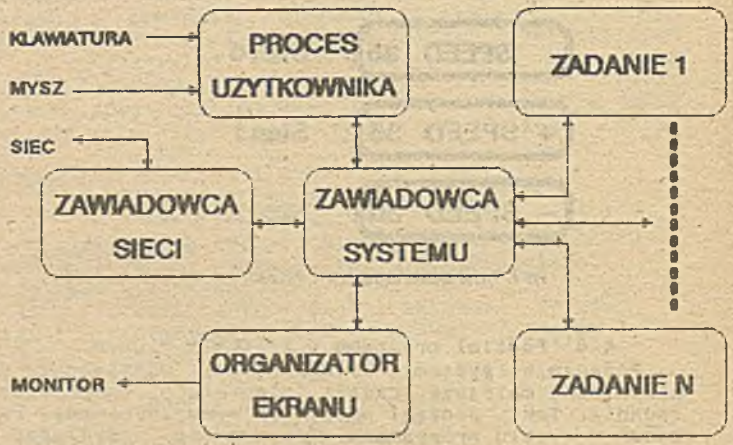
W ramach programu oprócz zadań zdefiniowanych przez użytkownika działają zadania Systemu. Są nimi:

Proces Użytkownika - odczytuje i interpretuje działania podejmowane za pomocą urządzeń wejściowych przez użytkownika. Obsługuje ruch kursora systemowego po ekranie.

Organizator Ekranu (SM - Screen Manager) - jest odpowiedzialny za wszystko co się dzieje na ekranie, realizuje między innymi otwieranie i zamykanie obiektów dialogowych, odświeżanie obrazów z nimi związanych itp.

Zawiadowca Systemu - zajmuje się interpretacją zdarzeń ze strumienia zdarzeń. Zna aktualnie aktywne zadanie, okno i receptor. Wyszukuje adresatów zdarzeń, ustala zbiór obowiązujących aktualnie reguł dialogu i wykonuje akcje z nich wynikające.

Zawiadowca Sieci (w przygotowaniu) - organizuje przesyłanie komunikatów pomiędzy różnymi zadaniami w sieci.



Rys 4. Zadania w systemie ViDE-2.

5. Zakonczenie.

Stworzony system ViDE-2 wraz z biblioteką graficzną i systemem zdarzen był wykorzystywany do pisania wielu programow graficznych, za każdym razem pozwalając na poprawienie szybkosci ich tworzenia i jakości komunikacji z użytkownikiem.

W trakcie prac powstało wiele obiektów dialogowych pozwalających na składanie nowych aplikacji z gotowych fragmentów. Ze względu na użycie DOS-a System w obecnej wersji nie pozwala na rownoległą pracę wielu zadan. Ze względu na rozwiązania przyjęte w ViDE-2 ta niedogodnosc będzie mogła być usunięta przy pracy pod kontrolą innych systemów operacyjnych.

Literatura:

1. Fundamental Algoritms for Computer Graphics, praca zbiorowa pod red. Rae A. Earnshaw, Springer-Verlag 1985
2. Handbook of Human-Computer Interaction, praca zbiorowa pod red. M. Helander, North-Holland 1988
3. Hearn D., Baker M. P. : Computer Graphics, Prentice-Hall 1986
4. Theo Pavlidis : Algoritms for graphics and image processing. Computer Scince Press, Inc. 1982.
5. Scheifler R. W. , Gettys J. : The X Window System , ACM Tran. Graph. 5, 2(April 86), str. 79-109.
6. System Organizacji Komunikacji z Użytkownikiem, praca zbiorowa pod red. Agnieszki Wojciechowskiej. IIUW 1989.

DNUK USPOL 106/90/110.