WIOSENNA SZKOŁA PTI
ŚWINOUJŚCIE '94

JĘZYKI 4GL
i
ZARZĄDZANIE PROJEKTAMI

Organizowana przez
POLSKIE TOWARZYSTWO INFORMATYCZNE

Świnoujście 16-20 maja 1994

WIOSENNA SZKOŁA PTI

ŚWINOUJŚCIE '94

JĘZYKI 4GL
i
ZARZĄDZANIE PROJEKTAMI

Organizowana przez

POLSKIE TOWARZYSTWO INFORMATYCZNE

Świnoujście 16-20 maja 1994

SPIS TREŚCI

Słowo wstępne

Witold Staniszkis (RODAN-SYSTEM)
Projektowanie rozproszonych systemów informatycznych

OFICYNA INFORMATYCZNA
ACUCOBOL-85

COMPUTER SYSTEMS FOR BUSINESS INTERNATIONAL
Zintegrowane środowisko do tworzenia i utrzymania systemów informatycznych

Marek Miłosz (Politechnika Lubelska)
Prototypowanie we współczesnych SZRDB

TRADEX SOFTWARE COMPANY
ROSI-SQL
SZRBD INFORMIX

DIGITAL EQUIPMENT POLSKA
FORTE - narzędzie czwartej generacji przyszłości

COMPUTER ASSOCIATES
CA - Visual Objects
CA - Super - Project

Tezy do dyskusji "Programista roku 2000"

Słowo wstępne

Po rocznej przerwie ponownie spotykamy się na Wiosennej Szkole PTI. Spotkania te zwykle były i pozostaną monotematyczne poruszając naistotniejsze aktualnie problemy informatyki. Rozwój technologii informatycznych nie pozostaje bez wpływu na sposoby i metody pracy informatyków. Problematyka ta zwykle była podstawą szczegółowych tematów kolejnych Wiosennych Szkół PTI. Pragniemy pozostać w tym nurcie zainteresowań.

Tematyka Wiosennej Szkoły PTI Świnoujście'94 będzie się koncentrowała wokół problematyki Języki 4GL. Pragniemy przedstawić różne rozumienie tego terminu, który na trwałe wszedł do słownika terminów informatycznych, chociaż jest bardzo różnorodnie wykorzystywany. Chcemy przybliżyć nowe projekty, wspomaganie programowania, doświadczenia praktyczne użytkowania języków 4GL, uwarunkowania stosowania, efekty występujące w różnych fazach procesu wytwarzania systemu informatycznego. Mamy nadzieję na zderzenie różnych, często kontrowersyjnych poglądów, co powinno przybliżyć uczestnikom Szkoły rozumienie tego terminu i wyrobienie sobie własnej opini na temat języków programowania 4GL. Zapraszamy do prezentacji i wyrażania opinii na temat systemów zarządzania bazami danych, które oferują języki 4GL.

Zaprosiliśmy do prezentacji dystrybutorów dużych systemów zarządzania relacyjnymi bazami danych. Zaproszenie nasze przyjęli:

- Computer Systems for Business International dystrybutor systemu PROGRESS,
- Rodan System dystrybutor systemu INGRES,
- Tradex Software Company dystrybutor systemu INFORMIX.

Przedstawiciele tych firm przedstawią zalety i możliwości języków 4GL wmontowanych w systemy zarządzania bazami danych. Liczymy na ciekawą konfrontację i wymianę doświadczeń z stosowania nowych technik wytwarzania oprogramowania.

Uzupełnieniem dla języków 4GL wchodzących w skład dużych systemów zarządzania bazami danych są inne produkty, które mogą być stosowane samodzielnie i przy współpracy z różnymi bazami danych. W tym nurcie wykładowym przedstawione zostaną:

- ACUCOBOL-85, implementacja popularnego języka programowania z mechanizmami języka 4GL prezentowany przez Oficynę Informatyczną,
- CA - Visual Object następca popularnego w Polsce Cllipera z mechanizmami charakterystycznymi dla 4GL, prezentowany przez firmę Computer Associates,

- FORTE pakiet programowy reklamowany jako język 4GL przyszłości, prezentowany przez Digital Equipment Polska.

Drugim, współbieżnym tematem Szkoły będzie problematyka zarządzania projektami, w ramach tego nurtu wykładów przedstawione zostaną:

- ROSI przez Tradex Software Company,
- CA Super Project przez Computer Associates.

Na zakończenie Szkoły planowana jest poważna dyskusja o prowokacyjnym tytule Programista roku 2000, do której zaprosiliśmy kilku znanych informatyków reprezentujących różne poglądy i doświadczenie programistyczne. Liczymy bardzo na aktywny udział programistów i wszystkich uczestników Szkoły. Sądzimy, że warto podyskutować o zawodzie, który skupia na sobie stałą uwagę informatyków i często przez pryzmat programisty oceniana jast cała informatyka. Zawód programisty przechodził i przechodzi różne koleje losów w zależności od stanu rozwoju środowiska sprzętowo-programowego, w którym przyszło mu działać. Języki 4GL to nowe wyzwanie dla programistów. Jaka będzie ich przyszłość, jak należy szkolić nowych programistów ?

Te i inne pytania zostały postawione w załączonych na końcu materiałów konferencyjnych, tezach do dyskusji licząc na rozszerzenie tej listy i próbę odpowiedzi przynajmniej na niektóre z nich.

Zapraszam do aktywnego udziału w obradach Szkoły Wiosennej PTI.

Zdzisław Szyjewski

Szczecin maj 1994

PROJEKTOWANIE ROZPROSZONYCH SYSTEMÓW INFORMATYCZNYCH

Witold Staniszkis

RODAN-SYSTEM Sp. z o.o.
Warszawa

1. WSTĘP

Rozwój rozproszonych systemów informatycznych stanowił w okresie ostatniego dziesięciolecia podstawową tendencję ewolucji zastosowań informatyki. Rosnące wymagania użytkowników szły w tym zakresie w parze z eksponencjalnym wzrostem mocy obliczeniowej komputerów oraz dynamicznym rozwojem sieci transmisji danych pozwalających na budowę zdalnych i lokalnych sieci komputerowych. Prace badawcze i wdrożeniowe w dziedzinie protokołów obsługi transmisji danych (ISO/OSI, TCP/IP) doprowadziły do powstania globalnych sieci komputerowych (internet) integrujących wiele różnych zdalnych i lokalnych sieci komputerowych. W tej sytuacji realizacja rozproszonych systemów informatycznych, integrujących na poziomie logicznym (logiczny model danych, funkcje użytkowe) przy jednoczesnym rozproszeniu danych i procesów w sieci komputerowej, jest coraz częściej podstawowym wymaganiem użytkowym w stosunku do nowych zastosowań informatyki.

Projektowanie rozproszonych systemów informatycznych jest nową dziedziną informatyki wystarczająco różną od projektowania scentralizowanych systemów informatycznych by, mimo istnienia wielu wspólnych elementów, spowodować konieczność rozwinięcia konwencjonalnych metod projektowania. Naszym celem jest przedstawienie podstawowych problemów projektowania rozproszonych systemów informatycznych oraz wskazanie wynikających z nich nowych elementów metodyki projektowania. Po wprowadzeniu podstawowych pojęć i definicji w dziedzinie rozproszonych systemów informatycznych omówimy typowe wymagania technologiczne współczesnych systemów, a szczególnie zagadnienia związane z efektywnością techniczną w kontekście systemów rozproszonych, by w ostatnim punkcie przedstawić podstawowe elementy metodyki projektowania rozproszonych systemów informatycznych.

Nacisk jaki kładziemy na zagadnienia efektywności technicznej wynika z faktu, że podstawowe różnice procesów projektowania scentralizowanych i rozproszonych systemów leżą właśnie, przynajmniej z punktu widzenia użytkowych walorów systemu informatycznego, w dziedzinie efektywności technicznej.

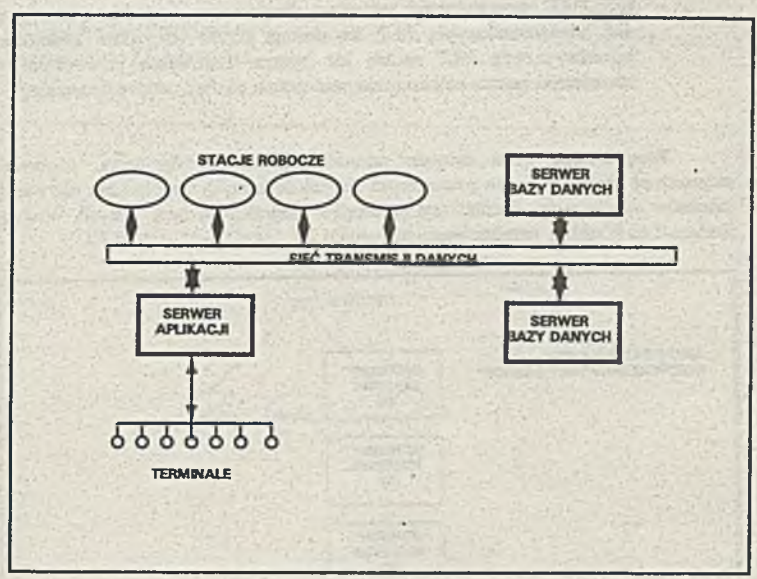
2. ROZPROSZONE SYSTEMY INFORMATYCZNE

Podstawowe pojęcia, poza definicją rozproszonego systemu informatycznego, obejmują takie zagadnienia jak rozproszona baza danych, rozproszone przetwarzanie transakcyjne, oraz podział i rozmieszczanie danych i procesów.

Rozproszony system informatyczny obejmuje zbiór połączonych komputerów przetwarzających współdziałające oprogramowanie użytkowe, przy czym

- **Współdziałanie oprogramowania (interoperability)** polega na utrzymaniu wymiany danych i sterowania pomiędzy różnymi procesami w oparciu o wspólny protokół komunikacyjny. Istnienie takiego protokołu powinno być warunkiem koniecznym i dostatecznym współdziałania bez narzucania dodatkowych wymagań na środowisko narzędziowe współdziałających procesów użytkowych.
- **Rozproszony system operacyjny** stanowi środowisko oprogramowania traktujące zbiór połączonych komputerów jako jeden, wspólnie zarządzany, system komputerowy.
- **Sieciowy system operacyjny** steruje współdziałaniem systemów operacyjnych i oprogramowania użytkowego autonomicznych systemów komputerowych połączonych siecią transmisji danych.

W praktyce zapewnienie współdziałania jest jedynym wymaganiem narzucanym na środowisko sprzętowe i programowe stanowiące platformę realizacji systemów rozproszonych. Rozproszone systemy operacyjne wychodzą dopiero z fazy eksperymentalnej, za wyjątkiem systemów operacyjnych w silnie związanych, wieloprocesorowych konfiguracjach komputerowych (tightly-coupled multiprocessor). Sieciowe systemy operacyjne wspomagają zarządzanie siecią komputerową i stanowią nieodłączny element konfiguracji sieci. Typowa konfiguracja lokalnej sieci komputerowej stanowiącej platformę realizacji rozproszonych systemów informatycznych jest przestawiona na rysunku 1.



Rys 1. Typowa konfiguracja lokalnej sieci komputerowej.

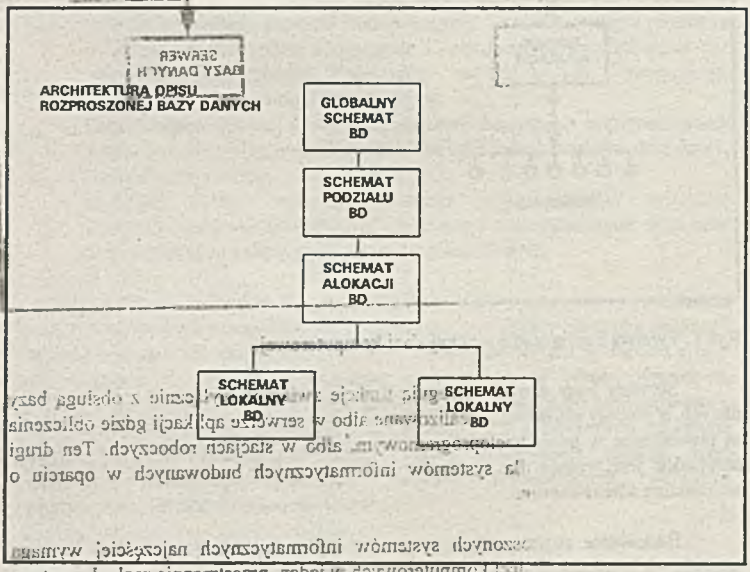
Serwery bazy danych obsługują funkcje związane wyłącznie z obsługą bazy danych, a procesy użytkowe są realizowane albo w serwerze aplikacji gdzie obliczenia są prowadzone w trybie wieloprogramowym, albo w stacjach roboczych. Ten drugi przypadek jest typowy dla systemów informatycznych budowanych w oparciu o architekturę **klient-serwer.**

Budowanie rozproszonych systemów informatycznych najczęściej wymaga łączenia wielu lokalnych sieci komputerowych w jeden, przestrzennie rozległy system. W Polsce możliwości takie już istnieją a powszechnie dostępne sieci telekomunikacyjne obejmują między innymi:

- **Polpak** stanowiący ogólnie dostępną sieć pakietową Telekomunikacji Polskiej S.A. obejmującej takie usługi jak transfer danych w oparciu o protokół X.25 i pochodne, poczta elektroniczna X.400/X.500 wraz z modułem EDI, praca w trybie sieci abonenckiej.
- **Telbank** służący przede wszystkim do łączności bankowej eksploatowany przez Bankowe Przedsiębiorstwo Telekomunikacyjne i umożliwiający transmisję danych w trybie pakietowym oraz wielomedialną łączność typu punkt-punkt.
- **NASK** obsługujące przede wszystkim środowisko naukowe i akademickie umożliwiająca stosowanie takich protokołów jak X.25, BSC/SNA, DECNET, i TCP/IP.

- KOLPAK stanowiąca sieć pakietową zrealizowaną w oparciu o istniejącą sieć telekomunikacyjną PKP do obsługi przede wszystkim systemów informatycznych PKP takich jak system kierowania przewozami i zarządzania, pocztę elektroniczną oraz system obsługi ruchu pasażerskiego.

Rozproszona baza danych stanowi logicznie zintegrowany, fizycznie rozproszony zbiór wzajemnie powiązanych danych dostępnych współbieżnie dla wielu procesów użytkowych. Architekturę schematów rozproszonej bazy danych, według Stefano Ceri (Ceri85), przedstawiono na rysunku 2.

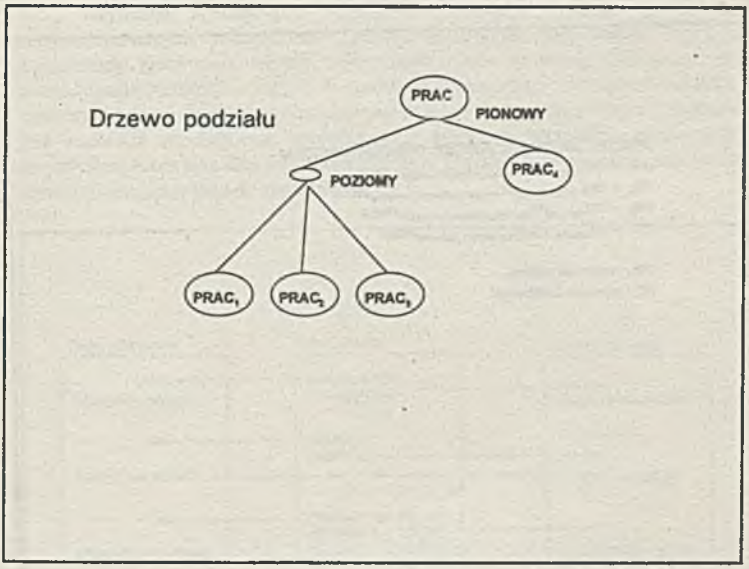


Rys. 2. Architektura schematów rozproszonej bazy danych.

Globalny schemat bazy danych stanowi odpowiednik logicznego schematu centralnej bazy danych, przy czym obejmuje on wszystkie elementy schematów lokalnych baz danych, które mają być widoczne w ramach logicznego modelu danych rozproszonego systemu informatycznego. Na tym poziomie nie są zazwyczaj definiowane, i naturalnie obsługiwane, takie poza relacyjne elementy modelu danych jak reguły wnioskowania, zdarzenia, i procedury bazy danych.

Schemat podziału bazy danych obejmuje definicję fragmentów rozproszonej bazy danych określone w oparciu o predykaty podziału. Typowe podziały tablic bazy danych to **podział pionowy** definiowany jako operacje rzutu tablicy, **podział poziomy** definiowany w oparciu o operacje selekcji, oraz **podział mieszany** wykorzystujący oba powyższe typy podziałów. Przykład drzewa podziału mieszanego

tablicy PRACOWNIK (PRAC) przedstawiono na rysunku 3, a predykaty tego podziału przedstawiono na rysunku 4.



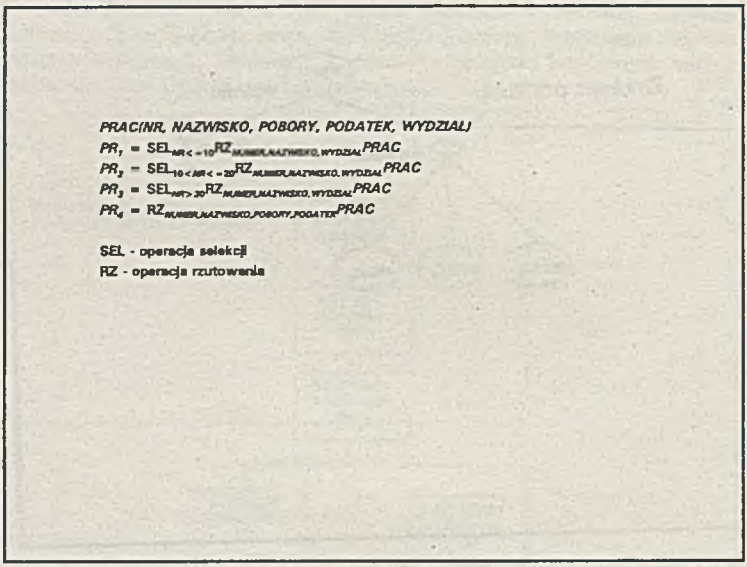
Rys. 3 Drzewo podziału tablicy PRACOWNIK.

Schemat alokacji służy do określenia zasad dystrybucji fragmentów rozproszonej bazy danych, stanowiących elementy podziału logicznego modelu danych, w ramach sieci komputerowej. Kryteria alokacji wynikają przede wszystkim z wymagań związanych z efektywnością techniczną. Fizyczna alokacja fragmentów pozwala na stosowania rozłącznej dystrybucji danych oraz redundancji danych. Spójność logiczna redundantnych danych w rozproszonej bazie danych jest zapewniana przez mechanizmy replikacji danych rozproszonego systemu zarządzania bazą danych.

Lokalne schematy bazy danych stanowią logiczne schematy autonomicznych lokalnych baz danych integrowanych w ramach systemu rozproszonej bazy danych. Możliwości integracji baz danych obejmują następujące przypadki:

- **Rozproszona baza danych** obejmująca kolekcję lokalnych baz danych zarządzanych tym samym systemem zarządzania bazą danych.
- **Homogeniczna wielobazowa (multidatabase) rozproszona baza danych** obejmująca kolekcję autonomicznych lokalnych baz danych zorganizowanych w oparciu o wspólny model danych (np. relacyjny model danych) zarządzanych przez różne systemy zarządzania bazą danych.

- **Heterogeniczna wielobazowa rozproszona baza danych** obejmująca kolekcję autonomicznych lokalnych baz danych zorganizowanych w oparciu o różne logiczne modele danych (np. relacyjny, sieciowy, hierarchiczny) zarządzanych przez różne systemy zarządzania bazą danych.



Rys 4. Predykaty podziału tablicy PRACOWNIK.

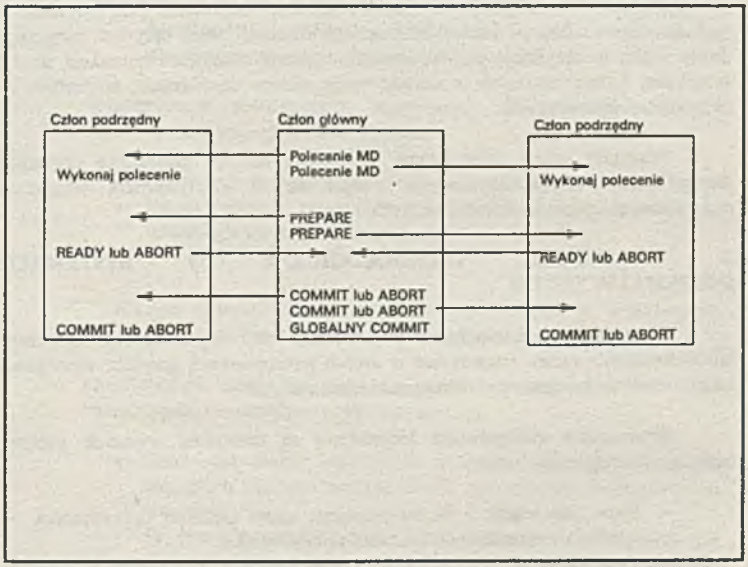
Przetwarzanie transakcyjne w obu środowiskach, to jest w środowisku centralnej bazy danych oraz rozproszonej bazy danych ma wiele wspólnych cech. Podstawową wspólną cechą jest definicja pojęcia **transakcji** w bazie danych wprowadzona przez J. Gray'a [Gray81]. Przyjmujemy następującą nieformalną definicję transakcji w bazie danych:

Transakcja jest uporządkowanym zbiorem operacji przeprowadzającym bazę danych z jednego logicznie spójnego stanu do drugiego takiego stanu, przy czym zakłada się następujące właściwości tego procesu:

- **Atomiczność** - transakcja jest wykonywana albo w całości albo, w przypadku awarii, nie pozostawia żadnych śladów w bazie danych.
- **Poprawność** - wynik działania transakcji jest zgodny z regułami spójności logicznej zdefiniowanymi w schemacie bazy danych.
- **Izolacja** - wyniki częściowo wykonanej transakcji nie mogą być widoczne dla innych współbieżnie realizowanych transakcji.

- **Trwałość** - wyniki poprawnie zakończonej (potwierdzonej) transakcji muszą na stałe pozostać w bazie danych.

Uzyskanie wymaganych właściwości transakcji wymaga zastosowania wyspecjalizowanych mechanizmów systemu zarządzania bazą danych, takich jak mechanizmy blokowania danych, wykrywania i rozwiązywania zakleszczeń, oraz potwierdzania transakcji (commit). Rozproszone systemy bazy danych wykorzystują te same mechanizmy, lecz ich implementacja jest znacznie bardziej złożona. Przykładem jest realizacja mechanizmu potwierdzania, który w przypadku rozproszonych transakcji wymaga dwu-fazowego potwierdzania. Przykład tego protokołu zakończenia transakcji przedstawiono na rysunku 5.



Rys. 5. Przykład dwu-fazowego potwierdzania rozproszonej transakcji.

Algorytm dwu-fazowego potwierdzania wymaga wymiany dużej liczby komunikatów, co może stanowić, szczególnie w przypadku systemów rozproszonych w sieciach rozległych, poważny problem w uzyskaniu wymaganych parametrów efektywności technicznej. Przykładem jest czas odpowiedzi rozproszonej transakcji (RT):

$$RT = T_{\text{człon-główny}} + MAX\,(T^{i}_{\text{człon-podrzędny}}) + T_{\text{transmisji}}$$

Zarówno czas realizacji członu głównego rozproszonej transakcji, jak i czasy realizacji poszczególnych jej członów podrzędnych, mogą być ograniczane poprzez

podejmowanie odpowiednich decyzji projektowych (algorytmy, elementy struktury danych). Natomiast czas transmisji, stanowiący sumę komunikatów wymaganych przez protokół realizacji rozproszonej transakcji jest, szczególnie w przypadku sieci rozległych, zmienna losową o dużej wariancji.

Innym istotnym problemem efektywności technicznej rozproszonych transakcji, związanym bezpośrednio z czasem odpowiedzi, jest stosowany w rozproszonych systemach zarządzania bazą danych algorytm wykrywania zakleszczenia transakcji. Jest on istotnie różny od powszechnie stosowanej w centralnych bazach danych metody wykrywania zakleszczeń opartej o graf oczekiwania. W przypadku rozproszonych systemów zarządzania bazą danych, ze względu na konieczność współpracy z innymi systemami w środowiskach heterogenicznych, powszechnie przyjmuje się metodę "limitu czasu" (timeout) polegającą na tym, że jeżeli nie zostanie uzyskana odpowiedź przewidywana w protokole komunikacji w ramach rozproszonej transakcji przed upływem przyjętego limitu czasu, to przyjmuje się, że wystąpiła sytuacja awaryjna i transakcja zostaje przerwana. Łatwo jest sobie wyobrazić wady takiego rozwiązania, szczególnie w przypadku sieci rozległych.

Powyższe cechy mają istotny wpływ na decyzje projektowe dotyczące szczególnie wymagań użytkowych związanych w efektywnością techniczną rozproszonych systemów informatycznych.

3. WYMAGANIA TECHNOLOGICZNE W SYSTEMACH INFORMATYCZNYCH

Wymagania technologiczne w stosunku do współczesnych systemów informatycznych można rozpatrywać w dwóch podstawowych grupach; wymagania efektywności technicznej oraz wymagania eksploatacyjne.

Wymagania efektywności technicznej są zazwyczaj wyrażane poprzez następujące miary efektywności:

- **Czas odpowiedzi** - liczba jednostek czasu (sekund) upływających od wysłania komunikatu do ukazania się odpowiedzi.
- **Przepustowość** - liczba transakcji obsługiwanych przez system w jednostce czasu (transakcje/sekundę).
- **Uwarunkowanie czasowe** - przekroczenie przyjętego limitu czasu odpowiedzi jest traktowane jako błędne wykonanie transakcji.

Wymagania eksploatacyjne obejmują następujące elementy:

- **Odporność na awarię** - parametr dostępności do elementów systemu (procent czasu osiągalności) oraz czas podniesienia elementu po awarii (średni czas naprawy).
- **Skalowalność systemu** - możliwość łatwego rozwoju mocy obliczeniowej w miarę rosnących wymagań użytkowych w zakresie przetwarzania.

Rozważając powyższe wymagania technologiczne z punktu widzenia projektowania rozproszonych systemów informatycznych wymienimy ich podstawowe zalety i wady. Do niewątpliwych zalet rozproszonych systemów informatycznych należą następujące ich cechy:

- Zwielokrotnienie zasobów konfiguracji systemów komputerowych wynikające z:
    - Wykorzystania "prywatnych" zasobów konfiguracji komputerowych (stacje robocze) w architekturze klient-serwer.
    - Wspólne wykorzystanie wybranych zasobów konfiguracji takich jak np. serwer bazy danych, serwer aplikacji itp.
    - Możliwość równomiernego rozkładania obciążeń (load balancing) elementów rozproszonej konfiguracji.
- Wykorzystanie równoległych algorytmów dla realizacji procesów użytkowych, w szczególności w zakresie:
    - Przetwarzania rozproszonych zapytań w relacyjnych bazach danych.
    - Przetwarzanie programów użytkowych wykorzystujących równoległe języki programowania.

Niestety powyższe zalety systemów rozproszonych są w poważnym stopniu równoważone przez ich niewątpliwe wady:

- Mogą wystąpić następujące problemy w realizacji systemów przetwarzania transakcyjnego uwarunkowanych czasowo:
    - Czas odpowiedzi, szczególnie w systemach realizowanych w rozległych sieciach komputerowych, jest zmienna losową o dużej wariancji.
    - W przewidywaniach czasu odpowiedzi trzeba uwzględnić tak trudno przewidywalne czynniki jak opóźnienie transmisji komunikatów, czas realizacji protokołu potwierdzania rozproszonych transakcji, opóźnienie wynikające z wykrywania i rozwiązywania rozproszonych zakleszczeń.
- Replikacja danych nie zawsze jest wystarczającym rozwiązaniem, a szczególnie w przypadku gdy mechanizm replikacji danych nie jest powiązany z algorytmem optymalizacji rozproszonych zapytań.

Typowe zadanie projektowe w dziedzinie projektowania efektywnych technicznie systemów rozproszonych jest zagadnieniem optymalizacji następujących alternatywnych funkcji celu:

- **Minimalizacja kosztów realizacji** rozproszonych transakcji, przy czym koszt realizacji jest liniową kombinacją kosztów jednostki centralnej, operacji dostępu do pamięci zewnętrznej oraz transmisji danych.
- **Minimalizacja czasu odpowiedzi**, który jest w przybliżeniu równy czasowi trwania najdłuższej ścieżki wykonania rozproszonej transakcji.

Wybór kryterium projektowego jest naturalnie wynikiem analizy wymagań konkretnego systemu użytkowego. Przy czym pierwsze kryterium jest typowe w sytuacjach gdzie zachodzi potrzeba równomiernego rozłożenia obciążeń przetwarzania w ramach rozproszonej konfiguracji.

4. PROJEKTOWANIE ROZPROSZONYCH SYSTEMÓW INFORMATYCZNYCH

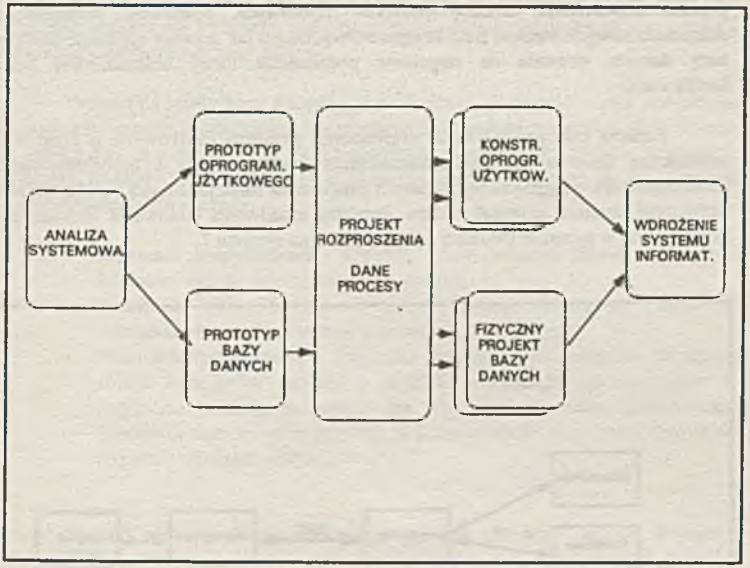
Prezentowany na rysunku 6 cykl realizacji rozproszonych systemów informatycznych stanowi rozwinięcie konwencjonalnej metodyki realizacji systemów wykorzystującej techniki "szybkiego prototypowania" [Rodan94]. Istotną zaletą tego podejścia jest znacznie lepszy kontakt zespołu realizującego z przyszłymi użytkownikami systemu informatycznego, którzy dzięki możliwości aktywnej partycypacji w procesie realizacji systemu stają się de facto jego współtwórcami. Fazy "szybkiego prototypowania" przebiegają zazwyczaj w środowisku realizacyjnym, a nie w docelowym środowisku eksploatacyjnym, tym samym mogą być realizowane jako system scentralizowany, ponieważ podstawowym celem próbnej eksploatacji jest weryfikacja interface'u użytkownika, funkcji użytkowych systemu, oraz jego zakresu informacyjnego. Istotną natomiast cechą operacyjnego prototypu systemu informatycznego jest jego próbna eksploatacja w oparciu o dane rzeczywiste.

Projekt rozproszenia, obejmujący zarówno bazę danych jak i procesy użytkowe, musi zostać zakończony, z przyczyn oczywistych, przed rozpoczęciem faz konstrukcji oprogramowania użytkowego i bazy danych. Projektowanie rozproszonych systemów informatycznych obejmuje następujące trzy podstawowe dziedziny:

- Projektowanie rozproszenia procesów użytkowych w oparciu o architekturę klient-serwer.
- Projektowanie rozproszonej bazy danych.
- Integracja istniejących baz danych.

Oczywiście konkretny projekt rozproszonego systemu może dotyczyć jednej z powyższych dziedzin lub ich dowolnej kombinacji. Projektowanie rozproszonej bazy danych odbywa się zgodnie z przedstawioną na rysunku 2 architekturą schematów rozproszonej bazy danych. Zgodnie z klasycznym podejściem do projektowania baz danych projekt schematu globalnego, stanowiącego logiczny model bazy danych jest poprzedzony projektem pojęciowego schematu bazy danych stanowiącego wynik fazy analizy. Tak więc projektowanie rozproszonej bazy danych odbywa się zgodnie z

klasycznym podejściem "od ogółu do szczegółu". W przypadku integracji istniejących baz danych punktem wyjściowym jest analiza semantyki logicznych schematów istniejących baz danych a następnie ich integracja. Podejście projektowe jest w tym przypadku podobne do problemu integracji cząstkowych schematów pojęciowych [Batini83]. Każda z powyższych dziedzin projektowania rozproszenia obejmuje inne problemy projektowe i wymaga zastosowania innego podejścia i technik projektowania.



Rys. 6. Cykl życiowy realizacji rozproszonego systemu informatycznego.

4.1 Projektowanie rozproszenia procesów użytkowych w oparciu o architekturę klient-serwer

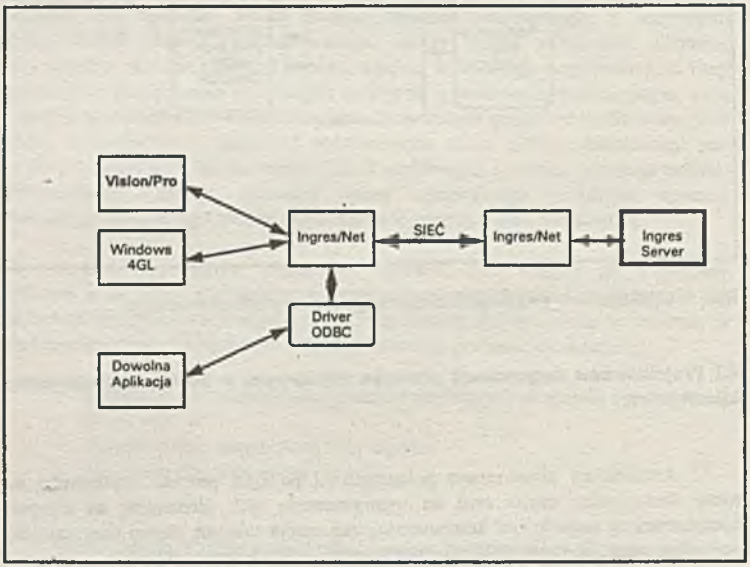
Architektura klient-serwer polegająca na podziale procesu użytkowego na różne funkcjonalne części oraz na rozmieszczeniu tych elementów na różnych komputerach w ramach sieci komputerowej (np. stacja robocza, serwer bazy danych) pozwala na wykorzystanie następujących zalet systemów rozproszonych:

- Wykorzystanie mocy obliczeniowych komputerów dostępnych w sieci przy jednoczesnym istotnym ograniczeniu liczby komunikatów.
- Równomierne rozłożenie obciążeń elementów rozproszonej konfiguracji.
- Stopniowy rozwój konfiguracji.

Przykładem wykorzystania mocy obliczeniowej dostępnej w sieci jest różnica pomiędzy konfiguracją serwer bazy danych - X-terminale a serwer bazy danych - stacje robocze. W tym drugim przypadku obsługa interface'u użytkownika nie wymaga żadnych komunikatów i wykorzystuje jedynie procesor stacji roboczej. W przypadku stosowania graficznego interface'u użytkownika takie rozłożenie obciążenia może mieć istotne znaczenie.

Tworzenie mieszanych konfiguracji, takich jak przykład przestawiony na rysunku 1, pozwala na równoważenie wykorzystania poszczególnych urządzeń poprzez odpowiednią alokację procesów użytkowych. Możliwość stopniowego dołączania nowych węzłów sieci komputerowej, takich jak serwery aplikacji, serwery bazy danych, pozwala na stopniowe podnoszenie mocy obliczeniowej całej konfiguracji.

Funkcją celu projektowania rozproszenia procesów użytkowych w oparciu o architekturę klient-serwer jest minimalizacja kosztu sprzętu i oprogramowania koniecznych dla osiągnięcia wymaganych parametrów funkcjonalności i efektywności technicznej systemu informatycznego. Przykład możliwości budowania architektury klient-serwer w systemie INGRES przedstawiono na rysunku 7.



Rys. 7. Schemat architektury klient-serwer w środowisku narzędziowym systemu INGRES.

4.2 Projektowanie rozproszonej bazy danych

Projekt logicznego modelu rozproszonej bazy danych reprezentowanego przez globalny schemat jest realizowany w konwencjonalny sposób poprzez odwzorowanie semantyki schematu pojęciowego. Elementy pozarelacyjne, takie jak reguły wnioskowania, zdarzenia, oraz procedury bazy danych stanowią elementy lokalnych schematów bazy danych i nie są widoczne na poziomie globalnym. Na tym poziomie projektu nie występują praktycznie żadne różnice pomiędzy projektowaniem centralnej i rozproszonej bazy danych.

Projekt podziału bazy danych jest analogiczny do projektowania perspektyw (views) użytkowych w relacyjnych bazach danych. Predykaty podziału są definiowane w języku danych systemu ZBD (SQL) określając odpowiednio pionowe, poziome, i mieszane podziały tablic bazy danych. Definicja fragmentów bazy danych powinna spełniać następujące warunki:

- **Warunek kompletności** - wszystkie dane globalnej tablicy muszą być odwzorowane we fragmentach bazy danych.
- **Warunek rekonstrukcji** - należy zawsze zapewnić możliwość rekonstrukcji globalnej tablicy w oparciu o jej fragmenty.
- **Warunek rozłączności** - fragmenty powinny być rozłączne, w sensie zbioru wierszy w przypadku podziału poziomego lub atrybutów w przypadku podziału pionowego (za wyjątkiem atrybutów kluczowych). Spełnienie tego warunku pozwala na kontrolowanie redundancji danych na poziomie schematu alokacji.

Projekt alokacji fragmentów bazy danych obejmuje decyzje dotyczące alokacji fragmentów bazy danych w sieci komputerowej oraz, jeżeli wystąpi potrzeba wprowadzenia redundancji danych, decyzje dotyczące replikacji danych. Projektowanie alokacji fragmentów jest realizowane w oparciu o następujące kryteria:

- **Lokalność przetwarzania** - umieszczanie danych jak najbliżej procesów użytkowych, które je wykorzystują, przy jednoczesnym spełnieniu ograniczeń technicznych i administracyjnych.
- **Dostępność i bezpieczeństwo danych** - tworzenie i rozpraszanie redundantnych kopii fragmentów tablic.
- **Rozkładanie obciążenia** - wykorzystanie możliwości przetwarzania równoległego i równomiernego obciążenia różnych węzłów sieci komputerowej.
- **Dostępność i koszt urządzeń pamięci zewnętrznej** - rozmieszczanie danych w węzłach sieci komputerowej w miarę dostępności pamięci zewnętrznej.

4.3 Integracja istniejących baz danych

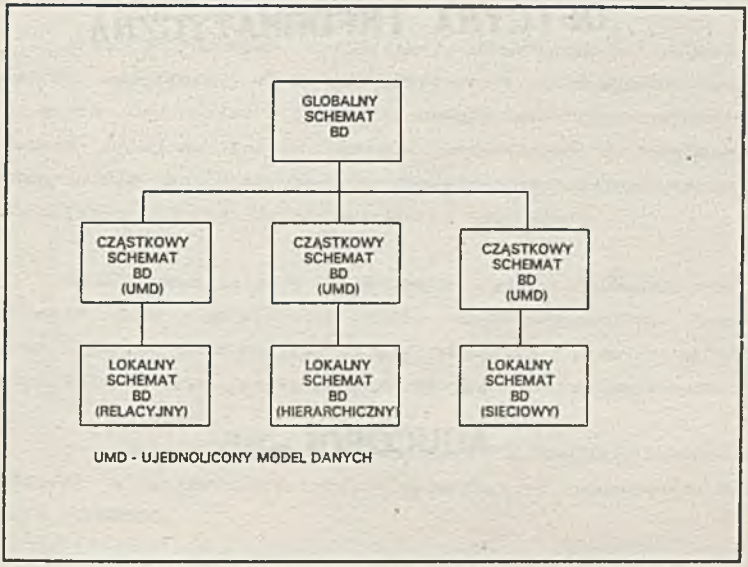
Architekturę schematów rozproszonego systemu informatycznego integrującego heterogeniczne bazy danych przedstawiono na rysunku 8. Proces tworzenia systemu rozproszonej bazy danych o architekturze wielobazowej (multidatabase) obejmuje następujące kroki:

- **Translacja modeli danych** - odwzorowanie heterogenicznych modeli danych w ujednolicony model danych. Najczęściej w tego typu systemach ujednoliconym modelem danych jest model relacyjny.
- **Integracja modeli danych** - połączenie wielu cząstkowych ujednoliconych modeli danych przy jednoczesnym rozwiązaniu istniejących konfliktów danych (np. nazwy, struktura).
- **Transformacja globalnego schematu** - analiza możliwych wariantów globalnego schematu w celu podniesienia jego walorów użytkowych i efektywności technicznej.

Rozwiązywanie konfliktów nazw obejmuje rozwiązywanie *synonimii* i *homonimii* oraz identyfikację *podobieństw* klas encji. Podobieństwo klas encji polega na występowaniu dwóch różnych klas encji zawierających te same atrybuty oraz ograniczenia spójności logicznej.

Rozwiązywanie konfliktów struktury odbywa się zawsze na poziomie klas encji i wymaga identyfikacji następujących warunków: *identyczne pojęcia* - ta sama struktura klasy i otoczenia (powiązania), *kompatybilne pojęcia* - inna struktura reprezentacji i/lub otoczenia przy jednoczesnym braku kontradykcji, *niekompatybilne pojęcia* - zawierają sprzeczne własności (np. różna funkcjonalność powiązań, różne identyfikatory, odwrotne relacje zawierania).

Transformacja globalnego schematu polega na kolejnych krokach restrukturyzacji schematu w celu uzyskania następujących pożądanych własności: *kompletność* - reprezentacja wszystkich istotnych elementów, *poprawność* - poprawność syntaktyczna i spójność semantyczna, *minimalność* - schemat jest minimalny jeżeli usunięcie jakiegokolwiek elementu obniża jego zawartość informacyjną, *ekspresyjność* - naturalna reprezentacja modelu rzeczywistości nie wymagająca dalszych wyjaśnień, *czytelność* - zachowanie estetycznych wymagań dotyczących graficznej reprezentacji (standardowe symbole, przebieg linii), *normalizacja* - zachowanie zasad normalizacji relacyjnego modelu danych.



Rys. 8. Architektura schematów systemu integrującego istniejące bazy danych.

LITERATURA

Batini85 Batini, C., Lenzerini, M., Moscarini, M., Views Integration, w Ceri S. (ed.), Methodology and Tools for Database Design, North Holland, Amsterdam 1983.

Ceri85 Ceri, S., Pelagatti, G., Distributed Databases - Principles & Systems, McGraw-Hill Computer Science Series, 1985.

Gray81 Gray, J.N., The Transaction Concept: Virtues and Limitations, 7th Int. Conf. on VLDB, Cannes 1981.

Rodan94 Metodyka projektowania i realizacji systemów informatycznych, Rodan-System Sp. z o.o., Warszawa 1994 (w przygotowaniu).

Stan90 Mostardi, T., Staniszkis, W., Multidatabase system design methodology, Database Technology, Vol. 3, No 1, pp 27-37, Pergamon Press, 1990.

OFICYNA INFORMATYCZNA

ACUCOBOL-85

Opracowali:

Stanisława Ossowska
Andrzej Wiśniewski
Artur Lempkowski
Janusz Ludwiak
Artur Nurzyński

Warszawa, kwiecień 1994 rok

Wraz z rozwojem technologii programowania zmienia się zakres możliwości i sposób użytkownaia języków programowania, mających swój udział w historii rozwoju dziedziny programowania. Jednym z ojców takich języków stosowanych do komputerowego wspomagania prac ewidencyjno-sprawozdawczo-obliczeniowych jest COBOL (ang. COmmon Business-Oriented Language).

COBOL jest ciągle stosowanym językiem programowania na świecie i ma ciągle liczne rzesze swoich zwolenników. Żywotność tego języka jest możliwa dzięki prowadzeniu prac rozwojowych i włączaniu do tego języka nowych rozwiązań technologicznych.

W ostatnim okresie technologie 4GL znalazły zastosowanie również w języku COBOL, dzięki interfejsowi programowemu ACU4GL dla ACUCOBOL.

Możliwości współpracy języka proceduralnego COBOL z systemami 4GL poprzez budowę interfejsu programowego jakim jest ACU4GL, pozwala w znacznym stopniu rozszerzyć zakres aplikacji danego systemu. Dzięki niezależności programu źródłowego od konkretnego systemu 4GL jest możliwa realizacja programów w ACUCOBOL-85 we współpracy z różnymi systemami 4GL. Daje to nowe możliwości opracowywania i wdrażania aplikacji, szczególnie cenne dla programistów doświadczonych w stosowaniu języka COBOL do tworzenia programów.

S. Ossowska i M. A. Wiśniewski

1. Charakterystyka kompilatora ACUCOBOL-85

ACUCOBOL-85 jest zgodny ze standardem ANSI 85 dla kompilatora COBOL oraz zgodny z ANSI x3.23-1985 dla systemu wykonawczego. ACUCOBOL-85 wykonuje różne funkcje, przypisane poszczególnym modułom, zakwalifikowanym do określonych poziomów. Poniżej przedstawiono wybrane moduły i przyjete dla nich poziomy realizacji.

- Nucleus (1-2)	level 2
- Sequential I-O (1-2)	level 2
- Relative I-O (0-2)	level 2
- Indexed I-O (0-2)	level 2
- Inter-Program (1-2) Communication	level 2
- Sort-Merge (0-1)	level 1
- Source Text (0-2)	level 2
- Segmentation (0-2)	level 1

ACUCOBOL nie wspomaga opcjonalnych modułów takich jak: Report Writer, Communication, Debug.

W celu dostarczenia odpowiedniego rozwoju dla platformy COBOL współpracują ze sobą: szybki kompilator oraz debuger pracujący na poziomie źródłowym i dostarczający zrozumiałych informacji o błędach. Przenośny kod wynikowy, jednolity system plikowy i niezależne od urządzeń zewnętrznych terminale ułatwiają stosowanie, a tym samym zainteresowanie rozwojem aplikacji ACUCOBOLu-85.

Jedną z podstawowych zalet tego produktu jest przenośność programowa. ACUCOBOL-85 może pracować bezpośrednio na kodach źródłowych trzech popularnych kompilatorów, a mianowicie:

- Ryan McFarland's RM/COBOL
- Data General Corporation's Interactive COBOL (Icobol)
- Digital Equipment Corporation's VAX COBOL

co pozwala programom poruszać się między tymi środowiskami z minimalnym wysiłkiem.

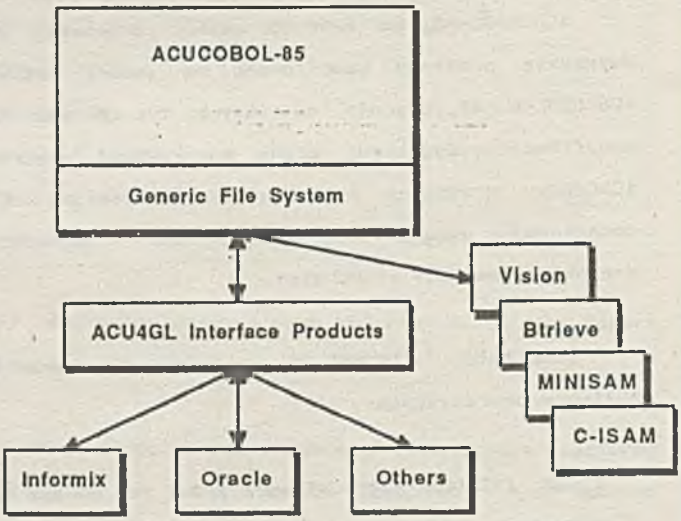
ACUCOBOL-85 ma również pełny, przenośny kod wynikowy. Wszystkie programy kompilowane na jednej maszynie z użyciem ACUCOBOL-a 85 pracują na innych typach maszyn bez potrzeby modyfikacji. Dodatkowo można modyfikować program napisany w ACUCOBOLu w czasie pracy w tym środowisku bez konieczności dodatkowej rekompilacji. Dzięki tym właściwościom ACUCOBOL szerokiego ma duży rynek zbytu.

Kompilator ACUCOBOL-85 pracuje pod nadzorem następujących systemów operacyjnych:

1. DOS (MS-DOS i PC-DOS wersja 2.1 lub nowsza lub każdy inny system, który może wykonywać programy w DOS-ie jako podzbiór),
2. OS/2,
3. UNIX (każdy system UNIX-owy wliczając w to ULTRIX i XENIX),
4. VMS (dla komputerów VAX),
5. AOS/VS (dla komputerów serii MV Data General).

2. Współpraca ACUCOBOL-85 z systemami zarządzania bazami danych

Pracując w ACUCOBOL-85 (ANSF-1985 COBOL) można mieć dostęp do systemów zarządzania bazami danych (DBMS). W celu umożliwienia dostępu do systemów zarządzania bazami danych ze środowiska ACUCOBOL-85 zastosowano interfejs ACU4GL. Zadaniem tego interfejsu (najogólniej rzecz ujmując) jest przyporządkowanie zdaniom I/O z COBOL-u odpowiednich instrukcji z języka danego DBMS (np. Informix, ORACLE i innych).



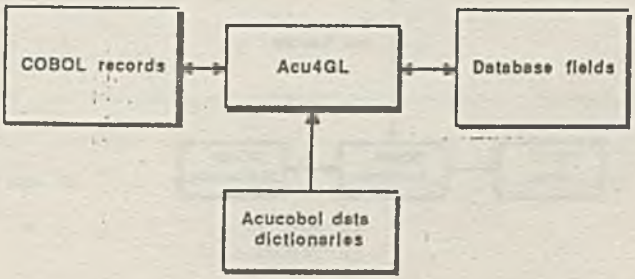
Rys. 1. Rodzaje dostępu do danych ze środowiska ACUCOBOL

Standardowo systemami plikowymi w ACUCOBOL-85 zarządza specjalny system plikowy Vision. Jeśli jest to VAX COBOL, to

rolę Vision pełni RMS, a dla ICOBOL (Interactive Cobol) jest to C-ISAM. Zgodnie z potrzebami użytkownika można stosować zamiennie albo poszczególne systemy plikowe, albo DBMS (np. Informix, Oracle). Osiąga się to dzięki przyjętemu rozwiązaniu polegającemu na tym, że wszystkie połączenia pomiędzy odpowiednimi zdaniami ACUCOBOL-85 a systemami zarządzania plikami, czy bazami danych, są realizowane przez Generator Systemów Plikowych.

Schemat współdziałania między ACUCOBOL-85, systemami plikowymi i systemami zarządzania bazami danych przedstawiono na rysunku 1. Pokazano tu również powiązania generatora systemów plikowych i interfejsu ACU4GL.

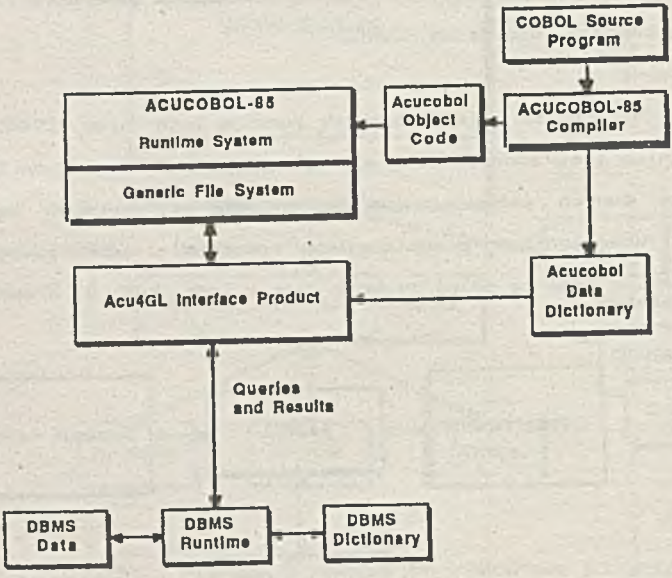
Wyróżniamy dwie podstawowe funkcje interfejsu ACU4GL. Jedną z nich jest *sporządzenie słownika danych*. Zważywszy na fakt, że baza danych pracuje na polach, a COBOL na rekordach, konieczne jest utworzenie połączeń pomiędzy rekordami i odpowiadającymi im polami. Rolę tę pełni słownik danych (zob. rys. 2). Pomaga on w



Rys. 2. Słowniki danych

rozpoznawaniu rekordów COBOLowych w polach bazy danych i odwrotnie. Oczywiste jest, że dla systemów plikowych zbiorów indeksowych nie ma potrzeby robienia słownika danych, co wynika z ich natury.

Inną funkcją ACU4GL jest *translacja między COBOLem a SQLem.* Otóż, dostęp do relacyjnych baz danych jest osiągany poprzez opisanie tego dostępu w języku SQL. Konieczna jest tu znajomość jego instrukcji odpowiednich dla specyficznych baz DBMS, do których zamierza się mieć dostęp.



Rys. 3. Interfejs ACU4GL jako rozszerzenie ACUCOBOL-85

Ponieważ wyżej wymienione zagadnienie jest związane z DBMS-em, instrukcje SQL-a powinny być zmieniane wówczas, gdy chcemy współpracować z innym DBMS lub innym indeksowym systemem plikowym. W praktyce więc mamy tyle produktów Acu4GL z iloma bazami danych chcemy współpracować.

Produkty ACU4GL posiadają "plastyczny" interfejs pomiędzy programem a relacyjną bazą danych. Nazwa "plastyczny" pochodzi stąd, że komunikacja pomiędzy programami a bazami przebiega gładko i nie wymaga ze strony użytkownika jakichkolwiek dodatkowych danych. Program w COBOL-u nie musi ulegać żadnej zmianie, jeśli chcemy korzystać z innej bazy danych lub z innego systemu plików indeksowanych. Wymiana informacji poprzez interfejs odbywa się w taki sposób, że wygląda to jakby baza danych i COBOL były jedną częścią tego samego procesu (rys. 3). Przykładowo, jeśli mamy program w COBOLu, który używa zdania READ, to jest ono automatycznie interpretowane przez interfejs jako odwołanie do bazy danych, przy czym czytana z bazy dana jest automatycznie przesyłana do właściwego rekordu COBOLowego.

***

Osoby zainteresowane działaniem ACUCOBOLu zapraszamy na prezentacje zarówno tu, w Swinoujściu, jak i w Warszawie (ul. Rzymowskiego 36, tel. 0*2 43-61-94, 43-67-51, fax 43-63-60).

PROGRESS 4GL

CSBI

Zintegrowane środowisko do tworzenia i utrzymywania systemów informatycznych

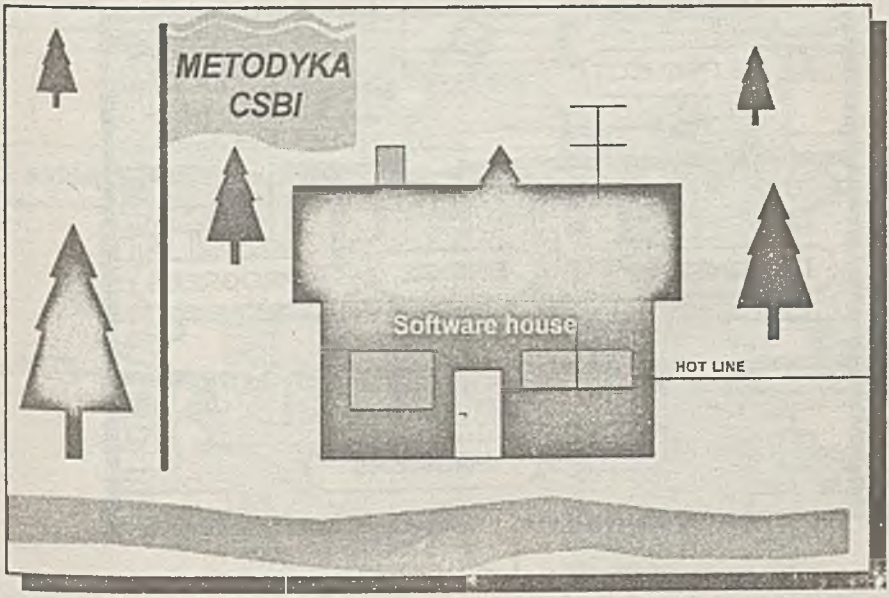
COMPUTER SYSTEMS FOR BUSINESS INTERNATIONAL LTD.
02-119 WARSZAWA, UL. PRUSZKOWSKA 17
TEL. (0-2) 659-04-16, FAX (0-2) 659-04-85, TLX 825085 CSBI
CSBI PROGRESS GROUP
00-215 WARSZAWA, UL. SAPIEŻYŃSKA 3
TEL. (0-22) 31-37-47, FAX (0-2) 635-65-58

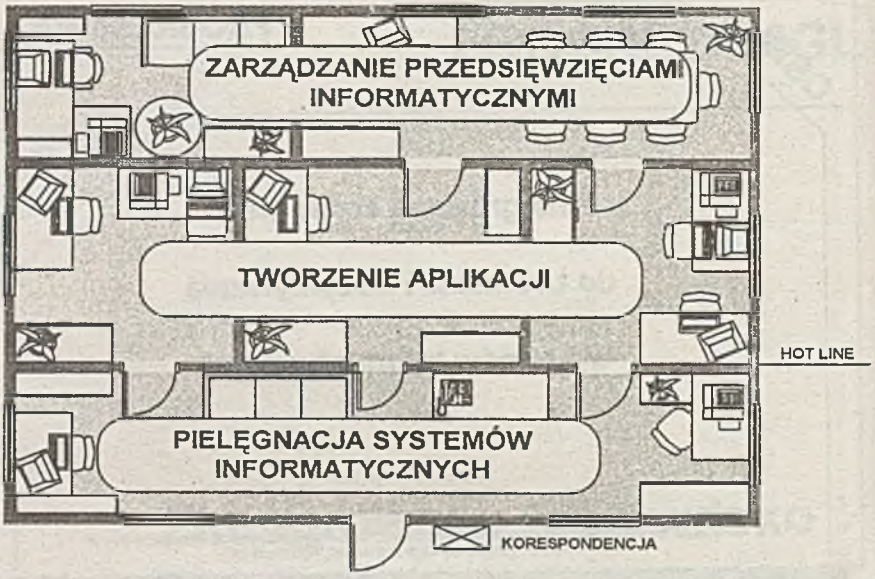
CSBI

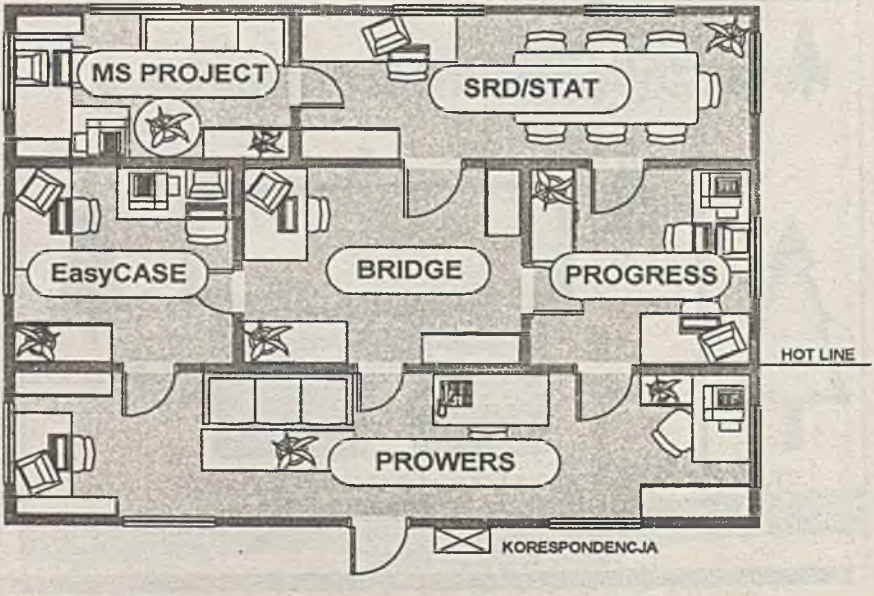
PROGRESS 4GL

Zintegrowane środowisko do tworzenia i utrzymywania systemów informatycznych

COMPUTER SYSTEMS FOR BUSINESS INTERNATIONAL LTD. 02-119 WARSZAWA, UL. PRUSZKOWSKA 17
TEL. (0-2) 659-04-18, FAX (0-2) 659-04-85, TLX 825085 CSBI




ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI
TWORZENIE APLIKACJI
HOT LINE
PIELĘGNACJA SYSTEMÓW INFORMATYCZNYCH
KORESPONDENCJA


MS PROJECT
SRD/STAT
EasyCASE
BRIDGE
PROGRESS
HOT LINE
PROWERS
KORESPONDENCJA

ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI

- strategia rozwoju systemów informatycznych
- struktura organizacyjna przedsięwzięć informatycznych
- planowanie i inicjacja przedsięwzięcia informatycznego
- prowadzenie i kontrola realizacji przedsięwzięcia informatycznego

ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI

strategia rozwoju systemów informatycznych

- plan strategiczny informatyzacji
- analiza (przed)wstępna
- SWOT, analiza ryzyka
- analiza kosztów

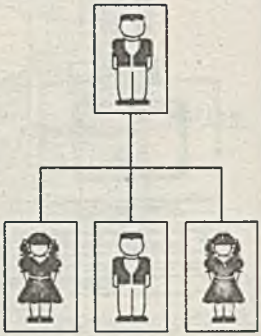
baza wiedzy o przedsięwzięciach informatycznych (zasobach, terminach, kosztach, zależnościach)

ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI

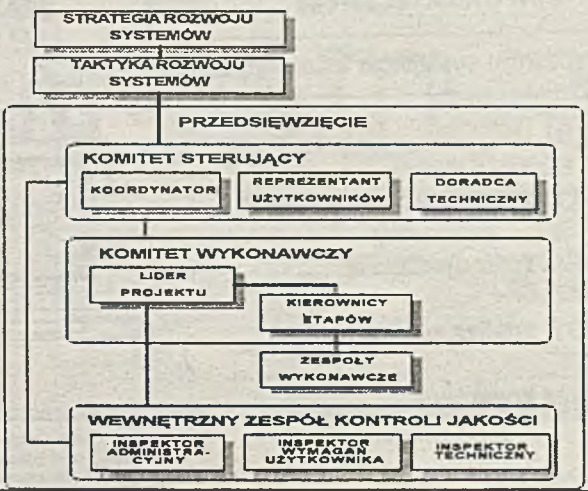
struktura organizacyjna przedsięwzięć informatycznych

organizacja pracy, raportowanie oraz przekazywanie i rejestracja poleceń (zadań) i wyników prac (realizacja zadań)

- komitety
- role
- odpowiedzialność
- kompetencje



ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI



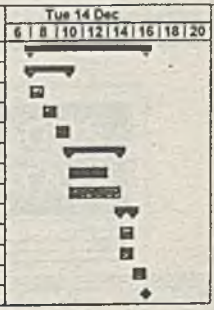
ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI

planowanie i inicjacja przedsięwzięcia informatycznego

- dekompozycja na fazy, zadania, czynności elementarne
- określenie terminów i produktów przedsięwzięcia
- przydział środków i zasobów (budżet, ludzie, sprzęt, pomieszczenia)
- opracowanie regulaminu przedsięwzięcia
- ustalenie punktów kontrolnych i punktów krytycznych przedsięwzięcia
- identyfikacja ścieżki krytycznej
- system zapewnienia jakości produktów i wykonywanych prac

proces planowania przedsięwzięcia, szacowanie kosztów, optymalizacja wykorzystywanych zasobów, aktualizacja terminów

ID	Nazwa zadania	Czas	Nazwiska osób realizujących zadanie	Poprz.
1	Przygotowanie prezentacji	1d		
2	Koncepcja	0,38d		1
3	Założenia	1h	Marcin Bańkowski,Ryszard Kasprzyk	2
4	Dobór przykładów	1h	Marcin Bańkowski,Ryszard Kasprzyk	3
5	Plan	1h	Marcin Bańkowski,Ryszard Kasprzyk	4
6	Przygotowanie przykładów	0,38d		5
7	Microsoft Project	2h	Ryszard Kasprzyk	6
8	EasyCASE	3h	Marcin Bańkowski	6
9	Opracowanie folii	0,13d		7,8
10	Folie Microsoft Project	1h	Ryszard Kasprzyk	9
11	Folie EasyCASE	1h	Marcin Bańkowski	9
12	Wydrukowanie folii	1h	Marcin Bańkowski,Ryszard Kasprzyk	10,11
13	Gotowa prezentacja	0d		1



Project: METODYKA CSBI
Date: 28/4/94

Critical — Progress — Summary
Noncritical — Milestone — Rolled Up

Page 1

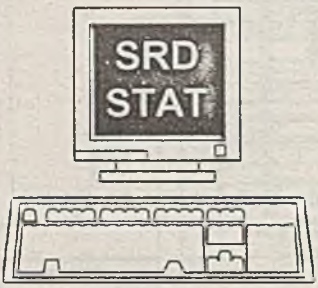
ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI

prowadzenie i kontrola realizacji przedsięwzięcia informatycznego

- koordynacja i integracja projektów
- śledzenie postępów prac, wykorzystania zasobów i poniesionych nakładów
- dokumentacja przedsięwzięcia, dokumentacja techniczna
- rejestracja zmian, aktualizacja harmonogramu, kontrola budżetu
- kontrola jakości produktów i jakości zarządzania przedsięwzięciem
- auditing przedsięwzięcia
- zmiany organizacyjne i technologiczne

ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI

prowadzenie i kontrola realizacji przedsięwzięcia informatycznego



bieżąca rejestracja i klasyfikacja prowadzonych prac,
ewidencja i raportowanie z podziałem czasu na zadania,
ocena czasochłonności zadań i zaangażowania osób i zespołów,
rozbudowa bazy wiedzy o przedsięwzięciach informatycznych

kontrola realizacji przedsięwzięcia,
aktualizacja planów, wariantowość planów,
analiza możliwych i symulacja przewidywanych
scenariuszy działań

SRD - System Rejestracji Działań

Raport ze zużytego czasu przez osoby na prace w poszczególnych tematach
od tygodnia 01 (01/03/94 - 01/07/94) do tygodnia 16 (04/18/94 - 04/22/94)

Lp.	Temat	Czas (w osobo-tygod.)
	Andrzej Trojnar	
1	Integracja systemu PROmiss	6.50
2	Prace związane z narzędziami typu CASE	2.25
3	Dystrybucja systemu PROMIS	2.20
4	Instalacje systemu PROMIS	1.60
5	Delegacje inne	0.80
6	Działalność marketingowa	0.65
7	Prowadzenie szkoleń	0.60
8	Rozwój systemu AD	0.40
9	Urlopy/Zwolnienia	0.40
10	Badanie i rozwój	0.40
11	Działania administracyjno-organizacyjne	0.20
Razem		16.00
RAZEM		16.00

ZARZĄDZANIE PRZEDSIĘWZIĘCIAMI INFORMATYCZNYMI

Etapy rozwoju zastosowań metodyki:

wiedza i doświadczenie

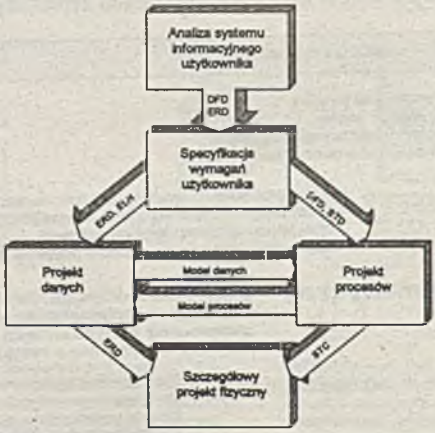
- **żywiołowy**
  - realizacja pod terminy
  - brak analizy ryzyka
  - brak sformalizowanych procedur
  - permanentna zmiana planów
- **prowizoryczny**
  - próby planowania, podział projektu na etapy
  - próby zastosowania narzędzi informatycznych
  - brak pełnej dokumentacji projektowej
  - brak kontroli jakości produktów pośrednich
- **schematyczny**
  - ustanowienie komórki METODYKA
  - bezpośrednie przejęcie dostępnej metodyki
  - zastosowanie technik i narzędzi do realizacji projektu
  - trudności w oszacowaniu kosztów i zasobów
- **adaptowalny**
  - przemysłowy proces realizacji projektu
  - zastosowanie narzędzi do zarządzania projektem
  - system zapewniania jakości procesu produkcji
  - koordynacja prac i integracja projektów

TWORZENIE APLIKACJI

- analiza
- projektowanie
- kodowanie
- testowanie
- reengineering

TWORZENIE APLIKACJI

METODYKA ANALIZY I PROJEKTOWANIA



TWORZENIE APLIKACJI

METODYKA REALIZACJI

- ANALIZA WSTĘPNA - BADANIE AKTUALNEGO SYSTEMU INFORMACYJNEGO: DD, OGÓLNE DFD I ERD
- UZGODNIENIE GRANIC ANALIZOWANEGO SYSTEMU, WERYFIKACJA DD, DFD I ERD
- ANALIZA POTRZEB I WYMAGAŃ UŻYTKOWNIKA, USTALENIE LISTY WYMAGAŃ NA SYSTEM
- MODEL LOGICZNY PRZYSZŁEGO SYSTEMU: ERD, DFD, ELH, STD
- WERYFIKACJA SPECYFIKACJI PRZEZ UŻYTKOWNIKA
- SCENARIUSZE ROZWIĄZAŃ TECHNICZNYCH, USTALENIE SPOSOBU IMPLEMENTACJI SYSTEMU
- MODEL IMPLEMENTACYJNY SYSTEMU: ZNORMALIZOWANY ERD, STC
- BUDOWA STRUKTURY BAZY DANYCH, TWORZENIE PROGRAMÓW I PROCEDUR
- TESTOWANIE PROGRAMÓW I MODUŁÓW, TESTOWANIE SYSTEMU
- WDRAŻANIE, ODBIÓR, PIELĘGNACJA SYSTEMU

TWORZENIE APLIKACJI

analiza

- analiza systemu informacyjnego
- analiza dziedziny
- analiza potrzeb
- analiza systemu informatycznego

TWORZENIE APLIKACJI

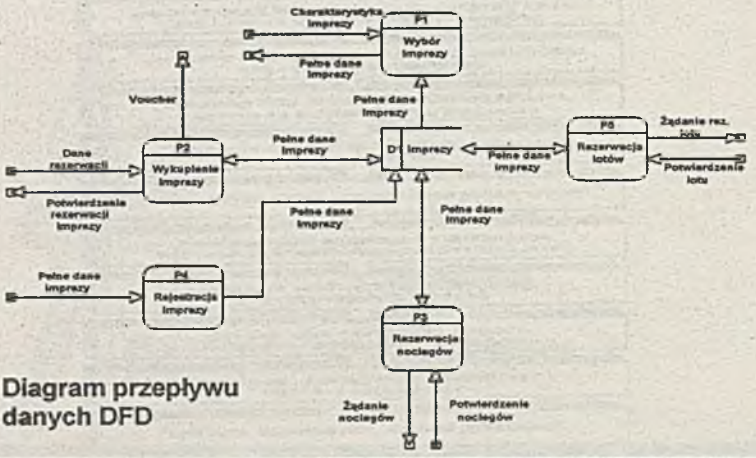


Diagram przepływu danych DFD

TWORZENIE APLIKACJI

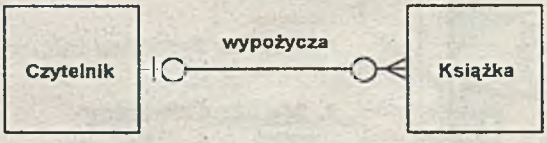


Diagram obiekt-związek ERD (poziom pojęciowy)

TWORZENIE APLIKACJI

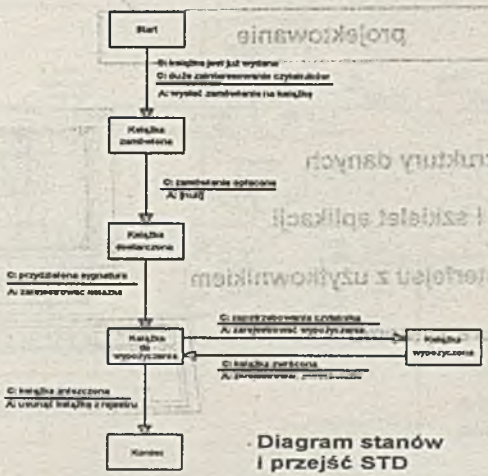


Diagram stanów
i przejść STD

TWORZENIE APLIKACJI

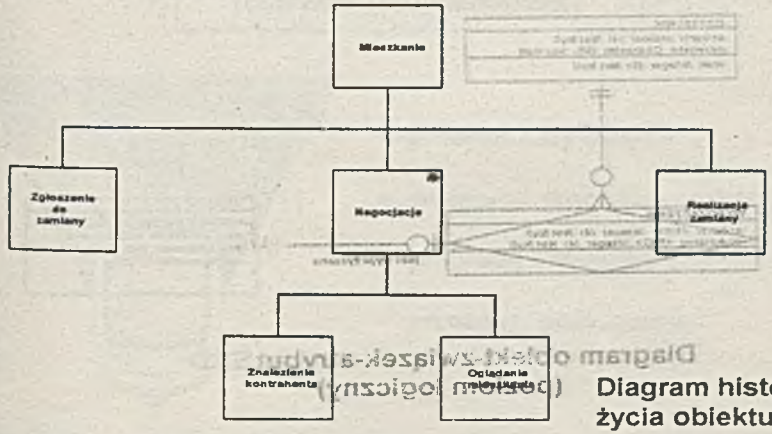
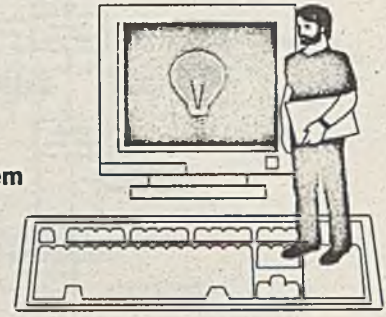


Diagram historii
życia obiektu ELH

TWORZENIE APLIKACJI

projektowanie

- projekt struktury danych
- struktura i szkielet aplikacji
- projekt interfejsu z użytkownikiem
- architektura systemu



TWORZENIE APLIKACJI

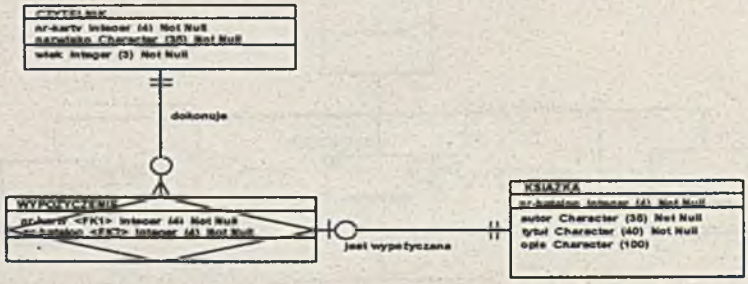


Diagram obiekt-związek-atrybut ERD (poziom logiczny)

TWORZENIE APLIKACJI

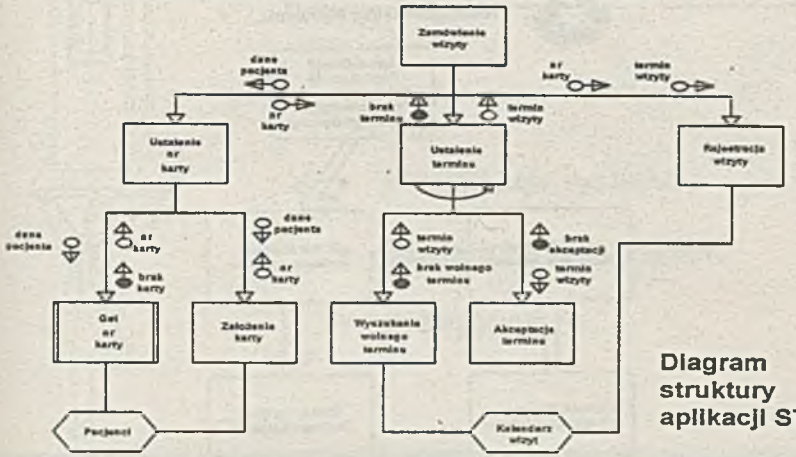
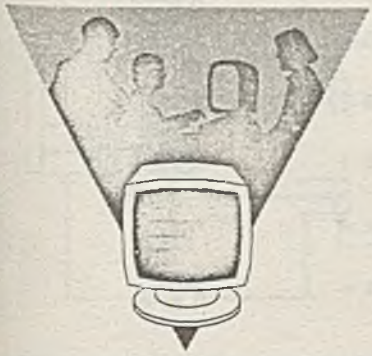


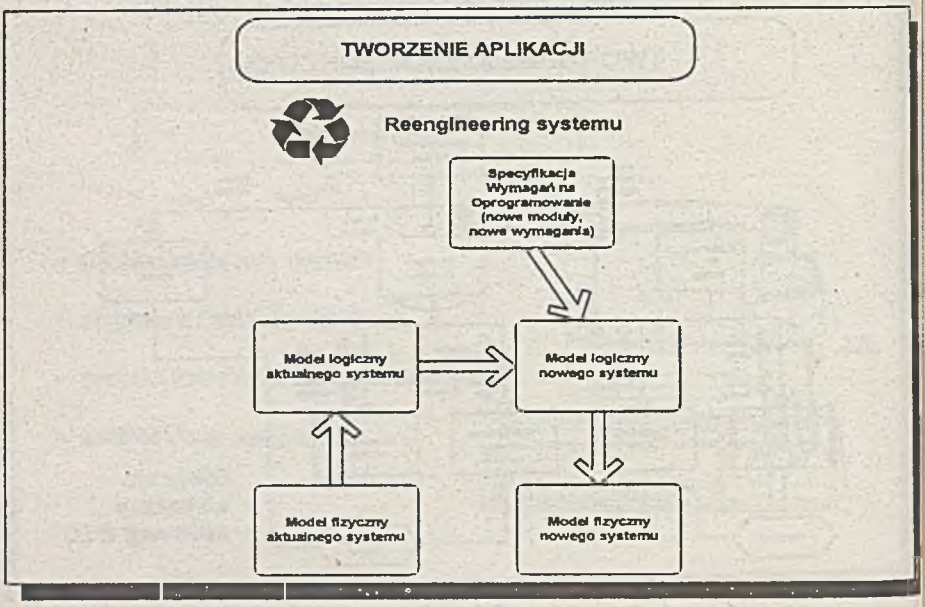
Diagram struktury aplikacji STC

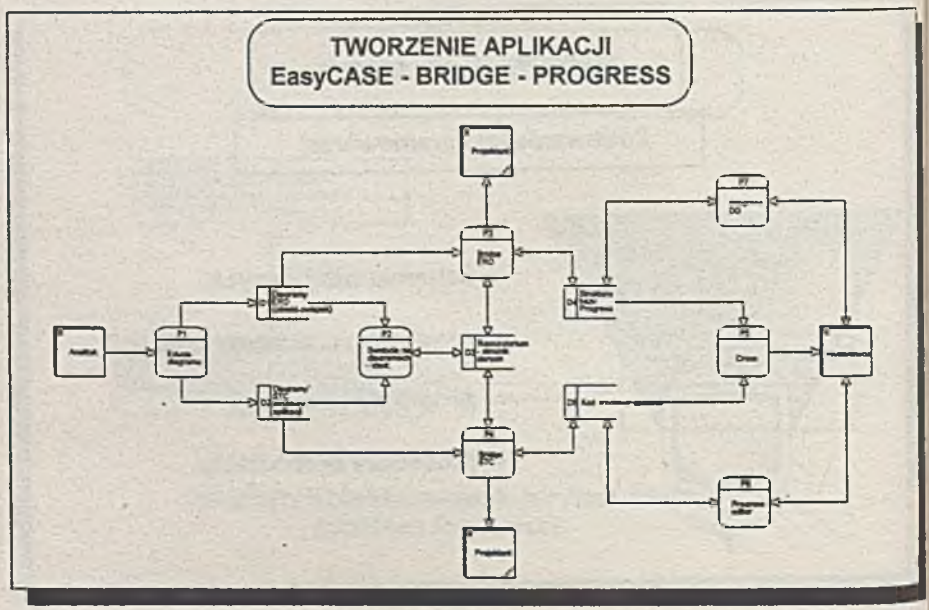
TWORZENIE APLIKACJI

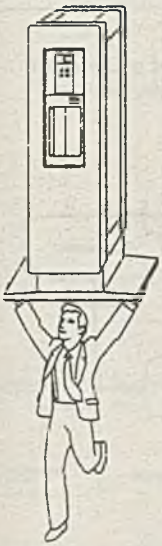
kodowanie (programowanie)



- schemat bazy danych
- programy, procedury aplikacji
- programy konwersji
- procedury pomocnicze


TWORZENIE APLIKACJI
Reengineering systemu
Specyfikacja Wymagań na Oprogramowanie (nowe moduły, nowe wymagania)
Model logiczny aktualnego systemu
Model logiczny nowego systemu
Model fizyczny aktualnego systemu
Model fizyczny nowego systemu


TWORZENIE APLIKACJI
EasyCASE - BRIDGE - PROGRESS

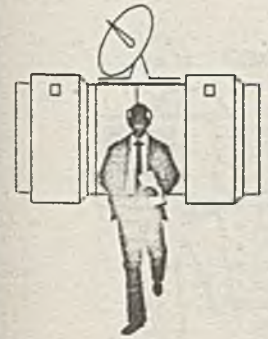


UTRZYMANIE I DYSTRYBUCJA SYSTEMÓW INFORMATYCZNYCH

- modyfikacje systemu
- powiązania informacyjne
- dystrybucja

UTRZYMANIE I DYSTRYBUCJA SYSTEMÓW INFORMATYCZNYCH

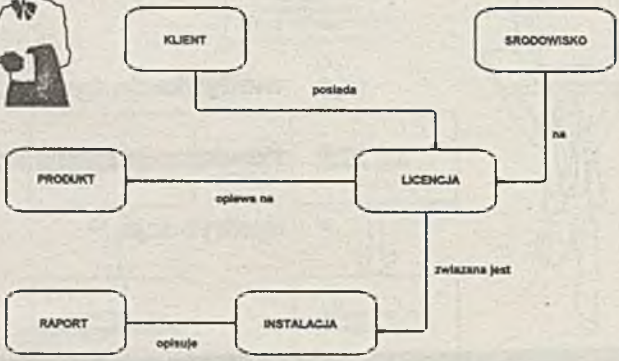
Modyfikacje systemu:



- rozwój
- zmiany w otoczeniu systemu
- uwagi i żądania użytkowników
- wady stwierdzone w czasie eksploatacji
- zmiany technologiczne

UTRZYMANIE I DYSTRYBUCJA SYSTEMÓW INFORMATYCZNYCH

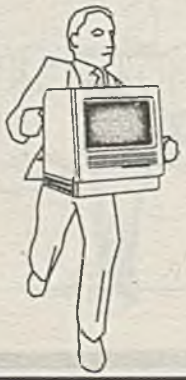
Powiązania informacyjne

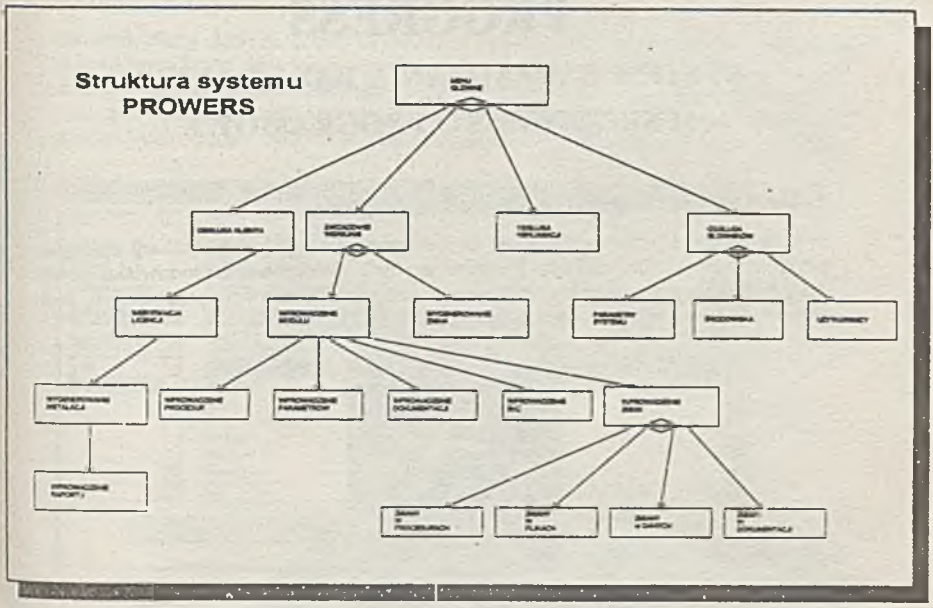


UTRZYMANIE I DYSTRYBUCJA SYSTEMÓW INFORMATYCZNYCH

Dystrybucja:

- problemy logistyczne
- kompletowanie wersji
- instalacje i wdrożenia
- zdalna dystrybucja





METODYKA CSBI

Usługi konsultingowe CSBI

Obszary zastosowań metodyki CSBI:

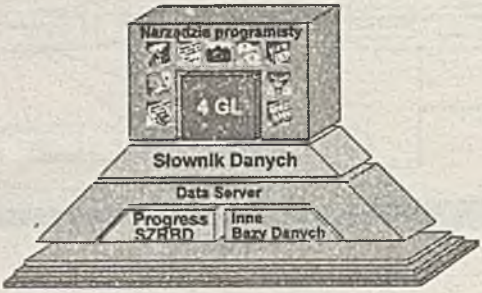
- organizacja przedsięwzięć informatycznych
- analizy systemów informacyjnych
- plany strategiczne informatyzacji
- projekty
- auditing przedsięwzięć informatycznych
- szkolenia z metodyki
- wdrożenia metodyki
- ekspertyzy porównawcze
- ekspertyzy techniczne
- oceny projektów
- analizy potrzeb i ograniczeń
- analizy ryzyka i warunków powodzenia przedsięwzięć
- analizy i oceny kosztów przedsięwzięć
- organizacja przetargów
- opracowywanie zapytań ofertowych

PROGRESS

SYSTEM ZARZĄDZANIA BAZĄ DANYCH I ŚRODOWISKO PROGRAMOWE

1. Ogólna charakterystyka systemu Progress

System Progress przeznaczony jest do tworzenia i eksploatacji aplikacji obsługujących relacyjne bazy danych. Progress zawiera zintegrowane środowisko pracy dla projektanta i programisty oraz system zarządzania relacyjną bazą danych, pod kontrolą którego wykonywane są aplikacje.

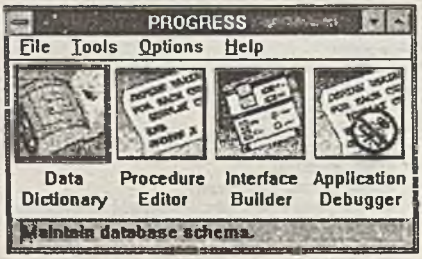


Rys. 1. Progress - struktura systemu

Progress jest systemem otwartym, co oznacza m.in., że możliwa jest współpraca z innym oprogramowaniem oraz tworzenie i użytkowanie aplikacji progressowych na różnych platformach sprzętowych, a także dowolne rozbudowywanie czy modyfikacja posiadanej konfiguracji bez potrzeby dokonywania zmian w aplikacji.

2. Tworzenie aplikacji

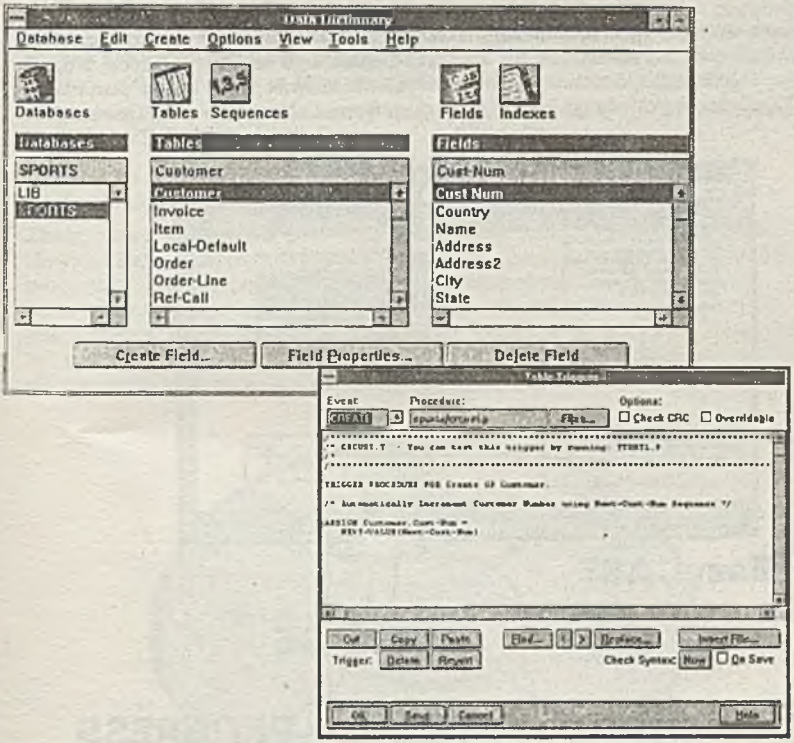
Progress zawiera narzędzia wspomagające pracę programisty i znacznie przyspieszające proces tworzenia aplikacji. Sam język programowania Progress 4GL jest językiem wysokiego poziomu, o czytelnej i zwięzłej składni, dzięki czemu łatwe jest zarówno uczenie się go, jak i stosowanie.



Rys. 2. Progress - pulpit roboczy

2.1. Słownik bazy danych

Słownik bazy danych (Data Dictionary) przechowuje szczegółowe informacje o strukturze bazy: jakie tablice zawiera baza, jaka jest struktura rekordu w każdej z tablic, jakie zachodzą powiązania relacyjne pomiędzy poszczególnymi tablicami, jaka jest struktura indeksów. W Słowniku zawarte są także reguły określające więzy integralności bazy i reguły walidacyjne na poziomie pól w rekordzie.

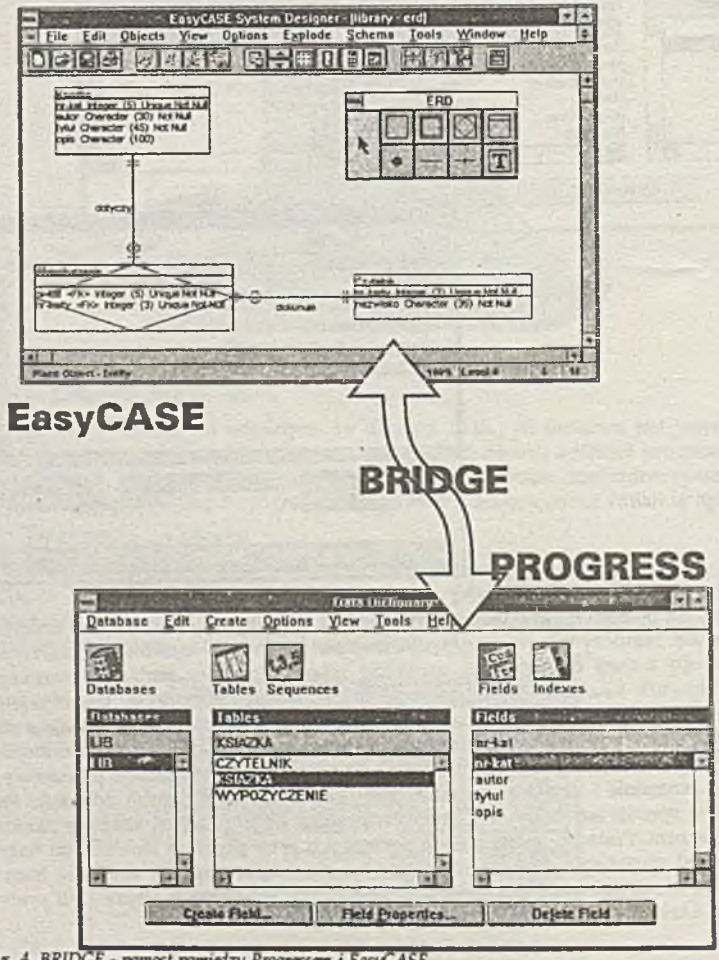


Rys. 3. Progress - Słownik Bazy Danych

Również na poziomie Słownika definiowane są triggery, czyli procedury uruchamiające się samoczynnie w przypadku zajścia pewnego określonego zdarzenia związanego z bazą danych (np. usunięcie rekordu czy też zmiana zawartości pola). Słownik zawiera też pewne informacje pomocnicze takie, jak etykiety i formaty używane przy wyświetlaniu danych, teksty podpowiedzi pojawiające się na ekranie przy wprowadzaniu lub modyfikowaniu danych czy wartości domyślne dla poszczególnych pól rekordu. Ponieważ wszystkie te informacje przechowywane są centralnie i każda z procedur operujących na bazie danych odwołuje się do nich w sposób automatyczny, proces tworzenia kodu aplikacji znacznie skraca się i upraszcza. Ponadto, jedną z opcji dostępnych przy pracy ze Słownikiem Bazy Danych jest możliwość drukowania raportów obrazujących pełną strukturę bazy; raporty te mogą stanowić dokumentację techniczną projektu, co usprawnia pracę zespołu, a także ułatwia późniejsze utrzymanie aplikacji.

2.2. Pomosty do programów CASE

Progress umożliwia korzystanie z dwukierunkowych pomostów *(bridges)* do programów CASE wysokiego poziomu. Po wykonaniu projektu aplikacji przy pomocy programu CASE następnym krokiem jest generacja (przy użyciu pomostu) schematu bazy danych Progress. Zastosowanie takiej metody daje szansę uniknięcia błędów popełnianych w fazie projektowej, a są to zwykle błędy najbardziej kosztowne. Pomosty umożliwiają także tzw. *reverse engineering*, czyli wczytanie gotowego (stworzonego ręcznie) schematu bazy danych do narzędzia CASE w celu weryfikacji lub też zmodyfikowania tego schematu. Użycie programów CASE daje w efekcie przede wszystkim łatwość utrzymania posiadanej aplikacji dzięki posiadaniu kompletnej i dostatecznie szczegółowej dokumentacji.



Rys. 4. BRIDGE - pomost pomiędzy Progressem i EasyCASE

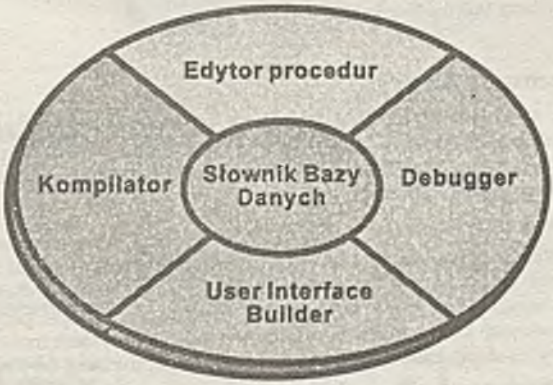
2.3. Język programowania Progress 4GL

Język Progress 4GL jest językiem kompletnym, co oznacza, że cała aplikacja może być napisana w Progressie bez konieczności odwoływania się do języków niższego poziomu. Jest to proceduralny język czwartej generacji; programista piszący w Progressie koncentruje się raczej na tym, co ma być zrobione, a nie w jaki sposób. Progress umożliwia programowanie sterowane zdarzeniami (*ang. event driven programming*), co daje w efekcie aplikację o bardziej płaskiej strukturze, a więc łatwiejszą do zarządzania, a z drugiej strony bardziej elastyczną i przyjazną dla użytkownika. Kod aplikacji napisanej w Progressie jest zwięzły i czytelny. Jak wynika z doświadczeń CSBI, programiści po dwutygodniowym kursie programowania w języku Progress są przygotowani do samodzielnej pracy w tym języku.

Progress 4GL zawiera wbudowany język SQL 89 Level 2 (certyfikat National Institute for Standards and Technology).

2.4. Środowisko pracy programisty

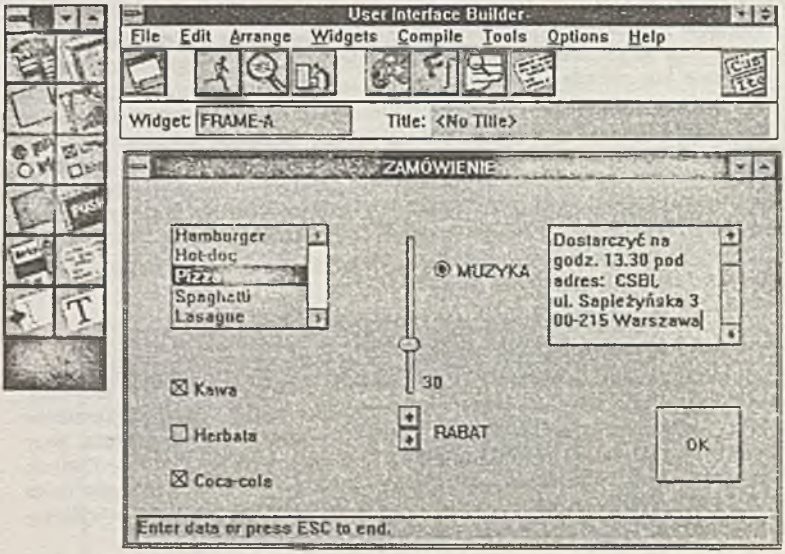
Zintegrowane środowisko pracy dla programisty zawiera następujące elementy: Słownik Bazy Danych, edytor procedur, kompilator oraz debugger. Tworzone procedury są kompilowane i uruchamiane bezpośrednio z pola edytora, przy czym brane są pod uwagę wszelkie informacje zawarte w Słowniku Bazy Danych. Jeśli program zachowuje się nieprawidłowo, pomocny w wykryciu błędu może być działający w trybie interakcyjnym debugger, umożliwiający śledzenie wykonania programu krok po kroku, ustawianie punktów kontrolnych itd.



Rys. 5. Progress - środowisko pracy programisty

2.5. Tworzenie interfejsu użytkownika

Narzędziem wspomagającym programistę przy tworzeniu graficznego interfejsu użytkownika jest moduł User Interface Builder (UIB). Pracując w trybie graficznym przy pomocy myszy można błyskawicznie projektować wygląd ekranu, a także definiować akcje związane z konkretnymi obiektami ekranowymi, takimi, jak przyciski, suwaki, listy wyboru itd. UIB generuje kod źródłowy w Progressie, który można również modyfikować ręcznie.



Rys. 6. Ekran roboczy User Interface Buildera

3. Bezpieczeństwo danych

Progress daje użytkownikowi pełną gwarancję zabezpieczenia przechowywanych danych przed zniszczeniem, utratą spójności czy też przed niepowołanym dostępem. Dzięki temu mogły w Progressie powstać systemy bankowe czy też systemy o zastosowaniach militarnych.

3.1. Archiwizacja danych

Progress umożliwia pełną i przyrostową archiwizację danych *(backup)*. Archiwizacja przyrostowa może być wykonywana z pewną zadaną redundancją, dzięki czemu nawet utrata jednego lub też kilku elementów ciągu kopii przyrostowych nie musi powodować utraty zawartości bazy danych. Jeśli system musi funkcjonować 24 godziny na dobę, możliwe jest wykonywanie archiwizacji on-line.

3.2. Zabezpieczenie przed niepowołanym dostępem

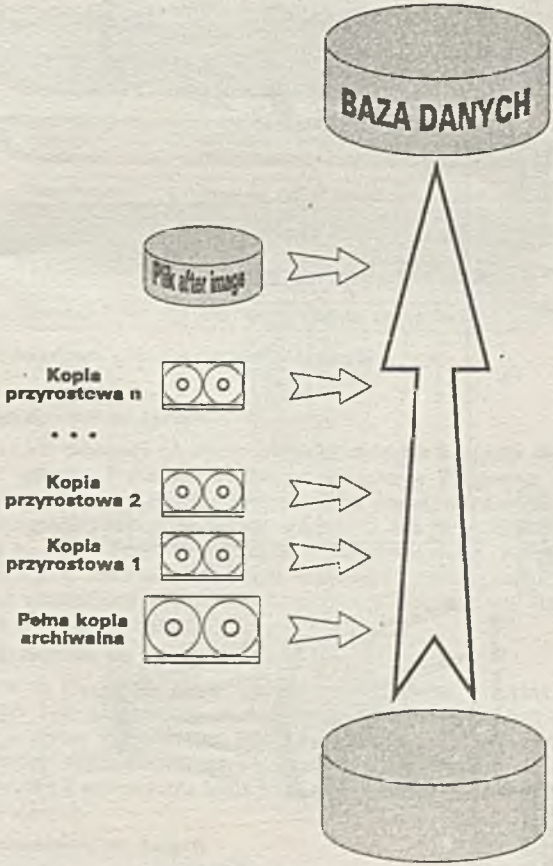
Dane są zabezpieczone przed niepowołanym dostępem na trzech poziomach. Są to: poziom systemu operacyjnego, poziom administracji bazą danych oraz poziom aplikacji.

3.3. Blokowanie zasobów

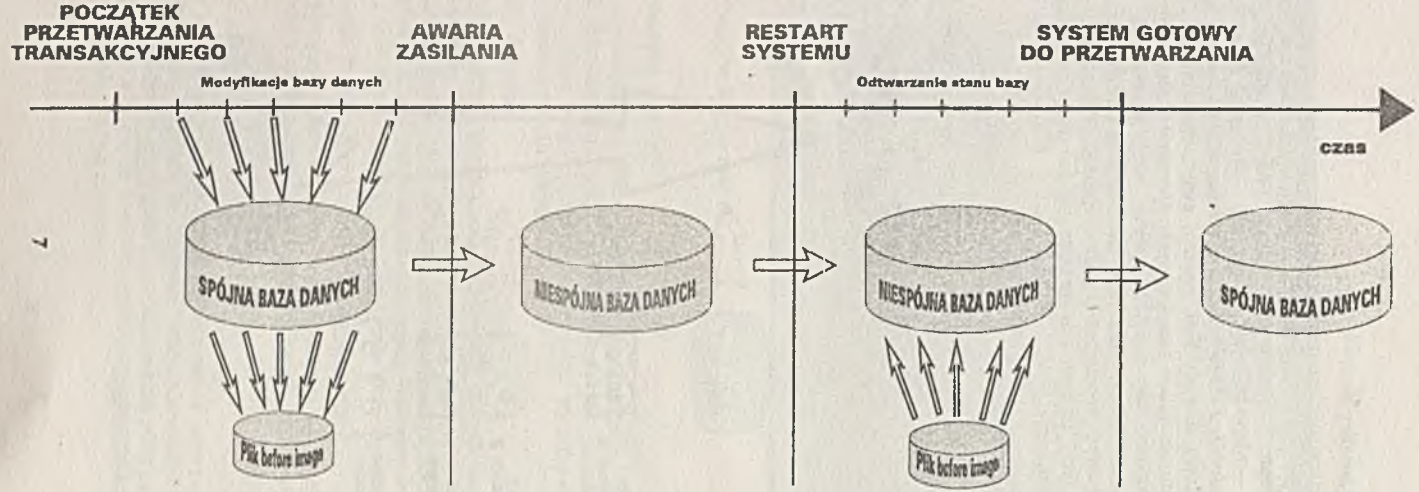
W trybie pracy wielodostępnej, gdy wielu użytkowników korzysta z tych samych danych, konieczne jest blokowanie zasobów aktualnie modyfikowanych przez jednego z użytkowników. W Progressie takie blokowanie odbywa się na możliwie najniższym poziomie, a więc na poziomie rekordu; dostępne są różne poziomy blokowania (wyłączność, dostęp dzielony).

3.4. Przetwarzanie transakcyjne

Progress jest ukierunkowany na przetwarzanie transakcyjne, przy czym zakresy transakcji mogą być *explicite* podawane przez programistę lub też są przyjmowane domyślnie przez system. Transakcje w Progressie posiadają standardowe własności ACID (*Atomicity* — niepodzielność, *Consistency* — spójność, *Isolation* — izolacja i *Durability* — trwałość). Przetwarzanie transakcyjne zapewnia utrzymanie spójności logicznej bazy danych. Jeśli nastąpi awaria systemu, po ponownym uruchomieniu systemu wszystkie niezakończone w momencie awarii transakcje są automatycznie wycofywane, aby utrzymać spójność bazy. Możliwe jest zapamiętywanie wszystkich pomyślnie wykonanych transakcji w oddzielnym pliku *after image*, co nazywamy zabezpieczaniem obrazu wtórnego. Wówczas w przypadku awarii sprzętu stan bazy danych odtwarzany jest na podstawie ostatniej kopii archiwalnej i pliku *after image*.



Rys. 7. Odtwarzanie utraconej bazy danych



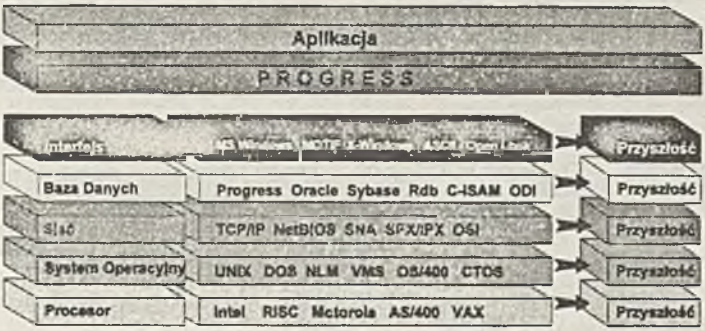
Rys. 8. Odtwarzanie stanu bazy po awarii

3.5. Transakcje rozproszone

Jeśli baza danych jest rozproszona pomiędzy kilka serwerów, pojedyncza transakcja może dotyczyć danych znajdujących się fizycznie na kilku maszynach. Aby zabezpieczyć spójność bazy w takim przypadku, stosowany jest dwufazowy protokół potwierdzeń *(two phase commit)*, co oznacza, że do prawidłowego zamknięcia transakcji niezbędne jest uzyskanie potwierdzenia jej wykonania od wszystkich serwerów biorących w niej udział.

4. Niezależność aplikacji

Użytkownik aplikacji progressowych ma pełną swobodę wyboru sprzętu i środowiska, w którym aplikacja będzie uruchamiana; może również korzystać z baz danych innych producentów.



Rys. 9. Niezależność aplikacji w Progressie od środowiska

4.1. Niezależność od sprzętu

Progress jest dostępny na komputerach wszystkich liczących się na rynku producentów sprzętu. Ponieważ aplikacje napisane w Progressie są stuprocentowo przenośne z jednej platformy sprzętowej na drugą, użytkownik nie jest w żaden sposób ograniczony przy wyborze sprzętu lub przy wymianie posiadanego sprzętu na nowy. Ponieważ aplikacja w Progressie może działać bez konieczności modyfikacji w nieomal dowolnym środowisku heterogenicznym, możliwa jest również rozbudowa posiadanej konfiguracji poprzez np. dołączenie nowego serwera czy stacji roboczych.

4.2. Niezależność od środowiska

Aplikacja w Progressie może funkcjonować zarówno w trybie jednostanowiskowym, jak i w środowisku wielodostępnym (serwer centralny z terminalami) a także w trybie klient-serwer. Tryb klient-serwer umożliwia optymalne wykorzystanie mocy obliczeniowej stacji roboczych (które zajmują się obsługą interfejsu użytkownika i wewnętrzną logiką aplikacji) i serwera, który obsługuje zapytania do bazy danych.

4.3. Niezależność od danych

Jeśli z różnych powodów użytkownik chce (lub musi) korzystać z danych znajdujących się w bazie danych innego producenta, Progress dostarcza tzw. *Dataservers*,

czyli serwery komunikujące się z "obcymi" bazami danych. Obecnie dostępne są serwery do baz Oracle, Sybase, Rdb, OS/400 oraz do Informixowych plików C-ISAM. Kwestia formatu danych jest niewidoczna dla użytkownika, który nie musi zdawać sobie sprawy, że dane do jego aplikacji pochodzą np. z dwóch baz danych, z czego jedna jest bazą progressową a druga oracle'ową.

W przypadku danych zapisanych w formacie dBase możliwe jest zaimportowanie całej zawartości bazy do bazy danych Progress i następnie praca na tych danych przy pomocy programów napisanych w Progressie.

5. Dodatkowe możliwości

Pewne dodatkowe możliwości, które mogą okazać się przydatne w niektórych zastosowaniach, to multimedialność, indeksowanie słów i możliwość tworzenia aplikacji wielojęzycznych.

5.1. Multimedialność

Aplikacja w Progressie może posługiwać się zarówno dźwiękiem, jak i grafiką. Tak więc np. w katalogu części mogą być przechowywane również rysunki techniczne, a w bazie zawierającej informacje o kompozytorach mogą znaleźć się nagrania ich utworów.

5.2. Indeksowanie słów

Jeśli rekordy bazy danych zawierają duże pola tekstowe, może okazać się przydatna możliwość zbudowania tzw. *word index*, czyli indeksu zawierającego odwołania do wszystkich słów występujących w polach tekstowych. Wówczas możliwe jest szybkie wyszukanie rekordów zawierających w polu tekstowym zadane słowo lub pewną kombinację słów.

5.3. Aplikacje wielojęzyczne

Jednym z produktów dostarczanych przez Progress jest Translation Manager. Zarządza on tworzeniem aplikacji wielojęzycznych, to jest takich, które mogą komunikować się z użytkownikiem w kilku językach, przy czym zmiana języka może następować "w biegu", bez przerywania działania aplikacji.

6. System zarządzania bazą danych

Progress RDBMS (*Relational Database Management System*) jest bardzo wydajnym systemem zarządzania bazą danych dla rzeczywistych aplikacji. Określenie *wydajny* w odniesieniu do rzeczywistych aplikacji oznacza nie tylko osiąganie wysokich wskaźników TPS (Transaction Per Second - liczba transakcji na sekundę) w testach szybkości baz danych przeprowadzanych na prostych transakcjach, ale również dużą szybkość pracy i krótkie czasy reakcji dla dłuższych, bardziej skomplikowanych transakcji, jakimi na ogół charakteryzują się aplikacje użytkowe.

Aby uzyskać wysoką efektywność przetwarzania przy jednoczesnym zapewnieniu niezawodności i gwarancji zachowania integralności bazy danych, zarówno w zastosowaniach opartych na dużych serwerach, jak i przy zastosowaniu architektury klient-serwer, Progress RDBMS posiada wiele własności, które minimalizują liczbę odwołań do dysku a zarazem zwiększają wydajność systemu.

6.1. Poziom blokowania zasobów

Dla długich, złożonych transakcji wąskim gardłem w dostępie do danych jest konieczność jednoczesnego korzystania ze wspólnych zasobów przez różnych użytkowników. W systemach baz danych, w których blokada następuje na

poziomie całych bloków bazy danych, powstają duże problemy jeżeli transakcja obejmuje więcej niż kilka rekordów. Aby zwiększyć wydajność i zredukować konflikty w dostępie do zasobów, Progress blokuje dane na poziomie rekordu. Jeżeli transakcja modyfikuje pojedynczy rekord, blokowany jest tylko ten rekord. Mikrotransakcje również blokują dane na możliwie najniższym poziomie, na przykład, jeżeli konieczne jest czasowe zablokowanie indeksu, blokowana jest tylko jedna strona, zamiast całego drzewa indeksowego.

6.2. Praca w trybie tylko do odczytu *(read-only)*

Aplikacje, które tylko czytają dane (na przykład raporty) mogą pobierać rekordy bez blokowania *(no-lock)*. Aplikacje takie nie mogą być też zablokowane przez innych użytkowników.

6.3. Inteligentne buforowanie

System Progress korzysta z efektywnego i elastycznego algorytmu buforowania, obejmującego tak zwany opóźniony zapis dyskowy oraz grupowe potwierdzenia, w celu ograniczenia do minimum liczby odwołań do dysku. Mechanizmy te stwarzają pewne ryzyko utraty części ostatnio wprowadzonych modyfikacji w przypadku wystąpienia awarii, jednak jest ono bardzo niewielkie, gdyż czasy opóźnień są rzędu ułamków sekund. Jednocześnie utrzymana jest stuprocentowa gwarancja integralności bazy danych.

Podczas pracy w architekturze klient-serwer, serwer bazy danych stosuje technikę nazywaną *optymistycznym buforowaniem rekordów*, polegającą na przesyłaniu wielu rekordów w jednym komunikacie sieciowym *(network message)*. W porównaniu do stosowanych często mechanizmów przesyłania rekordów w oddzielnych komunikatach sieciowych osiągana jest ogromna poprawa wydajności, gdyż liczba przesyłanych komunikatów jest znacznie ograniczona.

6.4. Zmienna długość rekordu

Przechowywanie danych ze zmienną długością jest wygodne nie tylko z punktu widzenia programisty. Podstawową korzyścią płynącą z zastosowania tej techniki jest oszczędność miejsca na dysku dochodząca w niektórych systemach do 60%. Rozmiar danych dyskowych ma duży wpływ na szybkość działania systemu, a więc uzyskiwane są również korzyści czasowe.

6.5. Kompresja indeksów

Drzewa indeksowe bazy danych Progress przechowywane są w bardzo skompresowanej postaci, dzięki temu więcej informacji może być pobranych lub zapisanych podczas jednej operacji dyskowej.

6.6. Zrzucanie buforów w tle *(Asynchronous Page Writers)*

Asynchroniczna obsługa buforów, to niezależne procesy, który okresowo (co kilka milisekund) przepisują uaktualnione bufory bazy danych z pamięci RAM na dysk. Prawdopodobieństwo, że zmodyfikowane bufory zostaną przepisane na dysk, zanim proces klienta będzie chciał je użyć ponownie, jest na tyle duże, że proces klienta praktycznie w ogóle nie musi wykonywać operacji dyskowych.

6.7. Skalowalność

Bardzo trudno jest przewidzieć docelową platformę sprzętową, na której będzie używana aplikacja użytkowa, dlatego też system obsługi bazy danych musi zapewnić proporcjonalną do zasobów systemu efektywność przetwarzania danych (skalowalność, *scalability*). System powinien być dostatecznie elastyczny, aby zapewnić optymalne warunki pracy z maksymalnym wykorzystaniem szczególnych własności różnorodnych platform sprzętowych i systemowych. System

Progress posiada dużą różnorodność i elastyczność architektury, konieczną do osiągnięcia wysokich parametrów przetwarzania w różnych środowiskach sprzętu i systemów operacyjnych. Progress został tak zaprojektowany, aby maksymalnie wykorzystać korzyści jakie dają różne systemy operacyjne i konfiguracje sprzętowe. Dlatego też wydajność systemu bazy danych Progress wzrasta proporcjonalne do tego, jak nowe zasoby - takie jak pamięć, czy procesor - są dodawane do systemu. W zależności od platformy sprzętowej Progress wykorzystuje jedną z trzech podstawowych konfiguracji:

- Autonomiczny klient (*self service client*)

W tej konfiguracji klient i serwer są częściami jednego procesu, eliminując w ten sposób opóźnienia wynikające z komunikacji między procesami, jak również z przełączania zadań. Jednoczesny dostęp do bazy danych dla wielu autonomicznych klientów osiągany jest poprzez pamięć dzieloną (*shared memory*). Dla systemów jednoużytkownikowych oraz jednoprocesorowych z lokalnymi klientami, konfiguracja taka zapewnia minimalne opóźnienia czasowe oraz najkrótszy czas dostępu do bazy danych.

- Jeden klient na serwer

W tej konfiguracji procesy klienta i serwera są oddzielone od siebie, ale każdy serwer obsługuje tylko jednego klienta. Taka konfiguracja jest najlepsza w systemach wieloprocesorowych, gdzie każdy serwer może być przydzielony do własnego procesora.

- Wielu klientów na serwer

Konfiguracja taka jest określana mianem *serwera wielowątkowego* (*multithreaded server*), gdyż jeden serwer obsługuje naraz wielu klientów. Pojedynczy serwer wielowątkowy zapewnia optymalne wykorzystanie CPU na maszynie jednoprocesorowej, podczas gdy wiele serwerów wielowątkowych zapewnia optymalne wykorzystanie zasobów maszyny wieloprocesorowej.

Administrator systemu może łączyć ze sobą różne konfiguracje podstawowe w ramach jednego systemu. Warto też wspomnieć, że procesy serwera dla klientów odległych są startowane automatycznie w miarę napływania kolejnych zgłoszeń, dzięki temu czas dostępu do bazy danych pozostaje niezmieniony, nawet przy bardzo obciążonym systemie.

7. Użytkowanie aplikacji

Jak wynika z powyższych własności systemu Progress, możliwe jest tworzenie przy jego pomocy wydajnych aplikacji z przyjaznym, graficznym interfejsem, bezpiecznych w użytkowaniu. Aplikacje te są również łatwe w utrzymaniu, a więc wszelkie zmiany wynikające np. ze zmian w obowiązujących przepisach są łatwe do wprowadzenia. W przypadku wymiany sprzętu lub rozbudowy posiadanej konfiguracji aplikacja w Progressie nie wymaga dokonywania w niej zmian.

9. Progress w Polsce

Jedynym autoryzowanym dystrybutorem systemu Progress w Polsce jest CSBI, współpracujące z siecią 32 Value Added Ressellerów, t.j. firm software'owych piszących oprogramowanie w Progressie. Sprzedano obecnie ponad 950 licencji na użytkowanie systemu. Działa Polski Klub Użytkowników Systemu Progress, będący sekcją Polskiego Towarzystwa Informatycznego, co roku odbywa się ogólnopolska Konferencja Klubu, wydawany jest biuletyn klubowy. Katalog Polskich Aplikacji w Języku Progress 4GL zawiera ok. 70 opisów gotowych aplikacji obsługujących wiele dziedzin gospodarki.

Prototypowanie we współczesnych SZRDB

dr inż. Marek MIŁOSZ dr inż. Elżbieta MIŁOSZ
Katedra Informatyki Katedra Zarządzania
Politechnika Lubelska, ul. Nadbystrzycka 36B, 20-618 Lublin

1. Wtęp

Informatyka, a w szczególności sprzęt komputerowy, oprogramowanie narzędziowe i użytkowe, to dziedziny, które rozwijają się szczególnie dynamicznie. Potrzeby informatyczne ludzi businessu są tak duże i zmienne w czasie, że aktywnie stymulują rozwój inżynierii budowy systemów informatycznych i podstawowych narzędzi zaspokojenia ich potrzeb - systemów zarządzania relacyjnymi bazami danych (SZRBD).

Aktualnie na "rynku informatycznym" równolegle istnieje kilka standardowych metodyk budowy systemów informatycznych (SI). Są one uporządkowanym, w sposób mniej lub bardziej naukowy, uogólnieniem długoletnich doświadczeń firm komputerowych, specjalizujących się w tworzeniu oprogramowania. Metodyki te zawierają konkretny zbiór metod i technik tworzenia poszczególnych elementów systemu, pozwalających realizować cały cykl życia [1] systemu informatycznego przez zespół wykonawczy (tj. w uproszczeniu przez analityków, projektantów, programistów, wdrożeniowców, administratorów i oczywiście użytkowników).

W procesie tworzenia SI stosowane są różne podejścia [2]. Do najważniejszych z nich należą:

- podejście sekwencyjne (klasyczne), pozwalające budować system w sposób szeregowy poprzez realizację kolejnych etapów projektowania;
- podejście strukturalne, pozwalające na częściowo-równoległą realizację procesów analizy, projektowania i implementacji;
- budowanie prototypu przyszłego systemu z późniejszą jego weryfikacją, modyfikacją itp.

W zasadzie każda z metod tworzenia SI w rzeczywistości jest procesem iteracyjnym, wielokrotnie realizującym te same etapy. Rzadko zdarza się bowiem w praktyce, by nie wprowadzano zmian w częściowo lub całkowicie zrealizowany projekt czy też oprogramowanie. Ponadto SI muszą być poddawane niemalże ciągłej, pernamentnej ewolucji, aby nadążyć za zmieniającą się (w naszych, polskich warunkach dosyć szybko i radykalnie) rzeczywistością, wciąż rosnącymi potrzebami i wymaganiami użytkowników, a także za rozwojem technologii informatycznych.

Obok tradycyjnych metodyk tworzenia systemów informatycznych, powstają nowe, które w mniejszym lub w większym stopniu doskonalą, upraszczają lub wręcz radykalnie modyfikują proces projektowania systemów informatycznych.

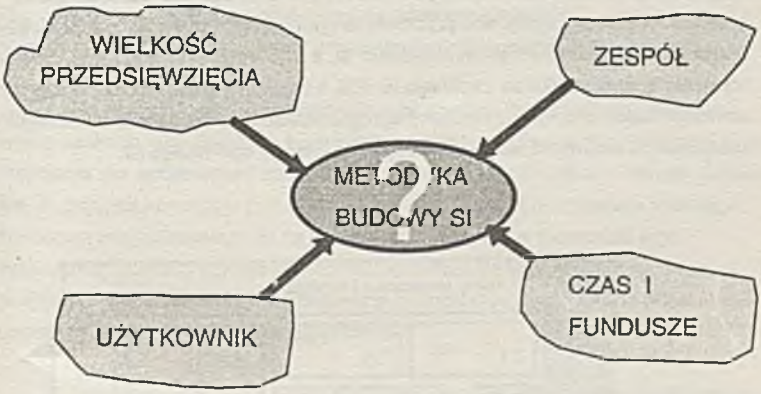
2. Czynniki warunkujące wybór metodyki budowy SI

Wybór konkretnego podejścia do budowy SI powinien być dokonany pod wpływem analizy warunków, w jakich ma powstawać i funkcjonować projektowany system.

Wydaje się, że należy przy tym wziąć pod uwagę następujące czynniki (rys. 1):

1) wielkość przedsięwzięcia:
   - zakres tematyczny SI (fragmentaryczny SI, dziedzinowy, złożony problem wielodziedzinowy, kompleksowy, rozbudowa istniejącego systemu itd.);
   - wielkość i złożoność struktury informacyjnej (liczba i rodzaj przetwarzanych danych, symbolika i systemy kodowania informacji, stopień złożoności algorytmów przetwarzania itd.);
   - ewentualne zmiany organizacyjne u użytkownika, wymuszane przez SI i jednocześnie warunkujące jego wdrożenie;
2) czasokres realizacji przedsięwzięcia:
   - czas przeznaczony na realizację systemu;
3) współpraca z użytkownikiem:
   - umiejętność ustalania założeń przyszłego systemu, wymagań i ograniczeń stawianych SI przez użytkownika;
   - identyczna interpretacja założeń systemu przez użytkownika i projektanta;
   - przezwyciężenie niechęci do zmiany tradycyjnego sposobu pracy. do pracy na komputerach, warunkujące wdrożenie SI;
4) wielkość i umiejętność zespołu realizującego SI:
   - liczba projektantów w zespole;
   - znajomość przedmiotu projektu przez zespół projektowy;

- umiejętność pracy zespołowej;
- dostępność narzędzi projektowych i programowych do budowy SI, a także wspomagających zespołowe tworzenie aplikacji;



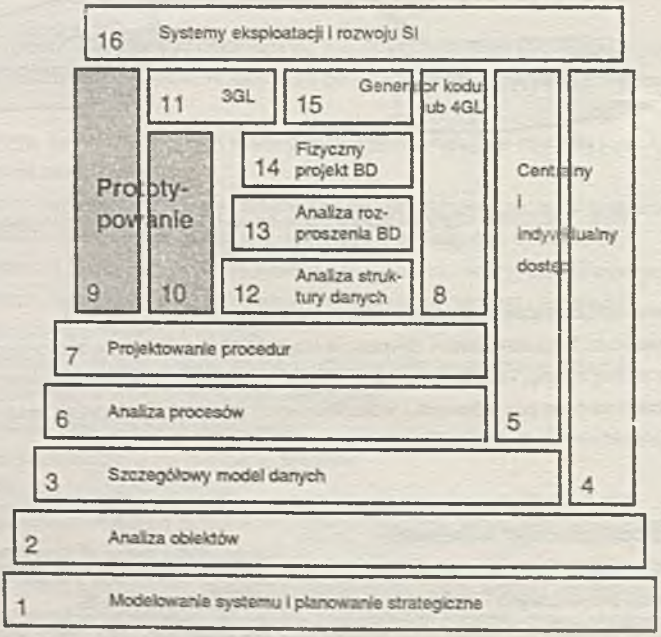
Rys. 1. Główne czynniki wpływające na wybór metodyki projektowania SI

5) cena opracowania systemu:
- wielkość funduszy, jakimi dysponuje użytkownik na realizację zleconego SI z jednej strony, i wielkość funduszy, niezbędnych na opłacenie konkretnego zakresu prac projektowych i wdrożeniowych realizowanych przez zespół projektowy, z drugiej.

Użytkownik oczekuje systemu, realizującego wskazane przez niego funkcje i "wyprodukowanego" w konkretnym czasie.
Przed projektantem zaś stoi problem wyboru metodyki realizacji przedsięwzięcia, narzędzi projektowych i programowych dla budowy konkretnego SI.
Jakimi przesłankami należy kierować się przy wyborze tego czy innego sposobu budowy SI? W jakich przypadkach celowe jest wcześniejsze budowanie prototypu, a nie od razu rzeczywistego systemu? Odpowiedzi na te pytania powinny być wskazówką wyboru podejścia do projektowania SI dla każdego konkretnego przypadku.

3. Miejsce prototypowania w inżynierii systemów informatycznych

W [3] zdefiniowane zostało 16 bloków - cegiełek tworzących współczesną inżynierię systemów informatycznych, tworzonych przy pomocy SZRBD. Układ bloków i związki pomiędzy nimi przedstawia rys. 2. Prototypowanie jest pomostem pomiędzy szeroko rozumianym projektem SI, a jego konkretną realizacją. Jest ono równoległą, w stosunku do zwykłego (w 4GL lub z wykorzystaniem generatorów aplikacji) kodowania, drogą realizacji oprogramowania. Może zatem w szczególnych przypadkach doprowadzać do gotowego i całkowicie sprawnego SI.



Rys. 2. Części składowe inżynierii systemów informatycznych [3]

Można wydzielić trzy podstawowe techniki, określające równocześnie obszary zastosowań prototypowania:
1) szybkie sprawdzenie koncepcji systemu (*proof-of-concept prototype*);
2) projektowanie poprzez prototypowanie (*desing prototype*);
3) budowa systemu przy pomocy prototypowania (*development prototype*).

Sprawdzenie koncepcji systemu przy pomocy jego prototypu stosuje się w przypadkach, gdy dziedzina przedmiotowa jest mało znana projektantom i duże znaczenie ma sprawdzenie opracowanego modelu tej dziedziny w celu upewnienia się, że przyjęta koncepcja jest słuszna. Łatwiej i szybciej jest bowiem sprawdzić koncepcję projektowanego SI za pomocą prototypu niż w momencie jego ostatecznej realizacji czy też wdrażania. Sprawdzenie takie stosuje się także w sytuacji, gdy dziedzina przedmiotowa jest słabo określona i trudno przewidzieć poprawny kształt projektowanego systemu.

Projektowanie przez prototypowanie jest alternatywą wobec tradycyjnego sposobu projektowania systemów i pozwala stworzyć aplikację, której dokumentacja i szczegółowy opis zawarty jest od razu w słowniku bazy danych, strukturach i układach menu, pomocy dla użytkownika i komentarzach do procedur przetwarzania.

Budowa systemu przy pomocy prototypowania pozwala opracowywać system (lub wybrane jego fragmenty) w wielokrotnie przeprowadzanych sesjach interakcyjnego kontaktu z użytkownikiem. W czasie sesji przedstawia się użytkownikowi do oceny kolejne wersje prototypu systemu, w celu zebrania jego uwag i uwzględnienia ich w toku dalszych prac. Technika ta pozwala także elastycznie rozszerzać zakres istniejącego systemu. Dzięki włączeniu do procesu budowy SI sprzężenia zwrotnego z użytkownikiem, można w sposób prawie doskonały zaspokoić jego oczekiwania, życzenia i żądania. Rezultatem jest podwyższenie jakości użytkowej powstającego systemu oraz, w pewnych przypadkach, zwiększenie wydajności zespołu projektującego SI.

Ogólnie rzecz ujmując możliwość sprawdzenia poprawności aplikacji na początkowym etapie jej opracowywania pozwala zwykle zmniejszyć koszty oraz ograniczyć liczbę pojawiających się problemów.

4. Obszary zastosowań techniki prototypowania

Wydaje się, że budowanie prototypu jest celowe w przypadkach systemów spełniających następujące warunki:

1) niewielki zakres tematyczny przedsięwzięcia:
   - szybkie zbudowanie prototypu przyszłego systemu możliwe jest dla niedużych i niezbyt skomplikowanych systemów;
   - w przypadku rozbudowy istniejącego SI o nowe funkcje modyfikacja może być sprawnie przeprowadzona przez zastosowanie techniki prototypowania;
2) krótki czas realizacji systemu:
   - użytkownik w przeciągu krótkiego okresu czasu może otrzymać prototyp przyszłego systemu lub jego części, może go przetestować porównując realizowane przez niego funkcje z własnymi oczekiwaniami;
3) ograniczone fundusze na realizację systemu;
4) niezdecydowany i trudny we współpracy "informatycznej" użytkownik:
   - prototyp pozwoli użytkownikowi zrozumieć cechy i założenia konstruowanego systemu, uzgodnić i skorygować je z projektantem tak, by interpretacja funkcji i założeń systemu przez użytkownika i projektanta była jednakowa;
   - ewentualne rozbieżności w zakresie projektownego systemu usuwane są na etapie tworzenia jego prototypu;
   - efekt wdrożenia systemu widoczny jest na jego prototypie, co stymuluje użytkownika do aktywnego uczestnictwa w procesie budowy SI, modyfikacji i uściślania swoich potrzeb;
5) szybka zmiana potrzeb informacyjnych użytkownika wymuszająca zmianę lub modyfikację jeszcze na etapie tworzenia aplikacji:
   - prototyp nowych funkcji systemu pozwoli szybko zmodyfikować istniejący SI i przewidzieć efekty modyfikacji.

Budowa prototypu w większości przypadków jest tylko początkowym etapem tworzenia SI, obejmującym sprecyzowanie założeń, analizę, projektowanie, programowanie i testowanie niedoskonałego jeszcze i dopuszczającego ewentualne błędy prototypu, aby następnie zweryfikować wymagania w stosunku do systemu, zaprojektować go w oparciu o poprawiony prototyp, zaprogramować i przetestować, tak by w końcu uzyskać produkt spełniający oczekiwania użytkownika.

Budowa prototypu ma zatem na celu:

- skrócenie czasu oczekiwania na konkretne efekty działania systemu;
- wspomaganie użytkownika w zakresie sprecyzowania potrzeb, wymagań i ograniczeń w stosunku do przyszłego systemu;
- zmniejszenie liczby błędów, wynikających z różnej interpretacji przez projektantów i użytkowników zakresu i sposobu działania poszczególnych elementów systemu;
- zacieśnienie współpracy między użytkownikiem a projektantem - aktywny udział użytkownika w procesie tworzenia systemu;
- szybka korekta i modyfikacja funkcji działania systemu realizowana w toku bieżącej pracy użytkownika i projektanta.

Należy podkreślić, że w procesie tworzenia prototypu dopuszcza się pozostawienie błędów w działającym prototypie, które nie umniejszają istoty problemu i zakresu realizowanych przez prototyp funkcji. Po akceptacji prototypu przez użytkownika i zweryfikowaniu wymagań w stosunku do przyszłego systemu, błędy zostają usunięte w momencie przekształcenia prototypu w ostatecznie funkcjonujący system.

Testowanie prototypu pozwala wychwycić wady i zalety projektowanego systemu i ułatwić użytkownikowi zrozumienie jego istoty. Prototyp jest też narzędziem porozumienia między użytkownikiem a projektantem w procesie budowy SI.

Pozytywny wynik testowania prototypu i jego akceptacja przez użytkownika pozwala projektantom na opracowanie ostatecznej wersji systemu rozbudowanej o funkcje pomocnicze i dodatkowe w stosunku do przyjętego prototypu.

Prototyp może być tworzony nie tylko dla całego systemu ale także dla jego części. Zwłaszcza rozbudowa, modyfikacja istniejącego systemu może być szybko wykonana przez projektanta jako prototyp dodatkowych funkcji sytemu. Prototyp części systemu pozwoli użytkownikowi lepiej zrozumieć daną funkcję czy moduł systemu, najbardziej dla niego istotny.

5. Prototyper - narzędzie wspomagające proces prototypowania SI

Szybka konstrukcja zrębów systemu i szybka jego realizacja programowa możliwa jest jedynie z zastosowaniem narzędzi wspomagających analizę, projektowanie i programowanie SI - CASE. Zestaw tych narzędzi tworzy **prototyper**, czyli system pozwalający w praktyce realizować technikę budowy prototypu.

W stosunku do prototypera można sprecyzować pewne ogólne wymagania. Powinien on mianowicie zawierać:

- elastyczny system definiowania struktury bazy danych, indeksów, relacji, reguł integralności, ograniczenia dostępu itp;
- efektywne narzędzia projektowania ekranów (układy menu, ekrany wprowadzania danych i wyprowadzania żądanych informacji, tworzenie odpowiedniej sekwencji ekranów wynikowych, zunifikowany, czytelny i przyjazny dla użytkownika sposób organizacji ekranu,...);
- narzędzia do tworzenia procedur przetwarzań (edycji bazy danych, wyszukiwań, algorytmów obliczeniowych itd.);
- efektywne narzędzia synchronizacji między ekranami prototypu a procedurami, realizującymi jego funkcje;
- narzędzia zapewniające jednolity i przyjazny sposób komunikacji z użytkownikiem;
- zestaw narzędzi programowych (np. w postaci odpowiednich bibliotek funkcji i procedur, generatorów itp.), pozwalających na generownie tekstu źródłowego programu, opisującego prototyp z możliwością późniejszej jego modyfikacji;
- zestaw narzędzi pozwalających dokumentować działanie prototypu.

6. Przegląd niektórych narzędzi do budowy prototypu systemu informatycznego

Budowa systemów informatycznych metodą prototypowania możliwa jest w praktycznie każdym SZRBD. Systemy te zawierają bowiem różnorodne dodatkowe narzędzia programistyczne wspomagające najbardziej pracochłonne i rutynowe elementy tworzenia aplikacji: projektowanie ekranów i wydawnictw, układów menu, zapytań do bazy danych itp. Prototypery są też elementami składowymi wielu systemów typu CASE.

ProKit*WORKBENCH

ProKit*WORKBENCH firmy McDonnell Douglas Information Systems to zaawansowane, zintegrowane narzędzie typu upper-CASE, wspomagające analityka i projektanta w trakcie całego cyklu życiowego SI.

Obok różnych modułów składowych (np. ANALYSER, DATA MODELER, DESIGNER) system ProKit*WORKBENCH zawiera również moduł PROTOTYPER, umożliwiający prototypowanie i symulację interfejsu użytkownika.

Moduł PROTOTYPER systemu ProKit*WORKBENCH pozwala [1] na:

- projektowanie ekranów SI, realizowane w całkowicie zintegrowanym środowisku tekstowo-graficznym;
- symulację interfejsu użytkownika (tworzenie sekwencji ekranów, warunków poruszania się po nich i sposobu ich wypełniania);
- generowanie wersji źródłowych definicji danych w jednym z 7 języków (Pascal, C, Cobol, Ada, Fortran, Basic, Assembler);
- generowanie map ekranów (dla systemów CICS, INS/DC);
- tworzenie pełnej dokumentacji (graficznej i tekstowej) zaprojektowanych ekranów;
- pełną synchronizację między ekranami prototypu a modelem funkcji.

PRO - IV

PRO-IV jest zintegrowanym środowiskiem programowym (4GL), umożliwiającym realizację i wspomagającym utrzymanie oprogramowania profesjonalnych SI. Ze względu na charakter realizacji aplikacji w tym środowisku należy go zaliczyć do prototyperów, tworzących gotowe, pełnowartościowe SI.

Podstawową jednostką programową i wykonawczą PRO-IV jest funkcja (typu: MENU, SCREEN, REPORT, UPDATE), którą tworzy się poprzez wypełnianie przedefiniowanych typów funkcji (*fill-in-the-blank*) określając ich parametry. Funkcje te mogą być samodzielnie uruchamiane, testowane i modyfikowane , co w szybki sposób pozwala stworzyć prototyp części lub całości realizowanego systemu. Tworzenie prototypu lub końcowej postaci aplikacji polega na logicznym wgraniu wcześniej stworzonych i przetestowanych funkcji.

PRO-IV pozwala na szybkie prototypowanie funkcji użytkowych tworzących aplikację. Funkcje te wraz z logiką SI, opisaną przy pomocy komend języka PRO-IV tworzą wersję wykonawczą aplikacji.

Aplikacja lub jej prototyp, stworzona za pomocą PRO-IV, może być przenoszona między różnymi systemami komputerowymi od mikro-, poprzez mini- do mainframe.

Oracle CASE

Oracle wraz ze swoim SZRBD dostarcza komplet narzędzi z rodziny Oracle CASE. Narzędzia te pozwalają na realizacje dowolnej z trzech strategii: tworzenie prototypu SI w celu szybkiego sprawdzenia koncepcji SI , projektowania SI poprzez prototypowanie lub też budowy SI przy pomocy prototypowania. Są to: CASE*Dictionary, CASE*Desinger i CASE*Generator.

Narzędzia prototypowania Oracle CASE zawierają [5]:

- środki do definiowania celu i zakresu (ekonomicznego, funkcjonalnego, informacyjnego) prototypu;
- narzędzia do szybkiego definiowania bazy danych (tabel, indeksów, widoków, relacji itp.) i specyfikacji modułów, wynikających z modelu analitycznego systemu (z analizy systemu);
- generatory do szybkiego tworzenia modułów;
- zestaw domyślnych parametrów interfejsu z użytkownikiem, który może pozostać niezmieniony lub też podlegać modyfikacjom, w celu dostosowania wyglądu aplikacji do bieżących potrzeb użytkownika;
- interaktywne manipulowanie definicjami modułów w celu wprowadzenia natychmiastowych zmian w czasie sesji budowy prototypu SI projektanta z użytkownikiem.

Pomocnymi narzędziami są także Oracle Data Browser, zapewniający przeźroczysty, interatywny dostęp do rozproszonych baz danych, Oracle Reports i Oracle Forms.

Moduły SI otrzymane metodą prototypowania mogą zostać włączone do istniejących SI, dając w efekcie nową aplikację. Mogą też być usunięte (wymienione) w momencie, kiedy zostanie zrealizowana pełna wersja danego modułu.

INGRES/Vision

INGRES/Vision jest bardzo rozbudowanym generatorem programów w Ingres' 4GL, którego funkcje kwalifikują go do roli prototypera. Umożliwia bowiem tworzenie kompletnej aplikacji, zawierającej formatki i menu w domyślnej postaci, procedury przetwarzań i ich przypisanie do poszczególnych opcji menu, kontrolę błędów, integralności bazy danych i układy pomocy dla użytkownika.
Podstawowymi elemetami środowiska INGRES/Vision są:

- edytor formatek (*frame*), umożliwiający graficzne projektowanie: menu, ramek operacji na bazie danych (*Append, Browse, Update*) i rapotów;
- katalog aplikacji (Applications Catalog), gromadzący informację o wszystkich częściach składowych tworzonej aplikacji;
- edytor struktury aplikacji (Application Flow Diagram Editor);
- edytor zapytań (Visual Query Editor);
- generator programu (Code Genarator).

W technologii GUI dostępne jest analogiczne środowisko pod nazwą INGRES/Windows4GL, które wspomaga tworzenie standardowych aplikacji i może być wykorzystane jako prototyper.

FAST TRACK

FAST TRACK jest programem napisanym w PROGRESS 4GL i wspomagającym tworzenie:

- wielopoziomowych układów menu (wraz z informacją dotyczącą uruchamiania procedur w wyniku wyboru opcji w menu);
- formatek ekranowych, umożliwiających komponowanie ekranu poprzez rozmieszczenie na nim pól bazy, zmiennych wraz z etykietami i komentarzami;
- raportów z bazy danych w różnych postaciach;
- zapytań do bazy danych przez formatkę (*query by form*) umożliwiających przegląd bazy (formatka po formatce) i wyszukiwanie żądanej informacji poprzez konstruowanie dowolnie złożonych warunków logicznych.

Nowa wersja PROGRESSa (ver. 7) posiada rozbudowany graficzny interfejs (*User Interface Builder*), umożliwiający w bardzo krótkim czasie zbudowanie w pełni sprawnej prototypu-aplikacji.

ROSI-SQL Prototyper

Jest narzędziem umożliwiającym interaktywne tworzenie układów menu i ekranów z późnieją ewentualną generacją kodu (w ROSI-SQL), który może być poddawany modyfikacjom. Oprócz typowych dla aplikacji okienkowych opcji zmiany parametrów tworzonych formatek-okien ROSI-SQL Prototyper umożliwia definiowanie układów menu i ekranów, przypisywanie im elemetów pomocy dla użytkownika i procedur przetwarzań, definiowanie sekwencji działań (w tym także sekwencji wprowadzania danych przez użytkownika).

Analogiczne narzędzia istnieją i w innych SZRBD np. INFORMIX (INFORMIX-4GL Forms, INFORMIX-4GL Menus i INFORMIX-SQL) lub Gupta (TeamWindows).

7. Podsumowanie

Prototypowanie jako jedna z metod budowy SI jest, w pewnych obszarach zastosowań, techniką wysoce efektywną. Działając w nieznanych lub nieokreślonych do końca dziedzinach, mając mało czasu lub funduszy na budowę skomplikowanej aplikacji czy też przewidując jej ustawiczne modyfikacje, projektant i programista może wykorzystać tą technikę w celu podwyższenia efektywności swojej pracy. Wspomagają go w tym liczne narzędzia programistyczne - prototypery.

Poza możliwością zmniejszenia pracochłonności wytwarzania SI prototypery zwiększają jego niezawodność, eliminując błędy programistów (kodowania).

Literatura

1. Piórkowska T., Puternicki D., Staniszkis W. Inżynieria systemów informatycznych - od strategii do realizacji. Informatyka, 8, 1992, str. 9-17.

2. Bańkowski M. Wprowadzenie do zagadnień tworzenia i utrzymania aplikacji. Mat. II Konferencji Polskiego Klubu Użytkowników PROGRESSa. Spała, 1993.

3. Martin J. Information Engineering. Vol. 1-3. Prentice Hall, NY, 1989.

4. Celmer P.J., Bruijn L. Projektowanie aplikacji w PROGRESSie przy użyciu pakietu "System Architekt". Referat, Kraków, SZP AM i UJ, 1993.

5. Johnson E. Oracle CASE and Information Engineering Mathodology. CASE Marketing, Bethesda. Oracle Part No: A10096.

Materiały informacyjno-reklamowe produktów informatycznych: Ingres, Oracle, PROGRESS, Gupta , INFORMIX, ROSI/SQL.

Wszystkie nazwy zastrzeżone zostały użyte wyłącznie w celu identyfikacji firm i produktów.

TRADEX
SOFTWARE COMPANY

Jesteśmy polskim przedstawicielstwem niemieckiej firmy Garmhausen & Partner GmbH. Firma ta, istniejąca od 1984 roku, jest na rynku niemieckim czołowym system - integratorem i dystrybutorem oprogramowania w zakresie systemów otwartych. Obszar jej działania obejmuje europejskie kraje niemieckojęzyczne, kraje Beneluxu oraz kraje Europy Środkowo-Wschodniej.

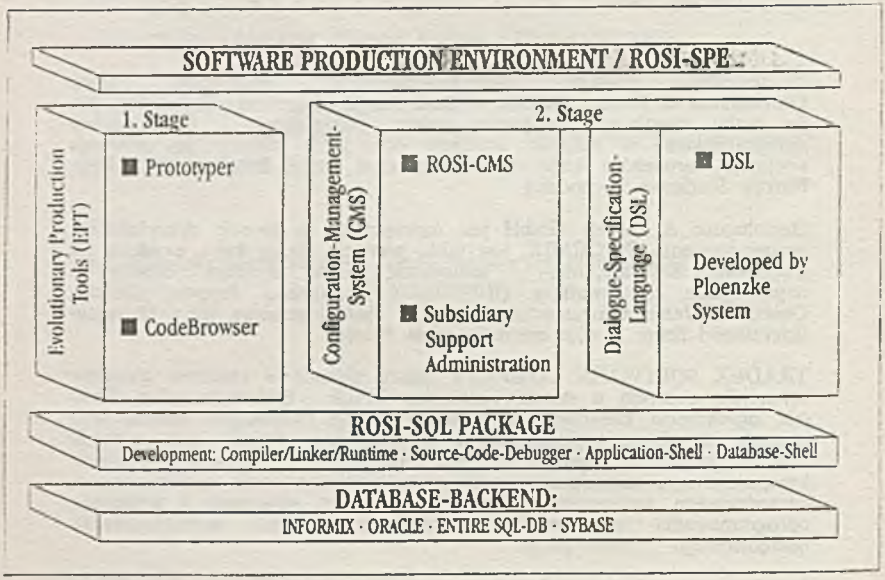
Garmhausen & Partner GmbH jest największym na świecie dystrybutorem oprogramowania INFORMIX, jest także pierwszą firmą, która uzyskała od INFORMIX Software, Inc. autoryzację jako centrum obsługi i wspomagania użytkowników (INFORMIX Authorised Support Centre). Obszar działania autoryzowanego centrum obsługi pokrywa się z obszarem działalności firmy, a więc obejmuje także Polskę.

TRADEX SOFTWARE COMPANY działa głównie w obszarze systemów otwartych, opartych o system operacyjny UNIX. Głównym celem firmy jest zapewnienie klientowi kompleksowej obsługi obejmującej analizę jego potrzeb, doradztwo organizacyjne i informatyczne, projektowanie systemu informacyjnego i informatycznego, dostawę i instalację sprzętu komputerowego, teleinformatycznego oraz oprogramowania systemowego i narzędziowego, wykonanie okablowania budynków, wykonanie i wdrożenie oprogramowania użytkowego, szkolenie w zakresie oprogramowania narzędziowego i aplikacyjnego.

Jesteśmy autoryzowanym dystrybutorem oprogramowania firm INFORMIX Software Inc., UNIPLEX Ltd., CONSENSYS Corporation, Structured Software Solution Inc., FourGen Inc., Century Software, urządzeń firmy EQUINOX i Thomas Conrad Corporation. Nasza firma jest również wyłącznym dystrybutorem urządzeń DYNATECH oraz terminali LINK w Polsce.

Pragniemy zaprezentować, obok znanego i powszechnie stosowanego w Polsce systemu INFORMIX, narzędzie mało u nas znane, a zasługujące na uwagę. ROSI - SQL jest to uzupełnienie, a zarazem rozwiązanie konkurencyjne dla modułów narzędziowych z pakietów baz danych. Może stanowić interesujący produkt np dla firm, tworzących komercyjnie systemy użytkowe, pozwalając pisać oprogramowanie bardziej uniwersalne, przeznaczone do zastosowania z różnymi bazami danych.

ROSI SQL



Halstenbach
ANGEWANDTE COMPUTER TECHNOLOGIE

WSTĘP

ROSI-SQL jest zestawem oprogramowania narzędziowego, niemieckiej firmy ACT Halstenbach. Podstawą zestawu jest uniwersalny język programowania czwartej generacji o nazwie ROSI-SQL, przeznaczony do współpracy z różnymi bazami danych (w standardzie SQL), z interfejsem do relacyjnych baz danych INFORMIX, ORACLE, SYBASE oraz ENTIRE SQL-DB. ROSI-SQL jest dostępny na platformie systemów UNIX-owych oraz pod MS-DOS/MS-Windows.

Wraz z językiem typu 4GL, pakiet oferuje szereg modułów, tworzących zintegrowane środowisko wspomagające projektowanie i programowanie systemów informatycznych. W klasyfikacji narzędzi typu *CASE* pakiet jest zbliżony do grupy narzędzi wspomagania cyklu życia systemu, a w tej grupie do klasy *back-end*. W klasyfikacji implementacyjnej jest to pakiet zintegrowany o charakterze *workbench*.

Jako uzupełnienie powyższego zestawu oferowane są pakiety: ROSI-SQL/Motif, umożliwiający pracę w trybie graficznym oraz ROSI-SQL/OLTP, przeznaczony do tworzenia rozproszonych aplikacji transakcyjnych.

ROSI-SQL Software Production Environment (SPE)

Środowisko tworzenia oprogramowania stanowi zestaw modułów, wspomagających różne fazy pracy - od prototypowania struktury systemu użytkowego, poprzez jego generację, do pielęgnacji i rozwoju oprogramowania (struktura narzędzi przedstawiona jest na diagramie z poprzedniej strony).

ROSI-SQL Prototyper

Jest to sprawne i wygodne oprogramowanie narzędziowe, wspomagające fazę projektowania aplikacji. Umożliwia interakcyjne tworzenie ekranów i systemów menu z opcjonalną modyfikacją.

Po przetworzeniu użytkowy system menu jest dostępny w formacie ASCII. Specjalistyczny interpreter umożliwia bezpośrednie użycie plików menu. Odpowiedni tryb pracy pozwala na szybkie poruszanie po złożonej hierarchii menu przez "Hot Keys". Ponadto system menu może być posortowany odpowiednio dla użytkownika lub grupy.

Zintegrowany interfejs pozwala doświadczonym użytkownikom przetwarzać pliki menu, źródłowe moduły ekranów czy pliki z komunikatami.

Istotne cechy edytora menu:
1. definicja okien w każdym żądanym rozmiarze i położeniu,
2. zmiany rozmiaru i położenia okien,
3. definicja/wymiana/wprowadzanie/kasowanie pozycji menu,
4. definicja nazw pozycji menu,
5. przypisywanie akcji do menu,
6. rozmieszczanie pozycji pionowe/poziome,
7. przypisywanie tekstów komentarzy/podpowiedzi,
8. generowanie systemów menu typu *pull-down*.

Istotne cechy edytora ekranów:
1. edytowanie zawartości ekranów,
2. zmiana rozmiarów okien, położenia, ramek i tytułów,
3. wprowadzanie, modyfikacja i kasowanie pól formularzy,
4. przesuwanie i kopiowanie pól i linii formularzy,
5. pionowe/poziome wprowadzanie linii, rysowanie linii,
6. przypisywanie/modyfikacja atrybutów pól formularzy,
7. definiowanie sekwencji wejściowych.

Zintegrowany generator kodu źródłowego tworzy moduły źródłowe z interakcyjnie budowanych menu oraz układów-projektów formularzy.

ROSI-SQL CodeBrowser

Jest to dodatkowe narzędzie, umożliwiające podwyższenie jakości kodów źródłowych w ROSI-SQL.

1. CodeBrowser umożliwia lokalizację wszystkich odwołań do obiektów ROSI-SQL w złożonej aplikacji o strukturze modularnej.
2. Obiekty, takie jak okna, kanały, bufory wydruku, mogą być wyświetlane z ich położeniem.
3. Poprzez mechanizm odwołań jest możliwe bezpośrednie rozgałęzienie do powiązanych modułów w odpowiednie miejsca kodu.
4. CodeBrowser umożliwia hierarchiczną ekranową reprezentację każdego elementu aplikacji - od modułu deklaracji do zmiennych lokalnych.
5. Eliminacja błędów w nieznanej aplikacji jest względnie prosta, ponieważ konsekwencje usunięcia błędu stają się widoczne.
6. Spójność listowanych obiektów jest zachowana poprzez bieżącą regenerację przez kompilator ROSI-SQL.
7. Dzięki odpowiedniej opcji menu możliwe jest listowanie hierarchii wołań procedur, co daje dobrą czytelność nawet złożonych struktur programowych.
8. Generacja odpowiednich znaczników umożliwia w edytorze "vi" pod systemem UNIX łatwą lokalizację procedur.

Programy źródłowe w ROSI-SQL mogą być standaryzowane i przejrzyście formatowane lub strukturalizowane przez CodeBeautifier.

CodeBeautifier przeszukuje źródła w ROSI-SQL pod kątem słów kluczowych i zmienia je na duże litery, dla łatwości odróżniania ich od nazw zmiennych czy etykiet np. okien, kanałów, kursorów lub buforów wydruku. Wykonuje również odpowiednie wcięcia tabulacyjne w kodzie dla lepszej identyfikacji struktury. Ułatwieniem czytelności kodu jest również wyróżnianie zewnętrznych odwołań.

CodeBeautifier może również być opcjonalnie dołączany dla wspomagania dokumentacji. W tym trybie generuje ramki dla wprowadzanej dokumentacji, używając wszystkich nagłówków aplikacyjnych z odpowiednimi komentarzami i deklaracjami parametrów.

ROSI-SQL Configuration-Management-System

ROSI-CMS umożliwia dialogowe korzystanie ze środowiska do tworzenia systemów informatycznych. Oferuje użytkowy interfejs do wszystkich potrzebnych narzędzi (kompilator, linker itp.).

Magazyny (słowniki danych) ROSI-CMS są organizowane i przechowywane przez systemy baz danych INFORMIX, ORACLE, SYBASE, ENTIRE SQL-DB. Jest zatem możliwość zmiany systemu baz danych.

ROSI-CMS jest konfigurowalny i może być adaptowany do wielu języków programowania (ROSI-SQL, COBOL, C itp.). Ta otwarta architektura daje możliwość obsługi systemów informatycznych, gdy kody źródłowe są pisane w różnych językach. Procedury wywołujące oprogramowanie narzędziowe (kompilator, linker itp.) są zapamiętywane w programach łączących. Indywidualne programy łączące mogą być tworzone stosownie do żądań.

ROSI-CMS stosowany jest głównie do realizacji fazy projektu oprogramowania. Istnieją interfejsy narzędziowe do wspomagania innych faz koncepcji (faza projektowania, faza serwisu), umożliwiające integrację ROSI-CMS z całym środowiskiem.

Duże systemy informatyczne, o modularnej strukturze, zbudowane są z wielu elementów ze złożonymi strukturalnymi współzależnościami. ROSI-CMS minimalizuje zadanie koordynacji stadiów realizacji pracy w wielodostępnym środowisku.

System kontroli wersji umożliwia ROSI-CMS dostęp do wszystkich wersji projektu, jak również wszystkich wersji źródłowych plików tekstowych, rozpoznawanych przez system.

ROSI-CMS może generować informacje dotyczące rozwoju systemu informatycznego, co stanowi ważną część dokumentacji projektu.

Zaletą ROSI-CMS jest także automatyczna generacja plików, niezbędnych dla instalacji systemu użytkowego.

ROSI-SQL Subsidiary Support Administration

ROSI-SQL/SSA - jest stosowane podczas fazy eksploatacji oprogramowania. Nowe żądania oraz komunikaty błędów są archiwowane i służą jako sprzężenia zwrotne przy poprawianiu błędów i dalszym rozwoju systemu informatycznego.

Wykorzystując powyższe informacje zwrotne , szef zespołu może definiować modyfikację wersji na równi z formułowaniem modyfikacji zadań dla projektanta. Sterowany zadaniami rozwój oprogramowania daje możliwość optymalizowania prac wykonawczych. Każda informacja zwrotna może być archiwizowana z dowolnie szczegółowym opisem, dla uniknięcia ewentualnych nieporozumień.

Sterowanie zadaniami przy wykorzystaniu informacji zwrotnych umożliwia elastyczne planowanie, dokonywanie odpowiednich korekt i dotrzymywanie terminów realizacji prac. Jednocześnie zadania podlegają sprawdzaniu przez zespół kontroli jakości, sporządzający rejestry aktualizacji obiektów itp.

Wymiana informacji dla wszystkich zainteresowanych uczestników (szef, programiści, zespół kontroli jakości) realizowana jest przez system poczty elektronicznej. Umożliwia to również koordynację zewnętrznych zespołów projektowych.

ROSI-SQL Dialogue-Specification-Language

DSL jest to narzędzie generacji oprogramowania, umożliwiające tworzenie systemów użytkowych w języku ROSI-SQL w sposób szybki i elastyczny.

Istnieją w nim instrukcje symboliczne dla tworzenia komponentów nadających się do wielokrotnego użycia w systemach użytkowych i dla włączenia wszystkich niezbędnych dialogowych komponentów, dostępów bazowych i ich sterowania. Składnia jest opisana w notacji gramatycznej Backusa-Naura.

Analizator składniowy sprawdza bardzo zwartą składnię dialogowych specyfikacji określających aplikację. Specyfikacje te zawierają wszystkie potrzebne informacje dla generacji przez kompilator DSL kodu źródłowego w ROSI-SQL. Generator kodu źródłowego konfiguruje preformatowane programowe ramki i uzupełnia je odpowiednią informacją.

Dialogowo-specyfikacyjny język bazuje na elastycznym języku docelowym z niezależną architekturalnie koncepcją, umożliwiającą połączenie indywidualnego dialogu i ustawianie parametrów, które w docelowym języku realizują żądania i zapotrzebowania nowoczesnego interfejsu użytkownika.

Dalsze korzyści z takiej koncepcji architektury polegają na usprawnieniu procesu modyfikacji oprogramowania i zachowaniu kompatybilności generowanych programów oraz na ułatwieniu dokumentowania oprogramowania.

ROSI-SQL/Motif

Uzupełnieniem zestawu ROSI-SQL jest ROSI-SQL/Motif, który jest interpreterem języka ROSI-SQL w środowisku graficznym X-Window z interfejsem OSF/Motif. Pozwala to na eksploatowanie tej samej wersji oprogramowania użytkowego, napisanego w ROSI-SQL, zarówno w środowisku znakowym, jak też w środowisku graficznym.

Graficzny interfejs użytkownika może być dowolnie konfigurowany przy pomocy Window-Managera. Możliwe jest również korzystanie z mechanizmu okien wirtualnych, szczególnie przydatnych przy dużych, rozbudowanych systemach informatycznych.

ROSI-SQL/OLTP (OnLine Transaction Processing)

ROSI-SQL/OLTP to wersja ROSI-SQL, specjalnie przygotowana do współpracy z maszynami transakcyjnymi z interfejsem spełniającym normę X/Open XA (np. TUXEDO, Top End, Encine). ROSI-SQL ma bogaty zestaw komend, umożliwiających, przy wykorzystaniu interfejsu X/Open XA, tworzenie i eksploatację oprogramowania użytkowego, wykorzystującego programowe maszyny transakcyjne, rozproszone bazy danych oraz architekturę klient-serwer.

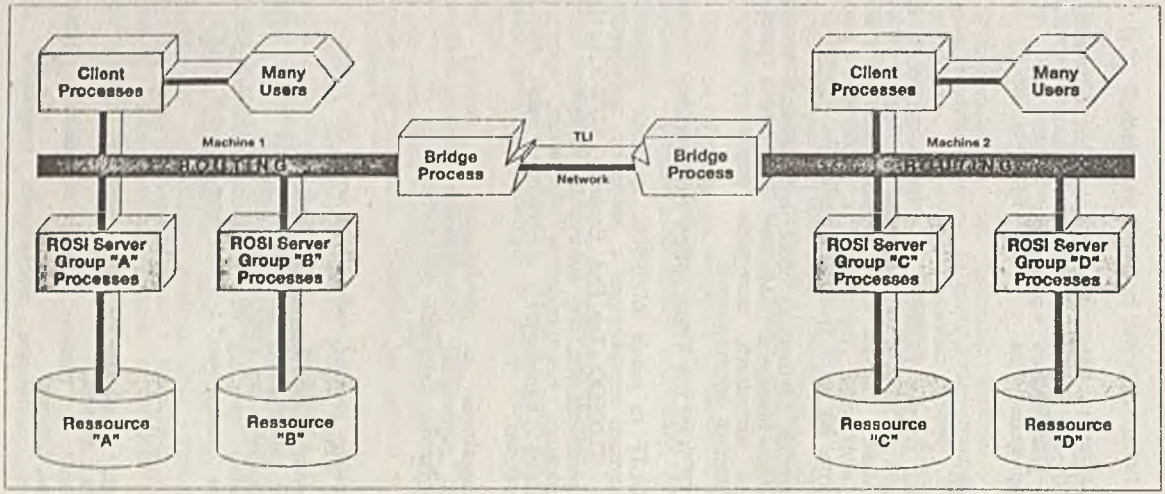
Maszyny transakcyjne są rozwiązaniem o wysokiej sprawności, przeznaczonym do zapewnienia podwyższonego poziomu bezpieczeństwa danych oraz podwyższonej dostępności tych danych.

Zalety:

- zoptymalizowany rozdział zadań użytkowych dla różnych systemów komputerowych,
- jednolite wykorzystanie dostępnych resursów,
- odtwarzanie integralności zasobów danych i stanu systemu po sprzętowym, programowym czy transakcyjnym błędzie,
- bezpieczeństwo transakcji dla wszystkich zasobów gwarantowane przez zatwierdzanie "commit work" i wycofywanie "rollback work".

Cechy ROSI-SQL/OLTP:

- udogodnienia eksploatacyjne dla wymiany danych pomiędzy programami klienta i serwera,
- współdziałanie z różnymi relacyjnymi bazami danych (np. INFORMIX-OnLine, Oracle),
- udogodnienia, ułatwiające uruchamianie programów klienta i serwera.



ROSI-SQL/OLTP (OnLine Transaction Processing)

ROSI-SQL - język programowania czwartej generacji

ROSI-SQL jest uniwersalnym językiem czwartej generacji. Stanowi bardzo sprawne i elastyczne narzędzie, zorientowane na wymagania profesjonalnych programistów. Posiada zintegrowany interfejs do relacyjnych baz danych i spełnia standard języka SQL.

Dialogowa struktura stanowi główną część nakładów programowych dla komercyjnej aplikacji. Część dialogowa jest również jednym z głównych źródeł błędów. Minimalizacja tych nakładów jest jedną z istotnych zalet ROSI-SQL.

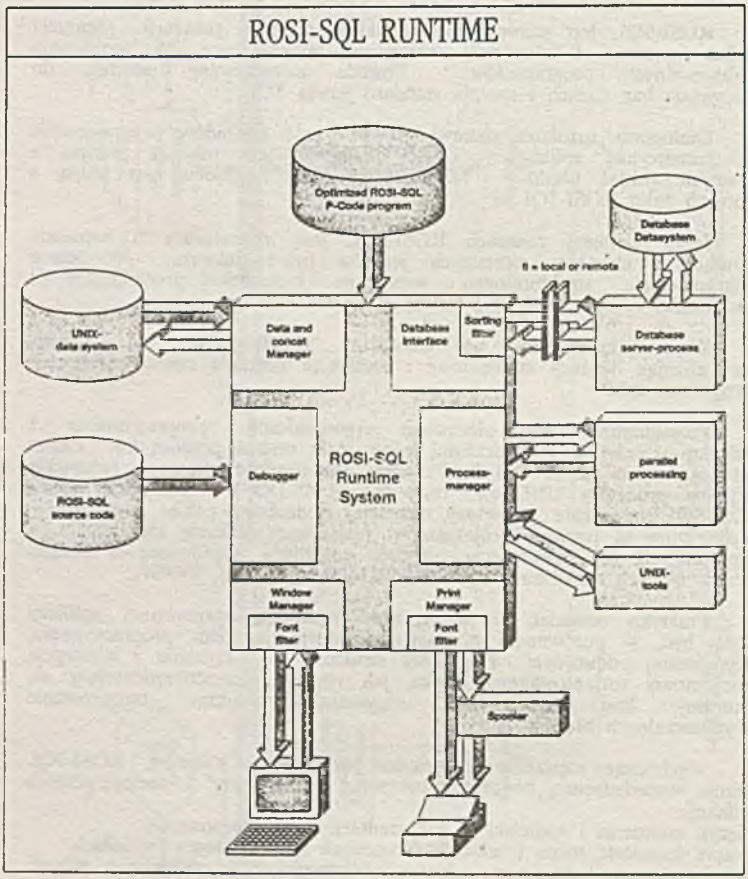
Język czwartej generacji ROSI-SQL jest uzupełniony o komendy sterujące i elementy aktualnych języków proceduralnych. W sensie programowania strukturalnego, wspomaga koncepcje proceduralne i modularne.

Samodokumentująca się składnia "mówi sama za siebie", minimalizując nakłady szkoleniowe i redukując znacznie koszty pielęgnacji oprogramowania.

Projektantowi jest oferowane ergonomiczne oprogramowanie z techniką okienkową, dynamicznymi menu, stałą on-line pomocą itp. Pakiet jest napisany w języku C i daje znaczną niezależność sprzętową (wszystkie aktualne procesory UNIX-owe oraz PC z MS-DOS). Koncepcję języka ROSI-SQL uzupełnia omówiony wcześniej dodatkowy pakiet narzędziowy. Architektura ta pozwala projektantowi realizować aplikację niezależnie od bazy danych, a zintegrowany interfejs umożliwia współpracę z bazami danych różnych systemów (np. INFORMIX, ORACLE, DDB4).

Praktyka dowodzi, że efektywność tworzenia komercyjnej aplikacji może być, w porównaniu z konwencjonalnym językiem programowania, przynajmniej podwojona. Przejrzysta struktura, w połączeniu z koncepcją problemowo zorientowanego języka, jak również samodokumentujący się naturalny kod programowy, zapewniają znaczne zredukowanie przypuszczalnych błędów.

Wychodząc naprzeciw aktywności partnerów i klientów, ROSI-SQL oferuje wszechstronną koncepcję tworzenia "third party" i wielojęzycznych aplikacji:
- język niemiecki i angielski w dokumentacji i oprogramowaniu,
- zapis formatek, menu i tekstów ekranowych w oddzielnych modułach,
- zapis pomocy, komentarzy, tekstów informacyjnych w oddzielnych plikach z komunikatami,
- wygodne dopasowanie narodowych/międzynarodowych kodów znaków.


ROSI-SQL RUNTIME
Optimized ROSI-SQL P-Code program
Database Datasystem
fl = local or remote
UNIX-data system
Data and concat Manager
Database interface
Sorting filter
Database server-process
ROSI-SQL source code
Debugger
ROSI-SQL Runtime System
Process-manager
parallel processing
UNIX-tools
Window Manager
Print Manager
Font filter
Font filter
Spooler

Kompilator ROSI-SQL generuje niezależny pseudo-kod, który może być przetwarzany bez rekompilacji w docelowych systemach na różnych platformach maszynowych przez ROSI-SQL Runtime.

Technika ta pozwala znacznie skrócić czas kompilacji (ok. 4 tys. linii < 6 sek.), a ponadto programy w p-kodzie są optymalizowane i bardziej zwarte niż porównywalne programy w tradycyjnej technice.

Kompilator ROSI-SQL umożliwia deklarowanie zmiennych ze słownika danych systemu bazy danych. Zmiany w słowniku danych będą automatycznie uwzględniane przez odpowiednią funkcję podczas retranslacji aplikacji.

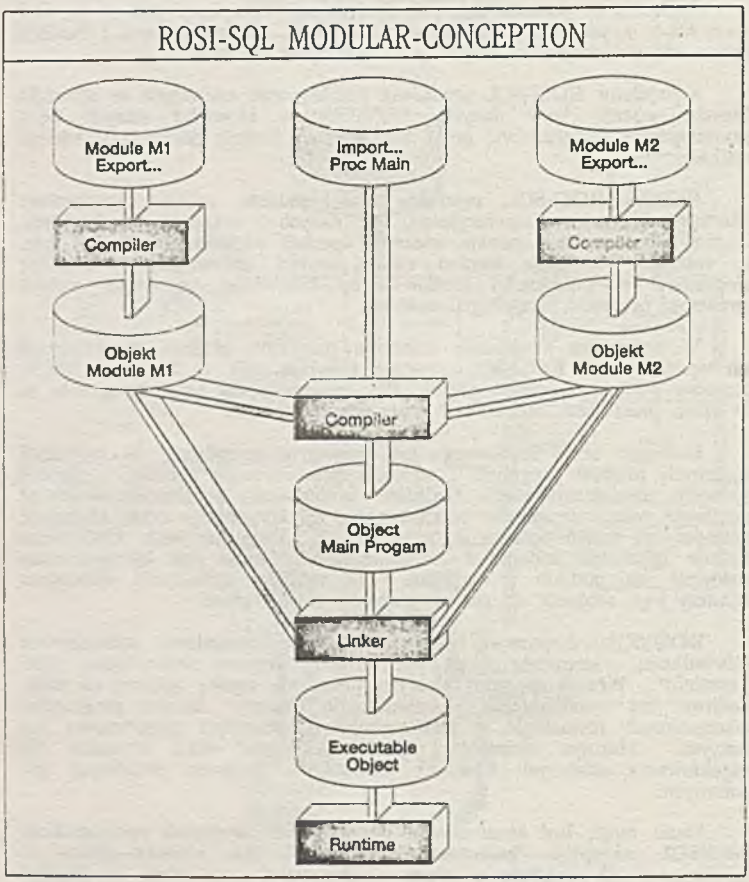
Runtime ROSI-SQL, centralny moduł pakietu, zawiera standardowe interfejsy do systemu operacyjnego, baz danych i urządzeń peryferyjnych. Godną uwagi cechą z punktu widzenia operacji wielodostępnych jest fakt, że runtime potrzebuje bardzo mało pamięci operacyjnej. Zwartość programów w p-kodzie i krótkie czasy ładowania zapewniają wysoką sprawność procesów modyfikacji aplikacji.

W przypadku przerwania działania programu błędem programowym lub operatorskim, ROSI-SQL runtime generuje plik w formacie ASCII, pomocny podczas analizy błędów. Plik zawiera pełny status programu aż do czasu przerwania, włącznie z kopią obrazu ekranu.

Debugger kodu źródłowego jest sprawnym narzędziem do lokalizacji logicznych błędów programu i strojenia przetwarzania aplikacji. Oprócz typowych mechanizmów dla śledzenia wykonywania programów, daje on możliwość mierzenia czasów przetwarzania, np. krytycznego czasu instrukcji, procedur czy żądań dostępu do bazy danych. Debugger kodu źródłowego zajmuje oddzielne zadaniowe okno, rozmiar którego jest konfigurowany stosownie do potrzeb i wymagań. O wysokiej sprawności debuggera świadczy jego zdolność do analizy procesów równoległych.

ROSI-SQL dysponuje inteligentnym, konfigurowalnym sterowaniem wyświetlania; terminale mogą być strojone poprzez odpowiednie pliki "Terminfo". Własny specjalny algorytm umożliwia szybką budowę ekranów. Możliwe jest symultaniczne wyprowadzanie obrazów ekranu programów wykonywanych równolegle, z przełączaniem sterowanym programowo lub ręcznym. Manager okienkowy pozwala definiować okna wirtualne dla projektowania złożonych formatek ekranowych, zarówno poziomych jak pionowych.

Menu mogą być konstruowane stosownie do wymagań operatorskich. ROSI-SQL akceptuje "pull-down" i "pop-up", jak również jedno- i wieloliniowe lub kolumnowe menu. Dostępnych jest wiele możliwości wyświetlania, z dołączaniem kontekstowej pomocy i komentarzy oraz wygodnym sterowaniem klawiszami funkcyjnymi. Specyficzną cechą jest możliwość dynamicznego tworzenia menu w czasie przetwarzania, przy użyciu tzw. "Matrix-Menus".


ROSI-SQL MODULAR-CONCEPTION
Module M1
Export...
Import...
Proc Main
Module M2
Export...
Compiler
Compiler
Objekt
Module M1
Objekt
Module M2
Compiler
Object
Main Progam
Linker
Executable
Object
Runtime

Kompilacja tekstów komunikatów pozwala szybko i spójnie przetwarzać teksty dialogowe. Kompilator komunikatów zabezpiecza rozdział tekstów komunikatów do różnorodnych plików. ROSI-SQL runtime zawiera hierarchiczny system, dopuszczając swobodę redundantnych definicji często używanych tekstów komunikatów w centralnym pliku komunikatów.

Złożone i wszechstronne projekty, tworzone przez zespoły robocze, wymagają ostrożnej, dokładnej dekompozycji i definicji częściowych zadań. Modularna koncepcja ROSI-SQL wychodzi naprzeciw tym wymaganiom.

Indywidualnie tworzone moduły - które mogą być również oddzielnie testowane - mogą być dołączane do końcowego programu przez ROSI-SQL linker. Powiązania pomiędzy modułami są definiowane przez interfejsy import/export. Każdy obiekt aplikacji może być zdefiniowany jako globalny lub lokalny moduł albo lokalna procedura.

ROSI-SQL zapewnia integralność aplikacji przez sprawdzanie importowych połączeń i spójności łączonych modułów zarówno przez kompilator, jak i linker. Jest wyposażony w optymalizacyjny algorytm, usuwający obiekty, do których nie ma odwołań, umożliwiając efektywne administrowanie standardowymi modułami bibliotecznymi.

ROSI-SQL pozwala przetwarzać bez kłopotów programy zewnętrzne. Umożliwia to wykonywanie np. programów obsługi systemu operacyjnego z zewnątrz aplikacji. Możliwości integracji różnych zewnętrznych aplikacji pozwalają tworzyć złożone administracyjnie i komunikacyjnie systemy pod standardowym interfejsem użytkowym. Odpowiednie komendy realizują bezpośrednie funkcje czytania/pisania dla tekstów i plików binarnych. Możliwe jest czytanie/pisanie danych przez pliki, potoki lub używając standardowych rozwiązań wejścia/wyjścia. Ponadto ROSI-SQL wspomaga wszystkie ważne plikowe funkcje systemu operacyjnego.

Dzięki właściwościom konsolidatora możliwe jest tworzenie i administrowanie plikami, konkatenowanymi ze zmiennej liczby plików danych z różnymi długościami linii i różną strukturą.

Użytkowa powłoka ROSI-SQL umożliwia interakcyjne operowanie narzędziami projektowymi. Ustawianie parametrów dla kompilatora, linkera i modułu runtime może się odbywać poprzez menu. Zdefiniowane parametry mogą być pamiętane oddzielnie dla każdego modułu.

INFORMIX SOFTWARE Inc. jest jednym z wiodących producentów systemów zarządzania relacyjnymi bazami danych oraz towarzyszącego oprogramowania narzędziowego. Od wielu lat INFORMIX zajmuje się głównie tą częścią rynku oprogramowania, która dotyczy systemu operacyjnego UNIX oraz współpracujących z nim poprzez połączenia komunikacyjne systemów operacyjnych MS-DOS i MS-Windows. Oprogramowanie INFORMIX jest oferowane w ponad 500 wersjach, dla komputerów ponad 120 producentów dla systemów operacyjnych UNIX (w różnych jego wersjach: SUN OS, ULTRIX, AIX, HP-UX, itd.), NeXT STEP, MS-DOS, Novell Netware, OS/2, Macintosh. Udział firmy INFORMIX w światowym rynku systemów zarządzania relacyjnymi bazami danych, pracującymi pod systemem operacyjnym UNIX, wynosi 37% (według International Data Corporation, 1992), a w Europie 54%. Powyższe wyniki są przede wszystkim efektem szybkiego wykorzystywania w swych produktach najnowszych osiągnięć technologii informatycznej, ścisłego przestrzegania norm i standardów (X/Open, ANSI, FIPS, ISO itd.) i oferowania, po konkurencyjnych cenach, wysokowydajnych serwerów baz danych oraz efektywnych, a zarazem prostych w obsłudze i nauce narzędzi do tworzenia oprogramowania użytkowego.

Od pewnego czasu firma INFORMIX realizuje strategię, zmierzającą do całkowitej otwartości i elastyczności oprogramowania. Jednym z jej celów jest udostępnianie otwartego, dobrze zdefiniowanego interfejsu OpenCase/ToolBus do oprogramowania INFORMIX producentom oprogramowania narzędziowego (w tym typu CASE). Istotnym elementem tej strategii jest także możliwość współpracy z programowymi maszynami transakcyjnymi serwerów baz danych INFORMIX-OnLine, jak również oprogramowania narzędziowego pozwalającego korzystać z heterogenicznych rozproszonych baz danych, pracujących z różnymi maszynami transakcyjnymi na różnych komputerach z różnymi systemami operacyjnymi.

Powyższe podejście zaowocowało szeroką ofertą oprogramowania typu CASE, współpracującego z oprogramowaniem INFORMIX, np.: UNIFACE (Uniface Corporation), ISEE (Westmount), ROSI-SQL (Halstenbach GmbH), PowerBuilder (Powersoft Corporation), ObjectView 2.0 (Matesys Corporation), Q+E Database Editor i Q+E Database Library (Pioneer Software) itd. Wykorzystanie programowych maszyn transakcyjnych spełniających interfejs X/Open XA, zarówno przez serwer bazy danych INFORMIX-OnLine, jak i przez oprogramowanie narzędziowe INFORMIX-4GL, stwarza zupełnie nowe możliwości w zakresie rozproszonych baz danych. Rozwiązanie takie pozwala, aby korzystając z oprogramowania INFORMIX, wykonać "globalną" transakcję obejmującą heterogeniczne rozproszone bazy danych, pracujące na różnych maszynach transakcyjnych (spełniających interfejs X/Open XA).

Przedstawiamy krótką charakterystykę pakietów systemu INFORMIX, wraz z kilkoma informacjami na temat wchodzącej niedługo na rynek wersji 6.

Narzędzia do tworzenia oprogramowania użytkowego

W systemie INFORMIX możliwe są dwa technologicznie różne podejścia do tworzenia oprogramowania (w pierwotnej koncepcji jedna metoda miała służyć do uruchamiania i testowania programów, a druga do konsolidacji oprogramowania przekazywanego do eksploatacji). W praktyce pierwsze rozwiązanie okazuje się być tak sprawne i wygodne również w procesie użytkowania oprogramowania, że, przy jego dodatkowych zaletach, stało się znacznie bardziej rozpowszechnione.

Rozwiązanie pierwsze polega na wykorzystaniu pakietu INFORMIX-4GL Rapid Development System oraz pakietu INFORMIX-4GL Interactive Debugger (opcjonalnie). INFORMIX-4GL RDS zawiera kompilator języka INFORMIX-4GL do pseudo-kodu oraz interakcyjny interpreter pseudo-kodu. Pakiet INFORMIX-4GL w wersji kompilacyjnej generuje (w kilkustopniowym procesie) kod wykonywalny. Długotrwały proces kompilacji i konsolidacji modułów programowych, wraz z dołączaniem bibliotek, czyni to narzędzie uciążliwym dla etapu uruchamiania oprogramowania. W wariancie pierwszym (I-4GL RDS + I-4GL ID) testowanie i modyfikacje oprogramowania przebiegają szybko i łatwo. W praktyce okazuje się, że rzeczywista szybkość działania oprogramowania aplikacyjnego, współdziałającego z INFORMIX-4GL RDS, wcale lub bardzo nieznacznie ustępuje szybkości działania wersji skompilowanej. Dlatego też większość oprogramowania użytkowego w języku INFORMIX-4GL została napisana i jest eksploatowana przy wykorzystaniu pakietu INFORMIX-4GL RDS.

Wielką zaletą INFORMIX-4GL RDS jest to, że generowany przez ten pakiet pseudo-kod można przenosić między różnymi komputerami i systemami operacyjnymi bez konieczności kompilacji, czego nie oferują inni producenci SZRBD. Z tej możliwości korzysta wiele firm tworzących oprogramowanie użytkowe, w ten sposób obniżając swym klientom koszty niezbędnego oprogramowania narzędziowego, ponieważ na komputerach użytkowników może być instalowane oprogramowanie INFORMIX w wersji Runtime, którego cena stanowi od 30 do 70% ceny oprogramowania w pełnej wersji (Development).

Uzupełnieniem pakietu INFORMIX-4GL RDS jest INFORMIX-4GL/GX będący interpreterem pseudo-kodu, umożliwiającym wykonywanie programów napisanych w języku INFORMIX-4GL w środowisku graficznym OpenLook, OSF/Motif i MS-Windows. Rozwiązanie to pozwala, aby jedna wersja oprogramowania użytkowego napisanego w języku INFORMIX-4GL była wykonywana zarówno w środowisku znakowym, jak też graficznym, bez konieczności zmian w kodzie źródłowym oprogramowania.

W grupie narzędzi typu lower CASE, INFORMIX oferuje pakiety do generowania pełnoekranowych menu (INFORMIX-4GL Menus) oraz formatek ekranowych (INFORMIX-4GL Forms), wraz z kodem źródłowym w języku INFORMIX-4GL, przydatne szczególnie na etapie prototypowania oprogramowania użytkowego.

Do grupy narzędzi do prototypowania, a także łatwego i szybkiego tworzenia prostych aplikacji, zaliczyć można pakiet INFORMIX-SQL. Jest to moduł bardzo popularny w eksploatowanych zestawach oprogramowania INFORMIX obejmujący: edytor schematów baz danych, edytor zdań języka SQL, generator i interpreter formatek ekranowych, generator i interpreter raportów, generator menu.

Poza wspomnianymi pakietami do oprogramowania narzędziowego INFORMIX należą również poniższe moduły:

INFORMIX-TP/Toolkit
to zestaw funkcji pozwalających na tworzenie i eksploatowanie oprogramowania napisanego w języku INFORMIX-4GL, wykorzystującego możliwości maszyn transakcyjnych oraz architektury klient/serwer.

INFORMIX-OpenCase/ToolBus
to kompleksowe środowisko do tworzenia oprogramowania pozwalające na integrację narzędzi typu CASE innych producentów ze środowiskiem oprogramowania narzędziowego opartego o język INFORMIX-4GL.

INFORMIX-OpenCase/Encapsulator
to narzędzie umożliwiające integrację ze środowiskiem OpenCase/ToolBus dodatkowych produktów innych producentów, a tym samym korzystanie z jednolitego środowiska przy tworzeniu oprogramowania użytkowego.

INFORMIX-4GL for ToolBus
to środowisko graficzne do tworzenia oprogramowania użytkowego przy wykorzystaniu języka INFORMIX-4GL, umożliwiające wykorzystanie narzędzi typu CASE poprzez interface OpenCase/ToolBus.

INFORMIX-4GL RF
to oprogramowanie narzędziowe pozwalające na tworzenie i eksploatowanie oprogramowania użytkowego przeznaczonego dla komputerów przenośnych, wykorzystujących radiolinie jako łącza komunikacyjne.

Język czwartej generacji INFORMIX-4GL, o bardzo dużych możliwościach, pozwala na napisanie całej aplikacji w tym języku. Tym użytkownikom, którzy z różnych powodów preferują programowanie w językach trzeciej generacji, INFORMIX oferuje pakiety INFORMIX-ESQL tj. język SQL zanurzony w języku trzeciej generacji: C, COBOL, FORTRAN, ADA.

Produkty sieciowe i dodatkowe

INFORMIX-STAR
uzupełnia możliwości serwera INFORMIX-OnLine o obsługę rozproszonych baz danych, pozwalając na:
- równoczesną pracę na kilku bazach danych zlokalizowanych na różnych serwerach w sieci,
- rozproszone transakcje wykorzystujące protokół "Two-Phase Commit".

INFORMIX-NET
uzupełnia funkcje serwera INFORMIX-Standard Engine o możliwość połączenia ze zdalną bazą danych, zlokalizowaną na innym serwerze sieci oraz przy pracy z wykorzystaniem architektury klient/serwer umożliwia dostęp z komputera-klienta do rozproszonych baz danych.

INFORMIX-TP/XA
to oprogramowanie umożliwiające współpracę pomiędzy serwerem bazy danych INFORMIX-OnLine a maszynami transakcyjnymi z interface'm spełniającym normę X/Open XA np. TUXEDO System T - UNIX System Laboratories, Top End - NCR, Encine - Transarc, UTM-Siemens-Nixdorf, AIX/CICS-IBM. Zastosowanie takiego rozwiązania umożliwia przetwarzanie transakcji pochodzących z heterogenicznych, rozproszonych baz danych, pracujących na różnych komputerach, z różnymi maszynami transakcyjnymi. Wykorzystanie maszyn transakcyjnych pozwala na zwielokrotnienie szybkości przetwarzania transakcji.

INFORMIX-OnLine/Optical
to oprogramowanie pozwalające na wykorzystywanie przez serwer bazy danych INFORMIX-OnLine optycznych jednostek dyskowych.

INFORMIX-DataExtract
to interface umożliwiający dostęp ze środowiska INFORMIX do następujących systemów zarządzania bazami danych: DB2, SQL/DS, IMS/DB, DL/I, IDMS, ADABAS, TOTAL, SUPRA, MODEL 20, DATACOM/DB oraz plików typu VSAM i SAM.

WINGZ
to graficzny arkusz kalkulacyjny współpracujący z serwerami baz danych INFORMIX. Rdzeniem pakietu WINGZ jest HyperScript - język programowania przeznaczony do tworzenia aplikacji graficznych oraz do budowy graficznych interface użytkowników.

Serwery baz danych

INFORMIX-OnLine

to serwer bazy danych o bardzo wysokiej wydajności, przeznaczony do zastosowań wymagających bieżącego przetwarzania transakcji (OLTP), ciągłej pracy, archiwizacji w czasie działania bazy danych w trybie on-line, wysokiego bezpieczeństwa danych oraz dużej niezawodności działania.

Charakteryzuje się następującymi cechami:

- baza danych istnieje poza systemem plików systemu operacyjnego, posiada bezpośredni dostęp do raw device, dostęp do bazy odbywa się w całości poprzez mechanizmy INFORMIX-OnLine, co pozwala na zwielokrotnienie szybkości działania;
- wykorzystywanie architektury klient/serwer;
- wielkość bazy danych jest ograniczona jedynie pojemnością pamięci masowych;
- monitorowanie i optymalizacja (strojenie) serwera bazy danych;
- wspólny segment danych w pamięci operacyjnej (shared memory);
- możliwość przechowywania danych w zmiennych typu Binary Large Object (BLOB) o wielkości do 2 GB, w których mogą się znajdować: zapis video, zapis dźwięku, mapa, zdjęcie itd;
- zarządzanie systemem dysków zwierciadlanych (disk mirroring);
- wykorzystanie możliwości komputerów multiprocesorowych poprzez własne mechanizmy;
- możliwość archiwowania, w czasie pracy serwera bazy danych, zarówno zawartości całej bazy danych jak też zmian w bazie danych;
- wysoka niezawodność bazy danych realizowana poprzez mechanizmy zabezpieczenia integralności bazy (Referential Integrity, Entity Integrity, Transaction Logging, Internal Consistency Checking, Triggering Operation);
- zastosowanie "Stored Procedures";
- wykorzystanie protokołu "Two-Phase Commit";
- zastosowanie architektury wielowątkowej (multithreading);
- wykorzystanie rozwiązań multimedialnych;
- wysokie bezpieczeństwo danych realizowane poprzez wielopoziomowy system praw dostępu i haseł oraz wykorzystanie "Stored Procedures";
- współpraca z maszynami transakcyjnymi poprzez interface X/Open - XA.

INFORMIX-OnLine/Secure

to serwer bazy danych INFORMIX-OnLine o najwyższej istniejącej klasie bezpieczeństwa tj. klasie B1 według United States Departament of Defense "Orange Book" oraz klasy F3/E3 określonej przez European Information Technology Security Evaluation Criteria (ITSEC). Jest to pierwszy serwer bazy danych spełniający wymagania klasy bezpieczeństwa B1, który uzyskał stosowny certyfikat U.S. National Computer Security Center (NCSC). INFORMIX-OnLine/Secure współpracuje z systemami operacyjnymi o klasie bezpieczeństwa B1 (np. UNIX System V Release 4.2 Enhanced Security), zapewniając bezpieczeństwo poprzez odpowiednią architekturę serwera bazy danych oraz wielopoziomowy system zabezpieczeń, wykorzystujący Discretionary Access Control i Mandatory Access Control.

INFORMIX-Standard Engine
to nie wymagający obsługi serwer bazy danych zalecany do małych i średnich baz danych, efektywny w systemach bez dużego obciążenia bieżącym przetwarzaniem transakcji.

INFORMIX-OnLine Dynamic Server v.6.0

Należy oczekiwać, że w niedługim czasie zaczną być handlowo dostępne pakiety wersji 6.0 INFORMIXa. Oto próba wstępnej charakterystyki nowego pakietu INFORMIX-OnLine Dynamic Server i zarys tendencji modernizacyjnych i rozwojowych.

W stosunku do wersji 5 nastąpiła istotna koncepcyjnie zmiana obsługi serwera bazy danych. INFORMIX-OnLine dynamic Server wykorzystuje architekturę wielowątkową w odniesieniu zarówno do klienta, jak też serwera bazy danych, oraz procesory wirtualne, pracujące zarówno na komputerach wielo-, jak też jednoprocesorowych. Każde zadanie klienta jest realizowane przez oddzielny wątek (thread). Serwer bazy danych rozdziela poszczególne wątki na dostępne procesory wirtualne, co umożliwia ich równoległe przetwarzanie oraz lepsze wykorzystanie zasobów systemu komputerowego.

Poszczególne procesory wirtualne mogą być przydzielone do określonych klas, realizujących odpowiednie zadania, np. operacje przetwarzania procesorowego, operacje czytania lub zapisu na jednostkach dyskowych, operacje wymiany danych między wątkami klientów i serwera bazy danych itd. Możliwe są dynamiczne zmiany przyporządkowania procesorów wirtualnych do poszczególnych klas.

Rozwiązania te doprowadziły w konsekwencji do przebudowy architektury systemu INFORMIX. Naturalnym stało się włączenie dotychczasowych funkcji INFORMIX-STAR do modułu INFORMIX-OnLine Dynamic Server. Rozwiązania te przenoszą w pewnym sensie część funkcji systemu operacyjnego na poziom funkcji serwera, co daje możliwośc lepszej specjalizacji rozwiązań. Przez obejście ograniczeń systemu operacyjnego i przełączanie kontekstów użytkownika w warstwie działania serwera zostają zmniejszone przeciążenia systemu, co daje odczuwalne zyski przy dużej liczbie użytkowników. Znikają ograniczenia związane z przełączaniem procesów w systemie. Uzyskuje się większą skalowalność, dynamiczne dostosowywanie zasobów systemu (np. pamięć współdzielona) i zmienną konfigurację bazy danych.

Zmodyfikowano funkcje obsługi dostępu do jednostek dyskowych, umożliwiając równoległe przetwarzanie zadań - zapytań do bazy, co pozwala zwiększyć sprawność i efektywność pracy. Zrealizowano możliwość optymalizacji przydzielania zadań, organizowania blokad itp., poprzez m.in. mechanizm priorytetowania zadań na poziomie SZBD.

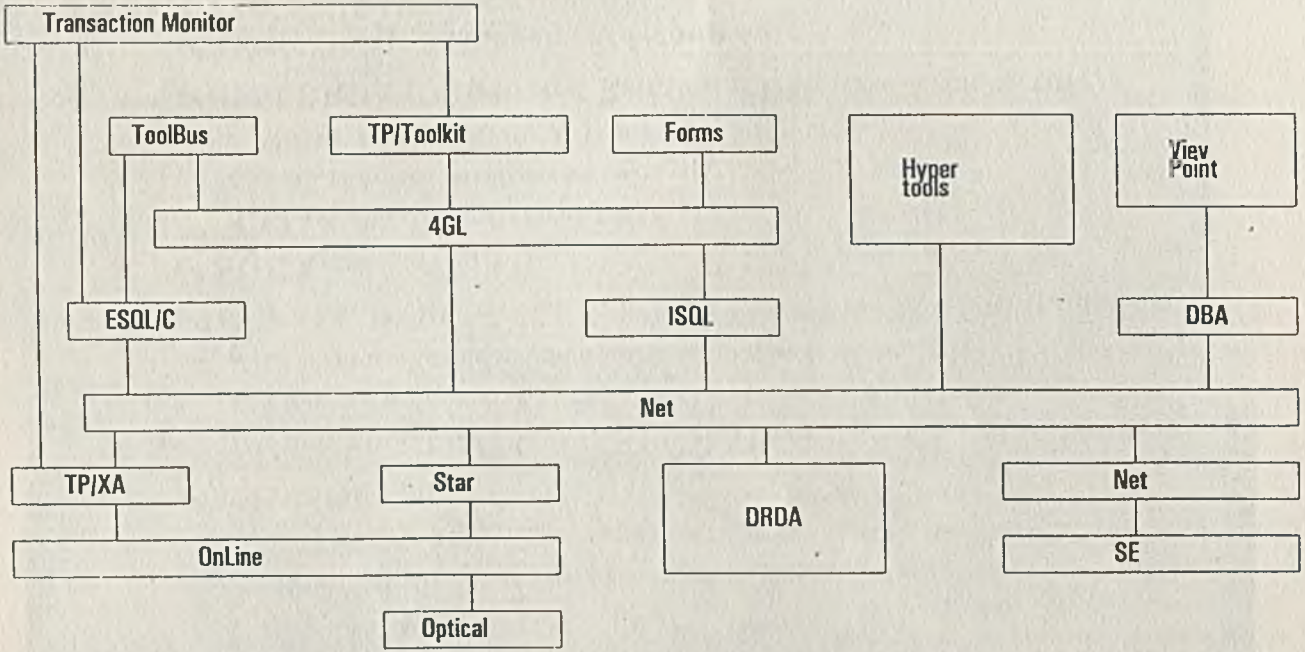
Asynchroniczny charakter we/wy uzyskuje się m.in. przez wykorzystywanie odpowiednich statystyk dostępów do dysków, w wyniku czego nie występuje zjawisko czekania na zasoby czy przechodzenia systemu w stan uśpienia. Umożliwia to wprowadzenie równoległego indeksowania, czytania wyprzedzającego, tworzenia histogramów rozmieszczenia tabel itp. Zoptymalizowano mechanizmy zwierciadlanego zapisu danych. Więcej jest narzędzi do monitorowania pracy systemu. Możliwe jest, częściowo automatyczne, dynamiczne strojenie parametrów INFORMIX-OnLine. W dodatkowej dzielonej pamięci typu cache przechowywane mogą być najczęściej używane struktury - tabele systemowe, procedury itp.

Rozbudowując rozwiązania więzów integralnościowych, znane z wersji 5, przeniesiono na poziom zdań rozszerzonego języka SQL, często stosowane kaskadowe usuwanie rekordów (kasowanie wraz z rekordem nadrzędnym wszystkich związanych z nim podrzędnych i kaskadowo dalej).

Dopuszczono operację archiwizowania fragmentów bazy, poszczególnych tabel i usprawniono mechanizmy archiwizowania równoległego na wielu urządzeniach. Wprowadzono elementy inteligentnych rozwiązań do obsługi nośników archiwizacyjnych (np. odpowiednie oznaczanie taśm, w celu łatwiejszego dynamicznego odtwarzania), stworzono możliwość ustalania momentów czasowych archiwizacji. Wbudowano opcję tworzenia kopii wielokrotnych, szyfrowania, kompresji zapisów, sum kontrolnych itp.

Wprowadzono możliwość automatycznej replikacji odpowiednich struktur, pod kątem wykorzystania w konfiguracjach sprzętowych typu *cluster.*

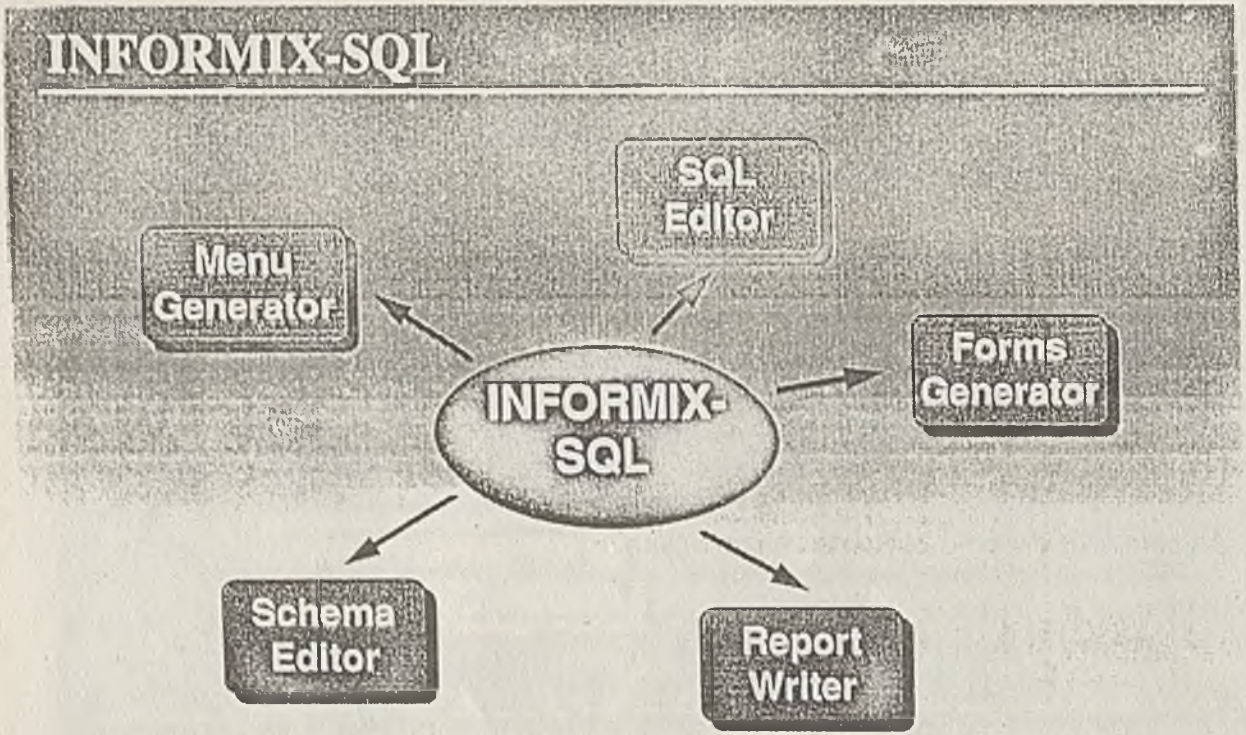
Zrealizowano możliwość pracy w standardzie NLS (Native Language Support). Zostało zaimplementowane wspomaganie działania w wersjach językowych w oparciu o specyfikację X/Open, XPG3.



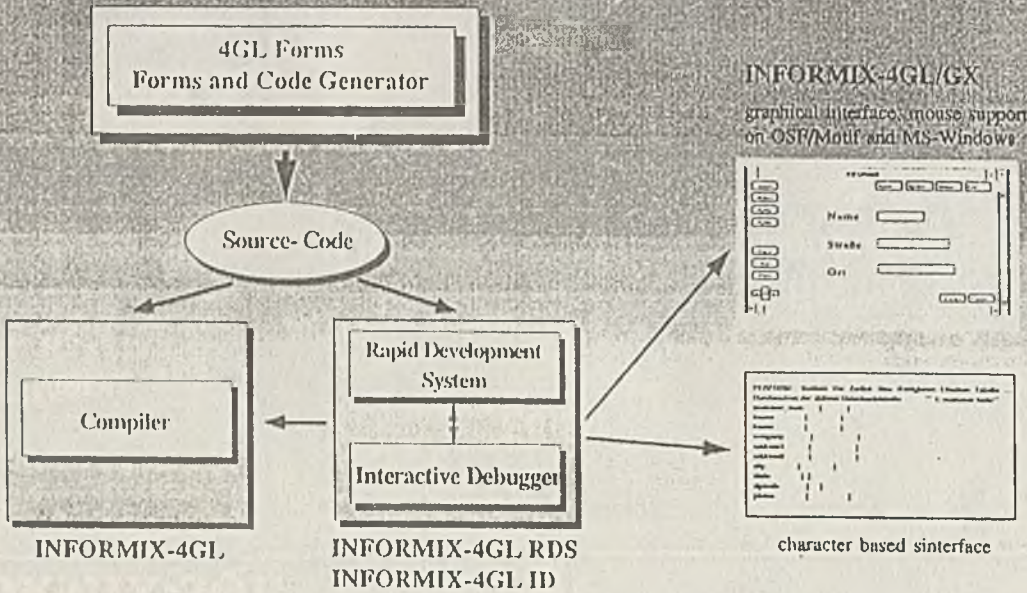
Architektura systemu INFORMIX

Informix and Standards

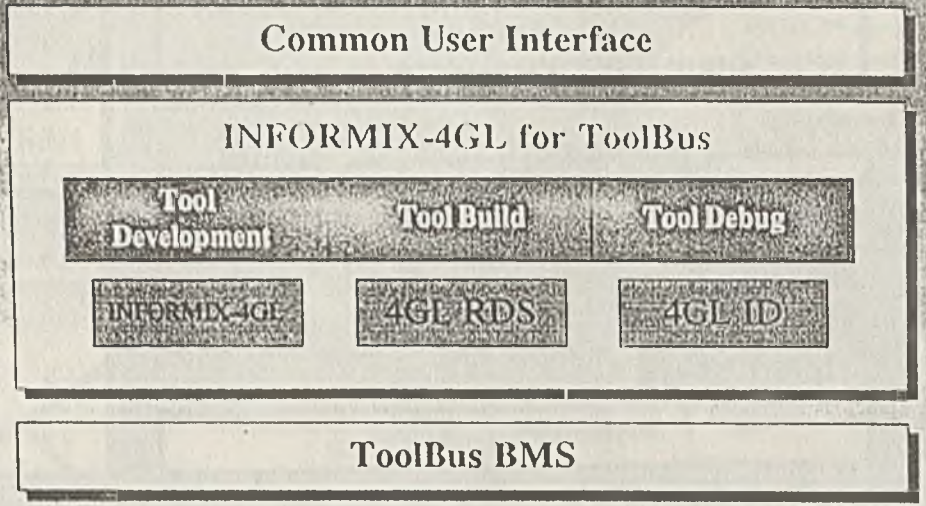
- ✓ ANSI-SQL89 Level II, ISO 9075:1989
  - ✗ ANSI declarative integrity constraints
  - ✗ NIST certificate
- ✓ ANSI-SQL2 entry Level
- ✓ X/Open Portability Guide XPG3, XPG4
  - ✗ X/Open dynamic SQL
  - ✗ XA-Protocol for distributed transaction management
- ✓ OSF, UI, ISO/OSI, IEEE, POSIX, AT&T Sys V Definitions
- ✓ SQL Access Group
  - ✗ CLI (Call Level Interface), UNIX
  - ✗ RDA (Remote Database Access)
  - ✗ ODBC, IDAPI, Microsoft Windows
- ✓ IBM DRDA (Distributed Relational Database Architecture)
- ✓ OMG Object Management Group


INFORMIX-SQL
SQL Editor
Menu Generator
Forms Generator
INFORMIX-SQL
Schema Editor
Report Writer

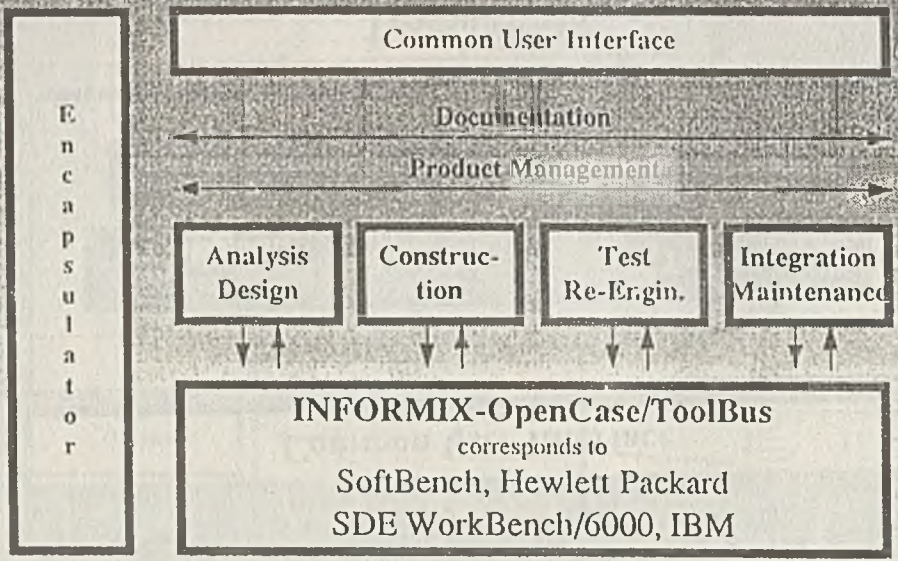
INFORMIX-4GL Product Family

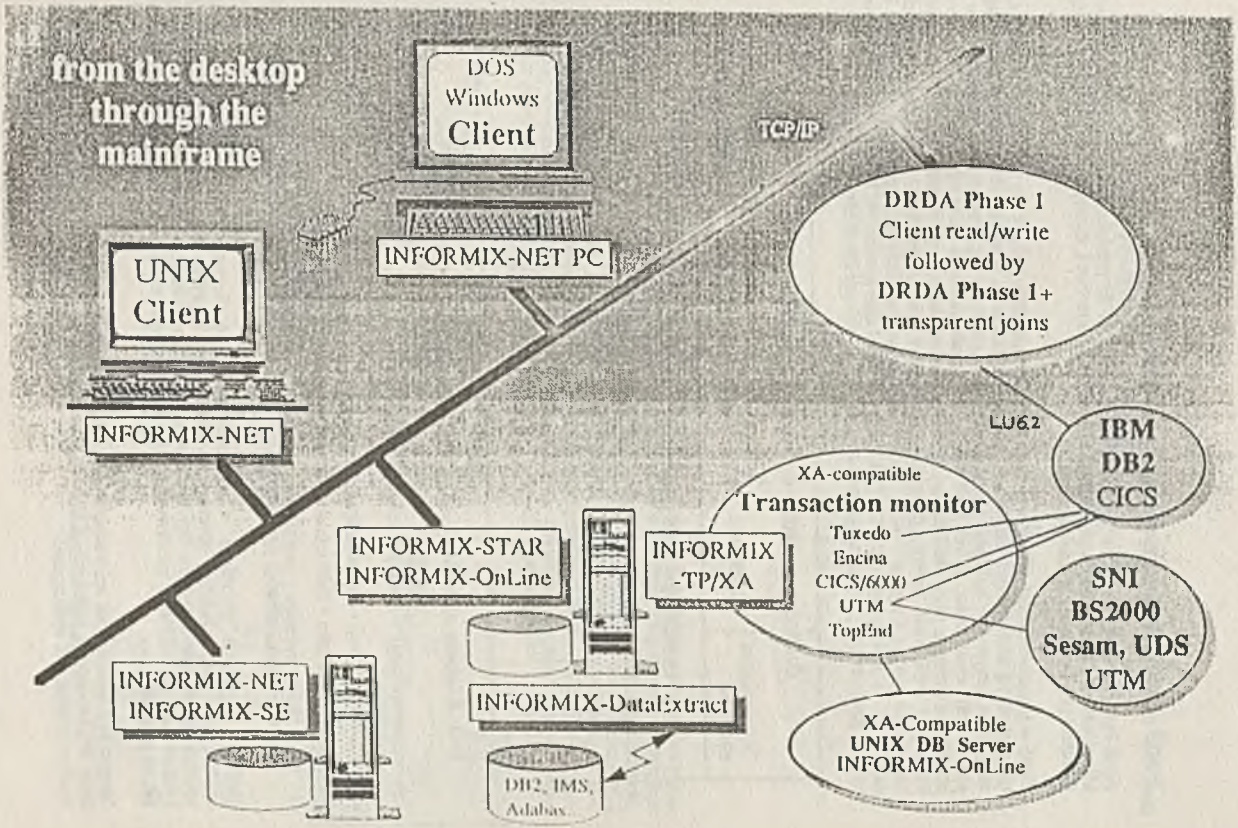


INFORMIX-4GL for ToolBus



INFORMIX-OpenCase/ToolBus: Integrated Development Environment

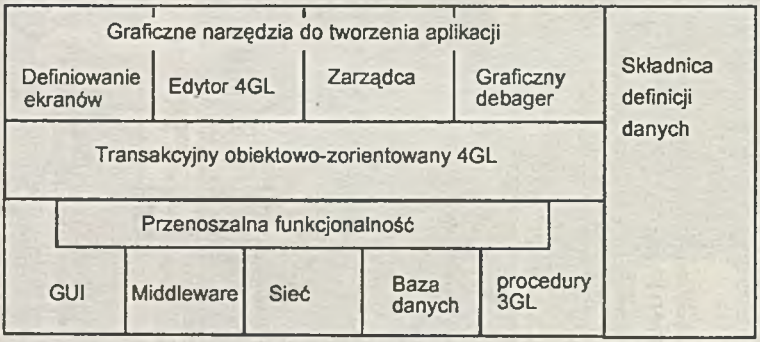



from the desktop
through the
mainframe
DOS
Windows
Client
TCP/IP
DRDA Phase 1
Client read/write
followed by
DRDA Phase 1+
transparent joins
INFORMIX-NET PC
UNIX
Client
INFORMIX-NET
LU6.2
IBM
DB2
CICS
XA-compatible
Transaction monitor
Tuxedo
Encina
CICS/6000
UTM
TopEnd
INFORMIX-STAR
INFORMIX-OnLine
INFORMIX
-TP/XA
SNI
BS2000
Sesam, UDS
UTM
INFORMIX-NET
INFORMIX-SE
INFORMIX-DataExtract
XA-Compatible
UNIX DB Server
INFORMIX-OnLine
DB2, IMS,
Adabas

Forté - narzędzie czwartej generacji przyszłości

Forté łączy w sobie cechy narzędzia czwartej generacji do produkcji rozproszonych aplikacji klient/serwer z interfejsem typu GUI (Graphical User Interface), dostępem do baz danych i elementami przetwarzania transakcyjnego.

Forté w przeciwieństwie do wielu innych pakietów nie służy jedynie do tworzenia aplikacji ale również do jej pielęgnacji i strojenia w okresie eksploatacji. Rozwiązanie to bierze pod uwagę fakt, że w przypadku aplikacji klient/serwer w środowisku sieciowym ciągle stykamy się z problemem wydajności. Forté daje możliwość skalowalności aplikacji wraz ze wzrostem obciążenia systemu przy wzrastającej liczbie użytkowników. Rozwiązaniem, które udostępnia Forté jest możliwość parcelacji aplikacji pomiędzy wiele serwerów. Dla jednego modelu logicznego można wygenerować w sposób automatyczny wiele implementacji fizycznych w zależności od posiadanej konfiguracji. Logiczny model daje możliwość łatwej przenośności pomiędzy platformami przy jednoczesnej optymalizacji aplikacji dla różnych platform.



Rys 1. Środowisko tworzenia aplikacji w Forté

Tworzenie aplikacji w Forté przebiega trzy etapowo:

1. definicja aplikacji - definicja funkcjonalności
2. generacja systemu - aplikacja zostaje rozparcelowana i wygenerowany kod
3. eksploatacja systemu - zarządzanie komunikacją pomiędzy obiektami aplikacji i ich stałe śledzenie oraz możliwość rekonfiguracji

Definicja aplikacji przebiega z wykorzystaniem graficznego edytora ekranów, wielożytkownikowej składnicy, obiektowo-zorientowanego języka 4GL oraz graficznego pakietu uruchomieniowego. Składnica daje możliwość zarządzania obiektami aplikacji podczas procesu tworzenia. Język 4GL daje możliwość dzielenia kodu i jego wielokrotnego użycia dzięki bibliotekom klas obiektów.

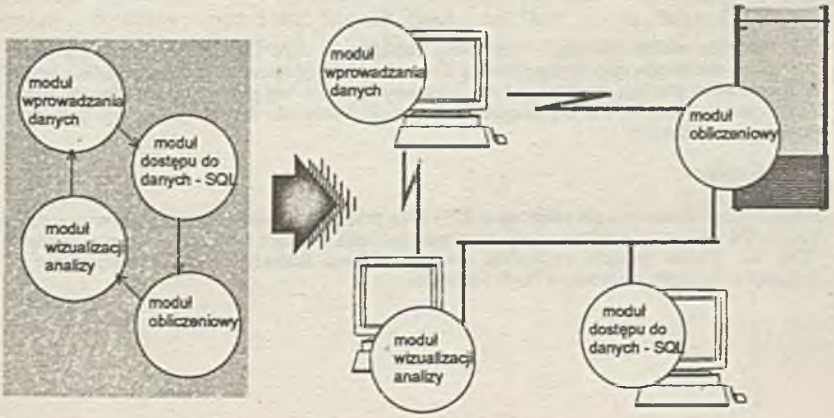
Integracja z istniejącymi aplikacjami jest możliwa poprzez pakiet ObjectBroker będący Digitalową implementacją standardu CORBA. Rozwiązanie to pozwala na integrację aplikacji w Forté z istniejącymi aplikacjami.

Forté z założenia jest produktem pracującym na wielu platformach. W wersji 1.0 Forté współpracuje z następującymi systemami:

- interfejs graficzny
  - MS Windows
  - Macintosh
  - Motif
- system operacyjny serwera
  - OSF/1
  - OpenVMS (VAX, AXP)
  - Sequent Dynix
  - IBM RS/6000 AIX
  - HP/9000 HP/UX
  - DG AViiON
  - Sun/OS i Solaris
- system sieciowy
  - TCP/IP
  - NetWare
  - DECnet
  - AppleTalk
  - PATHWORKS
- baza danych
  - Oracle
  - Sybase
  - DEC Rdb

Parcelacja aplikacji

Jednym z głównych powodów stosowania aplikacji klient/serwer jest maksymalne wykorzystanie mocy posiadanych maszyn. Do tej pory to projektanci i programiści tworzący aplikacje decydowali o tym, jaka część funkcjonalności ma być wykonywana na serwerze czy kliencie. Użycie Forté pozwala na przesunięcie takiej decyzji na czas późniejszy tj. do fazy eksploatacji i instalacji. Dzięki temu, podczas tworzenia aplikacji można wykorzystywać ulubione środowisko sprzętowo-systemowe a następnie przenieść przygotowany system na docelową konfigurację wykonawczą. Istnieje także możliwość zmiany docelowej konfiguracji poprzez odpowiednie przebudowanie partycji.



Rys 2. Aplikacja Forté składa się z obiektów, które można w sposób automatyczny rozproszyć w heterogenicznym środowisku sieciowym

Przenośność

Elastyczność związana z możliwością parcelacji daje nie tylko zwiększenie wydajności. Ponieważ tworzenie aplikacji jest niezależne od docelowej konfiguracji, daje to możliwość niezależnego konfigurowania sprzętu i oprogramowania w obu fazach. Wykorzystanie interfejsu graficznego jest w pełni niezależne od konkretnej implementacji. Forté zawiera nadzbiór funkcjonalności związanej z interfejsem GUI. Aplikacja zdefiniowana przy pomocy jednego interfejsu działa bez zmian na innym interfejsie GUI. Współdziałanie z bazami danych możliwe jest z wykorzystaniem dynamicznego języka SQL, języka SQL skompilowanego oraz procedur składowanych.

Możliwość integracji

Integracja jest poważnym problemem dla wielu narzędzi 4GL. Zawsze istnieje potrzeba integracji istniejących aplikacji oraz dodawania nowej, nietypowej funkcjonalności. Forté posiada możliwość integracji z wykorzystaniem pakietu ObjectBroker, mechanizm wywoływania procedur zewnętrznych lub obiektów C++. Dzięki zunifikowanemu modelowi obsługi zdarzeń możliwe jest integrowanie w ramach aplikacji zdarzeń z GUI, zdarzeń aplikacji oraz zdarzeń generowanych przez system.

Niezawodność

Dla każdej większej aplikacji problem niezawodności jest niezwykle ważny i powoduje, że poświęca mu się niezmiernie dużo czasu podczas tworzenia aplikacji. Forté pomaga w rozwiązywaniu tych problemów na wiele sposobów: Po pierwsze, Forté pozwala na zdefiniowanie obiektów jako obiektów transakcyjnych co powoduje, że gwarantowana jest atomowość ich wykonania. Dodatkowo istnieje możliwość zwiększenia odporności na wystąpinie awarii poprzez odpowiednią metodę parcelacji aplikacji. Możliwe jest zdefiniowanie zapasowej partycji aplikacji, która zostanie wykorzystana w przypadku awarii partycji głównej. Mechanizm ten jest w pełni przezroczysty dla użytkowników.

Wydajność

Forté poprzez analizę aplikacji i docelowej konfiguracji stara się za wszelką cenę zminimalizować obciążenie sieci komputerowej. Dla każdej partycji generowany jest kod C++ wraz ze standardową partycją. Kod ten jest kompilowany na docelowej platformie. Interfejs do bazy danych jest w stanie wybrać najkorzystniejszy sposób dostępu do danych (zanurzony SQL czy procedury składowane).

Forté Software

Firma Forté Software została założona w 1991 roku przez grupę znanych projektantów narzędzi CASE. Od samego początku zawarto silne porozumienia z takimi firmami jek Apple, Digital, IBM, itd. Digital posiada szczególne związki z Forté Software. Zespół projektantów i programistów Digitala pracuje w Forté Software.

COMPUTER ASSOCIATES

CA - Visual Objects

CA - Super-Project

Materiał przedstawiony w tym artykule nie jest dokładnym odwzorowaniem wykładów jakie zostaną przedstawione podczas Wiosennej Szkoły PTI w Świnoujściu. Zadaniem tego materiału jest jedynie przybliżenie słuchaczom produktów, które zostaną tam zaprezentowane.

Materiał dotyczący CA-Visual Objects opisuje jedynie idee tego produktu oraz założenia jakie zostały przyjęte w trakcie jego powstawania.

Opis CA-SuperProject przedstawia tylko jego główne cechy i możliwości, dokładniejsza prezentacja tego produktu miałaby objętość książki.

CA-Visual Express najnowszy produkt firmy Computer Associates podobnie jak CA-SuperProject został zaprezentowany w sposób bardzo ogólny. Powodem tego jest jednak fakt, że jest to produkt nowy, który do oficjalnej sprzedaży wejdzie w połowie tego roku i informacje na jego temat są jeszcze niepełne.

CA-Visual Objects
- przyszłość CA-Clipper i CA-dBFast

Pierwszym programem kojarzonym z terminem Xbase (wspólna nazwa dla programów dBase, FoxPro, CA-Clipper, CA-dBFast i podobnych) był niewątpliwie dBase II, program firmy Ashton-Tate Software Company. Jego pierwsza wersja dla mikrokomputerów wzorowana była na systemie stworzonym dla minikomputerów przez naukowców z Jet Propulsion Laboratory w Pasadenie (Kalifornia).
Polecenia dBase II i jego kolejnych wersji były interpretowane . Taka technika wpływała na szybkość wykonywania operacji. Zmienić to mógł tylko program, który umożliwiałby wykonywanie kompilacji przygotowanego zestawu komend. Takim programem był Clipper firmy Nantucket (obecnie część firmy Computer Associates). Wersja tego programu Summer '87 zdobyła sobie w Polsce olbrzymią popularność pod koniec lat osiemdziesiątych. Niewątpliwą jego zaletą była możliwość wykorzystania kodu @ródłowego, który był używany w programach dBase, a poprzez jego kompilację można było poprawić szybkość działania stworzonej aplikacji.

Kierunki rozwoju Xbase

Rozwój programów w tym okresie przebiegał w dwóch kierunkach: wprowadzania coraz wygodniejszych systemów komunikowania się z użytkownikiem (dBase III Plus Assistant) oraz zwiększaniu szybkości działania programu poprzez kompilację kodu (CA-Clipper). Oczywiście nie były to wszystkie sposoby rozwoju i modyfikacji baz typu Xbase, zmieniały się także możliwości samych baz danych (czas dostępu, metody indeksacji, wielkość zbiorów możliwych do obsłużenia przez program). Programy tego typu zaczęły pojawiać się również w innych środowiskach systemowych (np. FoxPro dla systemu XENIX). Wszystkie te przemiany wiązały się ze wzrostem mocy obliczeniowej komputerów osobistych, a co za tym idzie ich możliwości. Obecnie różnorodność dostawców tego sprzętu oraz popularność systemów otwartych tworzą nowe wymagania jakościowe i wydajnościowe w stosunku do użytkowanego oprogramowania. Wymaganiom tym muszą również sprostać programy typu Xbase.

Dalszy rozwój baz danych typu Xbase powinien zmierzać w trzech kierunkach: graficzne systemy wymiany informacji (GUI), programowanie obiektowe (OOP) oraz wieloplatformowość.

Przyszłość baz typu Xbase

GUI, OOP

wieloplatformowość

?

Programy Xbase (aplikacje)

Obecnie na rynku jest kilka baz typu Xbase (np. FoxPro, dBase, CA-Clipper, CA-dBFast), których rozwój w każdym z tych kierunków przebiega inaczej i uzależniony jest od polityki każdego z producentów. Program rozwoju przyjęty przez firmę Computer Associates, oparty jest o dwa produkty obecnie istniejące na rynku: CA-Clipper i CA-dBFast.

Dzień dzisiejszy: CA-Clipper i CA-dBFast

Rozwój każdego z tych produktów wiążę się z innym kierunkiem wytyczonym dla przyszłych baz typu Xbase. CA-dBFast umożliwia łatwe przejście do aplikacji pracujących w środowisku systemu Windows przy wykorzystaniu programów stworzonych dla systemu DOS. Baza danych wykorzystana w tym programie jest zgodna z bazą programu dBase III, łącznie z indeksowaniem oraz mechanizmami lokowania.
Rozwój programu CA-Clipper związany jest przede wszystkim z programowaniem obiektowymi oraz architekturą klient/serwer. Zmiany jakie są wykonywane w tym programie poczynając od wersji 5.0 czynią z niego coraz silniejsze narzędzie do rozwoju aplikacji. Obecnie dostępna jest już wersja 5.2 tego programu.

CA-Visual Objects: Przyszłość systemów Xbase

CA-Visual Objects jest efektem projektu naukowego Aspen, rozpoczętego przez Nantucket i kontynuowanego oraz rozszerzonego o nowe elementy przez Computer Associates po przejęciu firmy Nantucket. Obecnie przygotowania do wprowadzenia produktu powstałego w tym projekcie podążają dwoma drogami. Jedna jest związana z programem CA-Clipper, druga z programem CA-dBFast. W pierwszej rozwijane są techniki programowania obiektowego oraz architektury klient/serwer i RDD (Replacable Database Drivers), pozwalającej na łączenie z innymi bazami danych. W drugiej rozwijane są techniki graficznej wymiany informacji dostosowane do wymagań systemu Windows.

CA-Visual Objects

CA-dBFast 2.0 CA-Clipper 5.X

Ścieżka GUI Ścieżka OOP

CA-dBFast 1.7 CA-Clipper 5.0

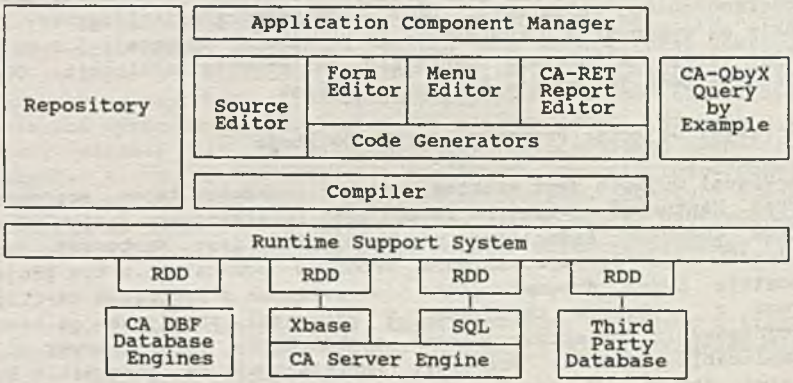
dBase III dBase IV FoxPro CA-Clipper S'87

Architektura

Założeniem dla programu Aspen (rozszerzonego o udział CA) było stworzenie bazy danych o architekturze otwartej, pozwalającej na pracę nie tylko na wielu platformach sprzętowych i systemowych, ale także na współpracę z innymi bazami danych. Aby sprostać temu założeniu przyjęto następujące wymagania w stosunku do tej bazy danych:

- silny język programowania
- programowanie obiektowe
- narzędzia projektowe i słownikowa baza danych
- wydajna baza danych
- język zapytań w standardzie SQL

Powstałą o te założenia i wymagania architekturę można przedstawić następująco



Do dalszego omówienia podzielimy ją na nastepujące kategorie:

- język : kompilator i system runtime
- system bazy danych : zawierający model DBF i klient/serwer
- zintegrowane środowisko tworzenia aplikacji : zawierające bazę słownikową i edytory
- pomoce do zarządzania bazą danych : administracja, przeglądanie i edycja łazy
- narzędzia dla użytkownika końcowego : język zapytań oraz raporty

Język

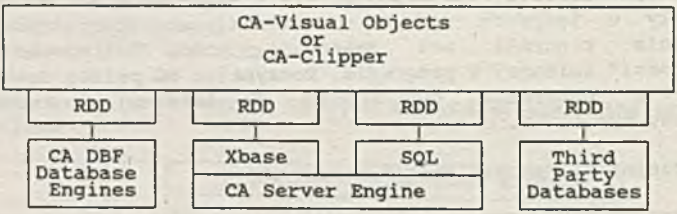
CA-Visual Objects jest pełną implementacją języka zorientowanego obiektowo zawierającego definicje klas i dziedziczenie. Jednocześnie umożliwia zachowanie tradycyjnych technik programowania wykorzystywanych w bazach typu Xbase, a w CA-Clipper w szczególności. Dodatkowo oprócz standardowych zmiennych typu liczbowego, znakowego lub tekstowego (string) można będzie definiować dynamicznie wielowymiarowe tablice zagnieżdżone, a także obiekty i bloki kodu. Jednocześnie CA-Visual Objects zapewni pełną kontrolę typów, głównie na poziomie kompilacji. Dostępne będą również typy, które do tej pory występowały w językach typu C - wska@niki, struktury itp.. W celu umożliwienia kontroli nad zmiennymi można definiować zasięg "widzialności" zmiennej w programie. Poczynając od pełnej dostępności zmiennej w programie, typ PUBLIC, do lokalnej dostępności w procedurze, typ LOCAL.

Tablica definicji "widzialności" zmiennych

Typ	Opis
Public	dostępna globalnie; dynamicznie alokowana oraz nieutypowiona
Private	dostępna w funkcji lub procedurze w której została zdefiniowana oraz w każdej funkcji która jest przez nie wywoływana; dynamicznie alokowana oraz nieutypowiona
Global	dostępna w całej aplikacji; statycznie alokowana oraz silnie utypowiona
Local	dostępna tylko w funkcji lub procedurze w której została zdefiniowana; statycznie alokowana oraz silnie utypowiona

Na wydajność bazy danych ma jednak największy wpływ kompilator. Dotychczasowe kompilatory stosowane w narzędziach typu Xbase pod tym względem pozostawały daleko w tyle za kompilatorami języków typu C. Przyczyną było generowanie przez nie "p-kodu", który następnie był wykonywany przez interpretator runtime, zamiast kodu maszynowego. CA-Visual Objects będzie dostarczał już kodu odpowiedniego dla danej platformy sprzętowej dzięki czemu wyniki wydajności są porównywalne z kodem generowanym przez kompilatory C lub Cobolu.
Kompilator CA-Visual Objects dodatkowo dostarcza wielu ciekawych narzędzi pomocnych przy tworzeniu aplikacji. I tak na przykład, w momencie tworzenia i testowania aplikacji często musimy dokonać niewielkiej zmiany kodu źródłowego, czasami jednej tylko linijki, a następnie kompilować cały program na nowo. Tutaj wystarczy skompilować procedurę, w której dokonano zmiany i ewentualnie procedury, w których ona występowała. Programista będzie miał również do dyspozycji bogaty zestaw bibliotek DLL (Dynamic Link Libraries) popularnych w systemach Windows i OS/2.

Schemat wykorzystania architektury RDD

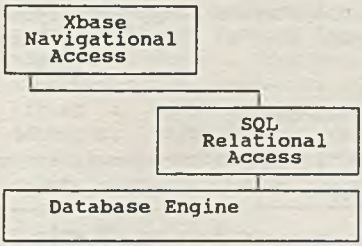


System bazy danych

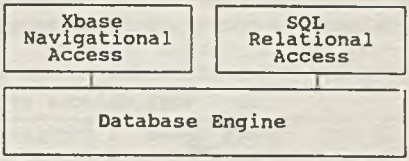
CA-Visual Objects oparty jest na otwartej architekturze bazy danych zawierającej RDD (Repalcable Database Drivers) poprzez które obsługiwane są bazy danych oparte o standardowe modele DBF, klient/serwer oraz bazy dostarczane przez innych producentów.

Do obsługi modelu DBF wykorzystywany jest program pracujący obecnie w CA-Clipper. Zapewnia on pełną obsługę tego typu bazy wraz z odpowiednim mechanizmem lokowania i indeksowania.
Przy wykorzystaniu modelu klient/serwer naturalne staje się

skorzystanie z architektury relacyjnej wraz z językiem SQL. Powstaje jednak problem, jak skorzystać z modelu Xbase przy takiej architekturze. Możliwe są dwa rozwiązania: pierwsze dostęp do bazy z poziomu języka manipulacji danymi (DML) właściwego dla Xbase poprzez SQL i odpowiednią emulację. Drugie, bezpośredni dostęp do bazy z poziomu DML.



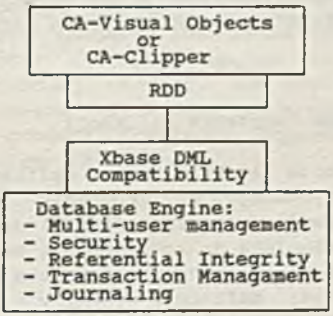
Tłumaczenie operacji Xbase na SQL



Bezpośrednie wykonywanie operacji z Xbase i SQL

Przy tłumaczeniu z języka DML na SQL mogą powstawać błędy jak również duże opó@nienia w reakcji na zapytanie. Dlatego najlepszym rozwiązaniem jest bezpośredni dostęp do bazy z każdej z tych metod. Rozwiązanie to oparte jest na prostej obserwacji: DML i SQL są metodami dialogu z bazą danych, zatem ten sam mechanizm obsługi bazy danych (database engine) może być wykorzystywany przez różne języki dostępu do bazy.

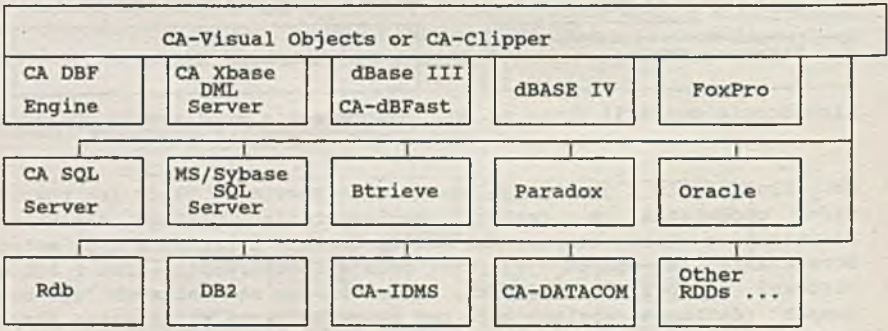
Baza danych o architekturze klient/serwer z językiem dostępu DML



Wybór odpowiedniej techniki DML albo SQL zależy od implementacji. W przypadku korzystania z modelu współdzielenia plików (DBF) należy korzystać z DML. Jeśli korzystamy z modelu klient/serwer należy wybrać technikę SQL.

Dzięki zastosowaniu architektury RDD, CA-Visual Objects może obsługiwać bazy danych tworzene w różnych standardach i przez różnych producentów. Bazy danych pracujące na komputerach PC pojedynczych i pracujących w sieci oraz znajdujących się w dużych systemach komputerowych.

Architektura RDD i możliwość dołączania baz danych



Computer Associates dostarczy szereg programów typu RDD umożliwiających obsługę różnorodnych baz danych, ale jednocześnie ma nadzieję na ich rozszerzenie przez inne firmy jak i przez samych użytkowników.

Zintegrowane środowisko tworzenia aplikacji

Środowisko do tworzenia i rozwijania aplikacji oparte jest na architekturze wykorzystanej w programie CA-dBFast 2.0 oraz narzędziach do tworzenia raportów (CA-RET) oraz zapytań (CA-QbyX). Narzędzia z tego środowiska są zorientowane obiektowo, dzięki czemu praca z nim jest bardzo efektywna. Korzystają one z wspólnej bazy słownikowej zawierającej definicję obiektów i powiązań między nimi. Do tworzenia ekranów umożliwiających wprowadzanie, modyfikowanie i

przeglądanie danych przez użytkownika służy Edytor Form Ekranowych podłączony do generatora kodu, który jest następnie przetwarzany przez kompilator CA-Visual Objects. Podobna jest zasada działania Edytora Menu. Przy definicji menu możemy określić: klawisze szybkiego dostępu (shortcuts), opis informacji wyświetlanej w linii statusu przy przechodzeniu przez użytkownika poprzez menu, akcje jakie mają być podjęte na początku i na końcu selekcji pozycji z menu oraz obsługę w przypadku pojawienia się błędów. Edytor Raportów ma takie same podstawowe funkcje jak poprzednie dwa edytory, pozwala jednak na edycję w postaci WYSIWYG. Dodatkowo, programista znajdzie w tym środowisku tak podstawowe narzędzia pracy jak edytor kodu źródłowego, linker i kompilator dostępne po wciśnięciu jednego klawisza oraz debbuger. Wszystkie wymienione powyżej narzędzia są ze sobą zintegrowane oraz zaprojektowane w sposób umożliwiający minimalizację czysto manualnej pracy programisty.

Pomoce do zarządzania bazą danych

Są one elementem zintegrowanego środowiska tworzenia aplikacji i umożliwiają bezpośrednią ingerencję programisty/administratora w strukturę tworzonego systemu, jak również w zawartość samej bazy danych. Pozwalają na kontrolę dostępu do bazy oraz jej integralności. Moduł ten wykorzystuje doświadczenia i elementy z programu CA-QbyX.

Narzędzia dla użytkownika końcowego

CA-Visual Objects jak również CA-Clipper oraz CA-dBFast są programami przeznaczonymi dla osób tworzących aplikacje, a nie dla końcowych użytkowników. Użytkowników, potrzebujących prostych narzędzi umożliwiających tworzenie zapytań lub raportów, czyli niestandardowego dostępu do bazy danych (nie oferowanego przez stworzoną aplikację). Computer Associates dostarcza takich narzędzi umożliwiających łatwy i szybki dostęp do bazy danych. Są to wspomniany już wcześniej CA-QbyX, ale jako już oddzielny program, oraz generator raportów użytkowany już w programie CA-dBFast. Jest nim również nowy produkt CA-Visual Express, który jest krótko przedstawiony na następnych stronach.

CA-SuperProject
- zarządzanie projektem.

CA-SuperProject jest silnym, elastycznym i łatwym w użyciu systemem do zarządzania projektami (przedsięwzięciami), pozwalającym na kontrolę wykonywania projektu oraz wykorzystaniu środków i ponoszonych kosztach. System może pracować w dwóch trybach Początkujący (Beginner) i Ekspercki (Expert) umożliwiając dobór dostępnych narzędzi odpowiednio do umiejętności użytkownika.

Wszystkie ekrany dostępne w programie CA-SuperProject, graficzne i tekstowe są wpełni interakcyjnie obsługiwane przy użyciu myszy lub klawiatury, umożliwiając w ten sposób proste wprowadzanie danych oraz analizę typu "co jeśli". Użytkownik ma pełną kontrolę nad kolorami, symbolami oraz formą wykresów. SuperProject pozwala stworzyć szkic projektu w sposób zwięzły lub rozszerzony opisując projekt ogólnie lub szczegółowo w zależności od potrzeb. Dzięki różnorodnym kryteriom sortowania i wyboru użytkownik może określić rodzaj informacji jaki chce mmieć wyświetlony na ekranie.

CA-SuperProject jest systemem do zarządzania projektami posiadającym cechę Wielowymiarowego Szkicowania MAO (Multi Angle Outlining). Pozwala ona na spojrzenie na projekt poprzez Zadanie (Task), Środki (Resource), Datę (Date) lub Koszty (Account). Dodatkowo użytkownik sam może sobie zdefiniować powyższe punkty widzenia wraz z nieograniczoną liczbą raportów.

Przeznaczenie środków dla zadań może być przeprowadzone wieloma metodami, dla jednego jak i wielu projektów oraz w całości lub tylko częściowo. Kalendarz projektu i wykorzystania środków uwzględnia święta publiczne i indywidualne dni wolne od pracy. Wykorzystanie i osiągalność oraz koszty środków mogą być zróżnicowane w zależności od czasu poprzez wykorzystanie kalendarza programu CA-SuperProject.

Przydatność programu do zarządzania projektami informatycznymi została potwierdzona w czasie ostatnich wyborów, podczas których CA-SuperProject był narzędziem pozwalającym na kontrolę prac związanych z przygotowaniem i przeprowadzeniem wyborów.

CA-SuperProject jest dostępny w wersji dla systemów DOS, MS-Windows, OS/2 i UNIX.

CA-Visual Express
- okno na bazy danych

CA-Visual Express jest narzędziem stworzonym w oparciu o architekturę klient/serwer pozwalającym na dostęp do różnych baz danych, tworzenie zapytań i raportów. Przezanczony jest dla użytkownika końcowego umożliwiając mu łatwy dostęp do bazy i analizę danych. Wszystkie operacje wykonywane są za pomocą graficznych metod wymiany informacji i nie wymagają wiedzy informatycznej, a jedynie intuicyjnego zrozumienia powiązań pomiędzy danymi. Dzięki wykorzystaniu Open Database Conectivity (ODBC), CA-Visual Express pozwala na dostęp do wielu standardów baz danych.

Podstawowymi elementami produktu są:

ODBC
umożliwia dostęp poprzez programy pośredniczące (drivers) dostęp do różnorodnych baz danych wykorzystywanych w firmie

ODBC database drivers
ponad dwadzieścia różnych programów pośredniczących, umożliwiających natychmiastowy dostęp do bazy danych. Wśród nich do baz:
CA-DATACOM, CA-IDMS, Btrive, DB2, dBASE, Excel, Informix, Ingres, NetWare SQL, Oracle, Paradox, Progress i inne.

Języki zapytań
w zależności od potrzeb i wiedzy użytkownika dostępne są trzy rodzaje języków zapytań o różnym stopniu złożoności
- Quick Query
- Query By Example
- Query By Forms

Lokalna baza danych
Baza pozwalająca przechowywać informacje uzyskane z różnych @ródeł.

Raportowanie
Wbudowane silne narzędzie pozwalające na tworzenie w pełni profesjonalnych raportów.

DDE
Narzędzie pozwalające na przekazywanie danych, może pracować jako klient i jako serwer.

Import/Eksport danych z plikami o formacie dBase, Excel, CSV
Dostęp do danych o formatach nie SQLowych.

Debbuger
Ułatwia znajdowanie błędów w stworzonych aplikacjach.

Administracja bazy danych
Pozwala na tworzenie i pielęgnację struktury bazy danych.

Materiał opracował: Marek Uszyński
Wszystkie nazwy programów występujące w artykule służą identyfikacji produktu i są zastrzeżone przez ich producentów.

Tezy do dyskusji Programista roku 2000

Rok 2000 w publikacjach, wypowiedziach, dyskusjach, symbolizuje najczęściej przyszłość. Przyszłość, o której wypowiadamy się w kategoriach wizji a nie konkretnego planu działania. Kiedy spojrzymy dokładniej w kalendarz zauważymy, że jest to przyszłość odległa tylko o kilka lat. Absolwent Uczelni w roku 2000, to dzisiejszy kandydat na studia, czyli dzisiaj kształtujemy tą przyszłość. Rok 2000 w tytule chcę potraktować zarówno w kategoriach symbolu przyszłości, jak i planu konkretnych działań ze ściśle określonym harmonogramem czasowym.

Spróbujmy określić wizerunek programisty roku 2000 oraz niezbędne działania zmierzające do realizacji praktycznej tego wizerunku.

Panuje dość powszechna opinia, że nasze Uczelnie opuszczają absolwenci nienajlepiej przygotowani do wykonywania zawodu informatyka. Narzekania na poziom absolwentów płyną z zakładów pracy, średnich i małych firm komputerowych. Z drugiej strony absolwenci naszych Uczelni zupełnie nieźle radzą sobie za granicą w różnorodnych działalnościach informatycznych. Absolwenci Uczelni tworzą zupełnie nieźle funkcjonujące firmy softwerowe w kraju i zaczynają narzekać na absolwentów.

Jaka jest prawda ? Czy jest to znany problem, że dzieci są niegrzeczne tak długo dopóki nie wyrosną i nie mają własnych dzieci, które oczywiście są gorsze i bardzo niegrzeczne ?

Mając świadomość, że żadna dyskusja nie jest w stanie rozwiązać problemu, sądzimy że warto wyspecyfikować żale płynące z tak zwanego przemysłu oraz wysłuchać głosu Uczelni. Pragniemy korzystając ze zgromadzenia dość reprezentatywnej próbki środowiska informatycznego w ramach Wiosennej Szkoły PTI, przeprowadzić dyskusję na temat wizerunku programisty roku 2000. Podkreślam programisty a nie informatyka, gdyż wydaje się, że ta grupa informatyczna przechodzi największe zmiany związane z nowinkami technologicznymi i nowymi obszarami zastosowań informatyki.

Z własnych doświadczeń programistycznych znam diagnozy głoszone blisko 20 lat temu, że "...za kilka lat zawód programisty przestanie istnieć w związku z rozwojem ...". Rzeczywistość okazała się inna. Dzisiaj można usłyszeć podobne opinie, czy są one równie nietrafne? Sylwetka, środowisko pracy programisty lat 70-tych są nieporównywalne z programistą lat 90-tych, ale czy aż tak bardzo ?

Chcielibyśmy w trakcie dyskusji odpowiedzieć na pytania:

- czy zawód programisty będzie istniał w roku 2000 ?
- jakie będzie środowisko pracy programisty w roku 2000 ?
- czy umiejętność algorytmicznego myślenia będzie konieczna ?
- jakie kwalifikacje powinien mieć absolwent Uczelni ?
- czy będziemy programować w językach proceduralnych ?
- jakie narzędzia i metody programowania będą stosowane ?

- czy absolwenci Uczelni będą natychmiast kierowani na kursy ?
- jakie systemy będziemy programować ?
- czy programiści w Cobolu będą bezużyteczni ?
- jak będą powstawały nowe zastosowania informatyki ?
- czy powstaną w Polsce silne domy wytwarzania oprogramowania ?
- jakie będzie zapotrzebowanie na programistów ?
- czy programista będzie tylko przygotowywał parametry ?
- jakie języki programowania będą dominować ?
- czy sensowne będzie optymalizowanie rozwiązań ?
- jaki będzie status programisty ?
- czy ... ?
- jak ... ?

oraz postawić nowe pytania, które będą dyskutowane w środowisku informatycznym nie tylko PTI.

Sytuacja kadrowa i finansowa Uczelni jest katastrofalna i na ten temat nie chciałbym rozwijać dyskusji, bo to inny ważny temat. Powstają jednak liczne szkoły prywatne, spółki zajmujące się szkoleniem informatycznym. Jak widzimy ich rolę w przygotowaniu programisty roku 2000 ?

Wydaje się, że środowisko jest w stanie samo określić rolę i niezbędne działania instytucji mających wpływ na wizerunek programisty za kilka lat. PTI wraz z innymi Stowarzyszeniami powinno wypracować model niezbędnych działań i regulacji, tak aby rok 2000 nie zaskoczył nas. Dotychczasowe działania i incjatywy środowiska pozwalają mieć nadzieję, że informatycy potrafią "brać sprawy w swoje ręce".

Zdzisław Szyjewski

Mariusz KLAPPER
ul.Sosnowskiego 6/7
31-139 KRAKÓW

UWAGI DO DYSKUSJI "PROGRAMISTA ROKU 2000"

MOTTO.

Czy zauważyliśmy, odurzeni pozycją, jaką zajmujemy w kosmosie, jaką sami sobie przyznaliśmy, że już przed ponad stu laty narodzili się dwaj bożkowie, a może tylko demony - Interes i Wydajność. Za jakiś czas zechcą zająć miejsce opustoszałe po prawdziwych, wielkich bogach archaicznych. Jaka wówczas będzie pozycja człowieka ?

(cytat) 1)

Odpowiadając na zaproszenie Organizatorów Wiosennej Szkoły PTI przesyłam kilka uwag do dyskusji "Programista roku 2000". Uwagi te piszę z perspektywy własnych doświadczeń zawodowych, oraz ze skromnej perspektywy małej firmy produkującej oprogramowanie użytkowe dla potrzeb zarządzania przedsiębiorstwami. Uwagi te są bardziej garścią luźnych, osobistych refleksji i poglądów, niż całościową koncepcją.

1. Przyszła rola programisty, w tradycyjnym rozumieniu tego pojęcia, będzie uzależniona od tendencji, które zdominują zastosowania informatyki. Jeśli zwycięży tendencja do używania dużych i uniwersalnych systemów oprogramowania narzędziowego, to rola programisty zostanie *de facto* ograniczona do zadań superoperatora tych systemów. Jeśli utrzyma się tendencja opracowywania systemów dedykowanych konkretnemu użytkownikowi, to rola programisty pozostanie nadal kluczowa.

   Obecnie na rynku oprogramowania można obserwować pęd do tworzenia coraz potężniejszych, większych i coraz bardziej uniwersalnych systemów i standardów oprogramowania narzędziowego i aplikacyjnego. Stopniowo narasta też podstawowa, moim zdaniem, bariera zastosowań tych systemów, - możliwości percepcyjne użytkowników. Coraz częściej można zauważyć, że największym problemem użytkownika staje się konieczność poznania i opanowania wszystkich możliwości systemu, które mają mu ułatwiać życie. Zazwyczaj użytkownik rezygnuje wtedy z "przebijania się" przez tomy dokumentacji systemowej i ogranicza się do wykorzystywania znikomej części prostych, lub znanych mu już możliwości systemu, albo wynajmuje informatyka, który wykonuje za niego "czarną robotę". W konsekwencji powstają liczne bariery pomiędzy rzeczywistymi potrzebami i oczekiwaniami użytkownika, a tym, co otrzymuje on od programisty i od systemu. Faktyczna rola programisty zostaje przy tym ograniczona do pośrednictwa pomiędzy oczekiwaniami użytkownika, a możliwościami systemu narzędziowego i umiejętnością ich wykorzystania.

1) Krzysztof KOWALSKI "EROS I KOSTUCHA" L.S.W. W-wa 1990 r.

Inaczej jest, gdy użytkownik otrzymuje oprogramowanie wykonywane specjalnie dla niego. System jest wtedy zazwyczaj opracowywany pod dyktando użytkownika, często jest realizowany i modernizowany przy jego czynnym współudziale. Rola programisty jest kluczowa, gdyż jego wiedza i umiejętności decydują o walorach użytkowych systemu. Wariant taki, wbrew pozorom, nie jest na ogół mniej efektywny ekonomicznie, często natomiast daje lepsze rezultaty praktyczne.

Można przypuścić, że, jak zwykle w życiu, przyszłość utrwali sytuację pośrednią: duże, bogate i ambitne firmy będą wykorzystywać duże pakiety oprogramowania standardowego. Firmy małe pozostaną zapewne przy systemach mniejszych, ale tańszych i lepiej pasujących do ich specyfiki. Programiści będą zatem mogli wybierać pomiędzy rolą trybiku w wielkiej machinie, lub rolą współtwórcy opracowań mniejszych, ale bardziej naznaczonych ich indywidualnym piętnem.

2. W zakresie modelu kształcenia informatyków należy oczekiwać szybkich i znaczących przewartościowań. Ciągły, lawinowy rozwój technologii i zastosowań pociąga za sobą opanowywanie przez informatykę wszystkich dziedzin życia. Już obecnie ludzie znający i rozumiejący możliwości i zastosowania technologii komputerowych są jedynymi ludźmi naprawdę rozumiejącymi wiele problemów współczesności. Informatyka rewolucjonizuje wiele nowych dziedzin życia i działalności człowieka (muzyka, film, plastyka). Umiejętność wykorzystywania technik informatycznych staje się podstawowym wymogiem kwalifikacyjnym współczesnego człowieka. Dlatego informatyka będzie zapewne stopniowo zmieniać swoje miejsce w kształceniu: w coraz większym stopniu będzie się stawać przedmiotem podstawowym (tak, jak np. matematyka), a nie, jak dotychczas, przedmiotem specjalistycznym. Sytuacja ta postawi całkowicie nowe wyzwania wobec programów kształcenia informatyków. Głównym problemem stanie się określenie podstawowego zakresu wiedzy i umiejętności informatycznych potrzebnych specjalistom różnych dziedzin. Nauczyciele informatyki będą zaś musieli znacznie poszerzyć swoją wiedzę interdyscyplinarną, aby mogli nawiązać kontakt i zrozumieć specyficzne problemy tych dziedzin wiedzy, które mają wspierać zastosowaniami informatyki. Problem wiedzy interdyscyplinarnej nauczycieli informatyki jest zresztą aktualny również dzisiaj. Liczne narzekania na absolwentów informatyki mają często swoje pierwotne źródło w zbyt skromnym doświadczeniu praktycznym nauczycieli, związanym ze specyfiką problemów, które będą musieli rozwiązywać w życiu ich uczniowie.

3. Można wątpić, czy w Polsce powstaną naprawdę duże firmy zajmujące się produkcją oprogramowania. Jest to bowiem działalność wymagająca sporych nakładów kapitałowych, o stosunkowo długim okresie zwrotu i relatywnie dużym ryzyku inwestycyjnym. Wobec bardzo silnej konkurencji dużych, bogatych i doświadczonych firm obcych działających na naszym rynku, szanse na przetrwanie rodzimej firmy angażującej się w produkcję własnego oprogramowania wydają się niewielkie. Można się też zastanawiać, czy w ogóle jest sens angażowania dużych nakładów na wytworzenie w kraju rzeczy, które kto inny już wytworzył wcześniej, szybciej, taniej i zazwyczaj lepiej.

4. Narzędzia i technologie programowania stosowane w przyszłości będą w decydującej mierze zdeterminowane wdrożeniami. Dlatego zarówno no. komputery IBM PC, jak język COBOL, będą nadal wykorzystywane, gdyż skala ich praktycznego stosowania przesądzi o ich przydatności, niezależnie od ich obiektywnych walorów.

5. Zastanawiając się nad bliską i dalszą przyszłością często nie można się oprzeć od poważnych refleksji natury ogólniejszej. Błyskawiczny rozwój zaawansowanych technologii w sposób tragiczne zmienia oblicze cywilizacji. Człowiek coraz bardziej traci kontakt ze źródłami, z których się wywodzi, ze swoją biologiczną tożsamością. W pogoni za rozwojem, ułatwianiem sobie życia, ludzkość gubi coraz bardziej podstawowe więzi, które ją uformowały. Informatyka odgrywa w tych procesach bardzo poważną rolę przyspieszającą. Równocześnie pogłębia się przepaść cywilizacyjna pomiędzy regionami rozwiniętymi i zacofanymi. Ostra konkurencja ekonomiczna niszczy podstawowe więzi społeczne. Narastają konflikty, których pierwotnym podłożem jest sprzeczność pomiędzy podmiotowością rozwiniętej cywilizacji, a biologicznymi uwarunkowaniami człowieka i przyrody. Dlatego pytania: gdzie właściwie zmierzamy ?, co znajdziemy tam, dokąd tak uparcie dążymy ?, czy będziemy jeszcze naprawdę ludźmi, kiedy tam dotrzemy ? są obecnie znacznie ważniejsze, niż się na ogół wydaje. Przewrót jakiekolwiek rozważania o przyszłości nie sposób tych pytań ominąć.

Marian Klapper

Kraków, 20 kwietnia 1994 r.