Wiosenna Szkoła PTI
Świnoujście'95

Narzędzia Programowania Obiektowego

Organizowana przez
POLSKIE TOWARZYSTWO INFORMATYCZNE

Świnoujście 8-12 maja 1995

Wiosenna Szkoła PTI

Świnoujście'95

Narzędzia Programowania Obiektowego

Organizowana przez

POLSKIE TOWARZYSTWO INFORMATYCZNE

Świnoujście 8-12 maja 1995

SPIS TREŚCI

Programowanie obiektowe - nowa jakość programowania

Gry komputerowe oprócz funkcji relaksujących stanowią doskonały sposób promocji informatyki, szczególnie wśród młodzieży, oraz wyzwalają u programistów nowe podejście do wytwarzania oprogramowania. Zastosowania animacji komputerowej biorą swój początek w grach komputerowych. Bez specjalnych badań można zaryzykować stwierdzenie, że wielu informatyków, szczególnie młodego pokolenia, pierwsze kroki w świat komputerów robiło poprzez gry komputerowe. Prawdą jest również, że gry komputerowe stanowią dla wielu, jedyne zastosowanie komputera. We wszystkim trzeba mieć umiar a wówczas gry komputerowe mogą spełniać inspirującą rolę w rozwoju metod wytwarzania systemów informatycznych.

Jedną z najwcześniej stosowanych zabaw jest prosta zabawa polegająca na układaniu klocków. Klocki różnej wielkości układa się według zadanego wzoru lub wykorzystując własną inwencję konstruuje się różnorodne budowle. Marzeniem układających są klocki typu lego, które pozwalają na maksymalne rozwinięcie inwencji twórców i pozwalają budować wprost niewyobrażalne konstrukcje. Atrybutem tego typu klocków jest standaryzacja budowy i możliwości łączenia w wynikające z potrzeb konstruktora konfiguracje. Nieograniczona inwencja jest dodatkowo wspomagana prostotą pojedynczych elementów i ich wielofunkcyjnością w zależności od konkretnych potrzeb konstruktora.

Zabawa w klocki w wydaniu informatycznym na potrzeby wytwarzania oprogramowania to programowanie obiektowe.

Każdy aktywny programista obok standardowych narzędzi programowania i bibliotek, posiada i stosuje własne moduły programowe, które wspomagają go w procesie wytwarzania oprogramowania niezależnie od tego czego aktualnie dotyczy wytwarzany program. Własne standardy, wytwarzane są w celu ułatwienia sobie programowania i przyspieszenia procesu programowania. Własne uniwersalne moduły programowe są różnej szczegółowości i najczęściej uwzględniają oryginalny styl programowania, przyzwyczajenia i potrzeby konkretnego programisty.

Szybki rozwój technologii informatycznych powoduje, że wytworzone biblioteki prywatne bardzo często muszą być modyfikowane pod nowe potrzeby zmieniającego się środowiska sprzętowego i programowego. Stosowany styl pracy programistów zmusza ich do częstych modyfikacji środowiska dorobku własnego lub wypracowywania nowego zestawu stosowanych własnych narzędzi wytwarzania oprogramowania. Zastosowanie standardów w programowaniu i wspomaganie ich uniwersalnymi modułami programowymi, niezależnymi od środowiska, wychodzi naprzeciw oczekiwaniu programistów.

Programowanie obiektowe to naturalny rozwój technologii procedur programowych, modułów programowych i innych metod wytwarzania oprogramowania, które swe źródła biorą w zasadzie "dziel i rządź". Podział większego problemu na mniejsze części pozwala na łatwiejsze zrozumienie i podział pracy wśród członków zespołu realizatorów. Części, na które podzielony zostanie problem, powinny być tak skonstruowane aby łatwo można było je łączyć w większe obiekty zgodnie z potrzebami rozwiązywanego problemu.

Posiadając odpowiednio dużą kolekcję klocków, obiektów programowych, które bardzo łatwo dają się łączyć, programista może skoncentrować się na problemach konstrukcyjnych rozwiązywanego problemu. Szczegółowe rozwiązania programowe zaszyte są w obiektach programowych i podobnie jak w przypadku konstruowania z wykorzystaniem klocków lego, konstruktor nie rozprasza się na szczegóły tylko buduje większą konstrukcje wykorzystując cechy mniejszych obiektów i możliwości ich łączenia dla celów tworzonej budowli, systemu informatycznego.

Taki styl wytwarzania oprogramowania wymaga nowego podejścia i innych predyspozycji indywidualnych poszczególnych członków zespołu realizatorów. Przeniesienie akcentów z problemów szczegółowych na problemy konstrukcyjne pozwala nie tylko szybciej wytwarzać ale również umożliwia rozwiązywać większe i bardziej skomplikowane problemy. Dotychczasowe metody wytwarzania powodowały, że informatycy ginęli w licznych, istotnych szczegółach rozwiązania, co uniemożliwiało zakończenie prac w wymaganych terminie.

Liczne i bardzo złożone są przyczyny nieudanych zastosowań informatyki ale eliminacja każdej z nich daje większe gwarancje sukcesu. Programowanie obiektowe przyspiesza prace programowe z równoczesnym podniesieniem niezawodności oprogramowania. Przeniesienie akcentów w procesie

wytwarzania na problemy konstrukcyjne pozwala na włączenie przyszłego użytkownika w proces wytwarzania. Wspólne konstruowanie kolejnych prototypów pozwala na bieżące modyfikowanie wymagań użytkownika i lepsze dopasowanie ostatecznej wersji systemu informatycznego do jego wymagań.

Podejście obiektowe do wytwarzania oprogramowania systemu przenosi akcenty na fazę projektową, co nie tyle upraszcza proces tworzenia systemu, ale wymaga nieco innych predyspozycji od autorów. Prace programistyczne wykonywane w środowisku proceduralnych języków programowania wymagały innych predyspozycji niż konstrukcja systemu oprogramowania z obiektów programowych układanych w system oprogramowania, podobnie jak układa się konstrukcję z klocków. Umiejętność oderwania się od szczegółowych rozwiązań algorytmicznych w połączeniu z inwencją konstruktorską pozwala na osiągnięcie nowej jakości wytwarzanych systemów informatycznych.

Programowanie obiektowe aby mogło być efektywnie stosowane musi dostarczać programistom odpowiednio dużą kolekcję gotowych obiektów programowych, proste mechanizmy łączenia obiektów oraz narzędzia wytwarzania własnych obiektów stanowiących niezbędne uzupełnienie konstrukcyjne. Programowanie obiektowe to ogólna metoda podejścia do procesu tworzenia oprogramowania, o jej sukcesie i zastosowalności decyduje otoczenie wspomagające. Narzędzia wspomagające programowanie obiektowe są przedmiotem wykładów tej edycji Wiosennej Szkoły PTI.

Życzę Państwu aby każdy programista mógł wzbogacić swój własny warsztat pracy o nowe narzędzia, które przybliżą go do obiektowego wytwarzania programów.

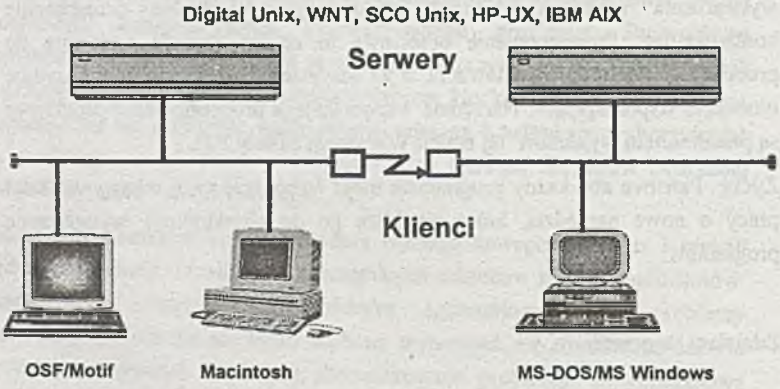
Zdzisław Szyjewski

Linkworks - obiektowo zorientowany system integracyjny

Sławomir Błaszczak
Digital Equipment Polska

1. Wprowadzenie

LinkWorks jest konfigurowalnym środowiskiem integracyjnym aplikacji oraz ludzi działających w zespołach (biurach, instytucjach, itp.). System ten zawiera wszystkie elementy stanowiące bazę do tworzenia specjalizowanych systemów, zapewniając możliwość dzielenia się dokumentami, kontroli w dostępie do nich, zarządzanie wersjami, pocztę elektroniczną, podpisywanie elektroniczne, oraz automatyzacje obiegu dokumentów. Jako system z otwartą architekturą typu klient-serwer, LinkWorks może pracować w środowisku składającym się z wielu różnego rodzaju serwerów oraz stacji roboczych, wykorzystując do komunikacji różne transporty sieciowe. Linkworks jest dostępny w kilkudziesięciu wersjach językowych (w tym również polskiej) . Własnością unikalną jest możliwość używania w jednej instalacji równocześnie różnych języków np. polskiego i angielskiego.



Rys.1. Architektura systemu LinkWorks

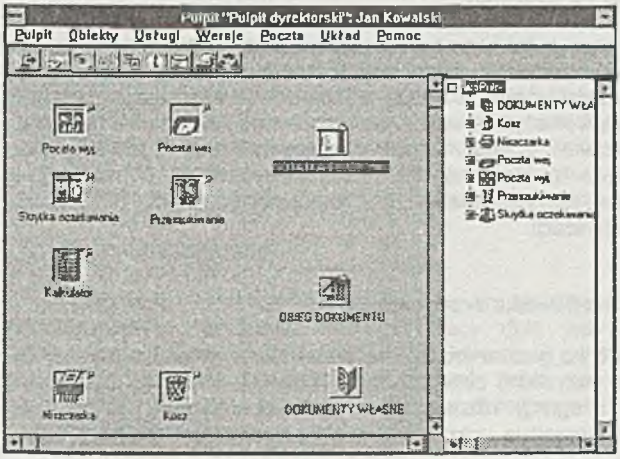
Opis systemu LinkWorks

LinkWorks oferuje użytkownikom "wirtualne biurko" - graficzne środowisko okienkowe, odzwierciedlajace rzeczywiste środowisko biurowe z

dokumentami, szafami na dokumenty oraz narzędziami biurowymi (kalkulator, kosz na śmieci, niszczarka).

W oknach umieszczone są ikony reprezentujące aplikacje, dokumenty lub zbiory dokumentów. Na przykład znajduje się tu ikona zegara, skrzynki pocztowej, teczki dokumentów czy szafki na teczki. Postać dokumentu może być różna: może to być tekst, rysunek, obraz czy nawet zapis dźwięku, lub dowolne złożenie tych elementów.

Wybranie ikony, poprzez wskazanie go kursorem i naciśnięcie klawisza myszy, powoduje uruchomienie aplikacji, która jest z nim skojarzona. Dla dokumentów oznacza to zaprezentowanie go przez odpowiedni program edytora np. MS Word for Windows. Wybranie zbioru dokumentów (np. teczki) powoduje jego otwarcie tzn. pokazywane są w oknie zawarte w nich dokumenty. Wszystkie obiekty pojawiające się w oknie mogą być w prosty sposób usunięte, trwale lub odtwarzalnie, poprzez przesunięcie ich ikony za pomocą myszy na ikonę niszczarki lub kosza na śmieci.



Rys.2. Widok pulpitu LinkWorks

Integralną częścią systemu jest podsystem poczty elektronicznej. Wysłanie dokumentu, podobnie jak jego likwidacja, polega na przesunięciu jego ikony na wyjściową skrzynkę pocztową i następnie określeniu adresu odbiorcy. W ten sposób mogą być przesyłane nie tylko pojedyńcze dokumenty, ale również całe ich zbiory. Można również przesyłać pocztę do innych systemów (nie LinkWorks) wykorzystując protokoły pocztowe takie jak X.400 lub SMTP.

Inną cechą związaną z pocztą, jest możliwość określania drogi przepływu dokumentów tzn. zdefiniowania listy osób lub organizacji, które mają dany dokument otrzymać wraz z podaniem kolejności, w jakiej ma się to odbywać.

Po drodze dokument może być modyfikowany lub uzupełniany oraz można go "podpisać", dołączając do niego odpowiedni, niewymazywalny znacznik. W ten sposób można określić globany przepływ dokumentów w ramach danej organizacji.

Wszelkimi pracami związanymi z konfigurowaniem systemu, definiowaniem użytkowników, tworzeniem nowych obiektów zajmuje się administrator systemu. Użytkownicy mogą działać jedynie na dokumentach własnych, do nich przysłanych lub określonych jako wspólne dla grupy osób lub całej organizacji. Mechanizmy protekcji oparte między innymi o hasła, uniemożliwiają dostęp do danych poufnych.

Każdy użytkownik może organizowac swoje "biurko" na ekranie w sposób najbardziej dla niego wygodny, dokładając nowe zbiory dokumentów lub nowe narzędzia potrzebne w jego pracy (np. kalkulator, szkicownik itp.). Wszystkie zbiory danych są przechowywane na maszynie serwera i jedynie na czas przetwarzania są przenoszone na maszynę klienta, co zapewnia, że dostęp do nich jest kontrolowany.

LinkWorks jest systemem bardzo elastycznym. Daje się on łatwo dostosować do wymagań danej organizacji. Można definiować dowolne klasy obiektów lub narzędzi wiążąc z każdą z nich dowolne aplikacje zarówno dostarczane z LinkWorks, jak również dostępne w ramach systemu (edytory, kalkulatory tablicowe,...) lub własne (programy finansowo-księgowe, kadrowe,...). Można również dowolnie modyfikować opisy, komunikaty i nazwy pojawiające się na ekranie np. dostosowując je do terminologi używanej w tej organizacji.

LinkWorks jako środowisko uruchomieniowe

LinkWorks to nie tylko gotowe środowisko pracy biurowej, jest to również (a właściwie przede wszystkim) zintegrowany system implementacji rozwiązań biurowych oraz integracji różnego typu aplikacji. W systemie istnieją mechanizmy oraz narzędzia umożliwiające implementatorom realizację tego typu prac.

LinkWorks jest środowiskiem w pełni obiektowo-zorientowanym i to zarówno od strony uzytkownika jak i implementatora. Informacje na pulpicie użytkownika są przedstawiane jako obiekty, reprezentujące takie rzeczy jak listy, rozliczenia wyjazdowe, szafki czy teczki. Obiekty te są definiowane w systemie z wykorzystaniem takich pojęć jak klasa obiektu z powiazanymi z nią metodami oraz atrybutami.

LinkWorks oferuje cztery techniki implementacji rozwiązań. Każda technika wymaga odpowiednich narzędzi, używa specyficznych mechanizmów systemu oraz jest stosowana przy specyficznych wymaganiach rozwiązania. W większości przypadków w rozwiązaniu wykorzystuje się kombinację niektórych lub wszystkich tych technik.
Techniki te to:

- Applikacje Plus Obiekty (APO)
- Język programowania klas obiektów
- Konfiguracja LinkWorks
- Język skryptów LinkWorks

Aplikacje Plus Obiekty (APO)

Dzięki technice APO aplikacje zewnętrzne mogą tworzyć oraz wykorzystywać obiekty systemu Linkworks. Pozwala to na ścisła integrację aplikacji indywidulanych i grupowych z systemem LinkWorks, dodając nową funkcjonalność do tworzonego systemu.

Twórcy oprogramowania, wykorzystując technikę APO, mogą tak konstruować swoje aplikacje, aby mogły one współpracować z LinkWorks (o ile jest on dostępny w czasie pracy aplikacji) w celu pobierania i zwracania przetwarzanych informacji.

Język programowania klas

Standardowa funkcjonalność LinkWorks może byc zmieniana i rozszerzana poprzez definiowanie nowych klas obiektów (jako podklasy klas istniejących) oraz zmianę lub rozszerzenie ich charakterystyk (atrybutów lub metod). Dokonuje się tego z wykorzystaniem Języka Programowania Klas oraz odpowiednich mechanizów (kompilator, debugger) dostępnych w systemie.

Konfiguracja LinkWorks

Wygląd okien LinkWorks może być w pełni dostosowany do wymagań użytkownika w zależności od jego pracy oraz zakresu obowiązków. Wykorzystując graficzne narzędzia wbudowane w system, administrator LinkWorks może zmienić lub zdefiniować nowe: menu, ikony, teksty komunikatów, używane aplikacje oraz prawa dostępu do obiektów własnych i innych osób.

Język skryptów

Język skryptów dostępny w systemie LinkWorks służy do automatyzacji wykonania powtarzajacych się czynności oraz do tworzenia niedużych aplikacji realizowanych w ramach tego systemu. Skrypty mogą być używane przez administratora oraz bardziej doświadczonych użytkowników.

Dystrybucja rozwiązania

Poza technikami implementacji i integracji rozwiązania, LinkWorks oferuje również unikalną koncepcję jego dystrybucji i instalacji. Jest to robione przy pomocy Komponentów Programowych (Software Components).

Wszystkie modyfikacje i rozszerzenia danego rozwiazania wykonane przy pomocy języka programowania klas oraz narzędzi konfiguracji, są automatycznie kojarzone z jednym, wybranym Komponentem Programowym. Komponent Programowy jest obiektem, który może być przeniesiony (wyeksportowany) na dysk lub dyskietkę stacji roboczej i następnie dostarczony użytkownikowi końcowemu samodzielnie lub razem z innymi aplikacjami (integrowanymi z LinkWorks). Użytkownik ponownie przenosi Komponent Programowy do swojego środowiska LinkWorks i następnie przy pomocy metody "przemieść i upuść" uruchamia odpowiednie narzędzie, które samoczynnie wprowadza modyfikacje do jego systemu.

2. Obiektowo-zorientowany LinkWorks

Wprowadzenie

LinkWorks jest całkowicie złożony z obiektów należących do odpowiednich klas. Klasa obiektu definiuje: jak obiekt jest widziany przez użytkownika, jego funkcjonalność oraz jak obiekt przechowuje informacje. Sam obiekt zawiera dane w sposób określony w definicji klasy. Użytkownicy widzą obiekty systemu LinkWorks, takie jak koperty, teczki, rysunki, jako ikony umieszczone w oknach na ekranie stacji roboczej.

Klasy obiektów

W LinkWorks każdy obiekt należy do pewnej klasy. W ramach klasy, sposób prezentacji obiektów (ikony, menu, przyciski, układ okna) jest określany poprzez *'własności'*, sposób przechowywania danych (struktura danych) poprzez *'atrybuty'*, natomiast funkcjonalność definiuje się przy pomocy *'metod'*. Zawartość obiektu jest to podzbór danych przechowywanych w atrybutach, może być ona prezentowana użytkownikowi podczas realizacji operacji edycji lub odczytu obiektu.

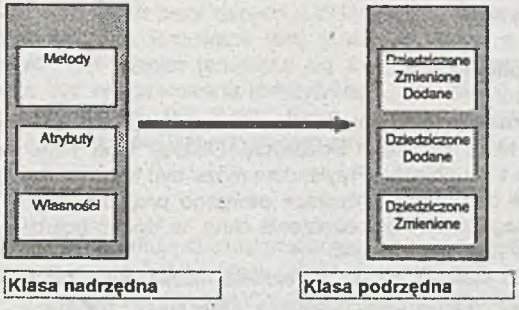
Dziedziczenie

Nowe klasy obiektów, dodawane do systemu są określane jako podklasy wybranej, istniejącej klasy. W takim przypadku nowa klasa dziedziczy wszystkie: własności, atrybuty i metody klasy nadrzędnej. Następnie programista może dodać nową lub zmienić starą metodę, dodać lub usunąć atrybuty oraz zdefiniować inne własności, dostosowując je do wymagań rozwiązania.

Taka nowa klasa może się stać klasą nadrzędną dla innych klas, dzięki czemu istnieje możliwość wykorzystania już istniejącego i sprawdzonego kodu w nowych aplikacjach.

W LinkWorks zrealizowany jest mechanizm *dynamicznych powiązań* (dynamic binding) klas obiektów, przy pomocy którego wszelkie nowe zmiany

w klasie nadrzędnej automatycznie propagują do wszystkich klas podrzędnych (nawet pośrednio) w stosunku do tej klasy. Np. zmiana metody czytania obiektu klasy Notatka powoduje zmianę tej metody w klasach podległych (np. Notatka Służbowa). Programista ma możliwość zablokowania tego mechanizmu dla niektórych elementów definicji klasy, np., gdy stworzył własną metodę czytania Notatki Służbowej i nie chce, aby została ona zamazana.



Rys. 3. Dziedziczenie metod, atrybutów i własności

Klasy systemowe i klasy użytkownika

Klasy dostarczane w pakiecie LinkWorks (inicjalnie) nazywane są klasami systemowymi. Klasy systemowe nie mogą być zmieniane, można jedynie dodawać do nich nowe metody oraz nowe atrybuty. Inne klasy obiektów, wywodzące się z klas systemowych, są nazywane klasami użytkownika i mogą być swobodnie modyfikowane. Takie podejście zapewnia twórców Komponentów Programowych, że , jeśli opierali się jedynie na klasach systemowych, to w każdych warunkach będzie można zainstalować i użyć ich oprogramowanie.

Hierarchia klas

Mechanizm dziedziczenia sprawia, że dla wszystkich klas w systemie można stworzyć drzewiastą strukturę hierarchii klas, w której węzeł reprezentujący pewną klasę jest powiązany z co najwyżej jednym węzłem klasy nadrzędnej i może być połączony z jednym lub więcej węzłami klas podrzędnych. Korzeniem takiej struktury jest klasa *Obiekt* definiująca ogólną funkcjonalność wszystkich obiektów w systemie (np. definiuje atrybuty takie jak *nazwa*, *właściciel* oraz metody takie jak *czytaj, edytuj, zlikwiduj* obiekt).

Rys. 4. Hierarchia klas

Klasy obiektów złożonych

Obok klas obiektów prostych takich jak *Notatka, Tekst, Rysunek, Kalkulacja*, w systemie istnieją klasy obiektów złożonych. Obiekty tych klas są to pojemniki zawierające inne obiekty. Przykładem może być teczka lub szafka zawierająca wszystkie dokumenty dotyczące pewnego projektu. Edycja lub odczyt obiektu złożonego powoduje otworzenie okna, w którym pokazywane są wszystkie obiekty w nim umieszczone.
Na bazie systemowych klas złożonych, użytkownik może definiować własne klasy np. klasa *Teczka* może stanowić podstawę nowej klasy *Projekty*.

Zagnieżdżenia

LinkWorks pozwala określić, które obiekty mogą być umieszczane wewnątrz danej klasy złożonej. Jest to jedna z własności klasy złożonej. Zagnieżdżanie umożliwia wprowadzenie ograniczeń porządkujących, np. w klasie *Projekty* mogą być tylko *Rysunki techniczne* i *Opisy tekstowe* a nie może być *Teczki*.

Atrybuty

Atrybuty służą do przchowywania danych należacych do pewnego obiektu. Atrybuty są przyporządkowywane do klasy obiektu. Mogą one być użyte do opisywania obiektów tej klasy (np. nazwa obiektu, właściciel, temat) lub być jedynie "pojemnikiem" do przechowywania informacji, bezposrednio nieudostepnianej użytkownikowi.
W każdym przypadku atrybuty mogą być mechanizmem wymiany informacji pomiędzy LinkWorks i aplikacjami zewnętrznymi.

Atrybuty systemowe

Pewien zestaw atrybutów, nazywanych atrybutami systemowymi, jest przypisany do każdej klasy systemu LinkWorks. Przykładami takich atrybutów są: *nazwa, właściciel, temat, data utworzenia* i inne. Poza *nazwą* oraz *tematem*, żaden inny atrybut systemowy nie może być zmieniony przez użytkownika. Specjalnym rodzajem atrybutu jest atrybut *Dokument* , który przechowuje plik z danymi.

Atrybuty użytkownika

Użytkownik (administrator) może zdefiniować własne atrybuty i dołączyć je do dowolnej klasy. Typ i przeznaczenie nowych atrybutów wynikają z przyjętego rozwiązania.

Atrybuty typu *Dokument*

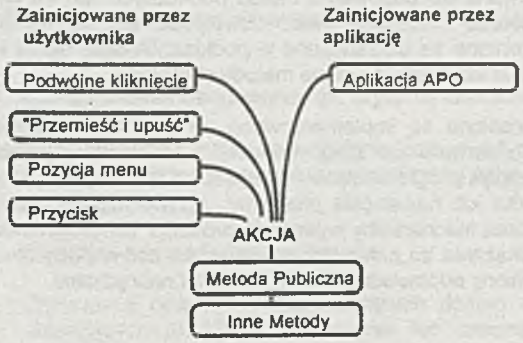
Atrybuty typu *Dokument* przechowują informacje w plikach (na serwerze) a nie w rekordach bazy danych. LinkWorks nie wnika w treść tych informacji. Mogą one być stosowane tam, gdzie nie można z góry określić rozmiaru danych lub rozmiar ten może się zmieniać. Zawartość atrybutu tego typu może być interpretowana (odczytywana, zmieniana) jedynie przez programy zewnętrzne, np. MS Word, Excel, zewnętrzne bazy danych. Systemowy atrybut *Dokument* jest przykładem atrybutu tego typu.

Własności

Własności definiują sposób prezentacji obiektów użytkownikowi. Jest to stały, jednakowy dla każdej klasy, zbiór charakterystyk takich jak wygląd ikony, domyślny sposób pokazywnia obiektów wewnątrz obiektu klasy złożonej itp.

Metody i akcje

Użytkownik lub aplikacja zewnętrzna może zainicjować akcję zwiazaną z pewnym obiektem (np. *edytuj obiekt*). Powoduje to realizację szeregu metod zdefiniowanych w ramach tej klasy. Metoda jest to "program" napisany w Języku Programowania Klas. Metody determinują funkcjonalność obiektu. Metody są dziedziczone przez podklasę danej klasy. Mogą one być następnie modyfikowane, jeśli wymagana jest zmiana funkcjonalności obiektu (np. czym innym jest edycja *Notatki* niż *Teczki*). Nowa funkcjonalność jest osiagana poprzez dodanie nowych metod do istniejących lub nowych klas.



Rys. 5. Akcje i metody publiczne

Metody systemowe i metody użytkownika

Metody dostarczane razem z LinkWorks są nazywane metodami systemowym, są one zdefiniowane dla klas systemowych jak również mogą być dziedziczone przez klasy użytkownika. Zmienione metody systemowe lub nowe, dodane metody są nazywane metodami użytkownika.

Istnieją 4 rodzaje metod, są to:

- metody publiczne
- wewnętrzne metody chronione
- zewnętrzne metody chronione
- metody prywatne

Metody publiczne

Metody publiczne są wywoływane przez użytkowników realizujących pewną akcję na obiekcie. Mogą one być również wywoływane przez zewnętrzne programy.

Metody publiczne są zbudowane z wywołań metod chronionych oraz innych metod publicznych.

O tym, czy użytkownik może wywołać daną metodę publiczną pewnego obiektu, decydują prawa dostępu do tego obiektu. Prawa te są określane w momencie tworzenia obiektu (mogą być jednak później zmieniane) przez jego twórcę.

Metody chronione wewnętrzne i zewnętrzne

Metody chronione (wewnętrzne i zewnętrzne) nie są dostępne spoza systemu LinkWorks. Są one używane do budowania metod publicznych lub innych metod chronionych. Metoda chroniona może odwoływać się do metod prywatnych. Metody chronione są dziedziczone w podklasach oraz mogą tu być zmieniane. Można również tworzyć własne metody chronione.

Zewnetrzne metody chronione są implementowane na zewnątrz systemu LinkWorks ,na maszynie serwera lub stacji roboczej. Do ich implementacji można użyć dowolnego język programowania, można również tu wykorzystąć gotowe biblioteki procedur lub nawet całe programy. Język Programowania Klas definiuje składnię oraz mechanizmy wywołania procedur lub programów zewnętrznych oraz przekazania im parametrów. Natomiast kod metody musi być utworzony i uruchomiony odpowiednimi, zewnętrznymi narzędziami.

Zewnętrzne metody chronione pozwalają rozszerzyć system LinkWorks o nowe funkcje, niedostępne wewnątrz systemu. Jest to również sposób integracji LinkWorks z innymi, zewnętrznymi systemami np. bazami danych.

Metody prywatne

Metody prywatne są to metody oferujące podstawową funkcjonalność systemu LinkWorks (odpowiednik biblioteki systemowej). Mogą one być użyte jedynie w systemowych metodach chronionych i nie mogą być zmieniane przez użytkownika. Sposób ich implementacji może ulec zmianie w następnych wersjach systemu.

3. Techniki implementacji rozwiązań

Gotowa, oferowana wraz z systemem LinkWorks funkcjonalność stanowi w większości przypadków główną część rozwiązania wymaganego przez użytkownika. W szczególności od razu dostępny jest: okienkowy interfejs użytkownika, poczta elektroniczna, definiowanie obiegów dokumentów, możliwość gromadzenia, przechowywania oraz wyszukiwania dokumentów w różnych postaciach, mechanizmy kontroli dostępu do informacji oraz szereg innych. Traktując tą funkcjonalność jako bazę, implementatorzy mogą skoncentrować się jedynie na realizacji specyficznych dla danego rozwiązania funkcji, np. *wprowadzanie i realizacja zamówień klienta.*

Istotną rolę w implemetacji rozwiązań odgrywa integracja istniejących lub nowych aplikacji pracujących na stacjach roboczych lub serwerach. Przykładem takiej aplikacji może być system finansowo-księgowy. Integracja z LinkWorks oznacza, że informacje generowane przez te aplikacje są przechowywane w systemie jako obiekty i można na nich wykonywać wszelkie operacje dostępne w LinkWorks takie jak przesyłanie pocztą, wspólny dostęp, opisywanie dodatkowymi atrybutami, podpisywanie itp.

Nowe rozwiązania, implementowane w LinkWorks, powinny być realizowane w dwóch krokach:

1. Konfigurowanie wbudowanej, generycznej funkcjonalności systemu

   Nie jest dodawana nowa funkcjonalność, a jedynie zmieniana istniejąca, np. zmiana praw dostępu lub zmiana menu pojawiającego się na pulpicie użytkownika.

2. Rozbudowa systemu o nowe funkcje

   Tworzenie nowych aplikacji mających dostęp do obiektów, integracja istniejących produktów, dodawanie lub zmiana metod związanych z obiektami.

W niektórych przypadkach implementacja rozwiązania może ograniczyć się jedynie do pierwszego kroku.

Aplikacje Plus Obiekty (APO)

Mechanizm integracyjny, który pozwala zewnętrznym aplikacjom pobierać i przetwarzać obiekty systemu LinkWorks nazywa się Przyłączem APO (APO Plugs).
Przyłącza APO pozwalają aplikacjom:

- wyszukiwać obiekty w systemie

  Mechanizmy Przyłacza APO pozwalają znaleźć obiekt(y) spełniające określone kryteria (np. obiekty klasy Notatka, których nazwa rozpoczyna się od litery 'K'). Został tu zdefiniowany język zapytań zbliżony do SQL pozwalający określać te kryteria.

- dostawać się do atrybutów obiektów

  Aplikacje zewnętrzne mogą czytać oraz modyfikować (jeśli pozwalają na to prawa dostępu) atrybuty systemowe lub użytkownika wskazanego obiektu.

- wykonywać metody publiczne obiektów

  Aplikacje zewnętrzne mogą wykonywać dowolną (z dokładnością do praw dostępu) metodę publiczną wskazanego obiektu. W szczególności aplikacja może utworzyć nowy obiekt, odczytać, zmodyfikować i zlikwidować jego zawartość.

- dostawać się do informacji konfiguracyjnych

  Aplikacje mogą odczytywać wszystkie informacje konfiguracyjne o całym systemie np. zarejestrowane stacje robocze, struktura organizacyjna, nazwy wszystkich zdefiniowanych klas, nazwy i typy atrybutów danej klasy.

Przyłacze APO może być dostępne z wykorzystaniem różnych interfejsów w zależności od platformy, na której działa aplikacja. Aplikacje w systemie MS Windows mogą wykorzystać w tym celu protokół DDE lub OLE Automation. Na każdej platformie (również na serwerze) dostępne są biblioteki dzielone (DLL). Ta ostatnia metoda pozwala wykorzystać technikę APO z prawie dowolnego języka programowania.

```
    Global Lnx As Object
    Global ObjList As String

    Set Lnx = CreateObject("Lnx.Application")

    ObjList = Lnx.ApoQuery("Select X Where X.Name
Like

""Folder%"" ")
```

Rys. 6. Przykład wykorzystania APO z Visual Basica

Technika Programowanie Klas

Technika programownia klas jest używana w celu zmiany lub rozszerzenia generycznej funkcjonalności systemu LinkWorks. Osiąga się to przez modyfikację lub dodawanie metod istniejących lub nowych (powstałych na bazie istniejących) klas obiektów. Nie można zmieniać metod klas systemowych.

Kod metody jest zapisywany w Języku Programowania Klas. Jest to obiektwo-zorientowany język wysokiego poziomu (zbliżony do C++) pozwalający deklarować zmienne różnych typów, dostawać się do atrybutów obiektów, wykonywać operacje numeryczne. Posiada również strukturalne instrukcje sterujące (if...else, while..) oraz możliwość działania na listach.

Metody nowe lub zmodyfikowane muszą być przekompilowane przy pomocy kompilatora Języka Programowania Klas dostępnego na serwerze lub w programie Workbench na stacji roboczej. Zakończona powodzeniem kompilacja (bez błedów składniowych) powoduje automatyczne wprowadzenie kodu nowej lub zmodyfikowanej metody do systemu. Od tej pory można ją wywoływać z innych metod lub z programów zewnętrznych (w tym ostatnim przypadku pod warunkiem, że jest to metoda publiczna).

Kod zewnętrznych metod chronionych istnieje na zewnątrz systemu LinkWorks. Może on być tworzony z wykorzystaniem prawie dowolnego środowiska programowego. Ponadto może to być już istniejący program lub biblioteka procedur. LinkWorks oferuje następujące mechanizmy wywoływania zewnętrznego kodu metody:

- DDE
- OLE Automation
- linia komendy (wywołanie programu)
- biblioteki dzielone (DLL)

Jeśli kod zewnętrzny znajduje się na serwerze, to można go wywołać tylko przy pomocy linii komendy lub DLL.
Metody zewnętrzne pozwalają, między innymi:

- dodać do systemu nowe funkcje niedostępne w samym języku programowania klas jak np. działania na tekstach (porównywania, modyfikacje itp.)
- dostawać się do informacji zawartych w zewnętrznych bazach danych, plikach czy systemach
- tworzyć nowe elementy interfejsu użytkowego, np. wyświetlenie dodatkowych informacji podczas tworzenia obiektów określonych klas
- uruchamiać programy zewnętrzne np. program archiwizacji na taśmie.
- itp.

```
Folder::SWCOMP:ExtSetObjName (String name)
{
    extern access MLOC_LOCAL (name)
        for WSTYPE_DOSWIN =
        MEXT_DLL("\"lib.dll\".GetObjName(OUT String
[100])")
}
```

Rys. 7. Przykład programu zapisanego w Języku Programowania Klas

Konfiguracja LinkWorks

Każdy element interfejsu użytkownika może być modyfikowany przez administratora systemu przy pomocy programu Konfiguratora (element systemu LinkWorks).

Konfiguracja pozwala zmienić lub dodać, między innymi, następujące elementy systemu LinkWorks:

- ikony skojarzone z obiektami
- menu
- przyciski
- teksty komunikatów
- prawa dostępu
- narzędzia (aplikacje) używane do edycji dokumentów
- narzędzia znajdujące się na biurku (kalkulator, kosz itp)

- nowe Komponenty Programowe

Przed wprowadzeniem nowego elementu konfiguracji należy wcześniej mieć zdefiniowany Komponent Programowy, do którego ten element będzie należał. Komponenty Programowe mogą być eksportowane i następnie importowane oraz instalowane gdzie indziej. Dzięki temu konfiguracja systemu może być przenoszana i powielona w innym środowisku LinkWorks.

Skrypty w systemie LinkWorks

Skrypty systemu LinkWorks są środowiskiem programowym dostarczanym wraz z tym systemem. Skrypty są tworzone w modularnym języku programowania podobnym do języka BASIC. Skrypty, podobnie jak programy zewnętrzne, mogą działać na obiektach poprzez Przyłącza APO. W odróżnieniu od tych programów, są one niezależne od platformy, na której są wykonywane.

Wraz z LinkWorks dostarczane jest narzędzie rejestrujące akcje wykonywane w systemie. Narzędzie to tworzy program w języku skryptów, który może być następnie wykonany i w ten sposób można powtórzyć zarejestrowane czynności. Program ten może być również zmieniony i np. uzupełniony o nowe funkcje. Skrypt, z punktu widzenia LinkWorks jest obiektem, może być więc przesłany pocztą lub w inny sposób udostępniony innym użytkownikom. Można również tworzyć biblioteki standardowych procedur wykorzystywanych przez wiele osób.

Skrypty dzielą się na: działajace na serwerze i działające tylko na stacji roboczej (skrypty działające na serwerze mogą również działać na stacji). Te drugie z reguły wymagają interakcji z użytkownikiem. Wszystkie skrypty powstałe w wyniku rejestracji czynności użytkownika należa do tego drugiego rodzaju.

Skrypty mogą być wykorzystywane do:

- automatyzacji prac w systemie LinkWorks; jest to szczególnie przydatne dla administratora systemu,
- tworzenia niedużych programów do manipulacji obiektami systemu, np. wyszukiwania w systemie określonych obiektów.

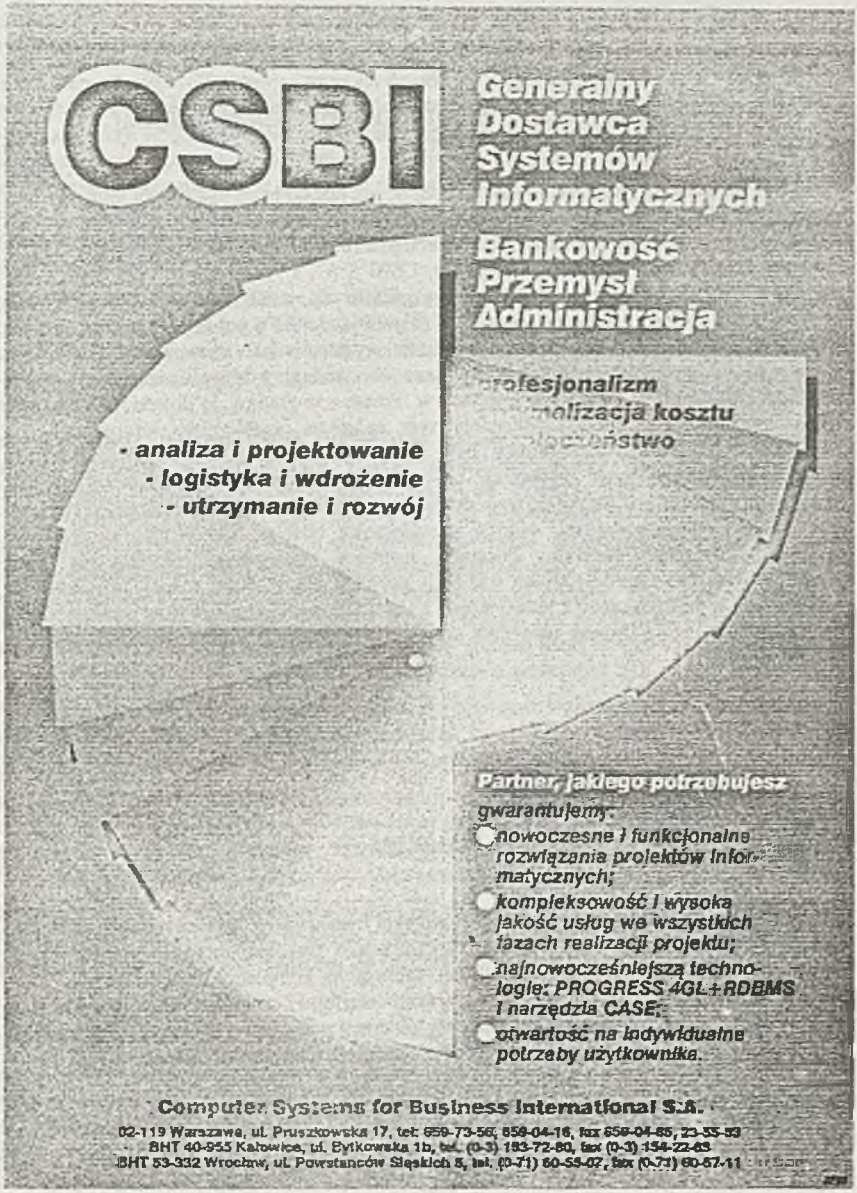
```
  objlist = (SELECT x FROM 1 WHERE x.ObjectClass =
"Note")
  obj = (SELECT x WHERE x.Name = "MyNote")
  IF obj IN objlist THEN
    PrintLn "MyNote found on Cell 1"
  END IF
```

Rys. 8. Fragment programu w języku skryptów LinkWorks

4. Podsumowanie

LinkWorks jest systemem, który w nowy sposób organizuje pracę w zespołach. System pozwala integrować ludzi, aplikacje oraz informacje. Użytkownicy systemu współpracują z nim poprzez jednolity interfejs okienkowy. Obiektowo-zorientowana koncepcja systemu przyjęta w LinkWorks, upraszcza zarówno sposób działania w systemie jak również jego rozbudowę. Jako system otwarty, LinkWorks może działać w różnych środowiskach sprzętowych i programowych (systemach operacyjnych). Otwartość oznacza również możliwość współpracy LinkWorks z innymi systemami pocztowymi , biurowymi i innymi pracującymi lokalnie na stacji roboczej lub w środowisku sieciowym.

Wszytkie te cechy sprawiają, że LinkWorks jest przodującym produktem w klasie oprogramowania pracy grupowej (groupware) i niewątpliwie jest przykładem programu, jaki będzie dominował w najbliższej przyszłości.


CSBI
Generalny
Dostawca
Systemów
Informatycznych
Bankowość
Przemysł
Administracja
- analiza i projektowanie
- logistyka i wdrożenie
- utrzymanie i rozwój
Partner, jakiego potrzebujesz
gwarantujemy:
nowoczesne i funkcjonalne rozwiązania projektów informatycznych;
kompleksowość i wysoka jakość usług we wszystkich fazach realizacji projektu;
najnowocześniejszą technologię: PROGRESS 4GL+RDBMS i narzędzia CASE;
otwartość na indywidualne potrzeby użytkownika.
Computer Systems for Business International S.A.
02-119 Warszawa, ul. Pruszkowska 17, tel: 659-73-56, 658-04-16, fax 659-04-65, 23-55-93
BHT 40-955 Katowice, ul. Bykowska 1b, tel. (0-3) 153-72-80, fax (0-3) 154-22-65
BHT 53-332 Wrocław, ul. Powstańców Śląskich 5, tel. (0-71) 60-55-07, fax (0-71) 60-57-11

Computer Systems for Business International

Computer Systems for Business International S.A. jest polsko-brytyjską spółką joint-venture, należącą do grupy CSB. Grupa bierze swój początek od firmy CSB Ltd, założonej w 1986 w Wielkiej Brytanii i specjalizującej się w komputerowych systemach zarządzania. Grupa CSB Plc obecnie to: trzy firmy brytyjskie, CSBI S.A., i założona w 1993 roku CSBI Eastern Europe (z siedzibą w St.Petersburgu). CSBI Sp. z o.o. - w czerwcu 1994 roku przekształcona w Spółkę Akcyjną - zaistniała na polskim rynku w połowie 1990 roku, przede wszystkim jako dostawca kompletnych, autorskich systemów informatycznych, opartych na zintegrowanych pakietach oprogramowania oraz jako generalny dystrybutor nowoczesnego narzędzia PROGRESS. Ponad 500 instalacji w Polsce i za granicą, to dowód, że CSBI jest firmą, która potrafi sprostać najtrudniejszym zadaniom współczesnej informatyki , w dziedzinie bankowości, administracji oraz przemyśle. CSBI jako generalny dostawca systemów informatycznych, wychodząc naprzeciw potrzebom jak również oczekiwaniom klientów podejmujących trudne decyzje dotyczące komputeryzacji i informatyzacji różnych sfer działalności gospodarczej, oferuje kompleksową obsługę wszystkich faz realizacji projektu. CSBI jako generalny dostawca systemów informatycznych jest nie tylko koordynatorem poszczególnych faz realizacji projektu ale w dużej części bezpośrednim wykonawcą, w wyniku czego z jednej strony gwarantuje profesjonalizm oraz wysoką jakość produktów i usług, z drugiej natomiast optymalizację kosztów i zmniejszenie ryzyka po stronie klienta.

Progress

Progress jest potężnym, graficznym, zintegrowanym środowiskiem do tworzenia przenośnych i skalowalnych aplikacji o najwyższych wymaganiach (*mission critical*), pracujących w architekturze klient/serwer, *host* i mieszanej, oferującym kompletny zestaw narzędzi: bazę danych, rozbudowany słownik danych, generatory interfejsu i raportów, silny, proceduralny i zdarzeniowy język programowania czwartej generacji połączony z ANSI SQL poziomu 2 oraz narzędzia dla użytkownika aplikacji. Współdziała z różnymi narzędziami CASE.

Jest niezależny od interfejsu użytkownika, systemu operacyjnego, bazy danych i protokołu sieciowego. Działa na prawie 1500 różnych komputerach i kilkudziesięciu systemach operacyjnych (UNIX, Novell, DOS/Windows, Windows NT, VMS, OS/400).

Podstawowe środowisko do tworzenia aplikacji stanowi system o nazwie ProVISION, na który składają się następujące moduły:

Słownik Danych, *Data Dictionary* - program do obsługi Słownika bazy danych Progress. Za pomocą tego narzędzia programista może nie tylko zdefiniować strukturę tabel i pól bazy danych, ale także wpisać informacje dotyczące sposobu wyświetlania danych: formatów, etykiet oraz tekstów podpowiedzi. Ponadto, możliwe jest napisanie procedur - *triggerów* - wywoływanych automatycznie w momencie zajścia określonych zdarzeń, na przykład próby odczytania rekordu, usunięcia, czy też modyfikacji jego zawartości. *Triggery* ułatwiają utrzymanie spójności referencyjnej bazy danych, pozwalają też na lepszą ochronę przed nieautoryzowanym dostępem.

Edytor Procedur, *Procedure Editor*, do tworzenia programu użytkownika. Edytor procedur pozwala na jednoczesną pracę z wieloma plikami, możliwe jest też wyświetlenie zawartości jednego pliku podczas pracy z innym plikiem w drugim oknie. Edytor procedur pozwala na łatwe uruchamianie pisanych programów: kompilacja następuje po naciśnięciu jednego klawisza.

Generator Interfejsu Użytkownika, *User Interface Builder*, *UIB* - narzędzie do graficznego tworzenia ekranów aplikacji. Praca z tym narzędziem polega na wskazywaniu i przemieszczaniu myszą gotowych elementów, takich jak okienka, przyciski, przełączniki, rozwijalne menu i inne. Umieszczanie pól do wyświetlania zawartości rekordów jest znacznie ułatwione, gdyż program korzysta z domyślnych formatów umieszczonych wcześniej w Słowniku Danych. UIB pozwala określić zachowanie aplikacji w odpowiedzi na różne zdarzenia przez umieszczanie fragmentów procedur w języku Progress 4GL. Następnie UIB generuje kod programu w języku Progress 4GL. Program ten może być uruchomiony z innego środowiska, na przykład z Edytora Procedur. Ręczne modyfikacje wygenerowanego programu nie przeszkadzają w późniejszym korzystaniu z generatora UIB.

Generator Raportów, *Report Writer* - narzędzie do tworzenia złożonych raportów w sposób graficzny. Wygenerowane raporty mogą zawierać rysunki oraz różne kroje czcionek, pozwalając w pełni wykorzystać możliwości drukarek laserowych.

Program Śledzący, *Application Debugger*, daje możliwość interaktywnego kontrolowania przebiegu wykonywanych programów. Pozwala ustawiać punkty kontrolne, śledzić krokowo wykonywanie procedur i triggerów, sprawdzać wartości zmiennych i buforów danych. Informuje o stanie transakcji i elementów interfejsu użytkownika

Administrator Bazy, *Database Administrator*, zarządza hasłami i prawami dostępu do bazy, wymianą danych pomiędzy bazą Progress a arkuszami kalkulacyjnymi, procesorami tekstu, programami graficznymi oraz dostępem do definicji nieprogressowych baz danych

Język Progress 4GL

Język PROGRESS 4GL jest kompletnym, proceduralnym językiem do tworzenia aplikacji, obsługujących bazy danych. Daje on możliwości kontroli każdego aspektu działania aplikacji, nawet na poziomie przyciśnięcia klawisza. Instrukcje i składnia PROGRESSa są bardzo zbliżone do naturalnej składni języka angielskiego, co zdecydowanie ułatwia zarówno nauczenie jak i posługiwanie się tym językiem. Kod źródłowy w PROGRESSie jest wyjątkowo

krótki, zwarty i tym samym łatwy w późniejszym utrzymaniu. Jego czytelność jest tak duża, że prawie wcale nie wymaga komentarzy. PROGRESS 4GL pozwala również na wbudowanie w aplikacje stworzone w języku PROGRESS instrukcji ANSI SQL. W jednej frazie można łączyć wyrażenia PROGRESS 4GL i SQL. Aplikacje w PROGRESSie są tworzone znacznie szybciej niż w językach trzeciej generacji (ok. dziesięciokrotnie szybciej, niż np. w COBOLU). PROGRESS bowiem dostarcza wielu funkcji, które w innych językach wymagają stworzenia odpowiednich podprogramów przez użytkownika. Wśród nich są na przykład mechanizmy dostosowywania szerokości okienka lub ramki do formatu wyświetlanych w niej danych i ilości wolnego miejsca na ekranie.

System zarządzania bazą danych Progress

PROGRESS 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 trybie wielodostępnym (host), jak i przy zastosowaniu architektury klient/serwer, PROGRESS RDBMS posiada wiele własności, które minimalizują liczbę odwołań do dysku a zarazem zwiększają wydajność systemu. Należą do nich:

Blokowanie na poziomie rekordu

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 poważne 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 jeden rekord. Mikrotransakcje również blokują dane na możliwie najniższym poziomie, na przykład jeżeli konieczne jest czasowe zablokowanie indeksu, blokowana jest jedna strona, zamiast całego drzewa.

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.

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.

Zmienna długość rekordu

Przechowywanie danych ze zmienną długością rekordu 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.

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.

Zrzucanie buforów w tle (Asynchronus Page Writers)

Asynchroniczna obsługa buforów, to niezależne procesy, które 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 ich użyć ponownie, jest na tyle duże, że proces klienta praktycznie w ogóle nie musi wykonywać operacji dyskowych.

System zarządzania bazą PROGRESS (RDBMS — Relational Database Management System) umożliwia równoczesny dostęp do bazy danych wielu użytkownikom, czuwając jednocześnie nad zachowaniem spójności bazy danych. System zabezpieczeń jest zorientowany na transakcje i obejmuje między innymi elastyczną kontrolę rezerwacji rekordów, dwufazowy protokół potwierdzeń oraz mechanizmy, umożliwiające automatyczne odtwarzanie stanu bazy po awarii systemu lub zaniku zasilania. Odtwarzając bazę danych po awarii, PROGRESS automatycznie wycofuje transakcje niezakończone.

Przenośność i wielopłaszczyznowość

Produkty PROGRESS mogą być instalowane na bardzo różnorodnym sprzęcie, pod wieloma systemami operacyjnymi (między innymi Unix i kilkadziesiąt jego pochodnych, OS/400, VMS, BTOS/CTOS, a także Novell NetWare, NetWare Lite, MS-Windows NT, MS-Windows, DOS i OS/2). Lista maszyn zawiera komputery wszystkich liczących się producentów, między innymi Apple, Bull, Compaq, Data General, DEC, Hewlett-Packard, IBM, ICL, Pyramid, Sequent, Silicon Graphics, Sun, Stratus, Unisys, Wyse oraz komputery kompatybilne z IBM PC. Są wśród nich potężne, kilkudziesięcioprocesorowe maszyny i komputery osobiste.

Napisane w PROGRESSie aplikacje są w pełni przenośne pomiędzy wszystkimi platformami sprzętowymi, przy jednoczesnym zachowaniu optymalnego wykorzystania szczególnych możliwości każdej z platform. Ta sama aplikacja działać może w środowisku wielodostępnym, klient/serwer i mieszanym. Przenoszenie aplikacji użytkowej z jednego systemu na drugi odbywa się bez zmiany nawet jednej linijki kodu. Przynosi to użytkownikowi ogromne korzyści w postaci oszczędności czasu i pieniędzy podczas zmiany posiadanego sprzętu komputerowego lub też przy dystrybucji stworzonej w PROGRESSie aplikacji. Niezależnie od możliwości instalowania aplikacji w dowolnym środowisku sprzętowym, bardzo istotna dla użytkownika jest możność uruchomienia aplikacji w mieszanym środowisku obejmującym komputery różnych producentów, pracujących pod różnymi systemami operacyjnymi, z różnymi interfejsami użytkownika, połączone siecią lokalną bądź o większym zasięgu. PROGRESS współpracuje z wszystkimi popularnymi protokołami sieciowymi, włączając w to TCP/IP, IPX/SPX, SNA, OSI, DEC Net, NetBIOS, CTOS Cluster i wiele innych. Przeniesienie aplikacji ze środowiska obejmującego pojedynczy komputer w środowisko złożone z wielu różnych maszyn z różnymi systemami operacyjnymi i połączonych różnymi protokołami odbywa się bez jakiejkolwiek modyfikacji napisanej aplikacji. Możliwe jest rozdzielenie przetwarzania — zarówno transakcji (środowisko klient/serwer), jak i obsługi samej bazy danych poprzez rozdzielenie bazy danych pomiędzy wiele serwerów. Operacja taka nie wymaga żadnych modyfikacji w programie obsługi bazy danych. PROGRESS - co istotne -działa nie tylko na własnej bazie, ale także na bazach innych producentów (są to: bazy Oracle, Sybase, Rdb, pliki Informixa SE oraz pliki RMS DECowskiego systemu VMS, a na maszynach AS/400 baza zintegrowana z systemem operacyjnym OS/400).

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 własności jakie posiadają różne systemy operacyjne i konfiguracje sprzętowe. Dlatego też wydajność aplikacji PROGRESS wzrasta proporcjonalne do tego, jak nowe zasoby - np. pamięć, czy procesor — są dodawane do systemu.

Progress wersja 7.
Elementy Programowania obiektowego

- Jerzy Sitarz
- CSBI BHT Katowice
- Świnoujście, maj 1995.

Co to jest PROGRESS?

- Baza danych typu PROGRESS (dostępna czasami przez SQL)
- Język 4GL
- Narzędzia dla tworzenia aplikacji bazo-danowych
- Środowisko dla pracy programów aplikacyjnych

Indeksy słowowe

"Przedsiębiorstwo Naukowo-Handlowe NaPro. Sp. z o.o."

```
....
FOR EACH dostawcy WHERE nazwa CONTAINS "NaPro":
     DISPLAY nazwa.
END.
```

Zmiany w serwerze

- Sekwencje
- Kontrola spójności
- Tablice tymczasowe
- Selekcja

Logika programu

- 4GL i preprocesor
- procedury wewnętrzne
- obiekty graficzne
- atrybuty, metody i obsługa obiektów
- fraza VIEW-AS.

Środowisko

- strony kodowe
- atrybuty łańcuchów
- atrybuty sesji

Preprocesor

```
&SCOPED-DEFINE moja_lista Janek Franek -
        Marek Andrzej Kazek Joasia Basia
...
&IF {moja_lista} NE "" &THEN
   DISPLAY  {moja_lista}.
&ELSE
   RUN pobranie_argumentów.
&ENDIF
```

Procedura wewnętrzna

```
DEFINE BUTTON rob_cos LABEL "Rób"
   TRIGERS:
         ON CHOOSE DO:
                RUN moja_procedura.
          END.
    END TRIGERS.
...
PROCEDURE moja_procedura.
    DISPLAY dostawca WITH FRAME {&FRAME-NAME}.
END PROCEDURE.
```

Zależności między obiektami graficznymi

Sesja
  Okna
    Okno dialogowe i ramki
      Grupy pól
        Pole
    Menu
      Pozycja menu
      Submenu
        Pozycja menu
        Submenu
    Okno komunikatów

Atrybuty obiektu graficznego

```
DEFINE VARIABLE zmienna
       AS CHARACTER
       FORMAT "x(20)"
       VIEW-AS FILL-IN
       NO-UNDO.
.......
DISPLAY zmienna WITH FRAME {&FRAME-NAME}
.......
DISPLAY
       zmienna : COLUMN
       zmienna : LABEL
       zmienna : ROW = 7.
```

Wywoływanie metod

zmienna = obiekt : metoda (parametr).

```
ADD-FIRST (element)          ADD-LAST (element)
INSERT (nowyel, element)     IS_SELECTED (element)
DELETE (element)             DELETE (indeks)
SCROLL-TO-ITEM (element)     SCROLL-TO-ITEM (indeks)
```

Zagadka

```
_ _ _ _ _ _ _ _ _ _ _ _ _ _
      _ _ _ _ _ _ _
      _ _ _ _ _ _ _
A C Ć D F G H J K L Ł M N Ń P
  Q R S Ś T V W X Z Ż Ź
```

Obsługa zdarzeń.

```
DEFINE zmienna AS CHARACTER VIEW-AS TEXT INITIAL "Abc
                    FORMAT "x(20)" NO-UNDO .
.....
ON LEAVE OF zmienna DO:
 zmienna:SCREEN-VALUE = "Sitarz" THEN STOP .
END .
 ..............
DISPLAY zmienna .
 ..............
ENABLE zmienna .
..............
DISABLE zmienna .
```

Fraza VIEW-AS

```
VIEW-AS
      TEXT
      FILL-IN

      SLIDER
      RADIO-SET
      TOGLE-BOX
      SELECTION-LIST
      EDITOR

      ALERT-BOX
```

Typowa struktura programu

- Definicje ...
- Deklaracje formatek ...
- Obsługa zdarzeń...
- Wyświetlanie formatek (umożliwienie obsługi)...
- Zawieszenie programu do jakiejś akcji...
- Zawieszenie obsługi...
- Podprogramy lokalne...

Łańcuchy tekstowe

```
"Łańcuch" [:[R|L|C|T] [U] [maks. długość] ]

DEFINE VARIABLE mam AS LOGICAL
                        LABEL "Mam":L12 NO-UNDO.
```

Plansza dla sesji

```
WINDOW:BGCOLOR                FULL-HIGH-CHARS
SESSION:TEMP-DIRECTORY        DATE-FORMAT
DISPLAY-TYPE                  CHARSET
WINDOW-SYSTEM
```

Pytania?

Danuta Kosecka, Stanisława Ossowska, Irena Radczyńska
Andrzej Wiśniewski, Artur Lempkowski

Acucobol-85

System Nowej Generacji

Oficyna Informatyczna

Siedziba:
ul. Błońska 62
05-830 Nadarzyn
tel. (0-2) 729-87-96
tel./fax (0-2) 729-87-97

Sprzedaż:
ul. Postępu 3
02-691 Warszawa
tel. 43-67-51, 43-61-94
fax 43-63-60

Jedyny dystrybutor Acucobol-85 w Polsce

2. Architektura systemu Acucobol-85 (rys.1)

Podstawowa część systemu zawiera:

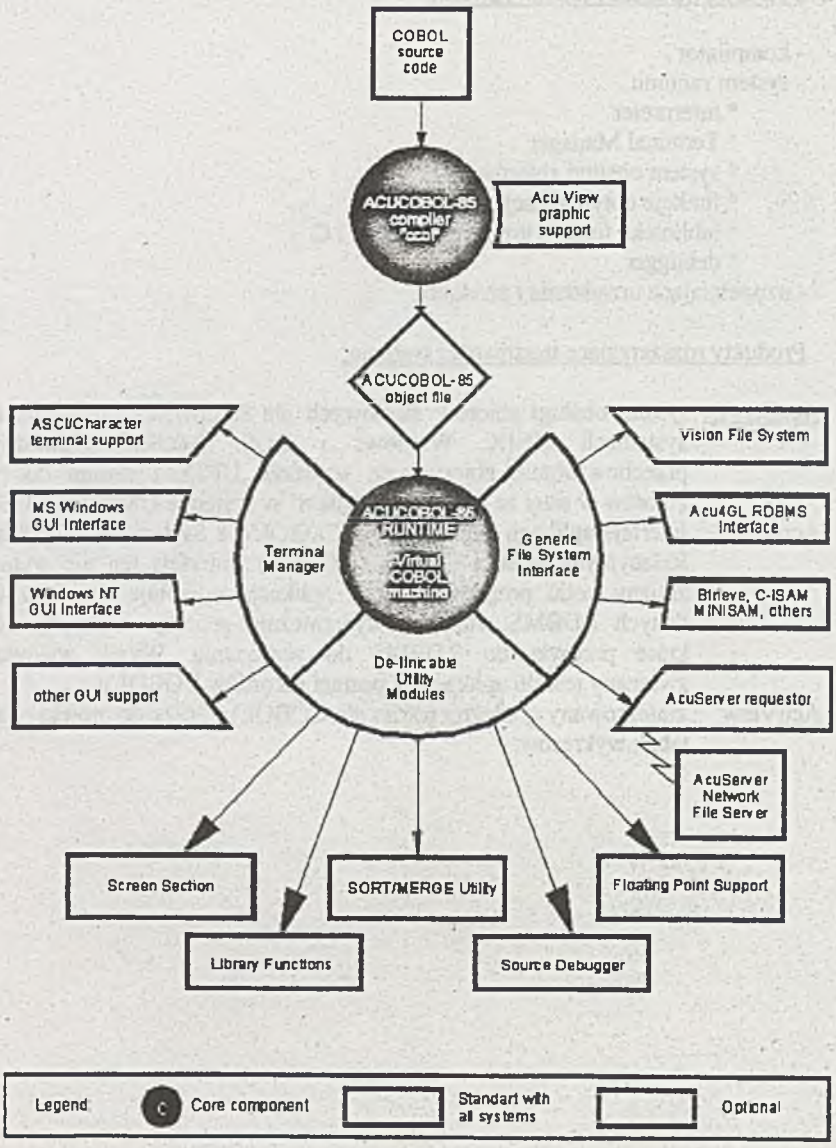
- kompilator
- system runtimu
    * interpreter
    * Terminal Manager
    * system obsługi zbiorów
    * funkcje optymalizacji
    * bibliotekę funkcji języków COBOL i C
    * debugger
- uzupełniające urządzenia i produkty

Produkty rozszerzające możliwości systemu:

AcuServer: system obsługi zbiorów sieciowych dla środowiska client/server w systemach UNIX, Windows i DOS. AcuServer umożliwia przechowywanie zbiorów na serwerze UNIX i dostęp do tych zbiorów w sieci ze stanowiska "client" w systemie UNIX lub DOS.

Acu4GL: interfejs aplikacji napisanych w COBOL'u z Systemami zarządzania Relacyjnym Bazami Danych (RDBMS). Interfejs ten nie wymaga zmiany kodu programu. Kiedy aplikacja odwołuje się do zbioru danych RDBMS, Acu4GL dynamicznie generuje komendy SQL, które przesyła do RDBMS do wykonania. Wynik wykonania zwracany jest do aplikacji w postaci rekordów COBOL'u.

AcuView: zintegrowany graficzny pakiet dla COBOL'u. Służy do projektowania tabel, wykresów.



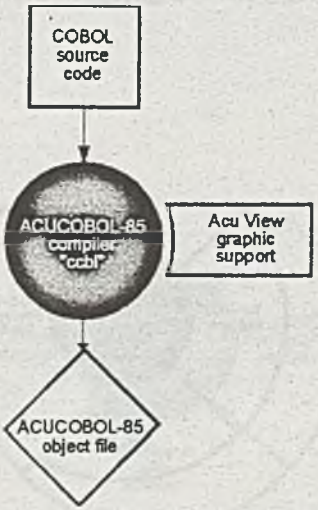
Rys.1 Architektura systemu Acucobol-85.

System Acucobol-85 jest w pełni kompatybilny z:
- DG ICOBOL
- VAX/COBOL
- RM/COBOL

Dodatkowo istnieje możliwość przy ustawieniu odpowiednich parametrów kompilowania pod:

- Micro Focus COBOL
- MS COBOL
- CA Realia

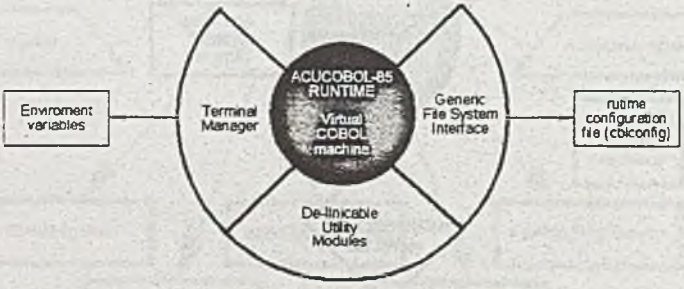
Kompilator Acucobol-85 jest jednoprzebiegowy i tłumaczy kod źródłowy na maszynowo-niezależny "object code". Zbiory w postaci "object" są gotowe do bezpośredniego wykonania i nie wymagają konsolidacji (linking)(np. 2).



Rys.2 Tłumaczenie programu źródłowego (source) na kod wynikowy (object).

Kod wynikowy (rys.3,4) jest interpretowany interpretowany przez system Runtime, który dostosowuje do danej platformy programowej i go następnie wykonuje

Rys.3 Wykonanie kodu wynikowego na maszynie wirtualnej



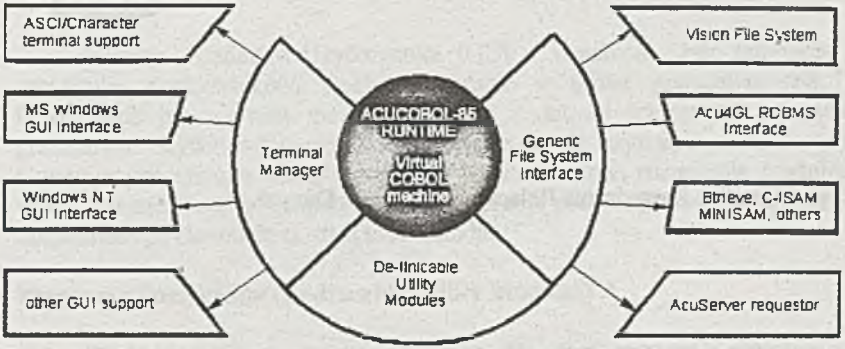
Rys.4 Przystosowanie przez maszynę wirtualną programu wynikowego do wybranego zbioru urządzeń i Systemu Obsługi Zbiorów.

Aktualnie istnieje przenaszalność pomiędzy następującymi platformami systemowymi:

- UNIX
- MS-DOS
- MS-DOS networks
- MS-DOS z Windows 3.x
- Windows NT
- IBM OS/2
- VAX/VMS
- Open VMS
- Data General AOS/VS
- Alpha Micro AMOS

Takie rozwiązanie pozwala na przenaszalność oprogramowania bez konieczności powtórnej kompilacji.

Przenaszalność pomiędzy różnymi konfiguracjami sprzętowymi odbywa się po przez (rys.5) zastąpienie zarządcy terminalami (Terminal Manager), istniejącymi urządzeniami np. urządzenia DOS-owe na urządzenia UNIX-owe.

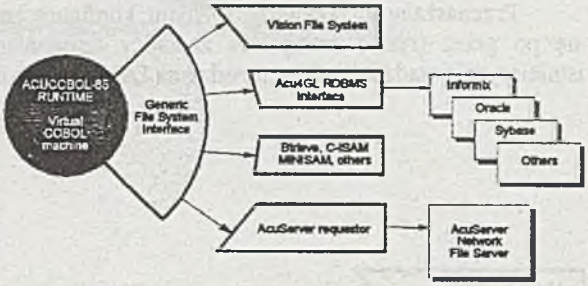


Rys.5 Możliwe grupy urządzeń systemowych; interfejs z Systemami Zarządzania Relacyjnymi Bazami Danych.

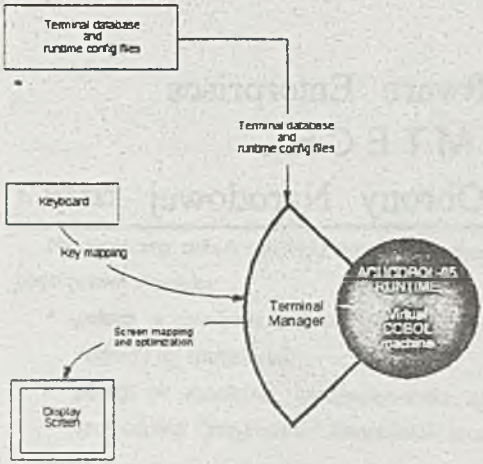
Acucobol-85 umożliwia korzystanie z procedur napisanych w języku C, co znacznie rozszerza możliwości systemu. Acucobol-85 wyposażony jest w system obsługi zbiorów indeksowych (Vision File System) (wykonywanych metodą B-tree). System ten umożliwia:

- definiowanie tablic sortowania
- pracę z rekordami zmiennej długości
- szyfrowanie danych
- odtwarzanie indeksów w uszkodzonych plikach.

Użytkownik Acucobol-85 ma także możliwość korzystania z baz danych 4GL takich jak ORACL, Informix, Sybase oraz baz danych tworzonych za pomocą Btrieve. Interfejs z tymi bazami jest realizowany za pomocą Generic File System (rys.6).



Rys.6 Interfejs Zarządzania Relacyjnymi Bazami Danych.



Rys.7 Zależność między menedżerem systemu a sprzętem i oprogramowaniem.

Interfejs Graficzny Użytkownika (GUI - Graphical User Interface) - umożliwia użytkownikowi pracę na ekranie w trybie graficznym (rys.7). Użytkownik ma również możliwość pracy z danymi tekstowymi w trybie graficznym. Użytkownik ma możliwość pracy z myszą, używania paska narzędzi i rozwijanych menu z narzędziami, ustawiania kolorów, rozmiarów i tytułów okien. Istnieje również specjalne oprogramowanie pozwalające na przenoszenie opracowanego ekranu do postaci pseudokodu.

Programy Pomocnicze (De-lincable Utility Modules)

Do dyspozycji programisty w Acucobol-85 jest również zbiór programów pomocniczych. Są to między innymi:
- debugger
- sort/marge
- biblioteki funkcji języków C i COBOL
- projektowanie ekranu
- oprogramowanie służące do ustawiania zmiennej przecinka

Magic Software Enterprises

KOMTECH

Akademia Obrony Narodowej

GENERATOR APLIKACJI MAGIC W PROCESIE PROJEKTOWANIA OBIEKTOWEGO

Opracował:
prof. dr hab. inż. Piotr SIENKIEWICZ
Centrum Informatyki AON
przy współpracy:
mgr Elżbiety MOLENDA
mgr Artura KUTKIEWICZA
KOMTECH – Radom

Wiosenna Szkoła PTI

Świnoujście Maj 1995

1. WSTĘP

Na społeczny odbiór rozwoju informatyki decydujący wpływ wydają się mieć dwa podstawowe zjawiska:

- *postęp w dziedzinie techniki komputerowej, który przyniósł masowość i różnorodność jej zastosowań,*
- *postęp w dziedzinie oprogramowania systemowego i narzędziowego, który spowodował "przyjazność" komputerów oraz wzrost kręgu ich użytkowników.*

Niejako w cieniu tych cokolwiek spektakularnych zjawisk odbywa się stały rozwój metodologii

- *analizy systemów informatycznych,*
- *inżynierii systemów informatycznych.*

W pierwszej z wymienionych dziedzin mamy do czynienia z doskonaleniem metod i technik analityczno-ocenowych służących przede wszystkim do analizy potrzeb i wymagań użytkowników oraz specyfikacji warunków systemowych koniecznych dla racjonalnej informatyzacji instytucji i organizacji gospodarczych. W drugiej zaś odbywa się proces doskonalenia metod i technik projektowania systemów informatycznych, w szczególności metod:

- *tworzenia i wdrażania standardowych systemów informatycznych dla podmiotów gospodarczych,*
- *tworzenia i wdrażania systemów informatycznych, specyficznych dla określonych użytkowników,*
- *kierowania zespołami i projektami systemów informatycznych.*

Stałemu procesowi doskonalenia podlega "warsztat" analityków i inżynierów systemów informatycznych.

Wzrost potrzeb użytkowników – organizacji gospodarczych oraz możliwości ich zaspokojenia przyniósł ewolucję systemów informatycznych obejmującą:

- systemy przetwarzania transakcji,
- systemy informowania kierownictwa,
- systemy wspomagania decyzji,
- systemy ekspertowe.

Obecnie, jak nigdy dotąd, efektywność zarządzania organizacjami gospodarczymi zależy od efektywności wspomagających je systemów informatycznych, ta zaś w coraz większym stopniu zależy od efektywności procesu ich projektowania oraz narzędzi wspomagających ten proces.

Należy zauważyć, że w ostatniej dekadzie w Polsce rozwój organizacji i technologii projektowania systemów informatycznych odbywał się niejako "w cieniu" techniki komputerowej (głównie PC i oprogramowania narzędziowego, o czym świadczy chociażby dominująca tematyka kursów i szkoleń informatycznych).

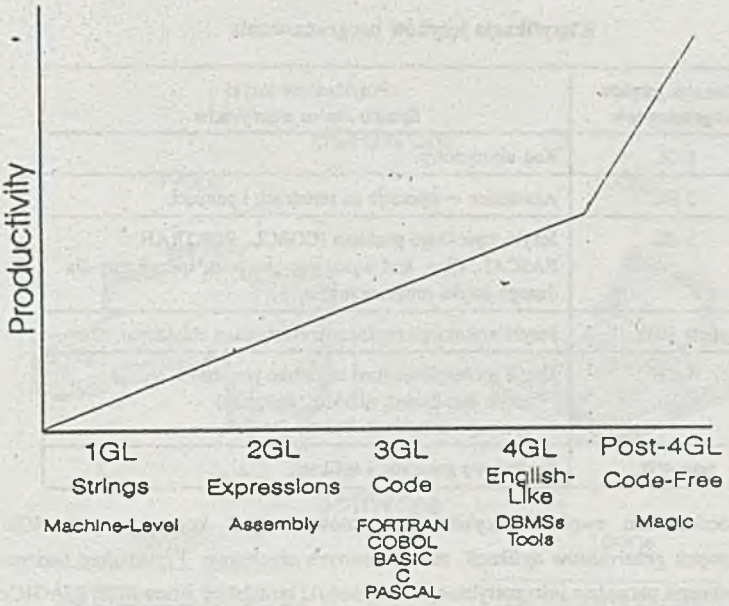
Wraz z pojawieniem się w połowie lat 80 pakietów CASE obejmujących następujące klasy:

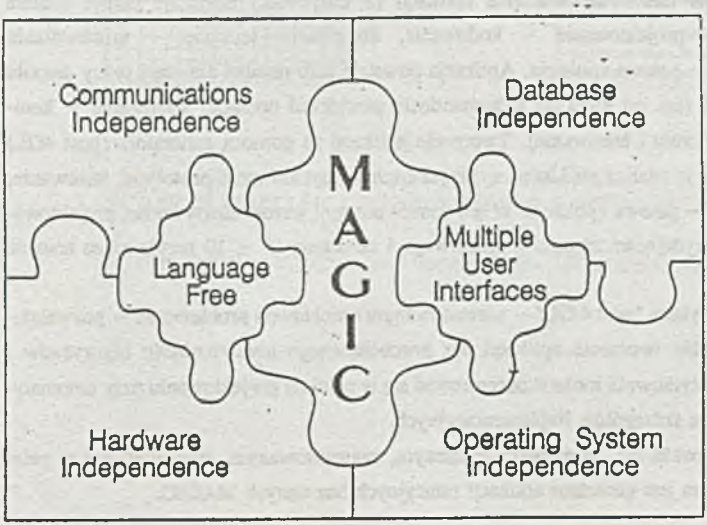
- *przechowywanie danych,*
- *modyfikację, adaptację systemów (korektę błędów i braków, stałe poszerzenie istniejącego oprogramowania dla zaspokojenia nowych potrzeb informacyjnych),*
- *wspomaganie cyklu życia systemu,*
- *sterowanie realizacją projektu,*
- *bieżące doskonalenie jakości systemu.*

Rozwój inżynierii oprogramowania determinuje rozwój języków programowania, obejmujący generacje:

I – języki maszynowe
II – asscemblery
III – języki niezależne od sprzętu
IV – języki "czwartej generacji"

Ogólnie można powiedzieć, że rozwiązując dany problem, w języku trzeciej generacji, należy wiedzieć, co się chce osiągnąć i w jaki sposób. Natomiast w języku "czwartej generacji można się bardziej skoncentrować na pierwszym aspekcie, gdyż sposoby rozwiązania znanych problemów znajdują się w wewnętrznej bazie danych.


Productivity
1GL
Strings
Machine-Level
2GL
Expressions
Assembly
3GL
Code
FORTRAN
COBOL
BASIC
C
PASCAL
4GL
English-Like
DBMSs
Tools
Post-4GL
Code-Free
Magic


Communications Independence
Database Independence
MAGIC
Language Free
Multiple User Interfaces
Hardware Independence
Operating System Independence

Klasyfikacja języków programowania

Generacja języków programowania	Przykładowe języki Sposób zapisu algorytmów
1 GL	Kod maszynowy.
2 GL	Assembler – operacje na rejestrach i pamięci.
3 GL	Języki wysokiego poziomu (COBOL, FORTRAN, PASCAL, C) – kod wysokiego poziomu, specyficzny dla danego języka programowania.
post 3 GL	Języki wysokiego poziomu zorientowane obiektowo.
4 GL	Języki problemowe (kod angielsko podobny): – języki baz danych (dBASE, Progress) – systemy zarządzania Bazami Danych.
post 4GL	Bezkodowe generatory aplikacji.

Uwieńczeniem ewolucji języków programowania jest koncepcja post 4GL – *bezkodowych generatorów aplikacji*, zorientowanych obiektowo. Przykładem takiego zaawansowanego narzędzia informatycznego jest produkt izraelskiej firmy MSE MAGIC. Odpowiada on w pełni wymaganiom inżynierii oprogramowania obiektowego (Object – oriented programming). Przypomnijmy, że jest to metoda implementacji, w działające zbiory obiektów, z których każdy jest przedstawicielem pewnej klasy. W metodzie tej wszystkie klasy są członkami hierarchii klas połączonymi przez związki dziedziczenia.

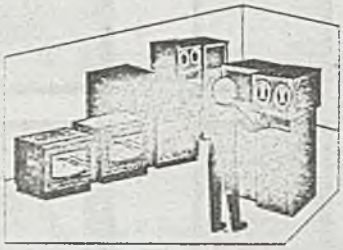
Tradycyjna metodyka tworzenia aplikacji (3 GL, 4GL) obejmuje etapy: analiza problemu – projektowanie – kodowanie, kompilacja, łączenie – uruchamianie i testowanie – gotowa aplikacja. Aplikacja powstaje jako rezultat żmudnej pracy zespołu projektowego (np. od kilku do kilkudziesięciu powtórzeń operacji: kodowanie – kompilacja – łączenie i testowanie). Tworzenie aplikacji za pomocą generatora (post 4GL) obejmuje etapy: analiza problemu – projektowanie, opracowanie prototypu, testowanie, modyfikacje – gotowa aplikacja. W ten sposób nastąpił wzrost efektywności projektowania: wzrost wydajności zespołu projektowego i skrócenia (6 – 10 razy!) czasu trwania procesu.

Dzięki językom "post 4 GL" – zorientowanym obiektowo i problemowo – pozwalającym na szybkie tworzenie aplikacji *bez pracochłonnego konstruowania algorytmów i kodowania*, użytkownik może skoncentrować się w pełni na projektowaniu przy automatycznej obsłudze szczegółów implementacyjnych.

Obecnie uważa się, że najbardziej znanym, zaawansowanym, rozwiniętym i w pełni profesjonalnym jest generator aplikacji relacyjnych baz danych MAGIC.

Hardware

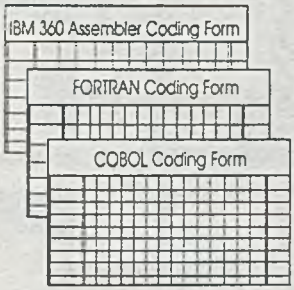
1960s



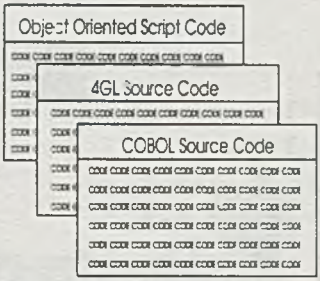
1990s

Software

1960s



1990s



The Facts - 1993

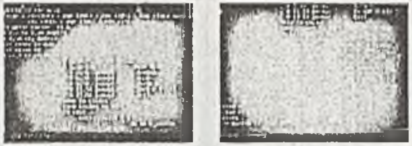
"... half of all current development undertaken is COBOL..."
"... 90% of all computer code in existence is written in COBOL (that's 78 billion lines!)"

(Source: DataPro International Rapport Newsletter, January 1993)

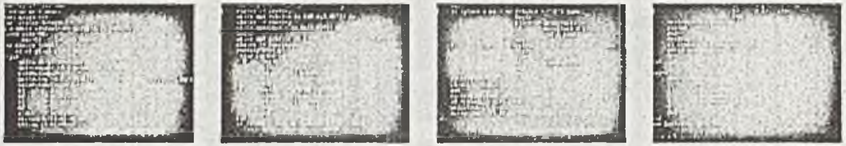
Magic's Object Oriented Code Free methodology eliminates the need for the massive volume of code found in conventional 3GL and 4GL programming. Establishing and designing menus, for example, takes just a handful of key strokes

Magic Table

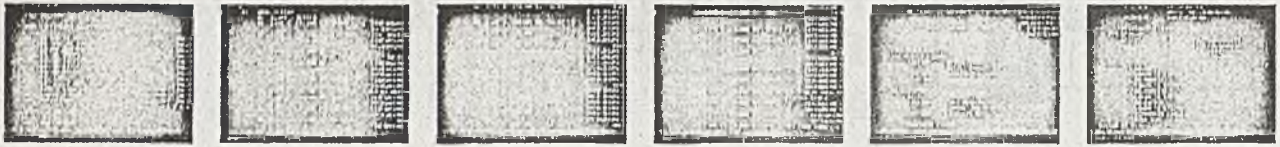
Conventional Database Language Code ►



Pascal Code ►



Cobol Code ►

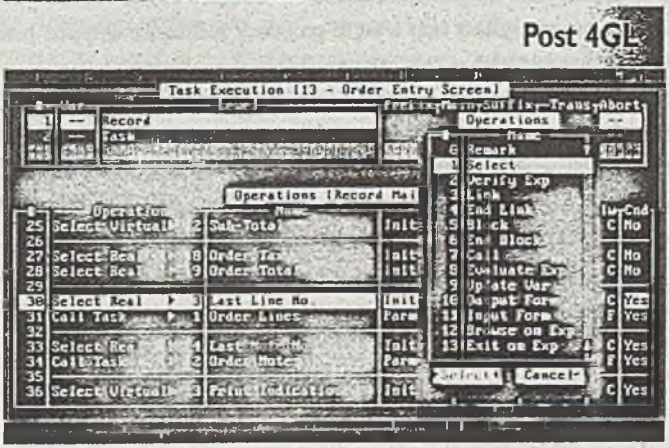


3GL

Database

4GL

Post 4GL



2. KONCEPCJA MAGIC'a

2.1. Uwagi ogólne

Magic jest pierwszym narzędziem programowym opracowanym dla zautomatyzowania procesu tworzenia aplikacji. Zastępuje on tradycyjne kody źródłowe tablicami proceduralnymi tworzącymi zintegrowane środowisko pracy. Ponieważ wszystkie programy MAGIC'a wykorzystują tablicowanie i są samodokumentujące, to są one podatne na modyfikacje, co zwiększa efektywność tworzenia aplikacji. MAGIC nie generuje programów w postaci wykonywalnego kodu (pliki. EXE), lecz stosuje maszynę wirtualną (engine) sterowaną danymi. Wszelkie informacje skojarzone z daną aplikacją są gromadzone jako dane w zintegrowanym słowniku, który można nazwać sterującą bazą danych. Maszyna czyta sterującą bazę danych i wykonuje odpowiednie, bardzo zaawansowane i kompleksowe operacje. Programowanie aplikacji polega nie tyle na tworzeniu od podstaw sterowania aplikacji, co na modyfikacji struktur standardowych, dostarczanych przez maszynę.

Tworzenie aplikacji w MAGIC'u składa się z dwóch podstawowych faz:

- *definiowanie danych i konstruowania ich struktur (bierne elementy aplikacji)*
- *tworzenie obsługujących je programów (elementy aktywne),*

MAGIC wyposaża od razu nowo tworzoną aplikację realizujących typowe operacje. W MAGIC'u nie ma listingu, jest natomiast wiele tablic i plansz, w których definiowane są właściwości programów i innych obiektów. Na aplikację składają się obiekty różnych kategorii: typy danych, bazy danych, środki do dialogu z użytkownikiem jak menu czy zlecenia klawiszowe, system help a także programy realizujące aktywne przetwarzanie danych. Maszyna wirtualna uniezależnia aplikację od konkretnej platformy. Pracuje ona w dwóch trybach: wykonawczym (runtime) i narzędziowym (toolkit). Programista ma kontakt z obydwoma trybami: w trybie narzędziowym tworzy i modyfikuje aplikacje w trybie wykonawczym – sprawdza ją i testuje. Użytkownik styka się wyłącznie z maszyną wykonawczą, realizując przeznaczoną dla niego aplikację. W ramach aplikacji istnieje wiele programów, dla których wspólnym elementem jest tylko baza danych, na której operują. Poza tym programy są od siebie niezależne. MAGIC traktuje programy jak "czarne skrzynki".

2.2. Cechy architektury

Architekturę MAGIC'a charakteryzują następujące podstawowe cechy:

a) nieproceduralne i niehierarchiczne rozwiązania:
- *nie wymagane jest predefiniowanie porządku tworzenia aplikacji,*
- *pozwala na swobodne przechodzenie pomiędzy definiowaniem struktury danych, konstrukcji zadań i formatek oraz menu,*
- *na dowolnym poziomie tworzenia aplikacji możliwa jest selekcja danych z listy dopuszczalnych opcji,*
- *można tworzyć menu, uruchamiać programy i aplikacje nawet wtedy, gdy zadania i dane pozostają częściowo zdefiniowane,*
- *modyfikacje następują automatycznie we wszystkich powiązanych ze sobą częściach programu.*

b) bezkodowe, obiektowo-zorientowane programowanie:
- *raz definiowane obiekty są łatwo wskazywane przez ich nazwę lub numer, bezpośrednio dostępne z dowolnego miejsca w aplikacji,*
- *model umożliwia spójną definicję i kontrolę danych , szybkie tworzenie aplikacji i bezpieczną, prostą konserwację,*
- *zadanie jest reprezentowane przez Tablicę Wykonawczą (z Tablicą Opcji),*
- *w której znajduje się wybrana sekwencja kroków programistycznych.*

c) szybkie i zautomatyzowane tworzenie aplikacji:
- *nie istnieją jawne komendy czy operacje (MAGIC na podstawie aktualnego trybu pracy końcowego użytkownika „wie". jaka operacja ma być wykonana).*
- *oferowane są narzędzia zwiększające produktywność (Automatyczny Generator Programów, Automatyczny Generator Formatek, Automatyczny Generator Raportów),*
- *użytkownik może zaprojektować bardziej wyrafinowane raporty użytkowe i modyfikować raporty, widok ogólny.*

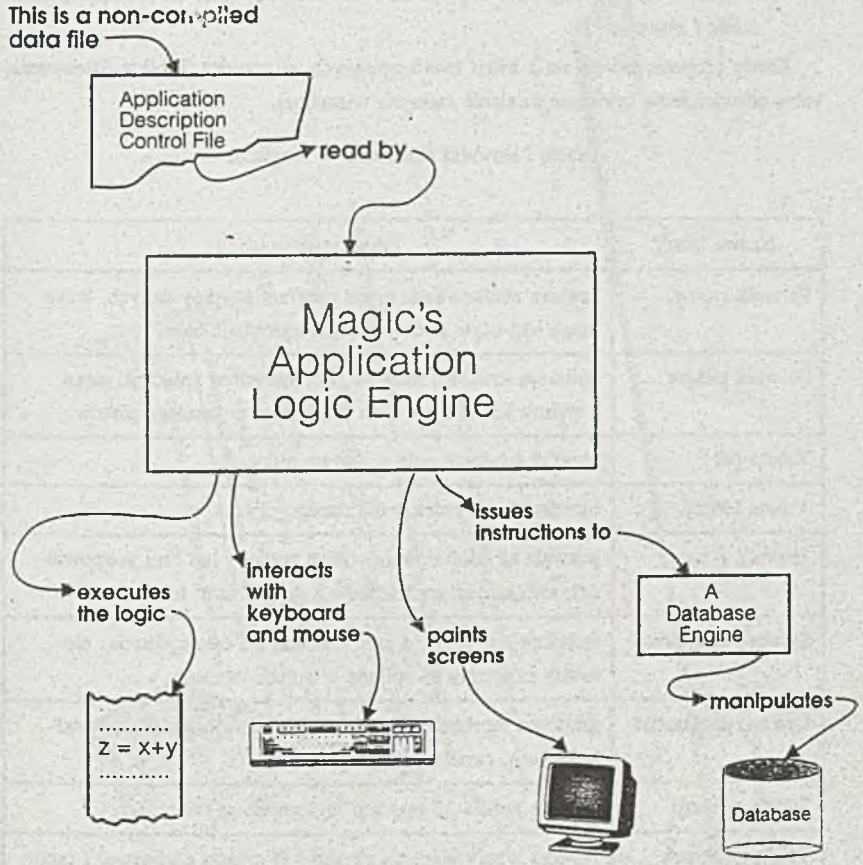
d) relacyjny model programowania:
- *dowolna relacyjna baza danych, może być przyjęta w celu manipulacji jej danymi.*

e) sterowanie przez menu:
- *programowanie jest zwykle kwestią prostych wyborów z listy dostępnych opcji i selekcją jednej z nich.*

f) maksymalnie elastyczny interfejs:
   - *ekrany i formatki mogą być automatycznie generowane lub ręcznie tworzone.*

g) niezależności od środowiska:
   - *możliwość szerokiego wyboru platform sprzętowych, systemów operacyjnych, baz danych, menedżerów zbiorów, menedżerów sieciowych (graficznych i tekstowych).*

h) architektura klient/serwer:
   - *wbudowana architektura klient/serwer, co pozwala na równoległy dostęp do tych samych danych i programów, nawet jeśli są znacznie oddalone,*
   - *czterowarstwowy system klienta:*
     - *interfejs użytkownika*
     - *jądro (steruje wykonywaniem aplikacji)*
     - *menadżer plików*
     - *bramy (bramy do baz danych i bramy komunikacyjne),*
   - *trzywarstwowy system serwera:*
     - *warstwa logiczna*
     - *menadżer plików*
     - *warstwy bram.*
   - *interoperatywność dzięki niezależności od platformy sprzętowej i baz danych (aplikacja może korzystać jednocześnie z różnych baz danych zlokalizowanych na różnych platformach).*


This is a non-compiled data file
Application Description Control File
read by
Magic's Application Logic Engine
executes the logic
interacts with keyboard and mouse
paints screens
issues instructions to
A Database Engine
manipulates
Z = X+y
Database

2.3. Metodyka

Programowanie polega na:

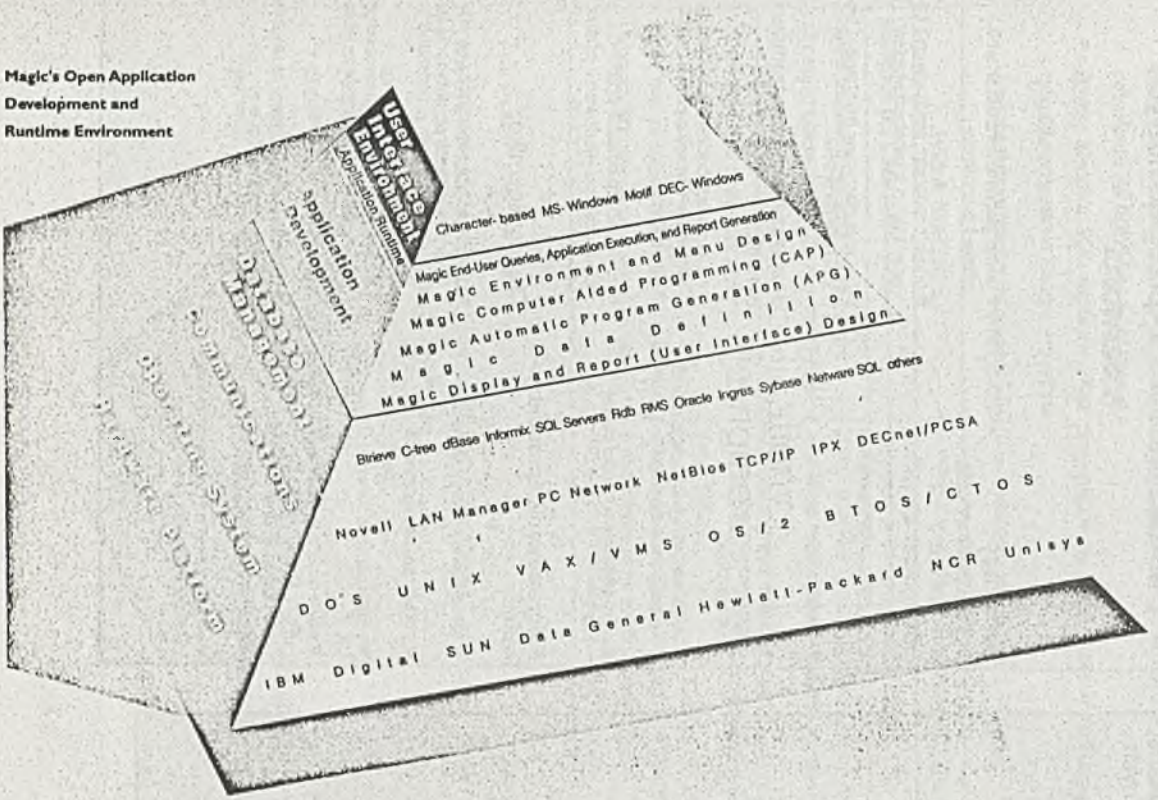
- *rozpoznaniu i opisie wymagań końcowego użytkownika,*
- *określeniu funkcji aplikacji i logiki programów (zadań),*
- *utworzeniu specyfikacji wysokiego poziomu za pomocą wypełnienia odpowiednich tablic i słowników.*

Każdy program składa się z kilku zadań opisanych za pomocą Tablicy Sterowania, która odzwierciedla cykliczne działanie maszyny wirtualnej,

Tabele i słowniki stosowane w Magic'u.

Nazwa tabeli	Przeznaczenie tabeli
Słownik typów	zawiera zdefiniowane przez programistę typy danych, które mogą być użyte w dowolnym miejscu aplikacji.
Słownik plików	definiuje strukturę zbiorów, używanych w aplikacji, wraz z typami RDBMSów i ich lokalizacją w katalogu plików.
Tabela pól	określa składowe pola w danym pliku.
Tabela kluczy	określa klucze-indeksy dla danego pliku.
Słownik pomocy	pozwala na zdefiniowanie okien pomocy lub linii podpowiedzi, związanych kontekstowo z programami lub polami.
Słownik programów	definiuje programy, z jakich składa się cała aplikacja: elementy programu są opisane w innych tabelach.
Drzewo programów	graficzna reprezentacja zależności między zadaniami i podzadaniami, tworzącymi program.
Tabela operacji	zawiera zestaw 13 operacji logicznych na rekordach.
Tabela formatek	zawiera wzory wyglądu ekranów (formatki ekranowe) i raportów dla każdego programu.
Tabela wyrażeń	definiuje arytmetyczne i logiczne wyrażenia używane w programie: składają się one ze zmiennych i wbudowanych funkcji.

Magic's Open Application Development and Runtime Environment



Obsługiwane bazy danych

- ADABAS,
- AS 400,
- Btrieve,
- C-ISAM,
- c-tree,
- Clipper,
- dBase,
- dBase IV,
- FoxBase,
- Informix,
- Ingres,
- MF Cobol,
- Microsoft SQL,
- Oracle,
- Paradox,
- Rdb,
- RMS,
- Sybase.

MAGIC wersja 5.6 - przegląd nowych możliwości

Dla twórcy oprogramowania:

Obsługa baz danych:

- dwufazowe potwierdzanie
- wbudowany język SQL
- funkcje słownika typów danych klient-serwer: ładowanie, sprawdzanie spójności
- wygodniejsze specyfikacje baz danych typu klient-serwer
- wprowadzenie wartości pustej NULL

Programowanie i produktywność:

- rozszerzona obsługa transakcji
- podwyższona wydajność
- współbieżność i wieloużytkowość
- nowa tablica atrybutów wydruku (Print Attributes Table) uniezależnia postać wydruku od użytej drukarki
- dziecięciokrotne zwiększenie pojemności słownika programów
- nowa, globalna kontrola słownika plików i programów
- nowy mechanizm wyszukiwania Locate/Search w tabelach Magica
- bardziej elastyczny mechanizm blokowania rekordów

Bezpieczeństwo programów i danych:

- zupełnie przebudowany system zabezpieczeń oparty o prawa użytkownika
- nowe zabezpieczenie oparte o klucz dostępu i kodowanie plików danych

Dla użytkownika:

- możliwość wprowadzania nowych rekordów w trybie przeglądania istniejących
- automatyczny wybór optymalnego klucza przy operacjach wyszukiwania
- przyrostowe wyszukiwanie rekordu w trybie zapytań (query)

3. WSPÓŁDZIAŁANIE

3.1. Bazy danych SQL

MAGIC obsługuje SQL w sposób:

- *niejawny: zlecenia SQL są generowane automatycznie przez MAGIC'a jako wynik odwołania do SQL-owej bazy danych;*
- *jawny: zlecenia SQL napisane przez programistę są wbudowane w aplikację na zasadzie wywołania innego programu.*

Podobieństwo między widokiem MIGIC'a a zbiorem wynikowym SQL'a.

Widok MAGIC'a	Wynikowy Zbiór Rekordów SQL'a
podzbiór pól i rekordów plików	podzbiór kolumn i rzędów tablic
może stanowić przekrój kilku plików, używając operacji LINK, tworzącej relację "jeden do jeden"	może stanowić przekrój kilku tablic, używając mechanizmu łączenie i klauzuli WHERE
może zawierać pola wirtualne	może zawierać pola wyliczane jako kolumny SQL'a
może być przedstawione użytkownikowi i uporządkowane w określony sposób	może być przedstawiony użytkownikowi i uporządkowany w określony sposób, używając klauzuli ORDER BY

3.2. Środowisko NetWare

MAGIC NLM zapewnia komunikację z nielokalnymi serwerami NetWare'u i udostępnia wszystkie ich zasady, a ponadto:

- *zmniejsza obciążenie sieci,*
- *przyspiesza przetwarzanie wsadowe,*
- *poprawia wydajność dostępu do danych o 100%,*
- *odciąża stacje robocze, co pozwala na uruchamianie dodatkowych zadań,*
- *pozwala na wstępne szeregowanie zadań i zapewnia zdalny dostęp do innych sieci.*

Specyfikacje Magic NLM*)
Składniki
Magic NLM Deployment (moduł wykonawczy) *Magic NLM Deployment Server (dostępny z późniejszymi wydaniami)*
Dedykowana brama baz danych
Magic Gate for Btrieve
Minimalna ilość użytkowników
5

*) *Wymaga Btrieve NLM firmy Novell. Magic NLM jest dostępny tylko w wersji wykonawczej (ang. runtime) serwera i nie zastępuje wersji wykonawczej Magic'a, wykonywanej na stacji roboczej.*

3.3. Środowisko UNIX

MAGIC funkcjonuje na wielu różnych UNIXowych platformach, systemach operacyjnych i systemów zarządzania bazami danych (DBMS). Pozwala użytkownikowi UNIXowych stacji roboczych, PC oraz VAX używać tych samych aplikacji i danych. Aplikacje tworzone w środowisku UNIXowym mogą być bez modyfikacji używane na PC i w sieciach lokalnych, a także, gdy jest to konieczne, aplikacje MAGIC'a mogą być wykonywane w DOSie, z danymi i programami zapisywanymi na dysku UNIXowym.

Specyfikacje dla systemu UNIX	
Elementy składowe	•Magic dla systemu UNIX •protokół TCP/IP
Platforma sprzętowe	•386/486 PC z uruchomionym systemem SCO UNIX •IBM RISC/6000 •Sun SPRAC •Hewlett Packard 9000 •Data General AViiON •Univel UnixWare •System V Wersja 4 •NCR 3000 •Olivietti •Silicon Graphic's
Obsługiwane bazy danych*	•c-tree, C-ISAM,Oracle, Informix SQL, Sybase.

* Pewne bramy mogą być wbudowane w system Magic, podczas gdy inne mogą być dostarczone na żądanie

3.4. SYNTHESIS

W ramach pakietu SYNTHESIS oferowany jest zaawansowany interfejs do generatora aplikacji relacyjnych baz danych MAGIC. Pozwala to użytkownikowi MAGIC'a na wykorzystanie pakietu SYNTHESIS do analizy i projektowania aplikacji. Wykonany i udokumentowany za pomocą SYNTHESIS'a projekt systemu można bezproblemowo przenieść w środowisko MAGIC'a i na odwrót – system utworzony w MAGIC'u można łatwo przenieść do SYNTEHESIS'a. System zaprojektowany za pomocą SYNTHESIS'a pozwala więc stworzyć podstawy aplikacji MAGIC'a – pozostaje do opisania logika programu.

4. ZASTOSOWANIA

Pełna lista referencyjna obejmuje ok.200000 użytkowników na całym świecie, w tym takich jak np. wszystkie rodzaje sił zbrojnych Armii Izraelskiej, elektrownie atomowe we Francji, IBM, COMPAQ, Hewlett Packard, ABB, JVC, NEC, SONY, Panasonic, Thompson, McDonald, Pepsi-Cola. Prezentację wybranych zastosowań przedstawiono w załącznikach. Wśród polskich użytkowników można wymienić: Bełchatów, Zespół Elektrowni Pątnów – Adamów – Konin, Elektrownia Kozienice, Bank Rozwoju Eksportu S.A., Bank Cukiernictwa oraz Wojskowy Instytut Informatyki i Akademię Obrony Narodowej.

W dalszej części przedstawiono ogólną charakterystykę zastosowań MAGIC'a dokonanych przez:

- *Centrum Informatyki AON:*
  - *wspomaganie planowania zajęć dydaktycznych;*
- *Wojskowy Instytut Informatyki:*
  - *ssystem gospodarowania zasobami WKU,*
  - *kontraktowanie kooperacji Przemysłowej "AGENCJA",*
  - *system ewidencji pojazdów "TRANSBEX".*

WYKORZYSTANIE GENERATORA APLIKACJI "MAGIC"
w Akademii Obrony Narodowej

Generator aplikacji "MAGIC" jest pakietem programów komputerowych przeznaczonych do tworzenia aplikacji (systemów komputerowych lub programów użytkowych), głównie w zakresie dużych baz danych, pracujących w sieciach komputerowych w systemach "klient-serwer". W porównaniu z innymi programami o podobnym przeznaczeniu, takimi jak "ORACLE", "INFORMIX" itp., wykorzystujące języki 4-tej generacji, posiada szereg zalet, do których jako najważniejsze można zaliczyć:

- 10-krotnie krótszy czas tworzenia aplikacji w porównaniu z innymi generatorami, dzięki temu, że wykorzystuje się mechanizm programowania poza językami czwartej generacji;
- możliwość użytkowania jednej i tej samej aplikacji na różnych platformach sprzętowych i programowych;
- łatwość modyfikacji aplikacji.

Pakiet oprogramowania generatora aplikacji "MAGIC" w wersji 5.2 na jeden komputer zakupiono w roku 1993. W roku 1994r. AON otrzymała od firmy "KOMTECH" wersję edukacyjną generatora aplikacji "MAGIC" wersji 5.6, przeznaczoną do użytkowania w sieci NOVELL. Wersja ta może być używana jedynie do celów edukacyjnych, nie można natomiast za jej pomocą tworzyć aplikacji "na sprzedaż" dla innych instytucji lub firm.

Do tej pory, w Akademii Obrony Narodowej, generator aplikacji wykorzystywano do następujących celów.

- do prowadzenia kilku odpłatnych kursów użytkowników generatora "MAGIC" w roku 1993;
- do prowadzenia zajęć na Kursie Zastosowań Informatyki oraz na Podyplomowych Studiach w Zakresie Analizy Systemowej i Informatyki;

- do opracowania i wdrożenia w AON programu do planowania zajęć dydaktycznych oraz zautomatyzowanego sporządzania rozkładów zajęć.

W przyszłości przewiduje się wykorzystanie generatora "MAGIC" do:

- rozbudowy programu do planowania zajęć dydaktycznych o elementy umożliwiające zarządzanie i ewidencjonowanie całokształtu działań organizycyjno-szkoleniowych w AON;
- tworzenia aplikacji z bazami danych do celów dydaktycznych oraz użytkowych z zakresu logistyki i dowodzenia wojskami;
- wykorzystanie do tworzenia aplikacji do wspomagania prac naukowo-badawczych;
- dalsze prowadzenie zajęć na kursach zastosowań informatyki oraz studiach podyplomowych z zakresu informatyki.

Komuterowy system gospodarowania zasobami WKU "SPIRALA-MU" powstał w dawnym Wojskowym Instytucie Informatyki. Został on zaimplementowany w środowisku genaratora aplikacji przetwarzania danych - MAGIC.

OPIS APLIKACJI

System ten przeznaczony jest do automatyzacji prowadzenia ewidencji, obiegu dokumentów i realizacji zadań bieżących związanych z działalnością Wojskowej Komendy Uzupełnień. Jest to system konwersacyjny, ukierunkowany na bezpośrednich użytkowników tj. komórki organizacyjne WKU.

Główne zadania systemu to:

- prowadzenie komputerowej ewidencji zasobów w trzech sekcjach WKU,
- realizowanie zadań związanych z działalnością poszczególnych sekcji WKU,
- automatyzacja obiegu dokumentów pomiędzy poszczególnymi sekcjami WKU,
- przesyłanie danych pomiędzy WKU a WSzW.

Ponadto koncepcja systemu zakłada mozliwość automatycznego przesyłania danych pomiędzy poszczególnymi poziomami, zgodnie ze strukturą organizacyjną odpowiednich organów.

System zakłada możliwość prowadzenia ewidencji na trzech poziomach:

1. Ogólnopolskim - Zarząd VII Szt. Gen.,
2. Wojewódzkim - Wojewódzki Sztab Wojskowy,
3. Rejonowym - Wojskowa Komenda Uzupełnień.

Dane podsystemu przechowywane są w bazie danych przechowywanej na twardym dysku komputera centralnego, oraz po zeskładowaniu, na zewnętrznych nośnikach magnetycznych takich jak dyskietki lub taśma magnetyczna.

Przetwarzanie w systemie odbywa się w trybie interakcyjnym - każdy uzytkownik systemu osobiście uczestniczy w jego użytkowaniu.

Oprogramowanie systemu umożliwia realizację m. in.: następujących zadań:

- utrzymywanie słownikowych danych stałych - poprzez zakładanie, przeglądanie, aktualizację i kasowanie danych w każdym słowniku;
- utrzymywanie opisu cech osobowych - poprzez wprowadzanie nowych zapisów, przeglądanie i aktualizację już istniejących oraz usuwanie zbędnych zapisów;
- utrzymanie danych o środkach transportu z gospodarki narodowej na uzupełnienie potrzeb mobilizacyjnych wojsk,
- drukowanie zestawień użytkowych - w postaci:

1) Raportów generowanych przez użytkowników systemu wdług dowolnych kryteriów na podstawie informacji znajdujących się w bazie danych,
2) Wydruków m.in.:Karty Powołania oraz Wezwania.

System umożliwia selektywny dostęp do poszczególnych funkcji użytkowych zależnie od funkcji użytkownika w systemie. Wybór odpowiedniego menu odbywa się automatycznie po wprowadzeniu nazwy użytkownika systemu i odpowiedniego hasła. Przyznanie użytkownikowi prawa dostępu do odpowiedniego menu (zakresu oprogramowania systemu) odbywa się drogą nadania użytkownikowi odpowiedniego numeru grupy użytkowników - podczas instalowania systemu u użytkownika.

Aplikacja Kontraktowania Kooperacji Przemysłowej "AGENCJA" dla Agencji Rozwoju Małych i Średnich Przedsiębiorstw powstała w dawnym Wojskowym Instytucie Informatyki przy użyciu generatora aplikacji MAGIC.

OPIS APLIKACJI

Sieć Agencji Kontraktowania Kooperacji SAKK jest wyspecjalizowanym systemem ukierunkowanym na problematykę kooperacji przemysłowej. Głównymi ogniwami systemu są Agencje Kontraktowania Kooperacji. Są one aktywnymi pośrednikami pomiędzy producentami wyrobów finalnych a podwykonawcami - potencjalnymi kooperantami. Podstawowym narzędziem w SAKK jest komputerowy bank informacji. Dane do banku informacji SAKK wprowadzane są z kwestionariusza SAKK. Kwestionariusz SAKK jest dokumentem zawierającym informacje o przedsiębiorstwie podporządkowane procesowi kojarzenia partnerów kooperacji przemysłowej. Integralną częścią bazy danych są niżej wymienione klasyfikacje:

- **EKD** - Europejska Klasyfikacja Działalności - do kodowania nazw rodzajów działalności społeczno-gospodarczej prowadzonej przez przedsiębiorstwo,
- **CN** - Scalona Nomenklatura Towarowa Handlu Zagranicznego - do kodowania nazw wyrobów finalnych, części składowych i materiałów produkowanych lub poszukiwanych przez przedsiębiorstwo,
- **KWK A (EWG-A)** - Klasyfikacja Wyrobów Kooperacyjnych - do kodowania nazw wyrobów kooperacyjnych,
- **KWK B (EWG-B)** - Klasyfikacja Operacji Technologicznych i Urządzeń Produkcyjnych - do kodowania nazw procesów technologicznych.

Obecna wersja aplikacji składa się z programów przeznaczonych do:

a) zakładania i aktualizacji bazy danych o firmach na podstawie danych zawartych w kwestionariuszach SAKK,
b) zakładania i utrzymania słowników kodów stosowanych w kontraktowaniu kooperacji (wg wymienionych powyżej klasyfikacji),
c) kontraktowania kooperacji (funkcje związane z wyszukiwaniem ankiet spełniających określone warunki).

Menu główne aplikacji składa się z następujących najważniejszych pozycji:

- **Słowniki** - zawiera zestaw funkcji służących do obsługi zbiorów słownikowych (dopisywnie, aktualizacja i usuwanie pozycji słowników),
- **Słowniki EWG** - zawiera zestaw funkcji służących do obsługi zbiorów słownikowych Terminologii Kontraktowania Kooperacyjnego,
- **Ankiety firm** - jest to menu, które zawiera pozycje:
  - **Wprowadzanie ankiety firmy,**
  - **Przeglądanie ankiet,**
  - **Aktualizacja ankiet,**
  - **Usuwanie ankiet,**
- **WYSZUKIWANIE** - jest najważniejszą funkcją procesu kojarzenia kooperacji. Daje ona możliwość zdefiniowania warunku wyszukiwania i wyszukania na jego podstawie ankiet firm, które spełniają wymagania tego warunku. Wyszukane ankiety zapisywane są na liście wybranych ankiet, którą użytkownik ma możliwość przeglądać. Wszystkie

umieszczone na tej liście ankiety można wydrukować lub wyeksportować (zapisać do pliku celem przesłania np. do innej agencji). Użytkownik ma też możliwość usunięcia ankiety z listy wyników jeśli po szczegółowym jej przejrzeniu uzna, że nie nadaje się ona do dalszej analizy jako ewentualny wynik kojarzenia kooperacji.

Opcji ta zawiera następujące pozycje:

a) Lista warunków - pozwala na przeglądanie i definiowanie warunków wyszukiwania, a następnie przeglądanie wyników poszukiwań.

b) Wyszukiwanie ankiet - powoduje bezpośrednie wejście do zdefiniowanych warunków z możliwością ponownego wyszukania (sprawdzenia dla aktualnej wersji ankiet).

c) Przeglądanie wyników - daje możliwość przeglądania listy ankiet - wyników wyszukiwania. Najważniejszą opcją jest "LISTA WARUNKÓW". Pozwala ona na przejście pełnego cyklu wyszukiwania, począwszy od zdefiniowania warunku poprzez przeglądanie wyników, aż do ich wydrukowania lub wyeksportowania.

Warunek składa się z podwarunków połączonych operatorami logicznymi : "i" , "lub".

Numeracja podwarunków i warunków jest automatyczna. Ilość podwarunków dla danego warunku nie jest programowo ograniczona. Przyjęta konwencja pozwala na budowanie warunków złożonych typu:

warunek nr1 = (podwarunek 1 lub podwarunek 2) i podwarunek 3, co oznacza, że możliwe jest sprawdzenie czy w bazie znajdują się firmy, które na przykład:

podwarunek1=prowadzą działalność w dziedzinie produkcji maszyn dla rolnictwa i leśnictwa (zakres działalności=29.3),

podwarunek2=produkują silniki spalinowe z zapłonem iskrowym (produkcja skatalogowana=8407 21),

podwarunek3=produkują kotły wytwarzające parę wodną płomiennorurowe (produkcja skatalogowana= 8402 19 90).

Znaczenie operatorów logicznych jest następujące :

a) "LUB" - podanie tego funktora przed warunkiem oznacza, że warunek będzie sprawdzony dla wszystkich ankiet zapisanych w agencji i numery ankiet spełniające ten warunek zostaną dopisane do listy wyników. Ten operator musi być ustawiony przed pierwszym podwarunkiem.

b) "I" - podanie tego funktora przed podwarunkiem (iloczyn logiczny) oznacza sprawdzenie warunku tylko dla ankiet umieszczonych na liście wyników i usunięcie z niej ankiet, które nie spełniają warunku. Sprawdzanie nie będize wykonywane dla listy pustej.

- **Administrator** - zawiera funkcje do administrowania systemem. Są to m.in.:
  - **Przeglądanie dziennika systemu** - pozwala na przeglądanie i usuwanie pozycji dziennika systemu. Dziennik systemu zawiera informacje o pracy systemu. Dla każdej wykonanej funkcji systemu przechowywane są w nim następujące informacje:
    a) Data - systemowa data rozpoczęcia wykonywania funkcji w systemie,
    b) Użytkownik - nazwa użytkownika, który wykonywał tę funkcję,
    c) Czas rozpoczęcia - systemowy czas rozpoczęcia wykonania funkcji,
    d) Czas zakończenia - systemowy czas zakończenia wykonania funkcji,
    e) Czynność (zadanie) - nazwa wykonanej funkcji.

System Ewidencji Pojazdów "TRANSBEX" został opracowany przez pracowników dawnego **Wojskowego Instytutu Informatyki** w Warszawie we współpracy z firmą "BEREX" **Sp. z. o. o.** z Warszawy.

OPIS APLIKACJI

Główne zadania opracowanego systemu to :

- prowadzenie ewidencji pojazdów będących w ruchu, na ewidencji , wycofanych czasowo i na stałe oraz zlikwidowanych;
- rejestrowanie pojazdów: po raz pierwszy,warunkowo(rej. próbna i czasowa),po zmianie właściwości miejscowej, po dopuszczeniu do ruchu po czasowym wycofaniu ,
- prowadzenie ewidencji zmian właścicieli i współwłaścicieli oraz zmian technicznych i w karcie pojazdu,
- Prowadzenie przychodu i rozchodu druków dowodów rejestracyjnych i tablic rejestracyjnych,
- Prowadzenie ewidencji wydanych zaświadczeń i otrzymanych potwierdzeń o ujęciu pojazdu na ewidencję,
- prowadzenie historii pojazdów (możliwość uzyskania pełnej informacji o poprzednich właścicielach pojazdu i historii zmian technicznych oraz zapisów w karcie pojazdu od momentu jego rejestracji Systemie),
- prowadzenie kroniki systemu (automatyczna ewidencja i kontrola wszelkich zapisów i zmian dokonanych w systemie wraz z rejestracją danych identyfikacyjnych dokonujących ich użytkownika),
- sprawozdawczość (12 zestawień w tym : Zestawienie T-10, zestawienie wydanych dow.rej, zestawienie wyd tablic rej),
- kontrola dostępu do aplikacji i do danych (oprócz systemu haseł i użytkowników system uprawnień pozwalający na upoważnianie użytkowników systemu do wykonywania odpowiednich jego funkcji).

System umożliwia prowadzenie bazy danych o pojazdach na trzech poziomach: wojewódzkim, rejonowym i punktów rejestracji pojazdów. Dane pomiędzy szczeblami mogą być przekazywane poprzez łącza modemowe z automatyczną modyfikacją zapisów w bazie wyższego szczebla lub na dyskietkach.

Dodatkowo system umożliwia POLICJI , STRAŻY GRANICZNEJ , Urzędom Celnym , Izbom Skarbowym i innym **upoważnionym** organom administracji państwowej oraz sprzedawcom pojazdów dostep do danych o pojazdach będących na ewidencji oraz o pojazdach i dokumentach skradzionych i poszukiwanych.

System oparto o obowiązujące standardy w zakresie słowników i kodów (katalog Instytutu Transportu Samochodowego, SI PESEL i REGON w zakresie ewidencji właścicieli, struktura zbiorów policji w zakresie bazy pojazdów i dokumentów poszukiwanych)

System "TRANSBEX" przeszedł pozytywnie testy kwalifikacyjne potwierdzające spełnianie wymogów stawianych przez Ministerswo Transportu i Gospodarki Morskiej, uzyskał odpowiednią homologację i jest na liście programów **rekomendowanych** przez to ministerstwo.

Opis Technologii

Zastosowana technologia spełnia następujące wymagania:

1. **Szybki i samodokumentujący** się proces wytwarzania aplikacji;
2. **Łatwość modyfikowania** i dostosowywania aplikacji do często zmieniających się uregulowań prawnych i potrzeb praktycznych oraz prostota w utrzymywaniu aplikacji po jej wdrożeniu;
3. **Zapewnienie przenaszalności** aplikacji i danych między różnymi :
   a) platformami sprzętowymi,
   b) systemami operacyjnymi,
   c) systemami relacyjnych baz danych;
4. Możliwość wytworzenia takiej aplikacji, która zapewniłaby **bezpieczne korzystanie z danych.**

Wygenerowany przy użyciu MAGIC'a Komputerowy System Ewidencji Pojazdów jest możliwy do uruchomienia , po nabyciu odpowiedniego do danej instalacji Run-Time MAGICA:

1. pod **DOS**'em w środowisku jednostanowiskowym (także graficznym pod **WINDOWS**), oraz pod **NOVELL**'em w wielostanowiskowym systemie sieciowym,
2. pod **UNIX**'em z wykorzystaniem komputerów jedno i wieloprocesorowych (np. WYSE, COMPAQ , ALR itd. zgodnie z zaleceniami obowiązującymi w administracji państwowej)
3. na komputerach VAX z systemem **VMS** i ALPHA firmy DEC.

Projektanci systemu zalecają stosowanie:

- na najniższych szczeblach (REP) instalacji DOS'owskich (w małych ośrodkach) i UNIX'owych tam gdzie zachodzi potrzeba jednoczesnej rejestracji danych na więcej niż jednym stanowisku;
- na szczeblach wyższych (WEP) systemów UNIX'owych
- na szczeblu centralnym systemu UNIX z wykorzystaniem komputerów wieloprocesorowych. Obecnie przeprowadzane jest testowane możliwości wykorzystanie komputera wieloprocesowego WYSE serii 7000 jako ewentualnego servera dla Centralnej Ewidencji Pojazdów.

Dobór wielkości pamięci i innych parametrów zależny jest od potrzeb lokalnych.

Opracowany system zapewnia automatyczne i bezproblemowe przeniesienie programów i danych między wymienionymi systemami. Pozwala to rozwijać system komputerowy wraz ze zwiększeniem ilości przetwarzanych danych i potrzeb użytkownika bez konieczności ponownego wydawania pięniędzy na wytworzenie i zakup programów i związaną z tym konieczność ponownego szkolenia personelu oraz zmiany przyjętych przez ludzi przyzwyczajeń. Z punktu widzenia użytkownika końcowego systemu praca i sposób korzystania z programu jest dokładnie taka sama przy wykorzystaniu każdego z w/w rozwiązań. Daje to możliwość jednolitego prowadzenia szkoleń w skali kraju i wykorzystywania raz przeszkolonych osób w wypadku ich przeniesienia do innego miejsca bez koniecznych przy dotychczasowych rozwiązaniach nakładów i opóźnień czasowych.

Kolejną cechą wynikającą z wyboru MAGICA jest szeroki zakres Systemów Relacyjnych Baz Danych jaki można wybrać do utrzymywania danych. Standardowo MAGIC korzysta :

z **BTRIEVE** pod DOS'em i

InfoViDE
partner w projektach klient/server

InfoViDE Sp. z o.o.
ul. Dembińskiego 10
01-644 Warszawa
tel. (48) (22) 33 83 65
faks (48) (22) 33 83 71

InfoViDE - wizytówka firmy

InfoViDE funkcjonuje na rynku informatycznym od 1991 roku. Stworzona została przez zespół pracowników Instytutu Informatyki Uniwersytetu Warszawskiego zajmujących się zagadnieniami Human - Computer Interaction. Początkowo firma specjalizowała się w tworzeniu aplikacji wykorzystujących siłę graficznych metod komunikacji z użytkownikiem - GUI.
W chwili obecnej misją InfoViDE jest propagowanie i wdrażanie nowoczesnych metod tworzenia systemów informatycznych. Od roku 1993 firma jest na polskim rynku przedstawicielem firmy Learmonth & Burchett Management Systems Plc - LBMS - dostawcy metodyk i narzędzi wspomagających tworzenie dużych projektów informatycznych. Chcąc oferować swoim klientom kompleksowe środowisko realizacji - nowoczesnych systemów klient/serwer, firma została w 1994 roku partnerem firmy Powersoft - producenta najlepszych narzędzi do konstruowania tego rodzaju systemów.

W ramach swojej działalności propaguje przemysłowe metodyki tworzenia systemów informatycznych, organizując seminaria i prezentacje, prowadząc szkolenia i kursy, publikując artykuły w prasie informatycznej.

Główne dziedziny działalności InfoViDE to: szkolenia z zakresu metod prowadzenia dużych systemów informatycznych, sprzedaż pakietu CASE firmy LBMS, konsultacje i pomoc we wdrażaniu nowoczesnej technologii informatycznej. W roku 1994 liczba zainstalowanych w Polsce stanowisk pakietu Systems Engineer przekroczyła 100, z oferty szkoleń do końca marca 1995 roku skorzystało ponad 300 osób z 50 organizacji z całej Polski.

Nasz strategiczny partner - Leamonth & Burchett Management Systems Plc

Firma LBMS powstała w roku 1977 stawiając sobie początkowo za cel świadczenie specjalistycznych usług informatycznych. W 1981 roku LBMS wygrał konkurs na standardową metodykę dla rządu Wielkiej Brytanii, czego wynikiem było powstanie Strukturalnej Metodyki Analizy i Projektowania Systemów" (ang. Structured Systems Analysis and Design Method - SSADM), którą wykorzystuje wiele organizacji publicznych na całym świecie. Linię produktów CASE - najpierw program, a później cały pakiet Systems Engineer wprowadzono do sprzedaży w 1990 roku. Od tamtej pory system rozwija się bardzo dynamicznie dostosowując się do pojawiających się nowych technologii informatycznych. W chwili obecnej LBMS ma roczne obroty wynoszące prawie 40 milionów dolarów i bazę danych wielu tysięcy użytkowników rekrutujących się spośród największych kompanii produkcyjnych, handlowych, rządowych i usługowych na całym świecie. Jest niekwestionowanym liderem na rynku metod, usług i oprogramowania CASE w Wielkiej Brytanii. Jednocześnie rośnie pozycja LBMS na rynku światowym. Ze względu na coraz silniejszą pozycję na rynku amerykańskim, główna siedziba firmy została przeniesiona do Huston w USA. Firma zatrudnia prawie 260 osób w 17 oddziałach w Wielkiej Brytanii, USA, Australii, Honk-Kongu, posiada partnerów w prawie wszystkich krajach Europy. LBMS podpisała wiele strategicznych porozumień z największymi dostawcami sprzętu i oprogramowania, między innymi z Gupta Technologies, Microsoft, Powersoft, Sybase, Ingres, Informix, ICL, Digital Equipment Corporation, NCR i Unisys.

LBMS Systems Engineer i Process Engineer

Systems Engineer, produkt firmy LBMS jest więcej niż zintegrowanym pakietem CASE przeznaczonym do pracy grupowej. Pomyślany jako zestaw programów działających w środowisku MS Windows zapewnia wspomaganie prac projektowych na wielu poziomach - od zarządzania, poprzez proces analizy i projektowania, po generowanie aplikacji i produkcję wysokiej jakości raportów i analiz.

Podstawowe cele, jakie postawiono sobie tworząc pakiet Systems Engineer to:

- elastyczne dopasowanie metodyki i narzędzi,
- wspomaganie pełnego cyklu rozwojowego systemu,
- kierowanie projektem i zarządzanie cyklem projektowym,
- zachowanie spójności projektu w środowisku wielu użytkowników,
- integracja z aplikacjami środowiska Windows

Process Engineer jest narzędziem wspomagającym Process Management. Dostarcza bogatą bibliotekę metodyk do prowadzenia różnego typu projektów informatycznych. Na podstawie danych z repozytorium SE, możliwe jest oszacowanie złożoności projektu, opracowanie planu projektu i kontrola jego wykonania. PE pozwala na modyfikację standardowych cykli projektowych na potrzeby konkretnych projektów. Wyjątkowość zestawu narzędzi dostarczanych przez LBMS polega na wspomaganiu całego procesu planowania, produkcji, dokumentowania i utrzymania systemów informatycznych.

LBMS efektywnie generuje kod dla większości "przemysłowych" baz danych i integruje się z produktami do tworzenia aplikacji klienckich takich jak, np. PowerBuilder, Visual Basic, SQL Windows. Praca w systemie MS Windows pozwala na pełną integrację narzędzi LBMS z pakietami tego środowiska. Dokumenty projektowe mogą być tworzone za pomocą takich narzędzi jak MS Word, czy MS Project korzystającymi na bieżąco z repozytorium LBMS poprzez mechanizmy DDE i OLE. Takie podejście umożliwia łatwe generowanie pełnej dokumentacji na każdym etapie projektu.

Nasz strategiczny partner - Powersoft

Na podstawie umowy podpisanej w grudniu 1994 roku firma InfoViDE, jako PowerChannel Partner, rozpoczęła współpracę z firmą Powersoft otrzymując prawa do dystrybucji pakietu PowerBuilder - pełnego środowiska do generowania wysokiej jakości aplikacji typu klient/serwer pracujących w systemie Windows 3.1™ i NT™. Powersoft - lider na rynku narzędzi do tworzenia aplikacji w środowisku klient/serwer, jest obecnie organizacją międzynarodową, posiada swoich przedstawicieli w 77 krajach świata. Jako organizacja młoda (powstała w 1991 roku), pochwalić się może zaskakującymi wynikami rynkowymi. Jej dochód w 1994 wzrósł o 131 % w stosunku do roku poprzedniego i wyniósł 132 miliony dolarów. Firma jest laureatem wielu nagród, między innymi Nagrody Czytelników Data Based Advisor na najlepsze narzędzie do tworzenia interfejsu użytkownika, Nagrody Głównej National Software Testing Labs i Software Digest, Nagrody Wydawców PC Magazine za najlepsze narzędzie do tworzenia aplikacji graficznych i wiele innych. Tak nieprawdopodobny sukces firma zawdzięcza faktowi, że w dobie rozwoju zorientowanej obiektowo technologii, kiedy złożone aplikacje klient/serwer wymagają coraz silniejszych i coraz bardziej elastycznych narzędzi, swoją ofertą zaspokaja potrzebę posiadania efektywnego środowiska pracy, wspomagającego projektowanie, począwszy od zarządzania pracą zespołu, po tworzenie funkcji dla użytkownika końcowego.

PowerBuilder Enterprise

PowerBuilder Enterprise jest środowiskiem narzędziowym do realizacji aplikacji w architekturze klient/serwer. Ukierunkowany jest przede wszystkim na tworzone zespołowo, wydajne aplikacje w pełni wykorzystujące nowoczesną trójwarstwową

architekturę rozproszoną (a więc zarówno serwery danych jak i serwery procesów) i architekturę 32-bitową procesorów Intel dla platform takich jak Windows 3.1, Windows NT i Win OS/2. Faktem jest, że ponad jedna trzecia aplikacji stworzonych tym narzędziem kwalifikowana jest jako Critical Transaction Processing (dane International Data Corp. z 1994)

Istnieją też warianty środowiska dla mniej wymagających użytkowników, wykorzystujące prostsze warianty architektury klient/serwer, protokoły ODBC, czy lokalne, jednostanowiskowe serwery baz danych.

InfoViDE
dostawca metodyki i wydajnych narzędzi

LBMS - metodyka i narzędzia

Metodyka projektowania oparta na standardach strukturalnych i dostosowana do profilu działalności producenta oprogramowania staje się powoli niezbędnym kanonem pracy poważnych firm zajmujących się tworzeniem systemów informatycznych. Systematyczny, zdyscyplinowany cykl rozwojowy ze ściśle zdefiniowanymi standardami produktów, dokumentacji i organizacji pracy pozwala na wypracowanie powtarzalnego procesu produkcyjnego, w którym udane wdrożenie dużego systemu nie jest przypadkowym wydarzeniem, ale stanowi przewidziany rezultat dobrze zaplanowanych działań o możliwych do oszacowania kosztach.

Cykl projektowy to proces przekształcania informacji - od notatek pochodzących z wywiadów z użytkownikami począwszy, a kończąc na gotowym produkcie z aktualną dokumentacją techniczną pozwalającą na pielęgnację i rozwój. Taki proces wymaga specyficznych narzędzi. Z jednej strony konieczna jest integracja dokumentacji w cyklu projektowym zapewniająca klarowną wizję kolejno podejmowanych decyzji. Z drugiej strony, faza implementacji systemu wymaga zastosowania narzędzi dostosowanych w optymalny sposób do docelowej architektury systemu. Naprzeciw takim właśnie wymaganiom wychodzi propozycja firmy LBMS.

Narzędzia dla osób zarządzających projektem pozwalają na wybranie cyklu projektowego i dostosowanie go do specyfiki zespołu. Szacowanie pracochłonności, oparte na różnego rodzaju metodach estymacji, można dostroić do faktycznych wyników osiąganych przez zespół. Pozwala to na stopniowe, kontrolowane wprowadzanie metodyki i mierzenie rezultatów tego wdrożenia.

Narzędzia dla analityków pozwalają na elastyczne tworzenie dokumentacji projektowej, przy zachowaniu logiki powiązań informacji charakterystycznej dla zagadnienia lub wynikającej z przyjętej, specyficznej metody analizy.

Dokumentacja koncepcyjna, uzupełniana o dane projektowe związane z wybraną platformą implementacji, pozwala na sprawne konstruowanie aplikacji z wykorzystaniem generatorów, składaczy aplikacji i graficznych edytorów modułów dostępu do danych.

Możliwość wprowadzania informacji projektowej do narzędzi konstrukcyjnych innych producentów pozwala na wybranie optymalnego narzędzia implementacji przy zachowaniu standardów dokumentacji projektowej i kontroli zgodności specyfikacji z faktyczną realizacją.

Podstawowe komponenty pakietu Systems Engineer

Systems Engineer Tool Set - bogaty zestaw narzędzi wspomagających analizę i projektowanie systemów informatycznych, ze specjalnym uwzględnieniem systemów klient/serwer. Przeznaczony dla wieloosobowych zespołów informatycznych działających w średnich i dużych organizacjach.

SE/Repository - wielodostępna, relacyjna baza danych projektu przygotowywanego przy użyciu SE. Pozwala na równoległy dostęp wielu użytkowników do informacji zgromadzonej w bazie danych projektu. SE/Repository pozwala, między innymi, na dynamiczny dostęp do danych za pomocą kwerend SQL i generatora raportów. Daje także możliwość integracji z wieloma powszechnie dostępnymi aplikacjami Windows, takimi jak edytory tekstu, arkusze kalkulacyjne, poczta elektroniczna, etc.

SE/Workplace - wyrafinowany generator raportów, pozwalający na integrację pakietu Systems Engineer ze środowiskiem biurowym Windows - z programami Word for Windows, Excel, etc. Dzięki mechanizmowi DDE można tworzyć dokumenty zawierające "żywe" odwołania do bazy danych projektu uaktualniane zawsze, kiedy zachodzi potrzeba wydrukowania kolejnego raportu.

SE/Server Builder& Reverser - generator schematów serwerów danych, np. definicji indeksów, praw dostępu do danych, więzów integralności, wyzwalaczy (ang. triggers), procedur wbudowanych (ang. stored procedures), etc., generowanie odbywa się na podstawie informacji projektowej zawartej w repozytorium Ponadto pozwala na szacowanie przyszłych rozmiarów baz danych i definiowanie ich organizacji (ang. storage logic), a także na pełny reengineering już istniejących baz danych.

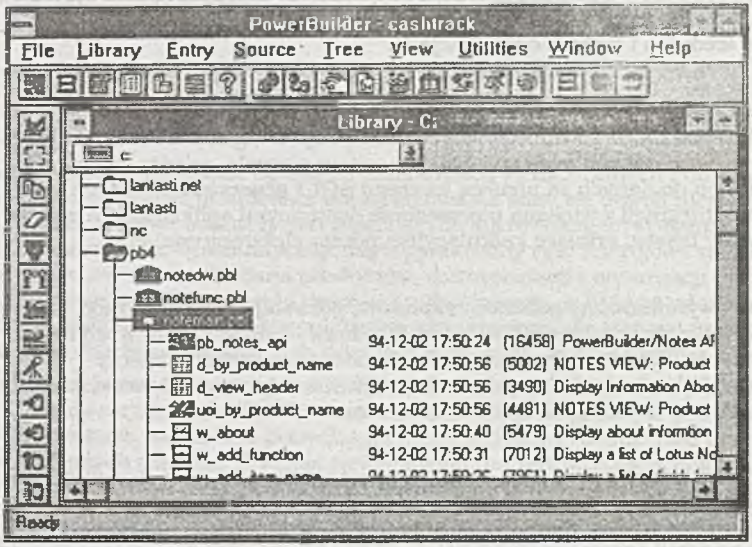
SE/Open for PowerBuilder, SE/Open for Visual Basic - narzędzie integracji danych środowiska SE i pakietów PowerBuilder i Visual Basic. Integrują środowisko analizy i projektowania pakietu Systems Engineer z pakietami PowerBuilder firmy Powersoft lub Visual Basic firmy Microsoft. Środowiska te mogą być wykorzystywane jako narzędzia do prototypowania i konstruowania aplikacji analizowanych i projektowanych za pomocą zestawu Systems Engineer. SE wzbogaca te pakiety o możliwości kontroli przebiegu prototypowania, kontroli wersji obiektów i przechowywania ich obiektów (formularzy, bibliotek, modułów) w wielodostępnym repozytorium SE.

Oprócz wymienionych, istnieją jeszcze moduły pozwalające na integrację środowiska Systems Engineer z takimi produktami, jak SQL Windows lub Informix New Era.

PowerBuilder Enterprise

Interakcyjne tworzenie aplikacji

Do tworzenia komponentów aplikacji wykorzystywane jest obiektowe środowisko implementacji - PowerBuilder for Windows. Pozwala ono na tworzenie za pomocą graficznych edytorów ("painterów") bibliotek komponentów aplikacji, takich jak okna, menu, obiekty manipulujące na bazach danych, funkcje, struktury danych i obiekty dialogowe. Z komponentów zawartych w takich bibliotekach korzystają aplikacje, mogące współdzielić różnego rodzaju obiekty - od obiektów związanych z interakcją z użytkownikiem, po struktury danych.



Logika działania obiektów kodowana jest za pomocą rozszerzalnego języka czwartej generacji o nazwie PowerScript. PowerBuilder spełnia oczekiwania wymagających miłośników obiektowości, umożliwiając wielopoziomowe dziedziczenie, przesyłanie komunikatów między obiektami, hermetyzację atrybutów i metod klas i obiektów.

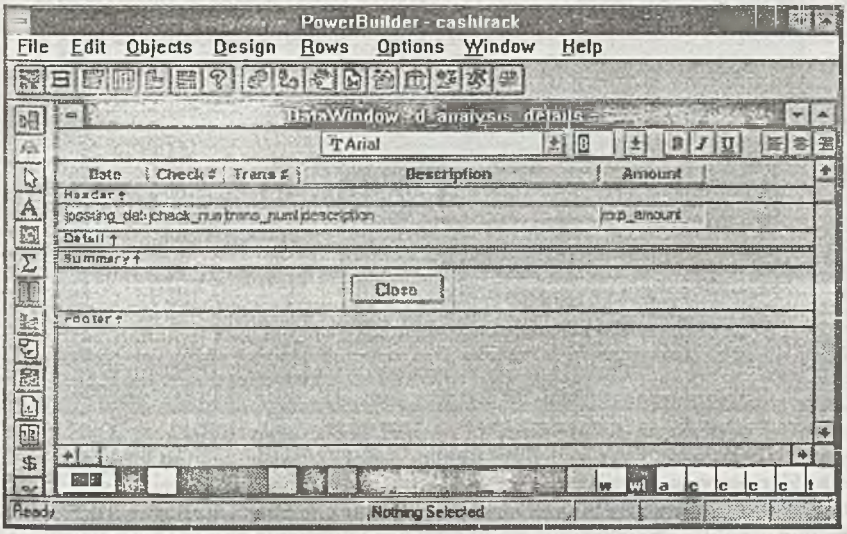
Konstruując warstwę dialogu z użytkownikiem, można zastosować w aplikacji wszelkie mechanizmy interakcji charakterystyczne dla Windows lub Windows NT, czy też generalnie - dla normy dialogu Common User Access. W szczególności PowerBuilder wspomaga tworzenie aplikacji wielodokumentowych (MDI), wspomaga bezpośrednią interakcję (ang. direct manipulation) i typowe udogodnienia w postaci pasków narzędzi, czy rozbudowanych systemów pomocy.

W ramach pakietu PowerBuilder Enterprise dostarczany jest jeden z najlepszych na rynku optymalizujących kompilatorów języków C i C++ - produkt firmy Watcom. Dzięki temu możliwe jest tworzenie własnych klas obiektów dla PowerBuildera (na

przykład implementujących kontakt z monitorami transakcyjnymi lub wykorzystujących nietypowe urządzenia peryferyjne).

Komunikacja ze źródłami danych

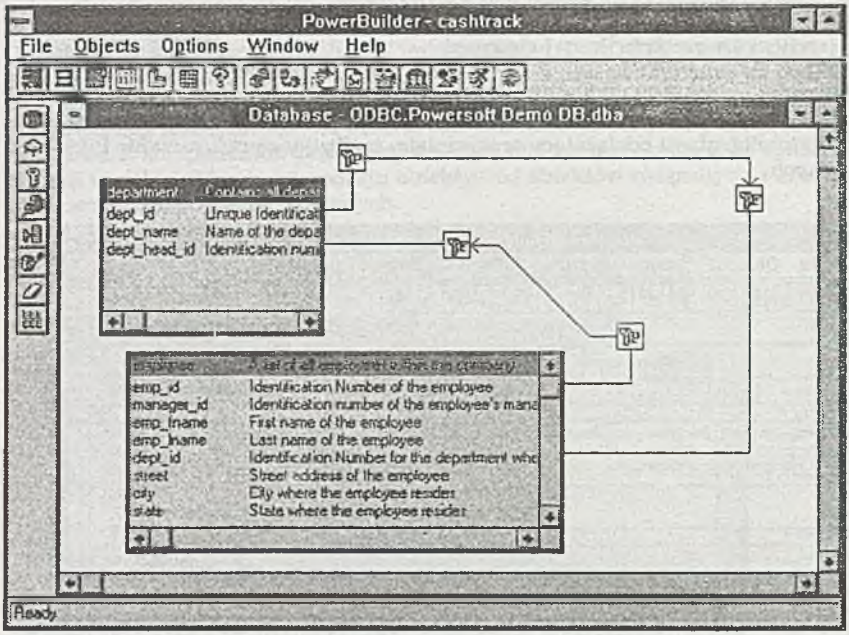
Podstawowymi obiektami do komunikowania się ze źródłami danych są w pakiecie PowerBuilder tak zwane okna danych (DataWindow™). Są to obiekty interakcyjne, w sposób inteligentny i odciążający programistę, realizujące przetwarzanie informacji z bazy danych.



Rozpraszanie aplikacji

Do łączenia się ze zdalnymi źródłami danych wykorzystywane są różnego rodzaju mechanizmy. Dla większości dostępnych na rynku baz danych Powersoft dostarcza dedykowane sterowniki (native database conectivity drivers). Dzięki własnym, dedykowanym sterownikom PowerBuilder omija pułapkę niskiej wydajności rozwiązań klient/serwer. Dla mniej popularnych baz, lub dla baz danych klasy desktop (xBase, Access, arkusze Excela, czy nawet pliki tekstowe) dostępne są sterowniki ODBC. Jednak dla obiektów DataWindow źródłem danych może być również inny obiekt - poprzez mechanizmy dynamicznej wymiany danych (OLE 2.0, DDE) po zdalne wywołanie procedur (RPC) w obiektach umieszczonych na różnych węzłach sieci. Obiekty PowerBuildera mogą być zarówno klientami jak i serwerami tych protokołów.

Graficzne konstruowanie struktury baz danych



PowerBuilder pozwala na definiowanie za pomocą graficznego edytora struktury relacyjnej bazy danych, wraz z więzami integralności i wieloma typami kluczy. PowerBuilder pozwala określać rozszerzone atrybuty kolumn w tablicach baz danych pokazujące sposób prezentacji oraz metodę weryfikacji wartości atrybutu. Te rozszerzone atrybuty wykorzystane są do szybkiego tworzenia spójnych od strony użytkowej obiektów do prezentacji i manipulowania informacjami z baz danych. Podobne graficzne mechanizmy uławiają i automatyzują tworzenie kodu dostępu do danych (SQL).

Migracja i replikacja danych

Elementem środowiska PowerBuilder Enterprise 4.0 jest pakiet InfoMaker. Jest to narzędzie dla użytkowników końcowych. Pozwala na łatwe tworzenie lub modyfikowanie zestawień, analiz i raportów realizowanych na podstawie źródeł danych lub ich, automatycznie tworzonych, lokalnych kopi.

InfoMaker pozwala też na łatwe tworzenie formularzy ekranowych umożliwiających modyfikowanie danych przez użytkowników końcowych. Wszystkie te możliwości są pod ścisłą kontrolą, zarówno pod kątem autoryzacji praw dostępu do danych jak i pod kątem wytwarzanego obciążenia dla serwera danych.

Zarządzanie relacyjnymi bazami danych

Pakiet PowerBuilder Enterprise wyposaża użytkownika w serwer SQL firm Watcom w wersji lokalnej. Wersja 4.0 tego serwera jest pierwszym produktem tego typu, który implementuje standard ANSI dotyczący procedur wbudowanych i triggerów. Poza tymi dynamicznymi mechanizmami, pozwala też na definiowanie statycznych więzów integralności referencyjnej. Serwer firmy Watcom pozwala na definiowanie dwukierunkowych, modyfikowalnych kursorów. Zawiera mechanizmy auto-optymalizacji (ang. self-tuning) i optymalizacji zapytań.

Integracja z pakietami CASE

PowerBuilder pozwala na wykorzystanie do tworzenia aplikacji i zarządzania konfiguracją projektu narzędzia CASE Systems Engineer firmy LBMS. Integracja pomiędzy pakietem Systems Engineer, a pakietem PowerBuilder obejmuje automatyczne tworzenie specyfikacji, transfer definicji baz danych, ich atrybutów, projektowanie i konstruowanie standardów interakcji i okien docelowej aplikacji.

Komponenty pakietu

- PowerBuilder (środowisko konstruowania aplikacji)
- Kompilator C/C++ firmy Watcom, wersja 10
- Advanced Developer Toolkit (biblioteki obiektów i narzędzia przydatne do profesjonalnej realizacji aplikacji)
- Serwer SQL firmy Watcom, wersja 4.0
- Configuration Management - interfejsy do narzędzi CASE (Systems Engineer, Endeavor, PVCS)
- InfoMaker

Główne zalety

- Kontakt z większością baz danych poprzez dedykowane sterowniki
- Możliwość wykorzystania protokołów RPC, OLE 2.0, DDE i prawdziwie trójwarstwowej architektury klient/serwer
- Graficzne narzędzia konstrukcyjne o spójnych mechanizmach obsługi
- Możliwości pracy grupowej i wielokrotnego wykorzystania komponentów aplikacji
- Integracja z narzędziami CASE

Wymagania sprzętowe (minimalne)

386SX, MS-DOS™ lub PC-DOS™ (wersja 3.3 lub wyższa), 8 MB RAM, Microsoft Windows 3.1 lub Windows NT, 19 MB przestrzeni dyskowej.

Obsługiwane serwery baz danych:

ALLBASE/SQL, DB2, DB2/2, DB2/6000, Informix, Microsoft SQL Server, Oracle, SQLBase, Sybase SQL Server, WATCOM SQL, XDB, Rdb.

Inne bazy danych dostępne poprzez protokół ODBC i sterowniki ODBC dostarczone przez producentów DBMS lub inne firmy.

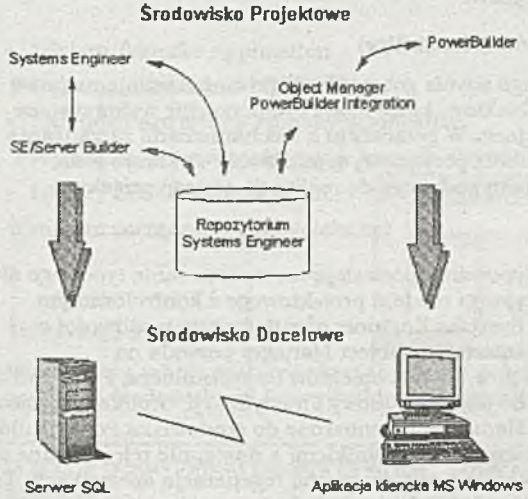
Obsługiwane serwery klasy Desktop

Paradox® for DOS 3.x, 4.x, 5.x, dBASE III, IV, V, Paradox for Windows, dBASE for Windows, NetWare SQL 3.x, Clipper Summer 87, 5.x, Access 1.x, FoxPro® for DOS 2.0, Btrieve 5.x, 6.x, FoxPro for Windows, ASCII text, Microsoft Excel.

Integracja LBMS Systems Engineer z pakietem PowerBuilder

Architektura Systemu

Integracja pakietu Systems Engineer z systemem PowerBuilder realizowana jest za pomocą komponentu pakietu o nazwie Object Manager. Object Manager sprawia, iż PowerBuildera staje się naturalnym narzędziem konstrukcyjnym pakietu, i odwrotnie - Systems Engineer stanowi naturalne środowisko projektowania, generowania, składowania i dokumentowania aplikacji PowerBuildera.



Systems Engineer dostarcza narzędzi do analizy wymagań, projektowania aplikacji i serwera. SE/Server Builder generuje definicje SQL DDL, kod triggerów oraz procedur wbudowanych dla serwera. PowerBuilder wspomaga swoim środowiskiem tworzenie interfejsu użytkownika aplikacji klienckiej. Wszystkie obiekty tworzone za pomocą tych narzędzi są przechowywane i zarządzane za pomocą systemu Object Manager, który umożliwia obszerną i elastyczną kontrolę nad wersjami i dostępem do obiektów.

Obszary funkcjonalne

Object Manager realizuje trzy fundamentalne funkcje o podstawowym znaczeniu dla rozwoju aplikacji w wieloosobych zespołach projektowych.

Zarządzanie obiektami

- Zarządzanie obiektami projektu- kontrola wersji, wariantowy rozwój aplikacji - w scentralizowanym, wielodostępnym repozytorium. Zarządzanie obiektami obejmuje obiekty PowerBuildera, modele analityczne i projektowe a także inne dokumenty stowarzyszone - plany projektu, raporty, dokumenty użytkownika itp.
- Koordynacja dostępu do komponentów tworzonych aplikacji poprzez kontrolę uprawnień i haseł członków projektu.

Ponowne wykorzystanie obiektów (reusability)

Udogodnienia dotyczące ponownego użycia gotowych obiektów pozwalają na łatwe modelowanie i przechowywanie obiektów, które mogą być ponownie wykorzystane i rozbudowywane w wielu aplikacjach. W połączeniu z mechanizmami zarządzania obiektami daje to organizacji i grupom projektowym możliwość wykorzystania istniejących elementów systemu jako podstawy do realizacji nowego projektu.

Prototypowanie

Mechanizmy wspomagania prototypowania pozwalają na zastosowanie typowego dla architektury klient/serwer, iteracyjnego modelu projektowego z kontrolowanym, interakcyjnym prototypowaniem. Systems Engineer oferuje bogate możliwości w zakresie projektowania struktury interfejsu, Object Manager pozwala na wygenerowanie prototypów aplikacji w postaci obiektów PowerBuildera, PowerBuilder zaś dostarcza wygodne narzędzia do jego rozbudowy i modyfikacji. Aplikacje projektu mogą być więc za pomocą Object Managera przenoszone do środowiska PowerBuildera w celu interakcyjnego prototypowania z użytkownikiem, a następnie rejestrowane z powrotem w repozytorium, z jednoczesną automatyczną regeneracją specyfikacji. Te unikalne mechanizmy pozwalają w pełni wykorzystać skuteczność prototypowania, z jednoczesnym utrzymaniem spójności produktu i jego dokumentacji.

Zarządzanie Słownikami

Zarządzanie słownikami umożliwia połączenie pomiędzy słownikiem danych przedsięwzięcia przechowywanym w Repozytorium Systems Engineer a słownikami typowymi dla każdej aplikacji PowerBuildera. Ta cecha stanowi centralny punkt dla definicji i kontroli każdego komponentu aplikacji i pozwala na utrzymanie spójności pomiędzy wieloma aplikacjami.

Elementy Oprogramowania

System PowerBuilder Integration zawiera trzy główne funkcje:

- Library Transfer Application

(Program do transferu bibliotek)

- Extended Attributes Transfer Application

(Program do przenoszenia rozszerzonych atrybutów)

- Library Services Application.

(Program zarządzający bibliotekami)

Integracja z PowerBuilderem

Następująca tabela opisuje zakres integracji pakietów i wzajemną odpowiedniość pomiędzy środowiskami PowerBuilder i Systems Engineer.

Obiekt w repozytorium pakietu Systems Engineer	Obiekt środowiska PowerBuilder
Specyfikacja okien wraz ze strukturą wywołań okien (Window Nawigation Chart)	Okna aplikacji
Specyfikacja klas wraz ze strukturą dziedziczenia	Okna biblioteki definiującej standardy interakcji
Model Danych	Struktura bazy danych
Forma prezentacji atrybutów (jako General Form klasy Extended Attribute)	Extended Attributes
Moduły dostępu do danych	Obiekty Query (moduły SQL)
General Form różnych klas odpowiadających obiektom PowerBuildera	Data Window, Function, Data Pipeline, itp.

Oracle Power Objects — obiektowe narzędzie do tworzenia aplikacji

Tomasz Traczyk

Opisano nowe narzędzie Oracle Power Objects[1], służące do tworzenia aplikacji obsługujących bazy danych, przeznaczone dla użytkowników komputerów osobistych. Przedstawiono możliwości narzędzia, sposób tworzenia aplikacji i połączenia z bazami danych, język Oracle Basic oraz cechy wbudowanej bazy danych.

1. Wprowadzenie

Nazwę Oracle kojarzono dotychczas z oprogramowaniem profesjonalnym przeznaczonym dla dużych projektów i wielkich organizacji. Ostatnio korporacja Oracle zainteresowała się jednak także rozwijającym się rynkiem oprogramowania dla grup roboczych i dla użytkowników indywidualnych. Dowodem jest strategia Oracle 2000, która zakłada dostarczanie narzędzi i serwerów baz danych skalowalnych od pojedynczego użytkownika aż po wielkie projekty.

Taka strategia wymaga baczniejszego skupienia uwagi na środowisku komputerów personalnych, najczęściej używanym przez użytkowników indywidualnych i małe grupy robocze. Dla takich użytkowników przeznaczone jest narzędzie Oracle Power Objects.

Oracle Power Objects (OPO) jest nowym narzędziem służącym do budowy aplikacji działających na komputerach PC lub Macintosh.

Za pomocą OPO można tworzyć aplikacje pracujące w środowisku graficznym, złożone z formularzy i raportów. Aplikacje mogą pracować po stronie klienta w architekturze klient-serwer i korzystać z relacyjnych serwerów danych Oracle, Sybase i Microsoft SQL Server. Przewidziano możliwość czytania i zapisywania danych w popularnych formatach baz danych i arkuszy kalkulacyjnych dla komputerów osobistych. Handlowa wersja programu będzie zawierała także lokalną bazę danych zwaną Blaze, przeznaczoną do tworzenia małych baz danych bez konieczności zakupu kosztownego serwera.

Narzędzie OPO jest proste w obsłudze, przeznaczone dla użytkownika nie mającego głębokiej wiedzy o bazach danych ani o języku SQL.

Opisano podstawowe cechy i możliwości OPO. Opis ogólny niekiedy uzupełniono szczegółami, chcąc czytelnikowi ułatwić wyrobienie poglądu o sposobie pracy z narzędziem i ukazać łatwość osiągania typowych celów.

2. Składniki programu

Projekt

Aplikację OPO projektuje się za pomocą narzędzia Oracle Power Objects Designer.

Projekt aplikacji składa się z dwóch części: sesji — będącej definicją połączenia z bazą danych i właściwej aplikacji, zawierającej formularze i raporty (rys. 1).

Sesja

Sesja odpowiada połączeniu z bazą danych przez jednego użytkownika. Dane identyfikujące bazę danych (np. nazwa pliku dla bazy Blaze, *connect string* dla bazy Oracle7) oraz użytkownika (uid i

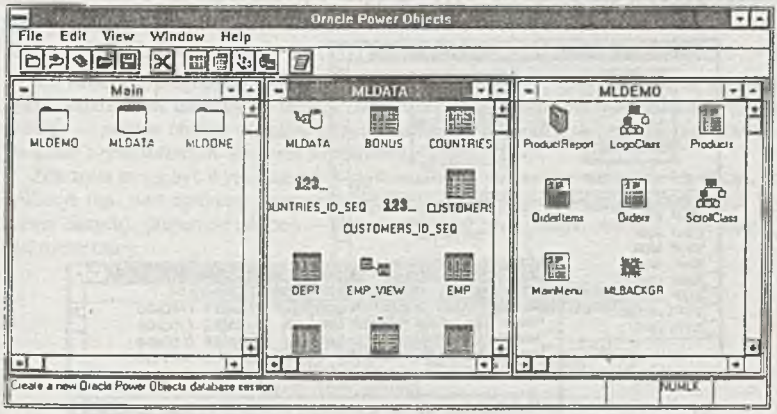
[1] Opis oparto na beta-wersji 0.1.1.40 udostępnionej przez Oracle Polska.

hasło) są zapamiętane w definicji sesji. Po otwarciu sesji następuje połączenie z bazą danych (*connect*) i uzyskujemy dostęp do obiektów bazy danych.

OPO udostępnia tabele, indeksy, sekwencje i perspektywy. Obiekty takie można tworzyć, usuwać, zmieniać ich właściwości (np. modyfikować definicję perspektywy) za pomocą prostych w obsłudze narzędzi graficznych, w przypadku tabel i perspektyw można także oglądać i edytować dane (rys. 2).

Definicje sesji są przechowywane w plikach o rozszerzeniu .XSE.

Jednocześnie można wykorzystywać kilka sesji, łączących z różnymi bazami danych, np. jedna sesja może dotyczyć bazy na serwerze Oracle7, druga — bazy lokalnej typu Blaze. Aplikacja może korzystać równocześnie z danych wielu sesji (chociaż nie można łączyć danych z różnych sesji w ramach jednego zdania SQL).



Rys. 1. Okna OPO Designera: główne (*Main*), sesji (MLDATA) i aplikacji (MLDEMO).

Aplikacja

Aplikacja składa się z formularzy i raportów. Dodatkowo aplikacja zawierać może także definicje klas obiektów oraz obrazy (mapy bitowe) wykorzystywane w raportach i formularzach.

Zarówno sama aplikacja jak i jej składniki są obiektami. Aplikację można więc widzieć jako drzewo obiektów. Każdy obiekt ma własności — dane związane z nim oraz metody — kod obsługujący ten obiekt. Projektant definiuje strukturę drzewa obiektów, określa wartości własności i pisze w języku Oracle Basic metody wykonujące niestandardowe czynności.

Definicje aplikacji przechowywane są w plikach o rozszerzeniu .XAP.

Biblioteki

Biblioteki OPO służą do przechowywania definicji klas oraz obrazów wykorzystywanych w wielu aplikacjach. Tworzy się je podobnie jak aplikacje. Biblioteki są przechowywane w plikach o rozszerzeniu .XLB.

W aplikacjach OPO można także wykorzystywać funkcje z bibliotek DLL.

Program wynikowy

Aplikację OPO kompiluje się i dostarcza użytkownikowi w jednej z dwóch możliwych postaci: samodzielnego pliku wykonywalnego .EXE albo pliku pośredniego .X, który może być wykonany przez oprogramowanie *runtime*.

Do działania aplikacji z odległą bazą danych jest także oczywiście potrzebne oprogramowanie komunikacyjne (np. Oracle SQL*Net).

Wymagania sprzętowe

Do uruchomienia gotowej aplikacji OPO potrzeba przynajmniej:

- komputera PC z procesorem 386SX, 4 MB RAM, kartą graficzną VGA, 2 MB przestrzeni na dysku oraz systemem MS Windows 3.1

albo

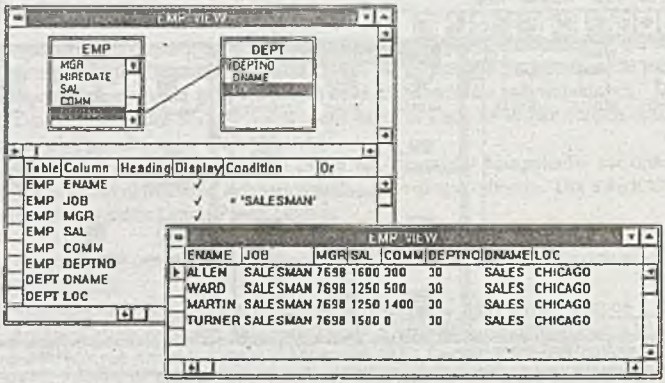
- komputera Apple Macintosh z procesorem 68020, 4 MB RAM, 2 MB przestrzeni na dysku oraz systemem Apple System Software Version 7.0.

Projektant aplikacji potrzebuje nieco więcej:

- 8 MB RAM, 30 MB przestrzeni na dysku dla PC

albo

- 8 MB RAM i 15 MB przestrzeni na dysku dla Macintosha.



Rys. 2. Edycja definicji perspektywy i przeglądarka danych: przedstawiono perspektywę ze złączeniem i warunkiem.

3. Obiektowa budowa aplikacji

Obiekty

Cała aplikacja jest hierarchią obiektów. Sesja i aplikacja są same obiektami i zawierają inne obiekty. Korzeń hierarchii stanowi aplikacja. Jej składnikami są formularze i raporty. Zawierają one obiekty-elementy graficzne, np. pola, listy, przyciski itp. Elementy te reprezentują dane (np. pola, listy), stanowią stałe tło graficzne (np. napisy, linie, prostokąty), służą do organizacji innych elementów (np. formularze podrzędne, grupy w raportach) albo do sterowania aplikacją (przyciski).

Obiekty w hierarchii mogą mieć charakter kontenerów — zawierać inne obiekty. Taki charakter mają oczywiście same formularze i raporty, ale także pewne ich elementy, np. formularze podrzędne.

W skład aplikacji wchodzą też obrazy (mapy bitowe), które mogą być włączane jako tło do formularzy i przycisków.

Własności

Każdy obiekt ma zbiór własności, określających jego nazwę i cechy. Obiekty reprezentujące dane mają wartość, która jest także własnością.

Do edycji własności obiektu służy specjalne okno — arkusz własności (rys. 3). Niektóre własności obiektów mogą być ustalone tylko w czasie projektowania, inne mogą być zmieniane w czasie wykonania programu. Własności z reguły mają sensowne wartości domyślne.

Projektant może zdefiniować nowe własności (*user defined properties*) i dodać je do zbioru własności obiektu aplikacji. Zbiór własności obiektów bazy danych (składników sesji) nie może być rozszerzany.

Metody

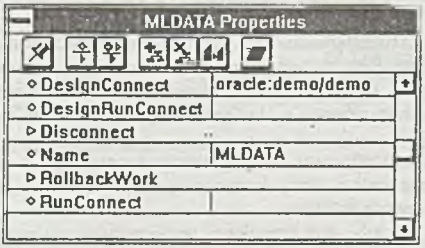
Obiekt aplikacji może mieć własne metody — funkcje i procedury związane z obiektem. Metoda opisuje wykonanie pewnej akcji, najczęściej wyzwolonej przez zdarzenie. Standardowe klasy obiektów są wyposażone w standardowe metody, wykonujące domyślne akcje. Można zmienić działanie metody, pisząc w języku Oracle Basic nowy kod metody. W kodzie tym można wywołać akcję domyślną. Metody edytuje się używając arkusza własności.

Projektant może definiować nowe metody (*user defined methods*), mające charakter procedur lub funkcji.

Zdarzenia i ich obsługa

Jak większość programów przeznaczonych dla środowisk GUI, aplikacje OPO są sterowane zdarzeniami. Zdarzeniom dotyczącym danego obiektu odpowiadają odpowiednie standardowe metody tego obiektu — metody obsługi są zatem sztywno powiązane z określonymi zdarzeniami. Domyślne akcje związane z tymi metodami stanowią standardową obsługę zdarzeń.

Zdarzenia mogą być wywołane przez użytkownika (np. kliknięcie, wciśnięcie klawisza), przez samą aplikację (np. start aplikacji, otwarcie formularza) lub przez środowisko (np. zgłoszenie błędu przez serwer danych). Repertuar zdarzeń — odpowiadający zbiorowi metod obsługi — jest stały i nie może być rozszerzany.



Rys. 3. Arkusz własności sesji: własności poprzedzone rombem, metody poprzedzone trójkątem.

4. Formularze

Podstawowymi składnikami aplikacji są formularze. Formularze umożliwiają przeglądanie i edycję danych. Umieszczone na nich przyciski sterują pracą aplikacji (OPO nie umożliwia definiowania menu). Zwykle formularz otwierany przy starcie aplikacji zawiera przyciski pełniące rolę głównego menu aplikacji.

Składniki formularza

Formularz zawierać może różnorodne obiekty graficzne:

- Elementy służące do wyświetlania i edycji danych: pola tekstowe, pola wyboru (*checkboxes*), listy stałe i nakładane, pola typu *combo* (połączenie listy nakładanej z polem tekstowym), przyciski radiowe.
- Elementy stałe: napisy, linie, prostokąty, owale. Tło formularza może stanowić pojedynczy lub powtarzający się obraz (mapa bitowa). Formularz może też zawierać osadzone obiekty OLE.
- Obiekty-kontenery, służące do grupowania obiektów: formularze podrzędne (*embedded forms*), wyświetlacze powtarzające (*repeater displays*) służące do wyświetlania wielu wierszy tabeli, ramy grup przycisków radiowych.
- Instancje klas zdefiniowanych przez projektanta.

- Elementy sterujące: przyciski, suwaki (służące do zmiany bieżącego wiersza), wskaźniki bieżącego wiersza.

Połączenie z bazą danych

Ponieważ podstawowym zadaniem formularza jest wyświetlanie danych z bazy, połączenie z tabelami lub perspektywami zawartymi w otwartej sesji jest realizowane automatycznie na podstawie własności formularza i jego obiektów. Obiekty-kontenery mają własność Record Source, która określa przyłączoną tabelę, będącą źródłem danych. Elementy wyświetlające dane mają z kolei własność Data Source, określającą kolumnę przyłączoną do elementu.

Właściwości obiektu-kontenera mogą także określać sesję z której pochodzi tabela oraz dodatkowy warunek umieszczany w klauzuli where zapytania pobierającego dane.

Kontenerowi połączonemu z tabelą przypisany zostaje bufor, tzw. *recordset*, do którego pobierane są dane. Bufor ten jest zarządzany przez specjalny obiekt OPO, zwany *Record Managerem*. Ukryty „za" formularzem *recordset* jest jako obiekt dostępny dla programisty: może być pobrany do zmiennej Oracle Basica. *Record Manager* pozwala na dostęp do danych *recordsetu* wiersz po wierszu. Metody *Record Managera* umożliwiają określenie cech *recordsetu* (np. numeru bieżącego wiersza, liczby wierszy i kolumn, nazw kolumn), zmianę bieżącego wiersza, kasowanie i wstawianie wierszy.

Formularze *master-detail* i synchronizacja

Jeden obiekt-kontener może służyć do edycji danych tylko z jednej tabeli. Można jednak bez trudu budować formularze typu *master-detail*. Najczęściej taki formularz konstruuje się umieszczając na nim obiekt-kontener typu *repeater display* albo *embedded form*. Ten obiekt zostaje połączony z inną tabelą niż formularz nadrzędny. Przykładowy formularz tego typu przedstawia rys. 4.

Aby tak złożony formularz *master-detail* funkcjonował poprawnie trzeba zapewnić synchronizację części podrzędnej z nadrzędną. Służą do tego odpowiednie własności obiektu podrzędnego: LinkMasterForm określa obiekt nadrzędny, LinkMasterColumn określa kolumny tabeli nadrzędnej uczestniczące w związku (zwykle tworzące klucz główny), a LinkDetailColumn — kolumny tabeli podrzędnej (zwykle tworzące klucz obcy).

W podobny sposób można zdefiniować związki wielopoziomowe, np. *master-detail-detail*. Ponieważ formularz może zawierać wiele obiektów-kontenerów, a każdy z nich może także zawierać obiekty-kontenery, tworzyć można złożone struktury.

Można zsynchronizować ze sobą dwa formularze, wykorzystując własności Link... jednego z nich. W prosty sposób można też zsynchronizować formularze z danymi z tej samej tabeli: wystarczy dla jednego z nich określić jako źródło danych ten drugi — do obu formularzy będzie wówczas przyłączony ten sam *recordset*.

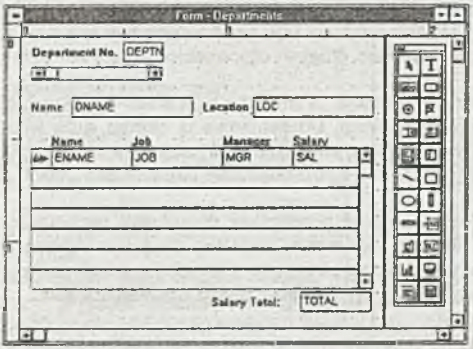
Projektowanie formularza

Formularz projektuje się używając specjalnego edytora graficznego OPO Designera. Do edycji własności i metod formularza i jego elementów używa się arkusza własności.

Typowy proces projektowania formularza *master-detail* przebiega następująco:

- Utworzyć nowy formularz.
- Wybrać z okna sesji ikonę odpowiadającą tabeli nadrzędnej (*master*) i przeciągnąć ją na formularz. Zostaną na nim umieszczone pola tekstowe dla wszystkich kolumn tabeli i będą utworzone odpowiednie połączenia z tabelą bazy danych.
- Skasować zbędne pola tekstowe i umieścić obiekty innych typów (np. listy, przyciski radiowe). Pola innych typów można umieszczać w łatwy sposób: wybrać odpowiedni typ obiektu z palety narzędzi, umieścić ten obiekt wskazując jego położenie myszką, otworzyć okno struktury tabeli lub perspektywy, wybrać odpowiednią kolumnę tabeli i przeciągnąć ją myszką na definiowane pole. Odpowiednie połączenie zostanie utworzone, należy jeszcze uzupełnić nazwę pola i umieścić na formularzu tekst opisu.
- Poprawić wygląd formularza: przenieść pola w wybrane miejsca, poprawić atrybuty wizualne itd.
- Umieścić na formularzu suwak (*scroll bar*) służący do zmiany bieżącego wiersza części *master*.

- Umieścić na formularzu *repeater display* i ustalić wymiary całego obiektu i jego sekcji (w sekcji mieszczą się dane z jednego wiersza *detail*).
- Umieścić pola dla kolumn tabeli podrzędnej w pierwszej sekcji obiektu *repeater display*; zastosować jeden ze sposobów jak dla części *master*.
- Poprawić pola dla części *detail*. Dodać wskaźnik bieżącego wiersza do obiektu *repeater display*.
- Ustanowić związek *master-detail* wypełniając własności Link... obiektu *repeater display*. Usunąć pola kolumn klucza obcego wiążącego część *detail* z częścią *master* — kolumny te będą automatycznie obsługiwane odpowiednio do zdefiniowanego związku.
- Umieścić na formularzu przyciski funkcyjne. Korzystając z arkusza własności wpisać kod metod obsługujących przyciski.
- Zapisać utworzony formularz.
- Uruchomić próbnie formularz (istnieje możliwość uruchomienia tylko bieżącego formularza, bez kompilowania i uruchamiania całej aplikacji), poprawić ewentualne błędy.



Rys. 4. Projekt typowego formularza *master-detail*. Pokazano także paletę narzędzi.

Wykonanie formularza

Otwarcie głównego formularza aplikacji zapewnia kod, który projektant wpisuje w metodzie obsługi startu aplikacji. Kolejne formularze otwiera się zwykle w wyniku obsługi wciśnięcia przycisków.

Otwarty formularz umożliwia przeglądanie i edycję danych. Jest wyposażony w standardowy pasek narzędzi, zawierający przyciski pozwalające wykonać zatwierdzenie zmian, odwołanie zmian oraz wywołać tryb wprowadzania zapytania. Tryb ten umożliwia wyszukiwanie danych za pomocą wbudowanego mechanizmu *Query-By-Form*.

Inne możliwości i cechy formularzy

Wartości domyślne Formularz może automatycznie dostarczać wartości domyślnych dla nowo tworzonych wierszy — służy do tego własność DefaultValue.

Szczególnie ważnym przypadkiem jest generowanie unikalnych numerycznych wartości dla kluczy głównych. W OPO można to osiągnąć na trzy sposoby:

- Użyć obiektu typu sekwencja z bazy danych: w tym celu należy w sesji powołać taki obiekt i dołączyć go do pola formularza ustawiając odpowiednio własności pola.
- Użyć lokalnego mechanizmu OPO: powoduje on wypełnianie pola maksymalną wartością z kolumny klucza głównego powiększoną o ustalony krok.
- Napisać własny kod w Oracle Basicu i umieścić go w specjalnej metodzie wypełnianego pola.

Pola wyliczane Na formularzu można umieścić pola, których wartości nie pochodzą wprost z tabel bazy, ale są wyliczane na bieżąco. W tym celu tworzy się nowe pole tekstowe i w jego własności DataSource wpisuje się wyrażenie wyznaczające wartość pola, poprzedzone znakiem równości.

Często przydatne jest wyliczanie podsumowania wartości z części *detail* formularza. Należy umieścić w części *master* pole tekstowe i ustawić jego własność DataSource na =Sum(*obiekt_detail.pole_sumowane*). Tak zrealizowano podsumowanie na formularzu z rys. 4.

Pola podglądane i listy translacji Często spotykana jest sytuacja, gdy klucz obcy jest numeryczny i wobec tego nie jest wygodny w użyciu. Warto wówczas do zaprezentowania tego klucza zastosować kolumny opisowe, pochodzące z tabeli do której odwołuje się klucz obcy. OPO pozwala odwoływać się do takich opisów poprzez mechanizm pól podglądanych (*lookup*). Są to pola wyliczane, których wartość pochodzi z funkcji SqlLookup(*sesja, zapytanie_SQL*).

Jeśli wartość klucza obcego ma podlegać edycji, to ten sposób nie wystarczy. Użyć należy listy wartości zawierającej opisy, a podstawiającej do pola odpowiadające opisom wartości numeryczne klucza. Taką listę translacji tworzy się wykorzystując pole typu lista (lub lista nakładkowa). Do własności Translation pola wpisuje się tekst zapytania SQL zwracającego dwie kolumny. Pierwsza z kolumn zawierać ma opisy wyświetlane na liście, druga — odpowiednie wartości klucza.

Integralność referencyjna i walidacja Formularze typu *master-detail* wykonują obsługę integralności referencyjnej na poziomie aplikacji. Do sterowania tą obsługą służą własności LinkMasterDel i LinkMasterUpd. Możliwy jest zakaz zmian klucza w części *master*, kaskadowa propagacja zmian klucza z części *master* do części *detail* albo dopuszczenie pozostawiania „sierot".

Formularz może także wykonywać programowe sprawdzanie poprawności danych na poziomie kolumny lub wiersza. Służą do tego metody Validate() dla pola i ValidateRow() dla wiersza. Jeśli metoda taka zwróci wartość logiczną „falsz", to wyświetlony zostaje przypisany przez projektanta komunikat alarmowy i użytkownik musi poprawić błędnie wprowadzone dane.

Do programowego wymuszania poprawności operacji wstawiania i kasowania wierszy służy zestaw metod: PreInsertCheck, PreInsert, PostInsert oraz PreDeleteCheck, PreDelete, PostDelete.

Transakcje i blokady Operacje na danych mają charakter transakcyjny. Zatwierdzenie (*commit*) lub odrzucenie (*rollback*) transakcji jest zwykle wywoływane przez naciśnięcie przycisków na standardowym pasku narzędzi towarzyszącym formularzowi. OPO dopuszcza obsługę transakcji z natychmiastowym lub opóźnionym przesyłaniem operacji do serwera danych. W pierwszej metodzie każda zmiana jest natychmiast przesyłana do serwera i on całkowicie zarządza zatwierdzaniem lub odwoływaniem transakcji. W drugiej metodzie zmiany są buforowane przez *Record Managera* i przesyłane do serwera danych dopiero po zatwierdzeniu.

OPO oferuje dwie strategie blokowania modyfikowanych wierszy: pesymistyczną — wiersze są blokowane od momentu modyfikacji aż do zakończenia transakcji i optymistyczną — wiersze są blokowane jedynie w czasie zatwierdzania transakcji. Użycie strategii optymistycznej wraz z opóźnionym przesyłaniem operacji pozwala zmniejszyć obciążenie sieci.

5. Raporty

Raporty definiowane w OPO są oparte na tabeli lub perspektywie. Raport projektuje się używając specjalnego edytora graficznego OPO Designera (rys. 5). W raporcie można używać takich samych obiektów graficznych, jak w formularzu, ale z oczywistych przyczyn zwykle nie używa się takich rodzajów obiektów, które służą głównie do interakcji (np. przycisków, list, grup radiowych). Najczęściej używane są napisy stale oraz pola tekstowe. Znaczne zastosowanie znajdują pola wyliczane (np. podsumowania) i podglądane (*lookupy*). Połączenie obiektów z bazą danych jest analogiczne jak dla formularzy.

Raporty OPO można drukować albo przeglądać na ekranie.

Sekcje

Raporty składają się z następujących sekcji:

- nagłówków i stopek całego raportu,
- nagłówków i stopek stron,
- nagłówków i stopek grup,
- właściwej zawartości (sekcji *detail*).

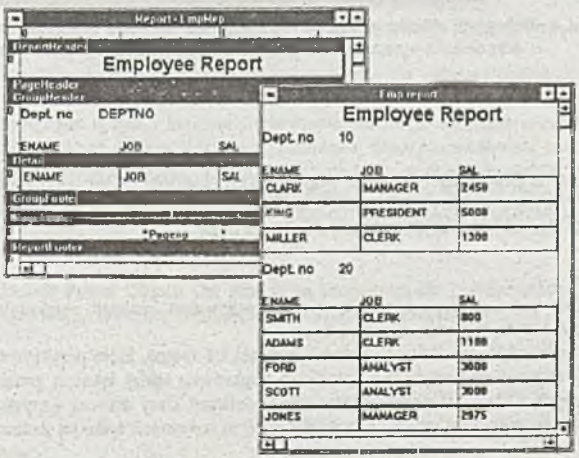
Grupy

Dodatkowym mechanizmem, umożliwiającym zredagowanie wyglądu raportu, jest grupowanie. Polega ono na opatrzeniu wierszy, mających jednakową wartość w wyróżnionej kolumnie, wspólnym nagłówkiem i stopką. Grupę definiuje się tworząc edytorem graficznym nagłówek grupy i wpisując do jego własności GroupCol nazwę kolumny według której grupujemy. Można utworzyć wiele zagłębionych poziomów grup.

Projektowanie raportu

Typowy raport z grupowaniem tworzy się tak:

- Utworzyć nowy raport.
- Ustawić nazwę i tytuł raportu.
- Otworzyć okno definicji tabeli, zaznaczyć kolumny, które mają znaleźć się na raporcie (bez kolumn użytych w grupowaniu!) i przeciągnąć je do sekcji *Detail* raportu. Zostaną utworzone pola tekstowe i opisy oraz odpowiednie połączenia z bazą danych.
- Wybrać z palety narzędzi obiekt *Report Group* i przeciągnąć go na raport. Powstaną sekcje nagłówka i stopki grupy.
- Zaznaczyć w oknie definicji tabeli kolumnę która ma być podstawą grupowania i przeciągnąć ją do nagłówka grupy. Wypełnić własność GroupCol nagłówka.
- W sekcji nagłówka raportu utworzyć stały napis — tytuł raportu.
- W sekcji stopki strony utworzyć pole tekstowe na numer strony i określić jego źródło danych na =pageNum()
- Uzupełnić raport o dodatkowe pola wyliczane (np. podsumowania) oraz podglądane (np. objaśniające wartości klucza użyte do grupowania).
- Poprawić wygląd graficzny raportu, zmieniając położenie, wielkość i atrybuty graficzne pól.
- Zapisać raport.
- Uruchomić raport i sprawdzić jego działanie.



Rys. 5. Projekt i wygląd prostego raportu z grupowaniem

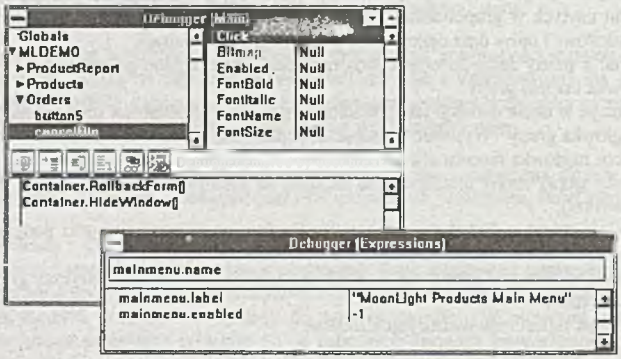
6. Programowanie

Metody obiektów OPO tworzy się używając języka Oracle Basic. Jest to nowoczesny strukturalny dialekt Basica, podobny do implementacji Microsoft Basic for Applications (znanej np. z programu Microsoft Access). Język ten zawiera pełny zestaw konstrukcji i funkcji typowych dla języków ogólnego przeznaczenia oraz wyposażony jest w szczególne mechanizmy umożliwiające jego integrację z OPO.

W Oracle Basicu można operować obiektami OPO. Służy do tego specjalny typ zmiennych Object. Dostęp do metod i własności umożliwiają nazwy kwalifikowane: *obiekt.własność* albo *obiekt.metoda*. Kwalifikator Self umożliwia dostęp do obiektu, którego metoda właśnie się wykonuje. Specjalny kwalifikator Inherited umożliwia wykonanie domyślnej akcji związanej ze standardową metodą.

W treści funkcji i podprogramów Oracle Basica można umieszczać zdania SQL — służy do tego konstrukcja EXEC SQL. W takich zdaniach *embedded SQL* można używać zmiennych Oracle Basica (tzw. *bind variables*).

OPO posiada debugger, który umożliwia przeglądanie struktury obiektów, oglądanie wartości własności i kodu metod, zatrzymywanie programu w punktach kontrolnych, wykonywanie krokowe oraz sprawdzanie wartości zmiennych i wyrażeń (rys. 6).



Rys. 6 Debugger: okienko główne — widać strukturę obiektów, własności i kod metody oraz okienko wyrażeń — do sprawdzania wartości zmiennych i wyrażeń.

7. Klasy

Projektant ma możliwość zdefiniowania własnych klas obiektów. Klasy te mają charakter kontenerów zawierających różne obiekty. Wszystkie klasy użytkownika pochodzą od formularza, do ich edycji jest też używane standardowe okno projektowania formularza.

Dziedziczenie

Definicja klasy jest wzorcem używanym przy powoływaniu instancji obiektu tej klasy. Obiekt-instancja dziedziczy metody i własności swej klasy.

W definicji instancji można dokonać zmian własności lub metod, które przykrywają cechy odziedziczone z klasy. Odpowiednie narzędzie umożliwia anulowanie takiej lokalnej zmiany i przywrócenie cech dziedziczonych z klasy. Zmiany dokonywane w definicji klasy dotyczą wszystkich obiektów powołanych z tej klasy, natomiast zmiany w instancji dotyczą oczywiście tylko tej instancji.

Klasa może dziedziczyć cechy innej klasy. W tym celu nową klasę powołuje się jako podklasę (*subclass*) klasy już istniejącej. Taki mechanizm dziedziczenia oznacza, że klasa może mieć tylko jednego „przodka".

Zasięg własności i metod

Dostęp do własności każdego istniejącego obiektu jest możliwy z każdego miejsca aplikacji — hermetyzacji nie ma. Odwołanie następuje przez nazwę kwalifikowaną.

Podobnie metody wszystkich obiektów mogą być wywoływane z każdego miejsca aplikacji poprzez użycie analogicznych nazw kwalifikowanych.

8. Blaze

Handlowa wersja Oracle Power Objects będzie wyposażona w lokalną bazę danych Blaze. Baza ta umożliwi tworzenie i przechowywanie tabel, indeksów, sekwencji, perspektyw i synonimów.

Blaze będzie transakcyjnym systemem zarządzania bazami danych, przeznaczonym dla niezbyt dużych baz danych (wg dokumentacji do 4 GB), do użytku lokalnego lub przy niezbyt intensywnym wielodostępie. Baza ta nie pracuje w architekturze klient-serwer, nie jest więc przeznaczona dla silnie obciążonych systemów wielodostępnych.

Blaze będzie używać języka SQL będącego podzbiorem SQL z systemu Oracle7 (z ważniejszych ograniczeń wymienić warto brak zapytań hierarchicznych PRIOR i typu CHAR). Nie będzie możliwości definiowania wyzwalaczy i procedur w bazie, ale będzie deklaratywna kontrola integralności referencyjnej. Przewidziano możliwość obsługi baz tylko do czytania (np. zapisanych na CD ROM).

9. Podsumowanie

Narzędzie Oracle Power Objects spełnia ważniejsze wymagania jakie może stawiać projektant systemów przeznaczonych dla indywidualnych użytkowników, małych biur i grup roboczych:

- jest proste w obsłudze i łatwe do nauczenia;
- umożliwia tworzenie zarówno bardzo prostych jak i — dzięki wbudowanemu językowi programowania — dość złożonych aplikacji;
- pozwala tworzyć w pełni funkcjonalne aplikacje w zasadzie bez programowania — napisania kodu wymagają niemal wyłącznie działania nietypowe;
- daje w wyniku wykonywalny program działający samodzielnie;
- umożliwia tworzenie aplikacji zarówno dla komputerów PC jak Macintosh;
- umożliwia zastosowanie skalowalnych SZRBD: od lokalnej bazy Blaze, przez serwer Personal Oracle, serwery dla grup roboczych, po wielkie systemy;
- w pełni wykorzystuje możliwości relacyjnych serwerów danych, łącznie z obsługą transakcji.

Konstrukcja aplikacji jest spójna i logiczna. Obiektowa orientacja narzędzia umożliwia tworzenie bibliotek klas, co ułatwia pracę profesjonalnemu programiście, dając mu możliwość stworzenia bibliotek często używanych rodzajów obiektów. Chociaż możliwości programowania zorientowanego obiektowo są ograniczone (brak hermetyzacji, dziedziczenia z wielu źródeł, nie ma klas funkcjonalnych — nie związanych z obiektami graficznymi), to najważniejsze potrzeby programisty typowych aplikacji zostały zaspokojone.

Wydaje się, że Oracle Power Objects jest atrakcyjną propozycją dla projektantów i programistów tworzących niewielkie i średnie systemy działające na komputerach osobistych.

Informacja u autorze:

dr inż. Tomasz Traczyk
Instytut Automatyki i Informatyki Stosowanej Politechniki Warszawskiej
e-mail: ttraczyk@ia.pw.edu.pl

Maciej Polakowski
IBM Polska

Środowisko generatora aplikacji VisualGen

1. Historia

Generator aplikacji VisualGen jest kolejnym z produktów serii AD/Cycle i bezpośrednim spadkobiercą pakietu CSP (Cross System Product). Pakiety CSP, przeznaczone do tworzenia systemów baz danych na duże serwery (działające pod kontrolą systemów operacyjnych MVS i VSE), mają już ponad dziesięcioletnią historię i są aktywnie wykorzystywane przez wielu klientów IBM. W miarę rozpowszechniania się mikrokomputerów, sieci lokalnych i graficznych interfejsów użytkownika, podjęto pracę mające na celu przeniesienie linii CSP na platformę PC. Pierwszym owocem tych prac była wersja generatora o nazwie CSP/2AD, działająca w środowisku systemu operacyjnego OS/2, a następnym krokiem pakiet VisualGen, który nie tylko poszerzył dostępne opcje ale również liczbę platform docelowych, na których mogą działać wygenerowane aplikacje. Do elementów takich jak: język czwartej generacji, interakcyjne narzędzie testujące czy generator kodu na systemy MVS, VSE, OS/2, dodano interakcyjny, obiektowy edytor graficznego interfejsu użytkownika, przy pomocy którego definiować można zarówno okna dialogowe aplikacji jak i jej logikę czy struktury danych. Daje to niespotykane dotoąd możliwości prototypowania, szybkiej modyfikacji aplikacji jak i również wielce podnosi produktywność tworzenia struktury systemu. Wraz z nowym produktem rozszerzono liczbę docelowych środowisk , które mogą stanowić cel generowanych aplikacji, co pozwoliło uzyskać wysoką przenośność kodu. W obecnej na rynku wersji pakietu, do opcji generacji dostępnych w poprzednich wersjach (MVS, VSE), dołączono możliwość generacji kodu dla systemów OS/2, AIX i MS Windows (wyłącznie aplikacja klienta). W zapowiadanej na drugą połowę roku nowej wersji pakietu dostępna będzie również opcja generacji dla systemu OS/400, co zamknie listę systemów dostarczanych obecnie przez IBM. Oprócz różnorodności docelowych środowisk, na które można generować napisaną aplikację, zaimplementowano kompleksowy zestaw mechanizmów komunikacyjnych pozwalający na pracę systemu w układzie klient-serwer. Inną ciekawą cechą pakietu jest dostępność obsługi wielu systemów baz danych (DB2, IMS, zbiory ISAM, GSAM itp.), co pozwala na wykorzystanie go w niemalże każdym istniejącym środowisku ES/9000.

Równolegle z generatorem VG, wprowadzono na rynek zintegrowane obiektowe środowisko tworzenia aplikacji klienta o nazwie VisualAge, oparte na popularnym obiektowym języku programowania Smalltalk. Podstawowe drzewo klas wzbogacone zostało zestawem wyspecjalizowanych typów "obudowywujących" podsystemy takie jak: systemy zarządzania bazami danych, komunikacji czy interfejsu użytkownika. Dzięki takiej strukturze

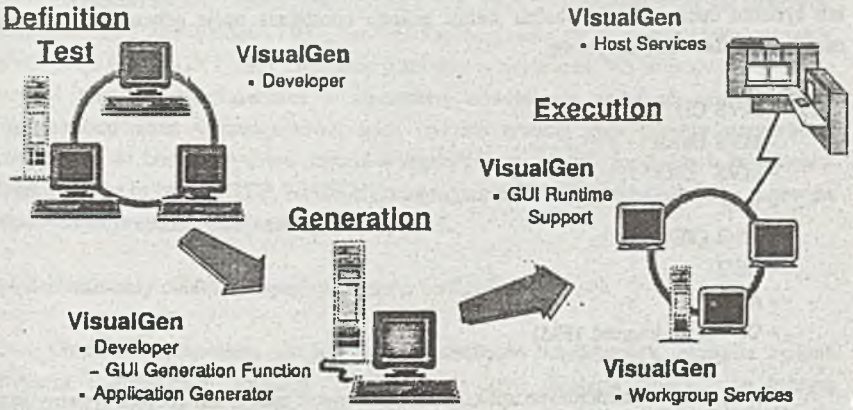
pakiet ten jest idealnym dopełnieniem środowiska VisualGen, pozwalając na pełne wykorzystanie możliwości pracy w układzie klient-serwer.

Na ukończeniu są również prace nad produktem CASE, którego wykorzystanie wspomagałoby proces analizy, projektowania i konserwacji systemów stworzonych przy pomocy VisualGen. Będzie on dostępny w drugiej połowie roku pod nazwą VisualGen Team Suite, i składać się będzie m.in. z: systemu do projektowania, oceny schematu koncepcyjnego i generacji schematu fizycznego bazy danych, systemu wspomagającego proces analizy wymagań i projektowania logiki aplikacji, pojemnika danych i definicji przedsiębiorstwa (*repository*) oraz pakietu służącego do kontroli i zarządzania kolejnymi wersjami kodu źródłowego aplikacji (*versioning software*).

2. Cechy charakterystyczne generatora VisualGen

2.1 Separacja środowisk definicji i wykonania aplikacji.

Jedną z najbardziej znamienitych cech generatora VisualGen jest rozdzielenie środowiska, w którym odbywa się cykl definicji systemu, od docelowej platformy (platform) wykonania. W systemie VG pełny cykl produkcji aplikacji (przedstawiony schematycznie na



Rysunek 1. Schemat definicji i generacji aplikacji w środowisku VisualGen

Rys.1), na który składają się: definicja składników systemu (logika, interfejs użytkownika, dane), testy wstępne, integeracyjne oraz generacja kodu pośredniego, odbywa się na platformie mikrokomputerowej w środowisku lokalnej sieci danych. Jedynie finalne kroki produkcji

modułów oprogramowania (kompilacja, łączenie itp.) z racji swojej specyfiki muszą odbywać się na platformie docelowej. Jednak i na tych etapach, procesy te odbywają się automatycznie.

Programiści po otrzymaniu zadań, pracują nad elementami systemu aplikacyjnego, korzystając ze wspólnych biblotek procedur i danych. W trakcie tworzenia nowych składników systemu, następuje proces ich testowania (wykorzystywane jest do tego interakcyjne narzędzie testujące środowiska VG), a po usunięciu błędów, elementy zachowywane są we wspólnych zbiorach, gdzie oczekują na dalsze przetwarzanie. Okresowo, są one udostępniane innym członkom zespołu, podlegają procesowi testowania integracyjnego, którego pomyślne zakończenie wyznacza moment generacji kodu dla środowiska docelowego. Warto podkreślić dużą swobodę możliwych konfiguracji środowiska rozwoju, gdyż proces testowania może przebiegać zarówno nad składnikami systemu w formie pierwotnej (nie poddanych generacji) jak i nad elementami znajdującymi się na platformie docelowej. Analogiczna sytuacja ma miejsce w przypadku dostępu do baz danych, mamy możliwość korzystania z danych lokalnych, lub umieszczonych w środowisku wielodostępnym (LAN, platforma docelowa np. ES/9000), gdzie operacje będą wykonywane w realnych warunkach testowych.

2.2 Wysoka przenośność kodu źródłowego.

Inną zaletą środowiska VisualGen jest wysoka przenośność kodu źródłowego aplikacji tworzonego przy pomocy tego narzędzia. Pozwala to na ochronę kapitału zainwestowanego w rozwój aplikacji i eliminuje konieczność modyfikacji, przy zmianach platform sprzętowych lub systemu operacyjnego. Obecna wersja pakietu udostępnia opcje generatora kodu na następujące platformy docelowe:

- MVS CICS
- MVS Batch
- MVS TSO
- VSE
- OS/2 CICS
- OS/2
- AIX
- OS/400 (3 kwartał 1995)

Jednocześnie w tworzonych aplikacjach wykorzystać można następujące systemy baz danch i metody organizacji zbiorów:

- DB2 (MVS,VSE,AIX,OS/2)
- IMS
- ISAM, GSAM, QSAM ...

Zalety przenośności kodu źródłowego są szczególnie doceniane w konfiguracjach

składających się z wielu współdziałających ze sobą platform sprzętowych (np. w wielopoziomowym środowisku przetwarzania klient-serwer), kiedy to bez ponoszenia dodatkowych kosztów, możliwe staje się optymalne rozproszenie składników aplikacji.

2.3 Wielopoziomowa architektura klient-serwer

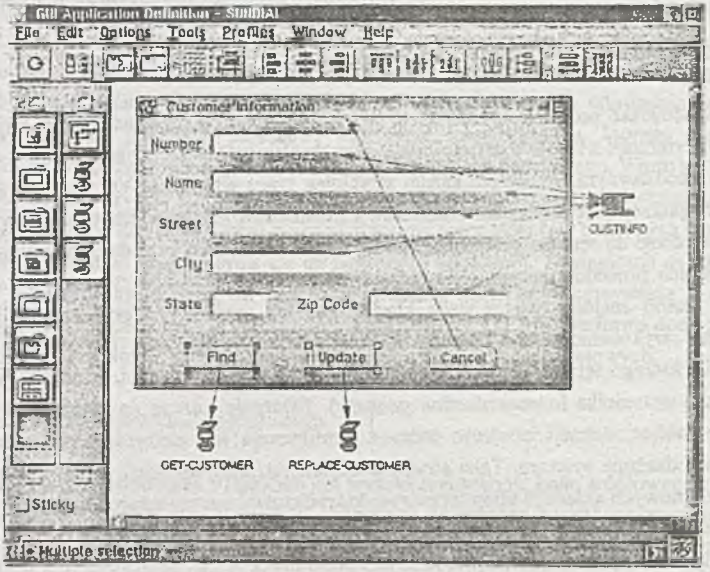
Środowisko produktów VisualGen, w porównaniu do poprzednich wersji linii CSP, zostało rozszerzone z myślą o definicji systemów działających w architekturze klient-serwer. Dlatego nieodzownym elementem każdej z biblotek czasu wykonania VG wspomagających działanie aplikacji jest podsystem komunikacji (*middleware*). Jego podstawowym zadaniem jest ustalić drogi do serwera potrzebnego zasobu (procedurę, plik, bazę danych) i wywołanie odpowiednich procedur dostępu. Niezależnie od tego czy odwołanie zaspokajone zostaje lokalnie (zasób znajduje się na stacji klienta) czy też musi zostać przesłane do innego komputera, wykonaniem tego zajmuje się podsystem komunikacji. Dodatkowo, każde odwołanie, którego serwer nie jest znany/dostępny przez lokalny system, zostaje przesyłane do systemu-pośrednika (*communication gateway*). Położenie i drogę do zasobów określają specjalne tablice alokacji zasobów, których konfiguracja jest elementem parametryzacji środowiska działania systemu. Taka struktura umożliwia pracę aplikacji mogą pracować w wielopoziomowych układach klient-serwer, w których rozmieszczenie zasobów jest proste do implementacji, a rekonfiguracja nieskomplikowana.

Składniki aplikacji spełniające zadania klienta mogą być wygenerowane dla środowisk systemów: OS/2, MS Windows i AIX, natomiast aplikacje serwera mogą działać w systemach: MVS, CICS OS/2, AIX i VSE. Wszystkie platformy za wyjątkiem MS Windows są w stanie spełniać rolę systemu-pośrednika w transporcie odwołań do odległych zasobów. Przy implementacji warstwy transportowej sieci, wykorzystywanej przy łączeniu komputerów działających w obrębie systemu można wykorzystać większość popularnych protokołów sieciowych (m.in. APPC, TCP, NetBIOS), a oprogramowanie VG automatycznie zapewnia odpowiednie procedury konwersji pomiędzy nimi.

2.4 Zorientowany obiektowo graficzny edytor aplikacji graficznych

Wraz z pojawieniem się graficznych interfejsów użytkownika, nastąpiła zmiana struktury i archtektury aplikacji. Dotychczasowy tryb działania aplikacji, w którym użytkownik był dopuszczany do konwersacji z komputerem jedynie miejscach i kolejności przewidzianej przez programistę ustąpił aplikacjom zorientowanym zdarzeniowo, w których sposób i kolejność prowadzenia konwersacji należy do człowieka. Interakcyjny tryb pracy wymusił inne metody organizacji architektury aplikacji i struktury jej logiki. VisualGen dostarcza programiście wyspecjalizowane narzędzie (Rys. 2) pozwalające definiować interfejs użytkownika, architekturę aplikacji, jej logikę i dane. Aplikacja przedstawiona jest w postaci graficznego obrazu , na który składają się: okna interfejsu użytkownika, ikony reprezentujące procedury, dane aplikacji oraz połączenia będęce reprezentacją zdarzeń i zależności pomiędzy

elementami. Przy definicji interfejsu użytkownika programista ma do wyboru zbiór gotowych elementów graficznych dostarczonych(okna dialogowe, przyciski, edycje) ale może także zdefiniować własne elementy, które bardziej spełniają jego wymagania. Zestaw powiązanych

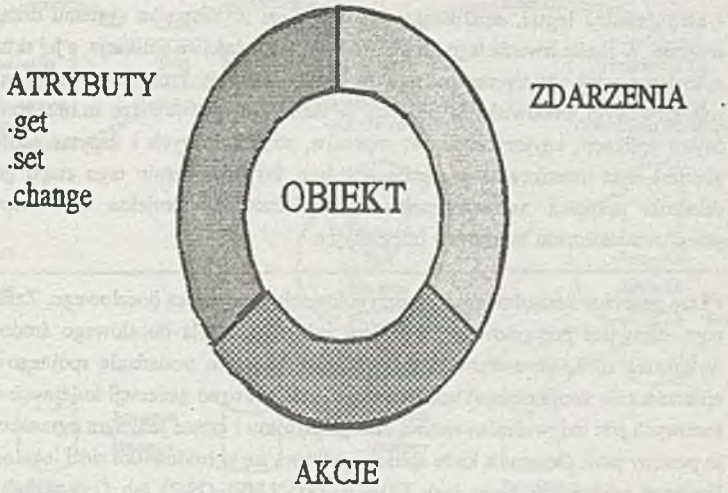


Rysunek 2. Obiektowy edytor aplikacji

składowych (np. okno dialogowe), który jest kandydatem do ponownego wykorzystania, może zostać dołączony do palety gotowych elementów.

Każdy element aplikacji posida specyficzny interfejs, za pośrednictwem którego może komunikować się ze światem zewnętrznym. Elementami takiego interfejsu są: atrybuty, akcje i zdarzenia (Rys. 3). Wartości atrybutów definiujące stan obiektu, są dostępne na zewnątrz i mogą być przez zmieniane przez zewnętrzne obiekty. Zdarzenia sygnalizują zmiany stanu obiektu i zależnie od typu element mogą być wynikiem interakcji z użytkownikiem (wciśnięcie klawisza, otworzenie okna dialogowego) lub współdziałania z innymi obiektami aplikacji (zmiana wartości atrybutu w wyniku wykonania procedury). Trzecim elementem interfejsu obiektu są akcje, będące implementacją protokołu komunikacji z obiektem (w terminologii obiektowej są one odpowiednikiem *metod*). Przykładami akcji są: żądanie wykonania proceury/aplikacji (obiekty logiki), nadanie wartości elementowi rekordu (obiekty danych), wyświetlenie okna dialogowego (obiekty interfejsu użytkownika) itd..

Powiązania pomiędzy obiektami przedstawione w edytorze przy pomocy połączeń reprezentują w rzeczwistości komunikujące się elementy interfejsów. Dozwolone są jedynie cztery typy połączeń (zdarzenie-akcja, atrybut-atrybut, atrybut-akcja, zdarzenie-atrybut), którym można nadać interpretację przepływu sterowania.



Rysunek 3. Obiekt i jego interfejs

Obiektowo zorientowany edytor aplikacji systemu VisualGen jest narzędziem pozwalającym na szybkie i proste definiowanie struktury aplikacji z wykorzystaniem predefiniowanych lub stworzonych przez użytkownika elementów. Graficzne przedstawienie zależności pomiędzy podsystemami i znaczące abstrakcje zwiększają przejrzystość aplikacji i w wyniku zmniejszają koszty jej utrzmania. Możliwość definiowania własnych elementów w celu wielokrotnego wykorzystania pozwala na tworzenie biblotek gotowych składników, co dodatkowo zwiększa produktywność programistów.

3. Zarys procesu definicji aplikacji

3.1 Proces definicji aplikacji

Typowy proces tworzenia aplikacji (Rys. 4) w środowisku VisualGen składa się z następujących etapów:

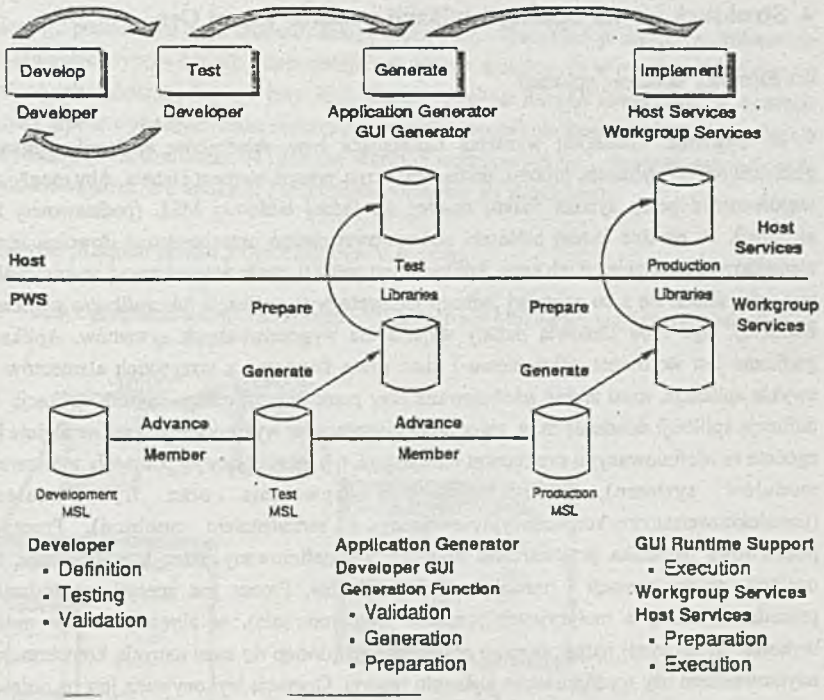
- Etap definicji schematu bazy danych; w ramach którego powstaje i zostaja zdefiniowany fizyczny schemat bazy (baz) danych aplikacji. Do pomocy mamy tutaj narzędzia ułatwiające definiowanie schematu, takie jak Query Manager (DB2/2) czy QMF (DB2). Już na tym etapie warto jest zapamiętać w bazie danych zestaw danych testowych, które udało się zgromadzić w trakcie etapu analizy (dane standardowe, odpowiadające sytuacjom wyjątkowym).

- Etap definicji logiki, struktur danych, interfejsu użytkownika systemu oraz testy wstępne. W czasie trwania tego kroku tworzona jest właściwa aplikacja, a jej składniki są testowane tak aby usunąć podstawowe błędy logiczne. Programista posługuje się zintegrowanym środowiskiem definicji w skład którego wchodzą m.in.: graficzny edytor aplikacji, edytory ekranów, raportów, struktur danych i zapytań (*database queries*) oraz interakcyjne narzędzie testujące. Po zakończeniu tego etapu gotowe składniki aplikacji są udostępniane administratorowi projektu lub zespołowi przeprowadzającemu testowanie integracyjne.

- Etap generacji kodu, kompilacji i przygotowania środowiska docelowego. Zadaniem tego etapu jest przygotowanie modułów ładowalnych dla docelowego środowiska wykonania aplikacji, a warunkiem jego wykonania jest posiadanie spójnego (choć niekoniecznie kompletnego) kodu systemu. Częstotliwość generacji kolejnych wersji testowych jest indywidualną sprawą danego projektu i często jedynym ograniczeniem są postępy prac. Generacja kodu aplikacji odbywa się w środowisku sieci lokalnej, zaś językami pośrednimi mogą być: COBOL (MVS,VSE,OS/2) lub C++ (OS/2,AIX). Kody źrodłowe aplikacji, wraz z odpowiednimi zbiorami, zadanie których jest kontrola kompilacji i wykonania, są przesyłane na platformę docelową, a następnie poddawane są procesowi kompilacji i przygotowania. Wraz z przebiegami kompilacji wykonywane są również inne zadania, do których należą np.: generacja planów dostępu do baz danych czy definicja zasobów systemu CICS. Wynikiem zakończenia kroku kompilacji są gotowe do wykonania moduły ładowalne (w zależności od docelowego systemu operacyjnego mogą one mieć postać bibliotek lub plików). Parametry procesów generacji i przygotowania można poddać wielu modyfikacjom pozwalającym ustalić indywidualne cechy każdego projektu. Modyfikacje te przeprowadzane są na dwa sposoby: poprzez podanie odpowiednich opcji (zestawów opcji) lub poprzez zmianę zawartości zbiorów służących jako elementy składowe wykorzystywane w procesie generacji. Przykładem takich zbiorów są pliki kodów źródłowych w języku COBOL, wykorzystywanych jako elementy kodu źródłowego podczas generacji aplikacji docelowej.

3.2 Biblioteki MSL

Elementy aplikacji niezależnie od typu są przechowywane w specjalnych zbiorach bibliotecznych MSL(*member specification library*). Elementami bibliotek MSL mogą być inne biblioteki MSL i wtedy mówić będziemy o złożonej bibliotece MSL (*concatenated MSL*), lub właściwe elementy aplikacji, kiedy to mówimy mamy do czynienia z biblioteką prostą MSL (*basic MSL*). Biblioteki złożone powstały z myślą o aplikacjach tworzonych w zespołach wielosobowych, i wspomaganie procesu modyfikacji istniejącego kodu. Użytkownik pracujący nad biblioteką złożoną, ma dostęp do wszystkich składników zdefiniowanych w bibliotekach składowych, lecz prawo zapisu (modyfikacji) do elementów jedynie do pierwszej na liście



Rysunek 4. Etapy procesu tworzenia aplikacji w systemie VisualGen

biblioteki podstawowej. Taka zasada powoduje, że każda modyfikacja widoczna jest widoczna tylko lokalnie i dopiero na wyraźne polecenie programisty może zostać udostępniona innym. W typowej sytuacji, każdy członek zespołu pracującego nad rozwojem/modyfikacją danej aplikacji, posiada własną bibliotekę podstawową, w obrębie której dokonuje zapisu zmodyfikowanych elementów, oraz bibliotekę złożoną, dzięki której ma dostęp do bibliotek elementów wspólnych. Pierwszm elementem bibloteki złożonej jest prywatna bibloteka podstawowa danego programisty, tak aby modyfikacje dokonywane przez niego były widoczne w obrębie jego sesji.

4. Struktura języka definicji aplikacji systemu Visual Gen

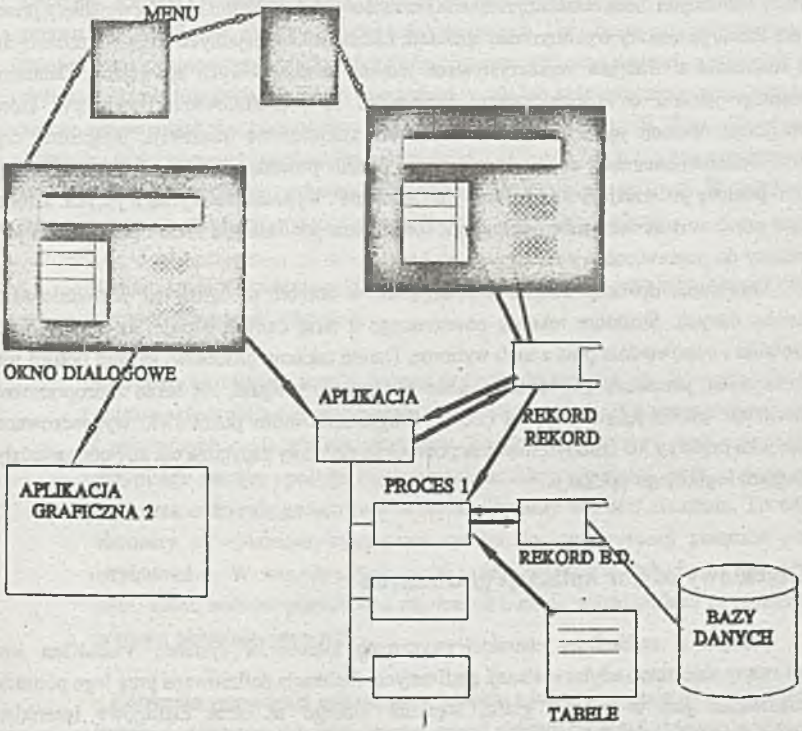
4.1 Elementy składowe aplikacji

Systemie VisualGen wyróżnia następujące typy składników aplikacji: aplikacja graficzna (GUI), aplikacja, process, procedura, ekran, rekord, element i tabela. Aby mogły one współtworzyć pełny system muszą należeć do jednej bibloteki MSL (podstawowej lub złożonej). W obrębie każdej bibloteki podstawowej można przechowywać dowolną liczbę elementów, niezależnie od ich typu. Aplikacja jest pełna tj. może wygenerować wykonywalny kod, gdy składa się z co najmniej jednego elementu typu *aplikacja* lub *aplikacja graficzna*. Elementy tego typu stanowią punkty wejścia dla wygenerowanych systemów. Aplikacja graficzna jest sterowana zdarzeniowo i choć może korzystać z wszystkich elementów co zwykła aplikacja, musi zostać zdefiniowana przy pomocy graficznego edytora aplikacji. Na definicję aplikacji składa się m.in. zbiór *procesów*/procedur wykonywanych sekwencyjnie lub zgodnie ze zdefiniowanymi strukturami sterującymi, typ (określający jej położenie w hierarchii modułów systemu), lista parametrów wywołania oraz tryb działania (pseudokonwersacyjny/konwersacyjny-związany z zarządzaniem zasobami). Proces - podstawowa jednostka przetwarzania aplikacji jest definiowany przez trzy elementy: typ operacji, obiekt operacji i procedurę obsługi błędów. Proces jest specjalnym rodzajem procedury (znanej z tradycyjnych języków programowania), w obrębie którego można wykonać co najwyżej jedną *operację zewnętrzną* np. dostęp do bazy danych, konwersację z użytkownikiem czy wydrukowanie elementu raportu. Operacja wykonywana jest na *obiekcie operacji*, którym może być rekord skojarzony z tabelami bazy danych (lub zbiorem), ekran do wyświetlenia na ekranie terminala czy też składnik drukowanego raportu. Jeśli operacja zewnętrzna zakończy się niepowodzeniem, wywoływana jest *procedura obsługi błędu* (w szczególności zdefiniowana przez użytkownika). Dostępnych jest 13 predefiniowanych typów procesów, większość z których zajmuje się obsługą operacji w systemach baz danych. Na przykład process typu ADD dodaje nowy rekord, process SETINQ przeszukuje wyspecyfikowane tabele baz danych, SCAN sekwencyjnie zwraca rekordy znalezione uprzednio przez SETINQ. Treść procesu definiowana jest instrukcjami języka 4GL poprzedzających i następujących po danej operacji. *Procedura* jest konstrukcją znaną z innych języków programowania i różni się od procesu jedynie tym, że nie operuje na obiektach zewnętrznych. *Ekran* jest wykorzystywany jako okno dialogowe lub jako składnik raportów. Definicja odbywa się przy pomocy specjalnego edytora ekranów, a składnikami są: (pola, napisy), atrybuty, docelowe urządzenia wyjściowe czy procedury sprawdzające poprawność wpisanych danych. *Rekord* to podstawowa struktura służąca do przechowywania danych, analogiczna do konstrukcji spotykanych w innych językach programowania. Wśród rekordów wyróżniamy rekordy robocze (*working storage records*), których zadaniem jest przechowywanie wartości lokalnych zmiennych aplikacji oraz rekordy baz danych skojarzone z rezultatami zapytań lub zawartością plików. Struktura rekordu roboczego może obejmować wielopoziomowe zagnieżdżone struktury i tablice *elementów*. Element zaś to analogia

zmiennej, posiadająca typ, długość, zasięg i atrybuty. VisualGen predefiniuje kilkanaście podstawowych typów danych odpowiadających typom spotykanym w systemach baz danych (znaki i ich łańcuchy, liczby, daty itp.). Zestaw struktur danych dostępnych w systemie zamyka *tabela* wykorzystywana najczęściej do przechowywania danych, które nie podlegają modyfikacji (treść komunikatów, zakresy wartości wejściowych itp.). Rysunek 5 przedstawia przykładową strukturę aplikacji i zależności pomiędzy jej składowymi.

4.3 Krótki przegląd języka programowania VisualGen

Język programowania 4GL systemu VisualGen jest blokowym językiem strukturalnym,



Rysunek 5. Struktura i zależności pomiędzy elementami aplikacji

którego składnia i semantyka zostały zoptymalizowane do obsługi operacji nad bazami danych. W rzeczywistości do składni języka należałoby dodać języki zapytań systemów baz danych (SQL, DL/I). Są to jednak standardy i nie wymagają omawiania.

Zestaw instrukcji i elementów języka został pozbawiony konstrukcji nadmiarowych,

co wymusza pewien styl pisania kodu aplikacji. Każda instrukcja języka musi rozpoczynać się od nowej linii i kończyć średnikiem. Pozostawiono jedynie trzy podstawowe struktury sterujące: złożenie sekwencyjne, wykonanie warunkowe (IF-ELSE) i iterację (WHILE). Wśród instrukcji występują oczywiście: przypisanie (MOVE, =), wywołanie procesu/procedury (PERFORM), ale także przeszukanie tabeli (RETR) czy przypisanie wielokrotne (MOVEA). Rozbudowane zostały wyrażenia logiczne, które muszą być w stanie obsługiwać współpracę z bazami danych jak i interakcję z użytkownikiem. Do tych kategorii warunków należą na przykład obsługa błędów/ostrzeżeń wygenerowanych przy dostępie do bazy danych (brak szukanego rekordu, blokada), do drugich zaś obsługa konwersacji z użytkownikiem (atrybuty pól, wyświetlanie komunikatów błędów). Aby ułatwić pisanie oprogramowania, w ramach samego języka zdefiniowano zestaw słów specjalnch i literałów o z góry określonym znaczeniu. Należą do nich m.in. identyfikatory najczęstszach błędów zwracanych przez systemy baz danych (brak rekordu), zmienna przechowywująca ostatni klawisz wciśnięty przez użytkownika, parametry wykonywanej aplikacji. Liczba słów specjalnych sięga kilkudziesięciu lecz większość z nich jest wykorzystywana jedynie w nietypowych sytuacjach. Ostatnim elementem języka, o którym należy wspomnieć są predefiniowane procedury, które udostępniają operacje systemowe takie jak jawne zakończenie tranzakcji, programu, czy zmianę aktualnej bazy danych. W czasie edycji i pisania procedur dostępny jest kontekstowy system pomocy pozwalający na interakcyjne tworzenie i wypełnianie wyrażeń języka. Zanim nastąpi uruchomienie narzędzia testującego, sprawdzana jest składnia kodu i programista jest zmuszony do poprawienia ewentualnych błędów.

Odwołania do bazy danych odbywają się w obrębie procesów za pośrednictwem rekordów danych. Strukturę rekordu powiązanego z bazą danych buduje się specyfikując tabele/pliki i odpowiednie pola z nich wybrane. Dzięki takiemu procesowi system potrafi nie tylko odczytać parametry pól rekordu, takie jak typ czy długość, ale także zaproponować odpowiednie zdanie języka zapytań (SQL). Programista może pozostawić wygenerowane zdanie jako prototyp lub zmodyfikować odpowiednie elementy zapytania tak aby odpowiadały wymogom logiki jego aplikacji.

5. Obiektowy edytor aplikacji graficznych

Jednym z najbardziej charaktetrystycznych elementów systemu VisualGen jest zorintowany obiektowo edytor aplikacji graficznych. Aplikacja definiowana przy jego pomocy przedstawiona jest w postaci grafu, węzłami którego są okna dialogowe interfejsu użytkownika, ikony przedstawiające logikę(aplikacje) i dane aplikacji(rekordy i elementy), zaś połączenia reprezentują przepływ sterowania i danych. Ten dość niezwykły sposób definiowania aplikacji jest próbą zaproponowania alternatywnego sposobu specyfikacji architektury systemu. Dotachczas cała logika systemu definiowana była przy pomocy struktur sterujących języków programowania. Bardzo często było to jednak mało przejrzyste i trudne do odtworzenia w czasie modyfikacji struktur systemu. Zaproponowana metoda graficznego

przedstawienia aplikacji pozwala na zwiększenie ilości informacji przy jednoczesnym uproszczeniu jej prezentacji. Implementacja bazuje na filozofii modelu View-Logic-Data, który zakłada że w każdym systemie przetwarzania danych można wyróżnić trzy podstawowe składowe: jego zewnętrzny obraz (postrzegany przez użytkownika), logikę (semantyka dostępnych akcji), dane (odpowiedzialne za stan systemu). W przypadku edytora systemu VisualGen zewnętrzny obraz aplikacji reprezentowany jest przez elementy interfejsu użytkownika (okna dialogowe, systemy menu), logika zarówno przez procedury aplikacji i połączenia pomiędzy składnikami rysunku, zaś dane przez rekordy, tabele i samą zawartość bazy danych.

Edytor aplikacji jest dostarczany wraz ze zbiorem predefiniowanych elementów, które służą do tworzenia okien interfejsu i logiki aplikacji. Należą do nich standardowe podokna takie jak przyciski (*buttons*), listy i pola edycji, ale również obiekty nie posiadające swojej reprezentacji graficznej jak na przykład kolekcja danych. Listę dostępnych elementów można powiększać dodając własne obirkty. Ich definicja może odbywać się na dwa sposoby: poprzez definicje elementów podstawowych (niepodzielnych), lub obiektów złożonych (składających się ze zbioru obiektów podstawowych i indywidualnego interfejsu). Specyfikacja nowych obiektów podstawowych musi oddbyć się w języku 3GL (C, COBOL), zaś elementy złożone definiowane są przy pomocy edytora - analiogicznie do zwykłych aplikacji. W skład obiektów złożonych mogą wchodzić również dane(rekordy) i logika(procedury).

Komunikacja z obiektem graficznym odbywa się poprzez jego interfejs, składający się z trzech typów składowych. Są nimi:

- *Atrybuty* określające stan, w którym dany obiekt się znajduje. W rzeczywistości stan obiektu składa się z danych prywatnych (niewidocznych dla elementów zewnętrznych) i publicznych czyli właśnie atrybutów. Każdemu atrybutowi wraz z jego definicją przypisane zostają operacje implementujące odczyt wartości (*get*), zmianę wartości (*set*) oraz zdarzenie generowane w wyniku zmiany wartości elementu. To właśnie te elementy są wykorzystywane przez system do implementacji połączeń pomiędzy atrybutami. W ramach edycji części, wiele atrybutów stałych, do których należą m.in. kolor, wartość początkowa czy rozmiar, może zostać podana za pośrednictwem zestawu okien dialogowych.

- *Zdarzenia* sygnalizują zmianę stanu obiektu i jako takie mogą być wynikiem zmian wartości atrybutów lub wewnętrznej części obiektu (w skład której wchodzą również elementy interfejsu użytkownika). Podczas definicji aplikacji mogą zostać one wykorzystane jedynie w połączeniach i być inicjatorami akcji. Jedynymi rodzajami zdarzeń definiowalnymi przez programistę są te, których przyczną jest zmiana wartości atrybutu.

- *Akcje* stanowią interfejs proceduralny obiektu, za pośrednictwem którego można

informować obiekt o zdarzeniach zewnętrznych. Podobnie jak to miało miejsce w przypadku zdarzeń jedynymi akcjami które mogą zostać zdefiniowane przez programistę są operacje zmiany wartości atrybutu. Wszystkie pozostałe akcje są predefiniowane i specyficzne dla danego obiektu.

Istotne jest, że również obiekty takie jak logika (procedury, aplikacje) i dane mają swój interfejs. Aplikacja lub procedura może zostać wykonana (akcja), powiadomić o swoim zakończeniu (zdarzenie) i otrzymać parametry (atrybuty). Rekord składa się natomiast z elementów, które posiadają wartość (atrybuty).

Logika wysokiego poziomu w aplikacji graficznej systemu VG jest opisana za pomocą połączeń. Połączenie realizuje jedną z dwóch funkcji: synchronizuje dane lub opisuje przepływ sterowania. Pierwszy typ połączeń zapewnia propagacje wartości pomiędzy elementami, drugi ma semntykę zwykłego wywołania procedury/funkcji. Aplikacja może korzystać z czterech typów połączeń:

-*Atrybut-atrybut*, jest połączeniem synchronizującym wartości atrybutów i występuje w dwóch odmianach: jednokierunkowym kiedy propagowana jest zmiana tylko jednego z atrybutów, i dwukierunkowym gdy propagacja zmian odbywa się w obu kierunkach. Przykładem połączenia atrybut-atrybut w rzeczywistej aplikacji jest połączenie pomiędzy zawartością pola tekstowego i rekordem danych.

-*Zdarzenie-Akcja*, to najbardzej oczywiste połączenie pomiędzy elementami interfejsu opisujące sterowanie. Należy nadmienić, że jeśli w definicji danego elementu występuje wiele połączeń zdarzenie-akcja zależnych od tego samego zdarzenia, to kolejność ich wykonywania jest zdefiniowana na podstawie ich sekwencji.

-*Atrybut-Akcja* jest odmianą połączenia zdarzenie-akcja i definiuje sytuację, w której zmiana wartości atrybutu ma być inicjatorem wywołania procedury. W tym przypadku atrybut (zdarzenie oznaczające zmianę jego wartości) traktowany jest jako zdarzenie.

-*Zdarzenie-Atrybut* jest również odmianą połączenia zdarzenie-akcja, którego uaktywnienie daje w wynikumodyfikację wartości atrybutu. Nowa wartość atrybutu specyfikowana jest w postaci parametru połączenia jako wartość stałą lub dodatkowe połączenie atrybut-atrybut.

W tym miejscu warto przedstawić krótki opis interfejsu jednej z perdefiniowanych części. Jako nasz przykład niech posłuży standardowa część interfejsu użytkownika jakim jest podokno edycji (*text edit*) występujące w edytorze VisualGen pod nazwą *text*. W skład jego interfejsu wchodzą między innymi następujące atrybuty, zdarzenia i akcje:

-Atrybuty: położenie, wielkość, początkowa zawartość, liczba akceptowanych znaków,

typ danych, aktywność, rodzaj i wielkość fontu, plik pomocy, aktualna zawatość jako łańcuch znakowy, aktualna zawartość jako dana (po konwersji).

-Akcje: wstaw tekst, usuń tekst, zmień pozycję kursora, odczytaj wartość, wywołaj okno pomocy.

-Zdarzenia: okno jest otwierane, okno zostało otwarte, okno jest zamykane, okno zostało zamknięte, błędne dane, wywołanie systemu pomocy.

Jak widać, nawet w przypadku tak prostego elementu jak okno edycji liczba opcji udostępnianych przez edytor jest znaczna i pozwala na zaimplementowanie wielu sytuacji.

6. Interakcyjne narzędzie testujące (INT).

Środowisko VisualGen dopełnia proces definicji aplikacji (pojedyńczych i pracujących w układzie klient-serwer) udostępniając możliwości testowania logiki tych aplikacji jeszcze przed ich generacją. Interakcyjne narzędzie testujące (w skrócie INT) jest integralną częścią środowiska VG, pozwalającą na testowanie zarówno aplikacji tekstowych jak i graficznych z wykorzystaniem zastosowanych baz danych i plików. Programista komunikuje się z INT za pośrednictwem okna (Rys. 6) monitora, które w zależności od konfiguracji może zawierać:

Rysunek 6. Interakcyjne narzędzie testujące

kod testowanego modułu aplikacji, stos wywołań procedur i/lub podgląd zawartości obiektów aplikacji. Dostępny jest również log operacji dokonywanych przez aplikację i bogaty zestaw filtrów wybierający interesujące użytkownika jego elementy. Dodatkowo mamy do dyspozycji wiele standardowych funkcji narzędzi tego typu, takich jak: pułapki (w tym warunkowe), praca krokowa, podgląd zawartości zmiennych aplikacji. Rzadko spotykana w innych systemach jest natomiast funkcja dynamicznego przemieszczenia licznika rozkazów, co pozwala na ominięcie lub powtórne wykonanie części aplikacji.

Zastosowanie narzędzia testującego jeszcze przed generacją kodu umożliwiło zrównoleglenie procesów pisania/testowania logiki aplikacji. Standardowe podejście, w którym testowanie odbywa się dopiero po kompilacji/łączeniu bywa czasochłonne i powoduje, że aby rozpocząć proces testowania należy dysponować spójnym kodem aplikacji.W przypadku systemu VG zasymulowanie działania aplikacji możliwe jest nawet gdy kod nie jest jeszcze kompletny (np. brak elementów tworzonych przez innych członków zespołu), co w odpowiednio zorganizowanym zespole pozwoli osiągnąć wysokie zrównoleglenie prac.

7. Prezentacje

W ramach trzech części prezentacji zostaną przedstawione elementy, opcje i możliwości systemu VisualGen. Tematami kolejnych prezentacji będą:

- Aplikacja tekstowe. Krótka prezentacja możliwości pakietu VG przy definiowaniu aplikacji posiadających tradycyjny, tekstowy interfejs użytkownika (terminale). VisualGen pozwala na generację kodu na systemy, które działają w tradycyjnych konfiguracjach (terminale-serwer) i nie wymagają graficznego interfejsu użytkownika. W ramach prezentacji pokazane zostaną składowe takiej aplkacji i dokonane zostaną pżroste jej modyfikacje.

- Aplikacje graficzne. Tematem tej prezentacji jest przedstawienie graficznego edytora aplikacji i filozofii definiowania aplikacji graficznych działających w układzie klient-serwer. W zależności od dostępnego czasu dokanamy definicji nowej (od początku) lub modyfikacji istniejącej aplikacji wraz z jej testowaniem.

-Procesy generacji i parametryzacji środowiska docelowego. W ramach tej prezentacji wyganerowany i uruchomiony zostanie system napisany w czasie poprzedniego kroku. Omówione zostany proces parametryzacji i konfiguracji docelowego środowiska działania aplikacji.