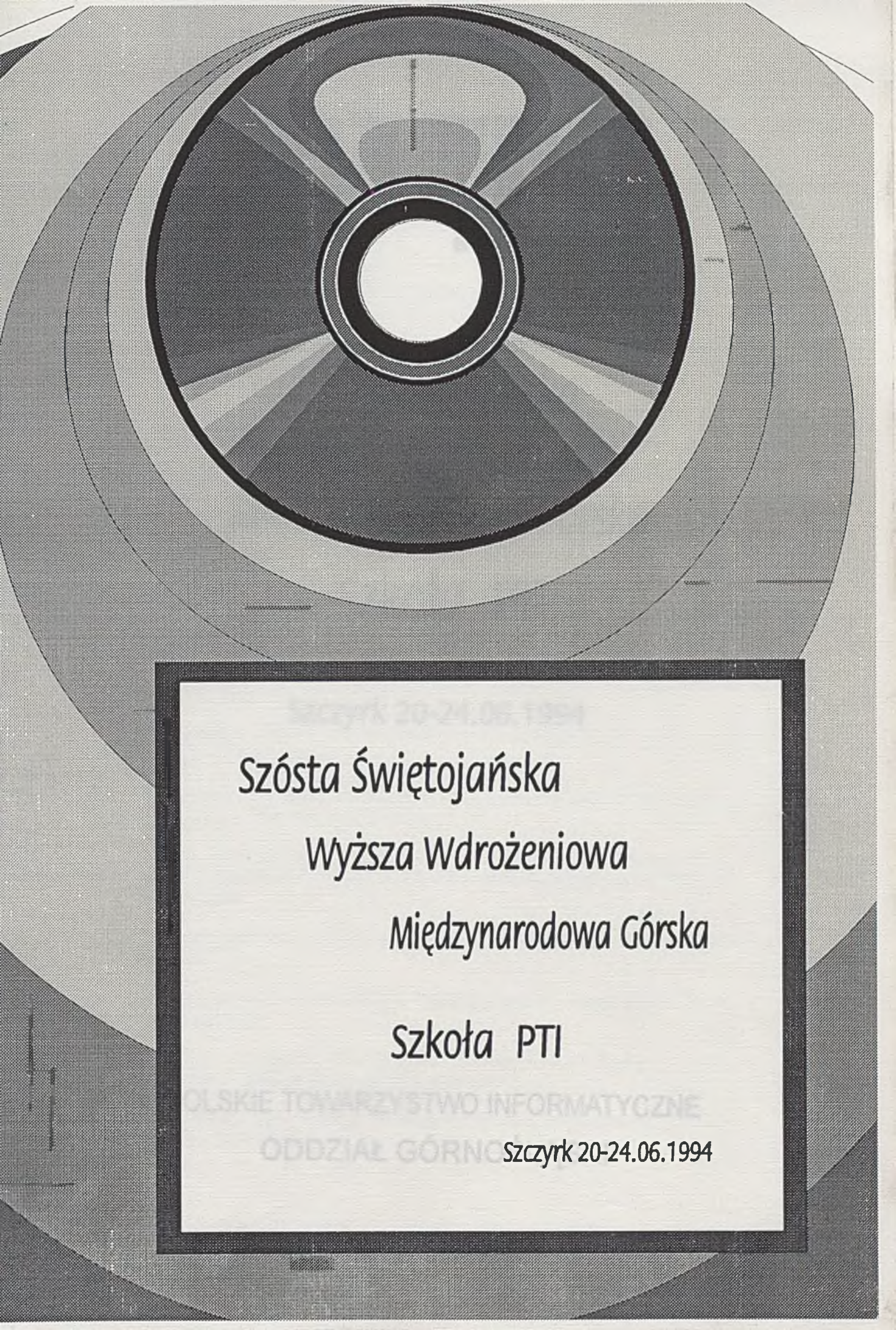
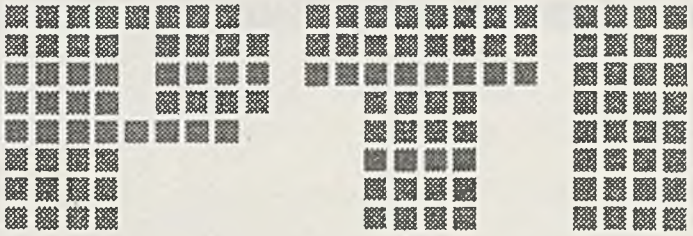


Szósta Świętojańska Wyższa Wdrożeniowa Międzynarodowa Górska Szkoła PTI

Szczyrk 20-24.06.1994



VI Świętojańska Wyższa Wdrożeniowa Międzynarodowa Górska Szkoła PTI

Szczyrk 20-24.06.1994

POLSKIE TOWARZYSTWO INFORMATYCZNE
ODDZIAŁ GÓRNOŚLĄSKI

Katowice 1994

Spis treści.

Spis treści

Janusz Krzemiński
MacroSoft Warszawa

Techniczne, organizacyjne i psychologiczne aspekty wdrażania systemów informatycznych

Reprezentuję firmę zajmującą się świadczeniem usług polegających na wytwarzaniu i wdrażaniu systemów informatycznych. Wraz z dojrzewaniem rynku usług komputerowych pojawiały się nowe wyzwania, którym należało stawić czoła. W miarę angażowania się w coraz większe projekty od problemów czysto technicznych takich jak wybór odpowiednich narzędzi programistycznych, ich konserwacja i rozwój wagi nabrały problemy organizacyjne: w jaki sposób kontrolować poszczególne etapy projektu? Jak zagwarantować zgodność funkcji realizowanych przez system z oczekiwaniami użytkownika? Jak prowadzić pracę by zmieścić się w planowanych ramach czasowych oraz zbytnio nie przekroczyć budżetu? Z doświadczeń wynika, iż nawet poprawnie organizacyjnie prowadzony projekt może zakończyć się niepowodzeniem. W takich przypadkach niebagatelną rolę odgrywają psychologiczne aspekty wdrożenia.

Jednoosobowa koordynacja.

Proces komputeryzacji dzieli się na kilka faz. Każda z tych faz, angażuje odrębne zespoły ludzi, z różnych poziomów zarządzania projektem i o różnych kwalifikacjach. Wszystkie prace są koordynowane przez Głównego Wykonawcę tematu. Do obowiązków Głównego Wykonawcy należy:

1. Faza planowania projektu.
   Konsultacje przy określaniu warunków merytorycznych zadania tzn.
   a. zakres czynności
   b. koszty
   c. terminy realizacji
2. Faza rozpoczęcia pracy.
   a. odbiór zadania (zapoznanie się z umową, załącznikami i aneksami)
   b. przygotowanie harmonogramu prac
   c. ustalenie punktów kontrolnych. Dla każdego punktu kontrolnego należy określić: co będzie zrobione, w jakiej postaci, kto i co będzie sprawdzał, termin odbioru pracy, warunki płatności.
   d. ustalenie kto będzie wykonywał określone w harmonogramie zadania.
   e. przekazanie prac wykonawcom zgodnie z harmonogramem.
3. Faza wykonywania zadania (analiza wymagań, projektowanie, implementacja, testowanie).
   a. kontakt z użytkownikiem w trakcie trwania prac.
   b. pilnowanie terminów płatności i przekazywania dokumentów.
      - przekazywanie dokumentów użytkownikowi.
      - odbieranie podpisanych kopii dokumentów.
      - uzupełnianie archiwum biura o podpisane kopie.
   c. uzgadnianie zakresu i kosztów nowych prac, sporządzanie nowych załączników do umów.
   d. sprawdzanie punktów kontrolnych.
   e. cotygodniowe sprawozdania z przebiegu prac (rozliczenie czasu pracy i merytorycznych postępów).

f. reagowanie na nieprzewidziane sytuacje w czasie trwania kontraktu, wyprzedzanie negatywnych reakcji klienta, informowanie o trudnościach.
4. Faza zakończenia zadania.
   a. archiwizacja wyników (analizy, programy, dokumentacja)
   b. przekazanie uwag dotyczących kontraktu koszty reklamacyjne, dobre oszacowanie kosztów i czasu wykonania, uwagi o wykonawcach).
5. Faza po zakończeniu zadania (pielęgnacja).
   a. nadzorowanie prowadzenia prac reklamacyjnych w ramach gwarancji.
   b. reaktywowanie kontraktu w ramach następnych płatnych zadań.

Sytuacja w komputeryzowanej firmie.

Aby przybliżyć problemy psychologiczne związane z wprowadzaniem nowych systemów należy spróbować odpowiedzieć na pytanie w jakim fazie rozwoju organizacji dyrekcja najwyższego szczebla podejmuje decyzje rozpoczęcia kompleksowej komputeryzacji. W strategii każdego przedsiębiorstwa występują trzy kolejne fazy:

1. Utrzymanie się i stabilizacja. Podstawowym założeniem tej fazy jest przede wszystkim zachowanie istnienia przedsiębiorstwa, które kierownictwu i załodze daje określone korzyści. W tym celu strategia nakierowana jest na minimalizację ryzyka i unikanie przedsięwzięć wykraczających poza „normalny" zakres działań.
2. Umocnienie. Stabilizacja prowadzi do nagromadzenia rezerw, które powodują wzmocnienie wewnętrznej struktury przedsiębiorstwa. Kształtuje się tendencja pełniejszego wykorzystania rezerw i podwyższania wyników.
3. Rozwój i ekspansja. Umocnienie przedsiębiorstwa stwarza przesłanki jego dalszego rozwoju. Następuje uruchomienie nowych rodzajów produkcji i sięgnięcie po nowe rynki. Ekspansja przedsiębiorstwa zawiera w sobie ryzyko, którego wzrost zaczyna zagrażać utrzymaniu i stabilizacji. Następuje więc przystosowanie strategii do punktu wyjścia, z tym że znajduje się on już na innym wyższym poziomie.

Faza rozwoju i ekspansji nie sprzyja efektywnej komputeryzacji przedsiębiorstw. Naturalne, silne motywacje dla komputeryzacji występują w fazie umocnienia, czyli wówczas gdy przedsiębiorstwo przez gromadzenie rezerw dostatecznie umocni własną strukturę. Wkroczenie zespołu specjalistów zajmujących się wdrażaniem systemów komputerowych do stabilnej i lepiej lub gorzej, sprawdzającej się na rynku organizacji powoduje reakcje negatywne.

Działania ułatwiające wdrożenie.

Aby skutecznie przejść przez cały proces komputeryzacji systemu informacyjnego w przedsiębiorstwie należy zwrócić uwagę na następujące problemy:

1. włączenie użytkowników systemu do zespołu projektowego;
2. rozważenie kosztów systemu;
3. przedkładanie wagi i selekcji informacji nad jej ilością;
4. wypróbowanie systemu przed wdrożeniem;
5. staranne przeszkolenie operatorów i użytkowników systemu.

1. Włączenie użytkowników systemu do zespołu projektowego. Współpraca między kierownikami operacyjnymi korzystającymi z informacji a projektantami systemu jest konieczna. Użytkownicy wiedzą jakich informacji poszukują, kiedy ich potrzebują i jak będą one wykorzystywane do działań

kierowniczych i podejmowania decyzji. Projektanci systemów często myślą inaczej i mogą nie zdawać sobie sprawy z podstawowych czynników, które się uwzględnia w decyzji operacyjnej. Jeżeli użytkownicy są odsunięci od procesu projektowania, system może im nie zapewniać potrzebnych informacji, a jednocześnie przeciążyć ich informacjami bezużytecznymi.

2. Rozważenie kosztów systemu z punktu widzenia pieniądza i czasu. Projektanci powinni przedstawić sposób opracowania systemu, czyli: terminarze poszczególnych działań, kamienie milowe do osiągnięcia i preliminiarze kosztów. Zmniejszone w ten sposób zostanie ryzyko przekroczenia kosztów. Ponadto projekt powinien wskazywać jednostki organizacyjne odpowiedzialne za wdrożenie i funkcjonowanie systemu. W ten sposób można odpowiednim kierownikom przypisać odpowiedzialność za jego koszty.

3. Przedkładanie wagi i selekcji informacji nad jej ilością. Kierownikowi potrzebna jest dostateczna ilość informacji do podjęcia świadomej decyzji; większa ilość niekoniecznie jest lepsza, chociaż wielu kierowników woli mieć raczej zbyt dużo niż zbyt mało informacji.

4. Wypróbowanie systemu przed wdrożeniem. Nawet jeśli użytkownicy i projektanci współpracują przy opracowywaniu systemu, mogą przeoczyć ważne czynniki. W okresie prób ujawnią się luki i błędy. Są cztery sposoby sprawdzania nowego systemu, które wpływają w pewnym stopniu na czas trwania i koszty etapu prób.

a. Pełne bezpośrednie wdrożenie. System od razu całkowicie zastępuje poprzedni, powstaje całkowita zależność od nowego systemu. W takim przypadku system trzeba wszechstronnie wypróbować przed wdrożeniem, gdyż organizacja nie będzie w stanie wrócić do poprzedniego, jeśli nowy system zawiedzie lub wystąpią znaczne błędy w jego funkcjonowaniu.
b. Równoległe wprowadzenie polega na tym, że nowy system wdraża się i stosuje obok dotychczasowego. Pozwala to na porównanie informacji dostarczanych przez obydwa systemy i dokonanie poprawek przed zrezygnowaniem z poprzedniego systemu. W porównaniu z pełnym bezpośrednim wdrożeniem w równoległym wprowadzaniu systemu ogranicza się liczbę prób, ale ze względu na utrzymywanie przez jakiś czas dwóch systemów informacyjnych może to być kosztowne.
c. Pilotowe wprowadzanie pozwala na wypróbowanie funkcjonowania systemu w małej części organizacji (np. w jednej z filii). Można wtedy ujawnić problemy i wprowadzić poprawki przed całkowitym wdrożeniem.
d. Segmentowe wprowadzanie umożliwia etapowe wdrażanie systemu, w którym prowadzi się próby funkcjonowania i usuwa błędy w każdym segmencie przed przejściem do następnego. Wymaga to jednak szczególnej uwagi w fazie projektowania, gdyż w trakcie procesu wprowadzania nowy i dotychczasowy system muszą ze sobą współpracować.

5. Szkolenie i przygotowanie pisemnej dokumentacji dla operatorów i użytkowników systemu. Program szkolenia kierowników i operatorów systemu jest ważny z dwóch powodów. Po pierwsze, bez przeszkolenia i pisemnych instrukcji dotyczących prowadzenia i wykorzystania systemu organizacja znajdzie się w kłopocie gdy

odejdą doświadczeni pracownicy. Po drugie, operatorzy powinni rozumieć potrzeby informacyjne kierowników różnych szczebli. Najważniejsze jest, by kierownicy rozumieli sposób funkcjonowania systemu i mogli go kontrolować, a nie żeby system kontrolował ich.

Nawet przy najstaranniejszym zaprojektowaniu i wdrażaniu może się okazać, że system nie odpowiada potrzebom użytkowników albo dlatego, że trudno jest z niego korzystać, albo że nie dostarcza wszystkich żądanych informacji. W takich przypadkach podejmowane są próby dostosowania użytkowników do systemu. Próby takie nie przyniosą powodzenia, jeśli użytkownicy nie są życzliwie nastawieni do przetwarzania danych. Stąd potrzebne jest powszechne zapoznanie pracowników z nowym systemem, z urządzeniami oraz oprogramowaniem. Należy podać do wiadomości pracowników jak nowy system wpłynie na zmiany struktury organizacyjnej, na odpowiedzialność za poszczególne zadania i na pracę każdego pracownika.

Czynniki negatywne.

Można wyróżnić pięć głównych czynników, od których zależy, czy i w jakim stopniu wdrażanie nowego systemu informatycznego napotka na opór:

1. Naruszenie istotniejszych granic między działami. Ustanowienie nowego systemu często powoduje zmiany w kilku jednostkach organizacyjnych (połączenie, zmiany zakresu obowiązków itp.). Członkowie działów mogą opierać się takim zmianom z powodu niechęci do zmiany dotychczasowego sposobu pracy lub do ludzi, z którymi pracują.
2. Zerwanie systemu nieformalnego. Wprowadzenie systemu informatycznego może spowodować zerwanie nieformalnej sieci komunikacji w wyniku zmiany układów komunikacyjnych. Jeśli członkowie organizacji wolą dotychczasowe, nieformalne mechanizmy zbierania i przekazywania informacji, mogą przeciwstawiać się nowym, bardziej sformalizowanym kanałom ustanowionym dla nowego systemu.
3. Szczególne cechy indywidualne. Ludzie pracujący w organizacji od wielu lat znają układy i wiedzą, jak działać w istniejącym systemie. Stąd mogą z większym uporem przeciwstawiać się zmianom niż ludzie, którzy w organizacji są stosunkowo krótko i nie znają tak dobrze organizacji i istniejących w niej stosunków.
4. Kultura organizacji. Jeżeli naczelne kierownictwo utrzymuje otwartą komunikację, zajmuje się zażaleniami i -- ogólnie biorąc -- wprowadza kulturę charakteryzującą się dużym zaufaniem w całej organizacji, przypuszczalnie opór wobec wdrożenia nowego systemu będzie mniejszy. Jeżeli jednak trzyma się ono z dala lub izoluje od innych członków organizacji albo jeśli kultura organizacji popiera zachowania nieelastyczne, skuteczne wdrożenie systemu zapewne będzie utrudnione.
5. Sposób wdrażania zmian. Sposób zaprojektowania i wdrażania zmian wpływa na siłę sprzeciwu, jaki zmiany te napotkają. Ogólnie biorąc, jeśli decyzje o zmianach podejmowane są wspólnie przez kierowników i pracowników, istnieje większe prawdopodobieństwo zaakceptowania zmian.

Reakcje negatywne.

Frustracja związana z wdrażaniem nowego systemu może przejawiać się w trzech postaciach: agresji, projekcji i unikania. Agresja polega na ataku na przedmiot (lub osobę), który wywołuje frustrację. Agresja w stosunku do systemu komputerowego

przybiera nawet formę sabotażu – niewłaściwego użytkowania urządzeń, wprowadzania do systemu niepewnych lub nieodpowiednich danych albo niszczenia wyposażenia lub oprogramowania. Projekcja jest psychologicznym obciążeniem niepowodzeniami kogoś lub czegoś innego. Projekcja występuje wtedy, gdy pracownicy winią system komputerowy za błędy popełnione w wyniku ludzkich pomyłek lub z innych przyczyn nie związanych z systemem. Unik występuje wtedy, gdy ludzie bronią się przez wycofanie się lub unikanie frustrującej sytuacji. Kierownicy stosują uniki, gdy pomijają informacje uzyskiwane z systemu na rzecz własnych źródeł informacji.

Ludzie na różnych szczeblach organizacyjnych w różny sposób odczuwają skutki komputeryzacji. Częstotliwość występowania poszczególnych rodzajów zachowań zależy od szczebla zajmowanego w hierarchii przez osoby dotknięte zmianami wynikającymi z wprowadzania systemu komputerowego. Na przykład wprowadzenie systemu może utrudnić pracę pracowników pozabiurowych dostarczających dane do systemu wywołując agresję. Pracownicy biurowi zatrudnieni bezpośrednio przy nowym systemie szczególnie odczuwają skutki wprowadzenia komputerów. Eliminacja stanowisk pracy czy zmiana układu pracy mogą prowadzić do reakcji projekcji. Z punktu widzenia kierownictwa operacyjnego problem polega na tym, że system ogranicza ich wpływ na to kiedy i jak informację się przesiewa, interpretuje i przedstawia przełożonym. Utrata kontroli nad informacją jest źródłem zaniepokojenia kierowników operacyjnych. Innym powodem niepokoju kierowników jest możliwość większej centralizacji podejmowania decyzji przez naczelne kierownictwo wyposażone w aktualne i syntetyczne informacje otrzymywane z systemu. Sprawny system wzmacnia kontrolę nad kierownictwem średniego szczebla. Ponadto zawsze istnieje obawa, że usprawnienie organizacji pracy wyeliminuje lub w znaczny sposób zmieni niektóre kierownicze stanowiska pierwszej linii i średniego szczebla. Tak więc opór kierowników operacyjnych wobec nowego systemu informatycznego może obejmować wszystkie trzy rodzaje reakcji psychologicznych: zwalczanie systemu (agresja); ignorowanie go przez utrzymywanie swoich dawnych kanałów komunikacyjnych (uniki); albo obciążanie systemu winą za błędy, powodowane przez inne czynniki (projekcja).

Na ogół w wielu organizacjach naczelne kierownictwo nie odczuwa konsekwencji wdrożenia systemu i stosunkowo niewiele się nim przejmuje, choć dzięki niemu może podejmować bardziej świadome decyzje. Unika ono jednak czynnego zaangażowania się w system, być może ze względu na brak znajomości komputerów lub wiążące się z nimi uczucie niepewności. Ten brak zaangażowania i poparcia ze strony naczelnego kierownictwa często zaostrza trudności innych członków organizacji w związku z jego wdrożeniem.

W innych wypadkach naczelne i średnie kierownictwo mocno angażuje się w proces projektowania i wprowadzania systemu. Projekt jest realizowany w politycznym systemie organizacji, a pozaracjonalne względy projektantów oraz podgrup użytkowników mogą wywierać wpływ na końcowy projekt, wdrożenie, wykorzystanie oraz reakcje i zachowania organizacji wobec nowego systemu. Na przykład zapotrzebowanie na informacje dotyczące filii może być odmienne w przypadku średniego i naczelnego kierownictwa. Informacje najbardziej przydatne dla członków naczelnego kierownictwa ułatwiają też ocenę efektywności filii, to zaś może być traktowane przez kierowników średniego szczebla jako zagrożenie ich bezpieczeństwa. Obie grupy będą manewrować w trosce o ochronę własnych interesów. Po zakończeniu

projektowania zwycięzcy i zwyciężeni w tej batalii politycznej mogą przyjąć odmienne postawy wobec systemu. Reasumując można wymienić następujące powody dla których pracownicy różnych szczebli organizacyjnych przedsiębiorstwa są szczególnie skłonni stawiać opór wobec komputeryzacji:

1. zagrożenie bezpieczeństwa ekonomicznego,
2. zagrożenie pozycji lub władzy,
3. większa komplikacja pracy,
4. niepewność lub nieznajomość,
5. zmiana stosunków międzyosobowych lub wzorców pracy,
6. zmiana stosunków między przełożonym a podwładnym,
7. wyższy stopień formalizacji i terminowości,
8. poczucie zagrożenia.

Przezwyciężanie trudności.

Istnieje wiele sposobów przezwyciężania wymienionych przeszkód występujących przy wdrożeniu. Najlepiej dla każdej sytuacji z osobna sporządzić diagnozę i przepisać indywidualną kurację. Jako najważniejsze czynniki przy przezwyciężaniu trudności związanych z wdrożeniem można wymienić:

1. Orientacja na użytkownika. Być może najważniejszym krokiem w przezwyciężaniu tych trudności jest zapewnienie orientacji na użytkownika zarówno w fazie projektowania, jak i wdrażania. Jeśli wykorzystanie systemu wymaga od użytkownika więcej niż minimum dostosowań i uczenia się, uparcie i logicznie będą się oni trzymać swoich systemów. Mniejsze jest prawdopodobieństwo występowania takich trudności, jeśli od początku prac nad projektem aktywnie zaangażuje się przyszłych użytkowników. Projektantów należy nagradzać za zaspokojenie potrzeb użytkowników tak samo, jak za dotrzymywanie terminów.
2. Uczestnictwo. Wiele trudności da się wyeliminować lub w ogóle ich uniknąć, jeśli przyszli użytkownicy będą członkami zespołu projektowego. W szczególności kierownicy operacyjni powinni mieć znaczny głos w sprawach: dysponowania informacją, ewentualnych modyfikacji zadań poszczególnych stanowisk pracy. Udział kierowników i pracowników wszystkich szczebli zapewni to, że projektanci będą mogli dobrze poznać i uwzględnić ich potrzeby. Pracownicy mający możliwość przedstawienia swej opinii na temat zmian w ich układzie pracy i zadaniach będą mieli większe zaufanie do jakości i użyteczności nowego systemu.
3. Informowanie o systemie. Należy jasno określić i zakomunikować wszystkim użytkownikom systemu jego cele i cechy. Zadanie to jest szczególnie trudne, gdyż system ewoluuje w miarę jego projektowania, a zatem na początku jego ostateczna postać nie jest jeszcze dokładnie znana. Jednakże bez zrozumienia podstawowych cech i celów systemu występować będą ciągle różnice zdań między członkami zespołu projektowego, a użytkownikami systemu. Ustanowienie wyraźnych i czynnych linii komunikowania się jest kluczem do poznania potrzeb użytkowników, do ustalenia ewentualnych konsekwencji organizacyjnych i zapewnienia tego, by wszyscy zainteresowani pracownicy mieli poczucie uczestnictwa.
4. Redefinicja miar efektywności. Nowy system może tak zmienić zadania poszczególnych pracowników, że stare oceny metody efektywności staną się bezprzedmiotowe. Pracowników należy zapoznać z nowymi metodami oceny, aby wiedzieli, w jaki ich osiągnięcia będą mierzone i nagradzane.

5. Nowe wyzwania. Świadomość, że komputer może wykonać wiele z tego, co robi pracownik – i to szybciej, a być może gruntownie -- w dużym stopniu przyczynia się do poczucia zagrożenia, jakie może wywołać komputeryzacja. Jednym ze sposobów ograniczenia tego zagrożenia jest rozpowszechnienie wyzwań, które stają się możliwe dzięki systemowi. Może on na przykład uwolnić kierownictwo średniego szczebla od wielu nudnych, rutynowych zadań. Kierownicy mogą więc zyskać szansę odgrywania większej roli w takich sprawach, jak planowanie długookresowe, które dotychczas stanowiło raczej wyłączną domenę kierowników wyższych szczebli. System może też stwarzać możliwości korzystania z dostarczanych przez niego informacji bardziej twórczo i efektywnie.

Tomasz Kossut
"MacroSoft"

MacroBASE dla systemu UNIX

1. Architektura systemu użytkowego

Wszystkie programy użytkowe powstałe w oparciu o system zarządzania bazą danych **MacroBASE** mają jednolitą architekturę. Elementy takiego programu użytkowego przedstawiają się następująco:

1. Definicja systemu.
2. Moduł wykonawczy.
3. Maszyna bazy danych.
4. Podprogramy zewnętrzne.
5. Ochrona bazy danych.
6. Dane użytkownika.
7. Historia bazy danych.
8. Drajwery urządzeń zewnętrznych.
9. Teksty programu wykonawczego.
10. Deklaracja ścieżek systemu.

1.1. Definicja systemu

Definicja systemu użytkowego (inaczej słownik bazy danych) jest tworzona przez projektanta-programistę przy pomocy generatora zastosowań **MacroGEN.** Definicja systemu zawiera w szczególności:

- schemat bazy danych,
- sposoby prezentacji bazy danych,
- przepływ sterowania programu,
- mechanizmy ochrony,
- podprogramy w języku **FORMULA 4GL**

1.2. Synchronizacja dostępu

Podczas działania systemu użytkowego w środowisku wielodostępnym musi być zapewniony bezkolizyjny i sprawiedliwy dostęp do bazy. Żądania poszczególnych użytkowników wykonywania operacji na danych mogą nadchodzić częściej, niż możliwa największa częstotliwość ich realizacji. Takie żądania są z punktu widzenia bazy danych jednoczesne. Należy więc ustawić je w kolejce i sukcesywnie wykonywać. Musi istnieć mechanizm organizujący taką kolejkę.

W systemie zarządzania bazą **MacroBASE** jest to oddzielny program. Żądanie wykonania operacji na bazie danych jest przesłane do procesu *serwera*, wstawione do kolejki, pobrane i wykonane. Informacja o wyniku operacji jest przekazywana do procesu *klienta* żądającego jej wykonania.

Dzięki takiemu rozwiązaniu można zgrupować wszystkie elementarne operacje dostępu do bazy danych (do poziomu fizycznego włącznie) w jednym procesie, bez konieczności ich powielania w każdym z wielu procesów programu obsługującego końcowego użytkownika.

1.3. Program wykonawczy

Zbiór **MacroCLIENT** jest modułem wykonawczym uruchamianym przez końcowego użytkownika. Niezależnie od liczby wykonywanych definicji systemu i liczby obsługiwanych stanowisk do pamięci komputera ładowana jest tylko jedna kopia tego programu. Uruchomienie systemu użytkowego polega na wykonaniu komendy **MacroCLIENT** z nazwą słownika bazy danych jako parametrem.

Tak uruchomiony program wykonawczy interpretuje podaną definicję systemu. Zbiór **MacroCLIENT** tworzą między innymi następujące moduły:

- algorytmiczna i dialogowa warstwa metody dostępu,
- niezależne od systemowego zabezpieczenia danych przed nieuprawnionym dostępem,
- interpretator języka **FORMULA 4GL**,
- wykonawca wstępnie przetworzonych wzorców wydruków języka **REPORT**,
- pełnoekranowy edytor,
- wyświetlanie komunikatów i pomocy kontekstowej,
- zarządzanie okienkami i menu,
- obsługa terminali i drukarek,
- rejestrowanie historii bazy.

Procesy programu wykonawczego komunikują się z procesem maszyny bazy danych **MacroSERVER** poprzez przesyłane komendy. Oczekując na wykonanie komendy usypiają nie używając zasobów systemu komputerowego.

1.4 Maszyna bazy danych

Program **MacroSERVER** jest odpowiedzialny za dostęp do danych i kolejkowanie zadań. Wykonuje kolejno elementarne komendy dostępu do bazy danych pełniąc zarazem funkcję arbitra. Uruchomienie tego programu musi być wcześniejsze niż pierwsze wywołanie programu **MacroCLIENT**. Zakończenie jego pracy jest wykonywane z poziomu systemu operacyjnego i uniemożliwia dalszą pracę programom użytkowym. Zazwyczaj maszyna bazy danych działa jako proces uruchomiony w tle.

Proces programu **MacroSERVER** komunikuje się z procesami programu wykonawczego **MacroCLIENT** poprzez przesyłane komendy. Pomiędzy wykonywaniem komend usypia nie używając zasobów systemu komputerowego.

Program maszyny bazy danych zawiera między innymi mechanizmy:

- blokowania tabel na trzech poziomach wyłączności,
- definiowania transakcji z gwarancją zachowania spójności danych zarówno przy jej wycofaniu, jak i awarii systemu,
- blokowania zapisu i odczytu rekordów,

1.5 Podprogramy zewnętrzne

System zarządzania bazą danych **MacroBASE** pozwala dołączać zewnętrzne podprogramy pod warunkiem, że definicja systemu dopuszcza taką możliwość.

Formuły

Interpretator języka **FORMULA 4GL** może wykonywać procedury zawarte w zewnętrznych zbiorach tekstowych. Język ten pozwala na pełny dostęp do zasobów systemu użytkowego. Posiada zaawansowane mechanizmy obiektowego tworzenia procedur i bogatą bibliotekę funkcji standardowych.

Wzorce wydruków

Wzorce wydruków są przygotowywane w postaci plików tekstowych zgodnie ze składnią języka generowania sprawozdań **REPORT**. Język ten jest wyposażony w wygodny mechanizm opisu strony oraz efektywną iterację po tabelach bazy. Umożliwia włączanie procedur **FORMULI 4GL**. Sprawozdania są wstępnie przetwarzane przy pomocy translatora **MacroREPORT**, co pozwala wyeliminować błędy składni i znacznie przyspieszyć ich realizację na etapie wykonania. Skompilowane sprawozdania przechowywane są w osobnych zbiorach.

1.6 Poufność dostępu do bazy

Standardową możliwość zapewnienia poufności danych daje system ochrony przy pomocy haseł. Chronione są akcje dokonywane na bazie danych przez operatora.

Podczas tworzenia definicji systemu użytkowego można przypisać do akcji okienek i menu uprawnienia chronione hasłami. Powiązania nazw użytkowników z ich uprawnieniami i hasłami zapisywane są w specjalnym zbiorze ochron tworzonym podczas generowania słownika. Informacja o ochronie jest wpisana do definicji.

Użytkownik rozpoczynając pracę z programem użytkowym musi się zarejestrować podając swoją nazwę i hasło. System zostanie skonfigurowany tak, jak na to pozwalają prawa dostępu danego użytkownika. Może się to przejawiać na przykład w nieobecności pewnych pozycji w menu, czy też zamazaniu na ekranie zawartości niektórych pól.

Administrator programu użytkowego ma do dyspozycji narzędzie do zmiany haseł --- program **MacroPASS**.

1.7 Fizyczna organizacja danych

Dane użytkownika zapisywane są przy zachowaniu zasady jedna tabela --- jeden zbiór. Informacje związane z uporządkowaniem zapisów w każdej tabeli przechowywane są w zbiorach indeksowych. Nazwy zbiorów są nadawane na podstawie definicji systemu.

1.8 Historia bazy danych

Podczas działania systemu użytkowego zachodzą zdarzenia, których znajomość jest konieczna dla jego prawidłowej obsługi. Istnieje potrzeba tworzenia ich historii wraz z opisem kontekstu wystąpienia. Służy do tego osobny zbiór rejestrujący. Krótki opis każdego istotnego zdarzenia jest dopisywany na koniec tego zbioru.

W historii bazy danych między innymi jest zapisywane:

- zarejestrowanie i wyrejestrowanie każdego użytkownika,
- błędy wykonania,
- użycie wybranych funkcji administracyjnych.

1.9 Drajwery urządzeń zewnętrznych

Program **MacroCLIENT** obsługuje terminale i drukarki jako urządzenia logiczne posiadające:

- własności opisujące urządzenie fizyczne zainstalowane w systemie UNIX, takie jak sekwencje inicjujące, przemieszczanie kursora czy ustawienie fontu,
- własności specyficzne dla działania systemu **MacroBASE**, takie jak tablice translacji znaków, atrybuty okienek czy podgląd drukowania.

Wybór drajwera terminala dokonywany jest na początku pracy systemu użytkowego na podstawie wartości zmiennej systemowej **MACROTERM.** Właściwy drajwer jest niezbędny do działania programu użytkowego.

Wybór drajwera drukarki dokonywany jest na podstawie informacji zawartych we wzorcu wydruku przy uruchomieniu wbudowanego w program wykonawczy generatora sprawozdań. Prawidłowy wybór drajwera jest konieczny do wykonania wydruku.

Do tworzenia drajwerów służą odpowiednie translatory **MacroTERMINAL** i **MacroPRINTER**.

1.10 Teksty programu wykonawczego

Treści komunikatów i pomoc kontekstowa dla programu wykonawczego przechowywane są w zewnętrznych zbiorach. Możliwa jest ich wymiana (np. przy zmianie wersji językowej) bez zmiany programu wykonawczego **MacroCLIENT**.

1.11 Deklaracja ścieżek

Informacja o rozmieszczeniu zbiorów użytkowych w systemie plików jest zapisana w zbiorze deklaracji ścieżek. Po uruchomieniu programu **MacroCLIENT** zbiór ten jest poszukiwany w katalogu bieżącym.

2. Generator zastosowań

2.1 Opis ogólny i funkcje generatora

Czym nie jest MacroGEN?

Generator **MacroGEN** nie jest językiem programowania, lecz dialogowym narzędziem tworzenia i pielęgnowania gotowych systemów użytkowych. Nie jest on językiem programowania trzeciej generacji, a więc językiem takim, jak Pascal, C czy COBOL. Wynikiem jego pracy nie jest program wykonawczy, ale definicja systemu użytkowego zapisana w zbiorze binarnym.

Czym jest MacroGEN?

Generator **MacroGEN** jest narzędziem wspomagającym pracę programisty budującego komputerowy system użytkowy klasy *baza danych*. Ułatwia projektowanie i prototypowanie tworzonych systemów dzięki wykorzystaniu podejścia **4GL**, w którym projektant koncentruje się na funkcjach systemu, a nie na rozwiązywaniu problemów technicznych implementacji. Umożliwia zapisanie wszelkich struktur przechowywania i redagowania danych oraz struktur pozwalających na sterowanie przebiegiem pracy programu. Informacje potrzebne do zdefiniowania systemu są wprowadzane dialogowo. Pewna część informacji dotyczy przetwarzania i obliczania. Operacje te zapisuje się w języku **FORMULA 4GL** i wprowadza jako procedury. Wynikiem sesji pracy z generatorem **MacroGEN** jest tworzona przez ten program definicja, zawierająca wszystkie elementy budowanego przez programistę systemu. Działanie systemu użytkowego polega na interpretowaniu jej zawartości.

Słownik bazy danych

Słownik bazy danych (inaczej definicja bazy danych bądź definicja systemu) jest zbiorem binarnym niezbędnym do działania programu głównego. Zbiór ten zawiera informacje o:

- tabelach bazy danych, ich schematach i uporządkowaniu,
- złączeniach tabel,
- sterowaniu przebiegiem programu,
- procedurach języka **FORMULA 4GL**,
- sposobach prezentacji ekranowej.

Wymagania systemowo-sprzętowe

Generator **MacroGEN** jest programem utworzonym w technologii **MacroBASE** w wersji 7.0 dla systemu operacyjnego DOS. Definicja bazy danych, jako produkt tego programu jest zbiorem formatu DOS. Należy ją przenieść jako zbiór binarny w środowisko systemu UNIX w celu jej dalszego wykorzystania. Możliwe jest uruchomienie generatora bezpośrednio w systemie UNIX, przy wykorzystaniu odpowiedniego programu do emulowania środowiska systemu DOS (np. DOS Merge, VP/ix).

Minimalne wymagania sprzętowe, to komputer zgodny z IBM AT.

2.2 Definiowanie tabel i złączeń

Systemowe typy danych

Lista systemowych (wbudowanych) typów danych przedstawia się następująco:

- DATE --- typ definiujący datę,
- HEADER --- typ nagłówkowy (tylko do prezentacji ekranowej),

- INTEGER --- typ całkowitoliczbowy (zakres od -2.147.483.648 do 2.147.483.647),
- MEMO --- typ tekstowy zmiennej długości.
- REAL --- typ zmiennoprzecinkowy (zakres od -1.7E+308 do 1.7E+308),
- STRING --- typ znakowy o zadanej długości,
- TIME --- typ definiujący odstęp czasowy.

Typy te są dostępne dla programisty podczas określania typów dla pól tabeli lub zmiennej.

Tworzenie tabel

Generator **MacroGEN** pozwala zarówno na definiowanie tabel rzeczywistych służących jako bufory odczytywania i zapisywania danych na dysku, jak również zmiennych strukturalnych istniejących jedynie w trakcie działania programu wykonawczego. Tabela może być maskowana, czyli związana z wieloma zbiorami danych o jednakowej strukturze. W takim przypadku w trakcie działania programu użytkowego jest możliwe przełączanie się pomiędzy tymi zbiorami.

Utworzenie tabeli bądź zmiennej polega na podaniu jej pól. Opis pola zawiera:

- nazwę i ewentualnie długość pola (do prezentacji ekranowej),
- akronim (do wykorzystania przy odwołaniu programowym),
- typ (systemowy lub złączeniowy),
- wzorzec redagowania pola.

Dla zmiennych programista może zdefiniować okienka redagowania, zaś dla tabel dodatkowo okienka wertowania oraz klucze indeksowe.

Złączeniowe typy danych

Złączeniowe typy danych są tworzone i dopisywane do listy typów przez generator **MacroGEN**. Po zdefiniowaniu tabeli następuje automatyczne dopisanie typu reprezentującego tę tabelę do słownika typów danych. Złączeniowy typ pola tabeli lub zmiennej jest realizowany jako wskazanie na rekord połączonej tabeli. Pozwala wykonać złączenie typu „jeden do wiele". Pole typu złączeniowego może być użyte:

1. przy dołączaniu pola do klucza sortowania,
2. przy dołączaniu pola do okienka,
3. przy odwołaniu do pola w języku **FORMULA 4GL**.

2.3 Tworzenie kluczy uporządkowania

Klucz uporządkowania (klucz indeksowy, indeks) określa uporządkowanie danych dla jednej tabeli. Jedna tabela może mieć kilka indeksów. Można definiować klucze indeksowe oraz nadawać im atrybuty *unikalny* (wymaganie różnych kluczy dla różnych zapisów w tabeli) bądź *ukryty* (wybór klucza jako bieżącego niedostępny w trakcie dialogu operatora). Definiowanie polega na podaniu kolejnych pól klucza wybieranych spośród pól tabeli. Dostępne są zarówno pola bieżącej tabeli, jak i pola tabeli połączonej poprzez pole typu złączeniowego. Na etapie wykonania klucz sortowania konstruowany jest dynamicznie na podstawie zawartości bieżącego rekordu. W przypadku pola typu złączeniowego odczytywany jest najpierw rekord połączonej tabeli, na który wskazuje zawartość tego pola, a do klucza wstawiane jest odpowiednie pole tabeli w złączeniu.

2.4 Okienka

Okienko wertowania służy do przeglądania tabeli. W przypadku niepustej dziedziny wyróżnia rekord bieżący. Możliwe jest:

- wykonywanie akcji standardowych (Szukaj, Dołącz, Usuń, Popraw,

Kolejność),
- wykonywanie akcji standardowych zmodyfikowanych przez procedury **FORMULI 4GL**,
- wykonywanie akcji zaprogramowanych w **FORMULI 4GL**.

Okienko redagowania służy do redagowania pojedynczego rekordu tabeli.

W trakcie działania programu użytkowego kolejno wywoływane okienka są wyświetlane na poprzednich. Okienka wywołane wcześniej mogą być częściowo lub całkowicie niewidoczne aż do momentu zdjęcia okienek je przykrywających.

Definiowanie okienka polega na podaniu tytułu, zredagowaniu listy pól oraz zaprojektowaniu jego pozycji na ekranie. Pola mogą pochodzić z dowolnej tabeli lub zmiennej. Jeśli pole jest typu złączeniowego, należy dokonać wyboru pola reprezentującego je na ekranie spośród pól tabeli połączonej. Możliwe jest zarówno automatyczne jak i ręczne rozmieszczenie pól w okienku.

Słownikowanie pól

Dla pola typu złączeniowego reprezentowanego w okienku przez pole tabeli połączonej program wykonawczy udostępnia wygodny mechanizm redagowania słownikowanego. Jest ono wspomagane przez okienko wertowania tej tabeli. Po zatwierdzeniu redagowania takiego pola w buforze znajduje się wskazanie na rekord tabeli połączonej, której pole zostało wybrane. Natomiast przy usuwaniu rekordu tabeli połączonej program wykonawczy sprawdza, czy nie zostanie osierocone wskazanie na ten rekordu z pewnego pola typu złączeniowego. Dodatkowo przed wyświetleniem zawartości pola reprezentującego pole typu złączeniowego, jego zawartość jest odświeżana poprzez odczytanie rekordu, na który wskazuje pole typu złączeniowego.

Dla pól typu systemowego prowadzona jest historia ostatnio zredagowanych wartości.

2.5 Projektowanie sterowania

Tworzenie menu programu jest możliwe zarówno statycznie przy pomocy generatora **MacroGEN** jak i dynamicznie przy wykorzystaniu **FORMULI 4GL**. Osobne menu tworzy się również dla każdego okienka wertowania. Proponowany jest zestaw akcji standardowych, który może być dowolnie zmieniany bądź rozszerzany.

2.6 Ochrona przed nieuprawnionym dostępem

Generator słownika bazy danych pozwala zabezpieczyć przed nieuprawnionym dostępem wybrane funkcje programu.

Definiowanie listy uprawnień

Lista uprawnień zawiera nazwę i opis każdego użytkownika, jego hasło oraz klucz zabezpieczeń. Do uruchomienia programu wykonawczego konieczna jest znajomość nazwy i hasła. Klucz zabezpieczeń służy do ochrony przed nieuprawnionym dostępem.

Ochrona akcji menu systemu

Ochrona ta dotyczy menu każdego okienka wertowania oraz menu głównego programu. Związanie klucza zabezpieczeń z pewną pozycją menu powoduje, że ta pozycja oraz ewentualne inne w stosunku do niej podrzędne stają się niewidoczne a przez to niedostępne.

Ochrona dostępu do prezentowanych pól

Ochrona ta dotyczy możliwości redagowania bądź oglądania pól tabel i zmiennych. Można ją wprowadzić statycznie przy pomocy generatora **MacroGEN** lub dynamicznie podczas działania programu wykonawczego przy użyciu **FORMULI 4GL**. Celowe jest

wykorzystanie przy tym klucza zabezpieczeń bieżącego użytkownika.

3. Dostęp do danych

3.1 Fizyczna organizacja danych

W systemie **MacroBASE** dane użytkownika zapisywane są przy zachowaniu zasady jedna tabela — jeden zbiór. Plik danych składa się z rekordów stałej długości. Struktura rekordu jest utworzona na podstawie listy pól tabeli zdefiniowanej przy użyciu generatora **MacroGEN**. Kolejność pól w rekordzie jest zgodna z kolejnością pól na tej liście. Poszczególne typy pól mają następującą reprezentację:

- **DATE** --- 2 bajty rok, 1 bajt miesiąc, 1 bajt dzień,
- **HEADER** --- bez reprezentacji (tylko do prezentacji ekranowej),
- **INTEGER** -- 4 bajty,
- **MEMO** --- bez reprezentacji w bieżącym rekordzie, dane z takiego pola są przechowywane w osobnej tabeli, której rekordy zawierają opis miejsca odwołania oraz kolejne fragmenty tekstu.
- **REAL** --- 8 bajtów,
- **STRING** --- typ znakowy o zadanej długości zakończony dodatkowym znakiem separatora długości 1 bajta,
- **TIME** --- 2 bajty liczba godzin, 1 bajt liczba minut, 1 bajt liczb sekund.
- typ złączeniowy --- 4 bajty na numer rekordu w połączonej tabeli.

Informacje związane z uporządkowaniem zapisów w każdej tabeli przechowywane są w zbiorach indeksowych przy zachowaniu zasady jeden indeks --- jeden zbiór. W zbiorach indeksowych przechowywane są jedynie numery rekordów zbioru danych reprezentującego tabelę. Kolejność wg danego indeksu jest zapisana jako kolejność numerów w odpowiednim zbiorze indeksowym. Zatem pierwszym rekordem tabeli wg danego indeksu jest rekord, którego numer jest pierwszym w odpowiednim zbiorze indeksowym, a ostatni rekord ma numer, który w tym zbiorze indeksowym jest numerem ostatnim.

Usunięcie rekordu polega na zamazaniu jego zawartości, usunięciu jego numeru ze zbioru indeksowego oraz zapisaniu jego numeru w specjalnym zbiorze numerów rekordów usuniętych z danej tabeli. Natomiast przy dołączeniu rekordu sprawdza się w pierwszej kolejności czy jest miejsce po usuniętym rekordzie (tj. czy zbiór numerów rekordów usuniętych jest niepusty). Jeśli tak, to rekord zostaje zapisany na odpowiedniej pozycji w zbiorze danych, numer tej pozycji zostaje usunięty ze zbioru numerów rekordów usuniętych i zapisany na odpowiednim miejscu w zbiorze indeksowym. Jeśli nie, to rekord zostaje dopisany na koniec zbioru danych, a numer tej pozycji wstawiony w odpowiednim miejscu zbioru indeksowego.

Taka organizacja danych i indeksów ma szereg zalet:

1. Pozwala rozwiązać problem odzyskiwania miejsca po usuniętych danych. Miejsce po rekordzie usuniętym jest wykorzystane przy dołączeniu nowego.
2. Pozwala na znaczącą oszczędność pamięci masowej. Nie są przechowywane klucze sortowania zbiorów danych. Inne rozwiązania, w których klucze są przechowywane wymagają przeciętnie od 30% do 50% obszaru dyskowego więcej.
3. Pozwala na efektywny dostęp do danych bez użycia indeksów. Możliwe jest dołączanie i modyfikowanie rekordów z opóźnionym aktualizowaniem indeksów oraz przeglądanie tabeli i wyszukiwanie rekordów wg ich organizacji fizycznej.
4. Pozwala ograniczyć liczbę utrzymywanych indeksów. W szczególności nie jest potrzebny indeks do realizowania złączenia tabel. Numer rekordu w

złączeniu jest dostępny bezpośrednio.

5. Obsługa indeksów przez wydzielony proces maszyny bazy danych pozwala na ich efektywne buforowanie. W praktyce przez cały czas działania tego procesu wszystkie zbiory indeksowe są umieszczone w specjalnie do tego przeznaczonej stronicowanej pamięci podręcznej.

Program wykonawczy potrafi odtworzyć zawartość zbiorów indeksowych i zbioru numerów rekordów usuniętych na podstawie zawartości danych. Nazwy wszystkich zbiorów są nadawane na podstawie definicji systemu, a ich rozmieszczenie w systemie plików na podstawie zbioru deklaracji ścieżek.

3.2 Restrukturyzacja danych

Generator zastosowań pozwala na szybkie i skuteczne dostosowanie istniejącej definicji do wymagań końcowego użytkownika. Może się to okazać konieczne przy adaptacji standardowej wersji oprogramowania, a także przy pojawieniu się nowych potrzeb po pewnym okresie eksploatacji. Użytkownik ma prawo oczekiwać, że dane wprowadzone wg poprzedniej definicji będą dostępne również przy użyciu jej zaktualizowanej wersji. Powinno to być możliwe nawet po daleko idących zmianach w schemacie bazy danych.

System **MacroBASE** zawiera narzędzie do restrukturyzacji danych użytkownika. Służy do tego celu program **MacroTRANSFER**. Można przy jego pomocy dokonać na podstawie starej i nowej definicji aktualizacji struktury zbiorów danych łącznie z wykonaniem konwersji związanej ze zmianami typów. Możliwe jest wstępne uzyskanie raportu o zakresie potencjalnych zmian.

Prosta organizacja fizyczna w systemie **MacroBASE** sprawia, że restrukturyzacja danych jest wykonywana bardzo efektywnie. W przypadku błędu w trakcie działania programu **MacroTRANSFER** (błędny dostęp do zbioru czy brak miejsca na dysku) wszystkie zmiany w bazie są automatycznie wycofywane.

3.3 Blokowanie tabel

System zarządzania bazą danych **MacroBASE** pozwala na blokowanie całych tabel. Dostępne są trzy sposoby blokowania:

- tabela niedostępna dla innych (wyłączny dostęp).
- tabela dostępna dla innych tylko do odczytu (wyłączny zapis),
- tabela dostępna dla wszystkich tylko do odczytu (dzielony odczyt),

Proces użytkownika, który ma **wyłączny dostęp** do tabeli może odczytywać i zmieniać jej dane. Dostęp do tej tabeli przez pozostałych użytkowników jest zabroniony --- nie mogą jej odczytać, dokonać zmian ani zablokować. Blokowanie w tym trybie jest możliwe tylko wtedy, gdy tabela nie jest zablokowana ani w ten ani inny sposób.

Proces użytkownika, który ma prawo **wyłącznego zapisu** do tabeli może odczytywać i zmieniać jej dane. Dostęp do tej tabeli przez pozostałych użytkowników jest ograniczony do odczytu --- nie mogą w niej dokonać zmian ani zablokować. Blokowanie w tym trybie jest możliwe tylko wtedy, gdy tabela nie jest zablokowana ani w ten ani inny sposób.

W przypadku gdy proces jakiegoś użytkownika uzyskał prawo **dzielonego odczytu** do tabeli, odczytywać jej zawartość mogą wszyscy użytkownicy. Zmiany w tej tabeli są niedozwolone dla wszystkich łącznie z użytkownikiem, który tę blokadę nałożył. Tabela w tym trybie może być blokowana przez więcej niż jednego użytkownika. Jakiekolwiek zmiany będą możliwe dopiero wówczas, gdy wszystkie blokady zostaną zdjęte. W przypadku gdy tabela jest

zablokowana w innym trybie, to uzyskanie **dzielonego odczytu** nie jest możliwe.

Aby powiodło się nałożenie blokady w trybie **wyłącznego dostępu** lub **wyłącznego zapisu**, nie może być zablokowana ani cała tabela ani żaden jej rekord (w wyniku działania transakcji).

Blokowanie tabeli w trybie **dzielonego odczytu** jest możliwe, gdy żaden użytkownik nie posiada prawa **wyłącznego dostępu** ani **wyłącznego zapisu** oraz żaden jej rekord nie ma zablokowanego odczytu w trakcie transakcji. Dopuszczalne jest natomiast uzyskanie **dzielonego odczytu** tabeli, jeśli jej rekordy mają zablokowaną jedynie możliwość zmiany.

Blokowanie tabel jest dostępne z poziomu języka **FORMULA 4GL**.

3.4 Transakcje i blokowanie rekordów

Dostęp do danych w systemie **MacroBASE** pozwala na pracę w trybie transakcyjnym, który gwarantuje kompletność wykonania złożonych operacji na bazie danych. Transakcje wykonują się w sposób atomowy. Należy je stosować podczas wykonywania operacji, które muszą wykonać się w całości albo nie wykonać wcale. Na przykład przeniesienie rekordu pomiędzy dwoma tabelami powinno się wykonywać w trybie transakcyjnym. Od momentu rozpoczęcia transakcji wszystkie zmiany w bazie danych --- dołączanie, poprawianie i usuwanie rekordów w tabelach --- są rejestrowane. Dostęp do rekordów odczytanych lub zmienionych (dołączonych, poprawionych lub usuniętych) jest automatycznie blokowany dla innych użytkowników. Rekordy, które w trakcie transakcji zostały odczytane mają zablokowaną możliwość zmiany. Rekordy, które zostały dołączone, poprawione lub usunięte mają dostęp zablokowany całkowicie. Maksymalna liczba rekordów, które mogą być naraz zablokowane przez wszystkie aktywne transakcje jest ograniczona przez stałą systemu.

Transakcja może się zakończyć:

- zatwierdzeniem --- wszystkie zmiany w bazie danych są zatwierdzane, wszystkie nałożone blokady zdejmowane,
- anulowaniem --- wszystkie zmiany w bazie danych są wycofywane, wszystkie nałożone blokady zdejmowane,
- awarią systemu --- po ponownym uruchomieniu programu wszystkie zmiany w bazie danych są wycofywane.

Transakcja jest zatwierdzana w przypadku:

- zatwierdzenia programowego,
- normalnego zakończenia pracy programu.

Transakcja jest anulowana w przypadku:

- anulowania programowego,
- fizycznego błędu dostępu do danych,
- logicznego błędu dostępu do danych (zapis spoza dziedziny, podwojenie klucza w unikalnym indeksie),
- błędu składniowego **FORMULI 4GL**,
- przerwania programu,
- rezygnacji z oczekiwania na odblokowanie rekordu lub tabeli,
- przekroczenia limitu czasowego oczekiwania na odblokowanie rekordu lub tabeli.

Mechanizm transakcji jest dostępny z poziomu języka **FORMULA 4GL.**

3.5 Konflikty i ich rozstrzyganie

Blokowanie tabel na trzech poziomach wyłączności oraz automatyczne blokowanie rekordów w trakcie pracy transakcyjnej niesie znaczne niebezpieczeństwo konfliktów. System zarządzania bazą danych **MacroBASE** ma wbudowane mechanizmy rozstrzygania konfliktów i unikania zakleszczeń.

W trakcie standardowego trybu pracy, tj. poza działaniem transakcji, odwołanie do zablokowanego zasobu (rekordu czy tabeli) powoduje pojawienie się na ekranie okienka z informacją o blokadzie. Jednocześnie próby dostępu do zasobu są cyklicznie ponawiane. W przypadku zwolnienia blokady okienko informacyjne jest zdejmowane i program kontynuuje działanie. Operator może czekać na zwolnienie zasobu lub wcisnąć klawisz rezygnacji. Informacja o ewentualnej rezygnacji jest przekazywana do programu.

Odwołanie do zablokowanego zasobu w trakcie trwania transakcji ma analogiczne konsekwencje, przy czym:

- czas oczekiwania na zwolnienie blokady jest ograniczony, a jego upływ jest przedstawiany na ekranie operatora,
- po rezygnacji z oczekiwania po upływie przewidzianego czasu lub na podstawie decyzji operatora transakcja jest automatycznie anulowana, a program kontynuuje działanie.
- przekroczenie maksymalnej liczby zablokowanych rekordów jest traktowane tak samo jak próba dostępu do zablokowanego rekordu.

4. Język FORMULA 4GL

4.1 Opis ogólny

Język **FORMULA 4GL** jest językiem programowania wysokiego poziomu. Oprócz typowych konstrukcji jak instrukcje strukturalne, funkcje czy zmienne, zawiera szereg elementów programowania obiektowego. Instrukcje języka są interpretowane, można je utworzyć i wykonać także w trakcie działania programu. Program wykonawczy umożliwia, a **FORMULA 4GL** zezwala na rekurencyjne wywołania procedur. Do języka dołączona jest bogata biblioteka gotowych procedur i operatorów, w tym biblioteka klasy tabel bazy danych zawierająca metody dostępu do danych.

4.2 Typy wyrażeń

Wyrażeniami **FORMULI 4GL** mogą być stałe, zmienne lub wywołania funkcji oraz ich dozwolone kombinacje z użyciem operatorów i instrukcji strukturalnych. Każde wyrażenie języka zwraca wynik określonego typu. Lista możliwych typów przedstawia się następująco:

- typy proste:
  - **DATE** --- typ określający datę,
  - **FORMULA** --- typ procedurowy
  - **NUMBER** --- typ liczbowy zmiennoprzecinkowy
  - **STRING** --- typ napisowy
  - **TIME** --- typ określający odstęp czasowy
  - **VOID** --- typ pusty.
- typy złożone:
  - klasa tabel bazy danych,
  - tablice o elementach dowolnego typu,
  - klasy utworzone dynamicznie.

4.3 Zmienne

Język **FORMULA 4GL** wyróżnia i obsługuje zmienne:

- lokalne, powoływane przez wystąpienie, których znaczenie trwa tylko do końca instancji interpretatora, w której zostały użyte,
- globalne, powoływane przez wystąpienie, których znaczenie trwa do końca programu bądź jawnego usunięcia,
- dynamiczne, wyposażone w wygodny mechanizm redagowania, powoływane i usuwane przez odpowiednie funkcje języka,
- proste charakterystyczne dla definicji systemu, istniejące przez cały czas

działania programu, takie jak dostępne poprzez akronimy pola tabel i zmiennych,
- obiektowe charakterystyczne dla definicji systemu, takie jak dostępne poprzez akronimy tabele i zmienne,
- obiektowe użytkownika, w tym także tablicowe, powoływane i usuwane przez odpowiednie funkcje języka.

Dla pól typu złączeniowego język **FORMULA 4GL** umożliwia dostęp zarówno bezpośrednio do wartości pola w złączeniu, jak i pośrednio do pól tabeli połączonej. W drugim przypadku przed dostępem do pola w tabeli połączonej odczytywany jest najpierw rekord tej tabeli, na który wskazuje zawartość pola w złączeniu. W przypadku bezpośredniego dostępu do pola tabeli połączonej odczytywanie rekordu nie występuje.

4.4 Procedury

Procedura może być umieszczona w zmiennej typu **FORMULA** lub w zewnętrznym zbiorze tekstowym. Zawiera ona tekst podprogramu w **FORMULI 4GL**. Procedury mogą mieć zmienną liczbę parametrów. Dozwolone jest wywołanie rekurencyjne. Każdy napis, a dokładnie zmienna typu **STRING**, może zostać przekształcony do procedury. Poprawność składniowa jest rozstrzygana w momencie wykonania.

4.5 Tablice i obiekty

Język **FORMULA 4GL** pozwala definiować dwa rodzaje zmiennych typu złożonego:

1. Tablice o dowolnej liczbie elementów. Każdy element może być dowolnego typu prostego lub złożonego.
2. Obiekty zgodne z uprzednio zadeklarowaną klasą. Deklaracja klasy zawiera:
   - listę pól obiektu wraz z nadanymi wartościami początkowymi,
   - listę metod wspólnych dla wszystkich obiektów danej klasy zawierającą wyróżnioną metodę inicjującą obiekt,
   - listę metod wirtualnych, których treść dla każdego obiektu danej klasy może być zmieniana dynamicznie.

Zmienne typu złożonego są powoływane dynamicznie i po wykorzystaniu mogą zostać usunięte.

4.6 Operatory

Dla typów prostych zdefiniowana jest znaczna liczba jedno- i dwuargumentowych operatorów. Ich działanie zależy od liczby i typów argumentów. Można przy ich użyciu wykonywać zarówno typowe obliczenia i operacje na tekstach, jak i dokonywać konwersji danych różnych typów, a także typy te rozpoznawać.

Dla typów złożonych określone są jedynie operatory podstawienia i porównania. Istnieje możliwość definiowania własnych operatorów dla deklarowanej klasy. Operatory te podobnie jak dla typów prostych mogą być przeciążone.

4.7 Funkcje wbudowane

Część funkcji języka **FORMULA 4GL** jest dostępna zawsze, niezależnie od wykonywanego programu użytkowego. Można wśród nich wyróżnić następujące grupy:

- zarządzanie obiektami,
- funkcje matematyczne,
- operacje na dacie i odstępie czasowym,
- wywoływanie zewnętrznych podprogramów,
- zamiana danych do postaci tekstowej,
- definiowanie oraz wykorzystanie menu i dialogu,
- pełnoekranowy edytor,

- obsługa organizacji ekranu i klawiatury,
- zarządzanie i redagowanie zmiennych dynamicznych,
- definiowanie poziomu obsługi błędów,
- korzystanie z usług systemu operacyjnego.

4.8 Funkcje zależne od definicji

Funkcje, których działanie jest zależne od definicji programu użytkowego, dzielą się na dwie kategorie:

1. Ogólne odwołania do definicji:
   - zarządzanie transakcjami,
   - obsługa generatora sprawozdań,
   - konstruowanie menu programu użytkowego,
   - ochrona danych przy pomocy bieżącego klucza protekcji.
2. Biblioteka klasy tabel:
   - wywołanie okienka wertowania,
   - wyszukiwanie i odczytywanie rekordu,
   - dołączanie, poprawianie i usuwanie rekordu,
   - wypełnianie, wyświetlanie i redagowanie bufora,
   - zapamiętanie i odtwarzanie kontekstu,
   - obsługa zbiorów dla tabel maskowanych,
   - ustalanie poziomu wyłączności poprzez blokowanie,
   - ustalanie i konfigurowanie bieżących okienek,
   - wybór klucza uporządkowania i określenie dziedziny,
   - sortowanie i kasowanie,
   - import i eksport danych w formacie tekstowym.

5. Język wydruków REPORT

5.1 Opis ogólny

Język **REPORT** służy do sporządzania sprawozdań z bazy danych. W jego konstrukcjach wykorzystuje się procedury **FORMULI 4GL** oraz definicję bazy danych utworzoną programem **MacroGEN**. Za sporządzanie raportów jest odpowiedzialny wbudowany w program wykonawczy generator sprawozdań. Tworzone wydruki mają trzy możliwe miejsca przeznaczenia:

- drukarka,
- ekran,
- kopia.

Wybór miejsc przeznaczenia jest zaznaczony we wzorcu wydruku. Jeśli wzorzec na to pozwala, operator może dokonać zmiany. Nie ma ograniczeń w ich doborze --- w szczególności można podać wszystkie trzy lub nie podać żadnego.

5.2 Translator MacroREPORT

Wersja źródłowa wzorca wydruku zawarta w zbiorze tekstowym jest przetwarzana przez translator **MacroREPORT** do kodu pośredniego wykonywanego przez generator sprawozdań. Kompilacja pozwala wyeliminować błędy składniowe i znacząco przyspiesza wykonanie.

5.3 Obiekty języka REPORT

Obiekty te dzielą się na teksty cytowane w wynikowym sprawozdaniu dosłownie oraz konstrukcje specyficzne języka takie jak:

- słowa kluczowe,
- pozycja i format wyprowadzenia danej,
- wyrażenia **FORMULI 4GL**

5.4 Opis strony

Mechanizm opisu strony pozwala:

- definiować dynamicznie marginesy oraz paginę górną i dolną,
- dowolnie formatować i pozycjonować

dane w wierszu,
- swobodnie wybierać bieżący wiersz na stronie,
- żądać umieszczenia wybranego fragmentu w całości na stronie.

5.5 Instrukcje strukturalne

Język sprawozdań **REPORT** pozwala nie tylko przygotowywać dowolne formularze wydruków, ale również dzięki dużej mocy opisowej nadaje się do przetwarzania bazy danych. Decyduje o tym grupa instrukcji strukturalnych:

- warunek i iteracja sterowane wyrażeniami **FORMULI 4GL**,
- iteracja po rekordach tabeli — najsilniejsza konstrukcja języka, dodatkowo wyposażona w atrybuty pozwalające ustalić dynamicznie:
  - kolejność przetwarzania,
  - jego dziedzinę i zakres,
  - konstrukcję inicjującą przetwarzanie,
  - wzorzec dla rekordu,
  - grupę rekordów wskazaną przez operatora,
  - warunki zakończenia przetwarzania.
- podsumowania z możliwością zagnieżdżania wykonywane automatycznie dla instrukcji iteracyjnych,
- definiowanie i przywoływanie makrodefinicji,
- możliwość komentowania.

5.6 Sterowanie drukarką

Język **REPORT** pozwala we wzorcu wydruku dynamicznie dobierać czcionkę i rozmiar odstępu pomiędzy wierszami oraz sterować końcem wiersza i strony.

5.7 Podgląd ekranowy

Jeśli wzorzec wydruku tego nie zabrania, przed zatwierdzeniem i wysłaniem wydruku na drukarkę można obejrzeć go na ekranie.

5.8 Kopia sprawozdania

Kopia sprawozdania ma na celu utworzenie migawkowego obrazu bazy danych do późniejszego wykorzystania. Może być utworzona niezależnie od innych miejsc przeznaczenia wydruku. W zakresie zawartych w niej informacji kopia może być wykorzystana jako źródło tworzenia kolejnych sprawozdań.

5.9 Konfigurowanie parametrów wydruku

Operator, jeśli wzorzec wydruku na to zezwala, ma przed rozpoczęciem działania generatora sprawozdań dostęp do okienka dialogowego, które umożliwia konfigurowanie parametrów wydruku. Może wybrać:

- drukarkę i jej drajwer,
- standardową czcionkę i wysuw,
- źródło pochodzenia wydruku (baza danych, kopia),
- miejsca przeznaczenia wydruku (ekran, kopia, drukarka).

6. Obsługa urządzeń zewnętrznych

System zarządzania bazą danych **MacroBASE** posiada własną obsługę terminali i drukarek. Drajwery urządzeń są przygotowywane w postaci zbiorów tekstowych, a następnie przetwarzane do postaci binarnej przy pomocy translatorów **MacroTERMINAL** i **MacroPRINTER**. Rozwiązanie to przynosi istotne korzyści:

- uniezależnia od opisów zawartych w bazach **termcap** i **terminfo** czy skryptach pośredniczących drukarek,
- uniezależnia od wersji narodowej systemu operacyjnego,
- pozwala na łatwiejsze przenoszenie

oprogramowania pomiędzy różnymi platformami,
- dzięki wstępnemu kompilowaniu drajwerów eliminuje się błędy możliwe w tekstowych zbiorach definicyjnych, a dostęp do urządzeń staje się szybszy,
- pozwala rozszerzyć opis urządzenia o dodatkowe elementy przydatne przy prezentacji bazy danych.

6.1 Terminal

Zawartość drajwera

Drajwer terminala określa:

- inicjowanie dostępu do ekranu i klawiatury,
- rozmiary ekranu i kursora, przemieszczanie kursora, sygnał dźwiękowy itp,
- tablicę translacji wyświetlanych znaków,
- dostępne atrybuty znakowe,
- przypisanie atrybutów i ramek do menu i okienek,
- przypisanie atrybutów i znaków specjalnych do obiektów okienek dialogowych,
- definicję klawiatury,
- przypisanie znaczenia wybranych klawiszy.

Wybór drajwera terminala dokonywany jest na początku pracy systemu użytkowego na podstawie wartości zmiennej systemowej **MACROTERM**. Właściwy drajwer jest niezbędny do uruchomienia programu użytkowego.

6.2 Drukarka

Program wykonawczy **MacroCLIENT** współpracuje zarówno z drukarkami globalnymi podłączonymi do jednostki centralnej, jak i z drukarką lokalną podłączoną do terminala.

Drukarka globalna

Do obsługi drukarki globalnej używany jest podsystem drukujący **lp** (*line printer*). Program ten jest wykorzystywany jedynie w zakresie przechowywania i kolejkowania zleceń. Istnieje możliwość wysłania wydruku na konkretną drukarkę bądź aktualnie najmniej obciążoną. Skrypt pośredniczący w przesyłaniu danych powinien umożliwić wysłanie zbioru wydruku bez żadnych zmian.

Drukarka lokalna

Dostęp do drukarki lokalnej podłączonej do terminala nie wymaga arbitrażu ze strony systemu operacyjnego i polega na przełączeniu standardowego wyjścia na czas transmisji odpowiedniego zbioru wydruku.

Zawartość drajwera

Drajwer drukarki określa:

- inicjowanie dostępu,
- nazwę drukarki lub klasy drukarek w systemie operacyjnym,
- sekwencje sterujące nowym wierszem i nową stroną,
- szerokość i wysokość strony,
- nazwę do wyboru z menu,
- dostęp operatora do panela drukarki w programie wykonawczym,
- wyświetlanie wydruku na ekranie,
- wykonywanie kopii wydruku,
- fizyczne wykonanie wydruku i liczbę jego egzemplarzy,
- drukowanie nagłówka systemowego,
- tablicę translacji drukowanych znaków,
- dostępne oraz standardowe czcionki i odstępy pomiędzy wierszami,
- wyświetlanie echa opracowywanych wierszy wydruku.

Wybór drajwera drukarki dokonywany jest przy uruchomieniu wbudowanego w program

wykonawczy generatora sprawozdań na podstawie informacji zawartych we wzorcu wydruku. Prawidłowy wybór drajwera jest konieczny do wykonania sprawozdania.

7. Dostępne platformy i kierunki rozwoju

7.1 Wielodostęp

W wersji podstawowej **MacroBASE** dla systemu UNIX przeznaczony jest do pracy wielodostępnej na terminalach podłączonych do jednostki centralnej. Dostępne platformy systemowo-sprzętowe to:

- mikrokomputery oparte o procesor Intel z systemem operacyjnym:
  - SCO UNIX w wersji 3.2.2 bądź wyższej,
  - UNIX Systemu V w wersji 4.0 bądź wyższej,
- komputery wieloprocesorowe Hewlett Packard PA-RISC serii 9000 z systemem operacyjnym HP-UX wersja 9.0[1]

7.2 Architektura *client-server*

System zarządzania bazą danych **MacroBASE** przewiduje wykorzystanie sieci lokalnej. Obecnie prowadzone są prace nad synchronizacją działania maszyny bazy danych i programu wykonawczego poprzez protokół TCP/IP. Program użytkowy oparty o nową generację systemu **MacroBASE** będzie skalowany w środowisku sieciowym.
W przypadku nadmiernego obciążenia możliwe będzie dołączenie dodatkowej jednostki centralnej, która przejmie wykonywanie części zadań.[2]

7.3 Graficzny interfejs użytkownika

Nowa generacja systemu **MacroBASE** będzie wyposażona w graficzny interfejs użytkownika zgodny z wymaganiami stawianymi przez standard CUA (*Common User Access*). Przewidziane są wersje[3]:

- jednodostępna dla pojedynczej stacji roboczej *MS Windows* z możliwością uruchomienia kilku programów użytkowych jednocześnie,
- wielodostępna dla stacji roboczej *MS Windows* połączonej siecią lokalną z maszyną bazy danych pracującą pod nadzorem systemu UNIX,
- wielodostępna dla stacji roboczej *X Window* komunikującej się z maszyną bazy danych uruchomioną na tej samej stacji roboczej lub innym systemie UNIX połączonym siecią lokalną.

7.4 Zgodność definicji bazy danych

Definicja programu użytkowego może być bez żadnej zmiany przeniesiona w środowisko nowej generacji oprogramowania **MacroBASE**. Dotyczy to zarówno wersji rozproszonej, jak i graficznego interfejsu użytkownika.

8. MacroBASE dla systemu UNIX --- oferowane produkty

System zarządzania bazą danych **MacroBASE** jest oferowany w postaci dwóch rodzin pakietów:

1. **MacroBASE Development System** --- wersja uruchomieniowa dla programisty:
   - **MacroGEN** --- generator definicji programów użytkowych,
   - **MacroDEVELOP** --- program

[1] na początku III kwartału br.

[2] w IV kwartale br.

[3] na koniec br.

wykonawczy w wersji jednostanowiskowej do wytworzenia, uruchomienia i testowania programu użytkowego,
- **MacroREPORT** --- translator języka wydruków,
- **MacroTRANSFER** --- program do restrukturyzacji bazy danych,
- **MacroPASS** --- program do zmiany haseł,
- **MacroTERMINAL** --- translator drajwerów terminala,
- **MacroPRINTER** --- translator drajwerów drukarek,
- gotowe drajwery wybranych terminali i drukarek,
- podręczniki:
  - „Generator zastosowań MacroGEN",
  - „Język FORMULA 4GL",
  - „Język wydruków REPORT",
  - „Poradnik programisty",
  - „Podręcznik administratora",
  - „Obsługa programu użytkowego".

2. **MacroBASE Runtime System** --- wersja dla końcowego użytkownika:
   - **MacroSERVER** --- maszyna bazy danych dla odpowiedniej liczby użytkowników,
   - **MacroCLIENT** --- program wykonawczy w wersji wielodostępnej,
   - **MacroPASS** --- program do zmiany haseł,
   - gotowe drajwery wybranych terminali i drukarek,
   - podręczniki:
     - „Podręcznik administratora",
     - „Obsługa programu użytkowego".

Generator **MacroGEN** jest dostępny na platformie DOS. Pozostałe programy na platformie UNIX.

Agnieszka Gąsiorowska, Aleksander Laurentowski
AGH Kraków

Ramy i ich zastosowanie do enkapsulacji interakcji między procesami

1 Wstęp

Przełom lat 80-tych i 90-tych w informatyce cechował się dużym wzrostem wymagań wobec oprogramowania użytkowego. Wymagania te obejmowały zarówno stopień funkcjonalności konstruowanych programów, tj. ilość dostarczanych przez nie opcji i funkcji, jak również ich poprawność, niezawodność, wysoką jakość i wygodę kontaktu z użytkownikiem. Pakiety programowe mają dziś rozmiary niewyobrażalne jeszcze kilka lat temu - dziesiątki i setki megabajtów kodu. Tymczasem wydajność pracy programistów przy tradycyjnych metodach programowania jest bardzo ograniczona. Według Jonesa ilość linii kodu źródłowego, którą jest w stanie napisać, przetestować i udokumentować w ciągu jednego dnia programista posługujący się tradycyjnymi metodami programowania jest zaledwie rzędu kilku (w przypadku projektów przekraczających 16 tys. linii, a obecnie większość z nich liczy się w setkach tysięcy). Wynika to z ograniczonych możliwości ludzkiego umysłu - pojedynczy człowiek nie jest w stanie objąć całkowitą kontrolą systemu tej wielkości.

Rozwiązanie zaistniałej sytuacji mogą przynieść nowe metody inżynierii oprogramowania. Jedną z najważniejszych wśród nich jest idea wielokrotnego użycia raz opracowanych składowych programu (ang. *software reuse*).

Celem niniejszego artykułu jest prezentacja jednej z realizacji tej idei: pojęcia ramy (ang. *framework*) jako uniwersalnego komponentu programowego wielokrotnego wykorzystania, cechującego się znaczną elastycznością oraz zdolnością dostosowania do różnych warunków i wymagań. Rama, w zależności od rodzaju, stanowi szkielet bądź część systemu końcowego tworzonego z jej pomocą. Zawsze jednak ramy zawierają i ukrywają (inaczej mówiąc: enkapsulują) pewien schemat funkcjonalny lub strukturalny istniejący w tych systemach. Ich zastosowanie może znacznie uprościć i przyspieszyć konstrukcję oprogramowania, obniżając jednocześnie jej koszt.

Przedstawimy również metodę kostrukcji ram dla schematów interakcji w programach rozproszonych za pomocą mechanizmów programowania obiektowego.

Czujemy się w obowiązku zaznaczyć, że ramy (ang. *frameworks*), o których piszemy, nie mają nic wspólnego z ramami (ang. *frames*), służącymi do reprezentacji wiedzy w systemach ekspertowych [6].

2 Propozycja systematyzacji modeli ram

Istnieje szereg podejść do pojęcia ramy jako jednostki wspomagającej wielokrotne wykorzystanie oprogramowania (*software reuse*). Jednym z przykładów jest *Choices*, rama-wzorzec do tworzenia systemów operacyjnych na różne architektury sprzętowe. Składa się ona w zasadniczej części z ok. 300 klas abstrakcyjnych w C++, ukonkretnianych w zależności od wymagań sprzętowych i koncepcyjnych. Tak rozumiana rama jest modelem obrazującym architekturę systemu obiektowo zorientowanego. Z kolei deklaratywny język programowania równoległego PCN umożliwia tworzenie innego typu ram: tzw. komórek (ang. *cells*) i

szablonów (ang. templates). Komponenty te odwzorowują zadany algorytmem schemat interakcji między procesami na odpowiednią topologię wirtualną procesorów. Jeszcze innym typem ramy jest medium komunikacyjne, zapewniające współdziałanie niezależnych aplikacji. Przykładem jest tu ToolTalk firmy SunSoft, moduł zapewniający aplikacjom automatyczny dostęp do wszelkiego typu danych numerycznych, tekstowych, graficznych, akustycznych, itp. bez konieczności posiadania wiedzy o miejscu i postaci przechowywania tej informacji w systemie, lub bardziej znany mechanizm OLE z Microsoft Windows.

Podejścia te różnią się w wielu aspektach, posiadają jednak naszym zdaniem pewną dozę cech wspólnych. Nie zaproponowano do tej pory uniwersalnej definicji ramy, choć pojęcie to jest często używane przy opisie poszczególnych realizacji projektów z zakresu inżynierii oprogramowania. Nie podano również kryteriów systematyzacji już istniejących implementacji. Wynika to zapewne z faktu, iż problematyka ta jest rozwijana od niedawna i nie osiągnęła jeszcze wystarczającej dojrzałości. Jesteśmy przekonani, że próba ogólnego opisu i klasyfikacji ram może nie tylko pomóc w uchwyceniu samej istoty zagadnienia, ale także mieć pozytywny wpływ na rozwój tej dyscypliny.

W punkcie tym przedstawimy zatem propozycję definicji uogólnienia ram oraz kryteria, jakie można stosować do ich klasyfikacji. Pokażemy przy tym, jak za pomocą tych kryteriów można uszeregować wymienione podejścia.

2.1 Ogólna charakterystyka pojęcia rama

Rama jest komponentem programowym, posiadającym następujące podstawowe cechy:

1. Może być powtórnie użyty w innym programie.
2. Reprezentuje pewien schemat funkcjonalny lub strukturalny (a czasem oba z nich).
3. Jest uniwersalny i elastyczny, umożliwiając budowę programów różnorodnych, ale zachowujących ten sam schemat funkcjonalności lub struktury (patrz punkt poprzedni).
4. Jest jednostką samodzielną pod względem funkcjonalnym, tzn. posiada ściśle określony zestaw funkcji, który samodzielnie realizuje.
5. Jest mechanizmem enkapsulacji, ponieważ sposób realizacji funkcji dostarczanych przez ramę jest z reguły niewidoczny dla ich użytkownika.

Rama powinna być maksymalnie uniwersalna i elastyczna. Elastyczność ramy oznacza, że możliwe jest dostosowywanie konstruowanych z jej pomocą systemów do wymagań środowiska, ukonkretnianie zawartych w niej założeń abstrakcyjnych, redefiniowanie składników konkretnych, a czasem nawet ingerencja w jej wewnętrzną strukturę. Odróżnia to ramę od procedury bibliotecznej, która jest dobrze znanym i powszechnie stosowanym narzędziem wielokrotnego wykorzystywania oprogramowania. Procedura posiada ściśle określony zestaw parametrów wejściowych i wyjściowych, a implementacji realizowanego przez nią algorytmu nie sposób zmienić.

Uniwersalność rozumiana jako wielofunkcyjność jest trudnym do zapewnienia kryterium i skłaniamy się do poglądu, że rama nie musi być w ten sposób uniwersalna. Stworzenie "Uniwersalnej Ramy Wszystkiego" jest z całą pewnością informatyczną (i filozoficzną) mrzonką. Jeżeli jednak uniwersalność rozumiemy jako możliwość zastosowania dla rodziny pokrewnych potrzeb, to rama powinna z definicji cechować się tą właściwością.

Rama kryjąca projekt (szkielet) całego programu, jak np. *Choices*, różni się od

tradycyjnego (czasem istniejącego jedynie w postaci półformalnego zapisu na papierze) projektu tym, że jest **instancją** ogólnego wzoru, a nie zestawem wytycznych oraz tym, że zawsze jest zapisana w konkretnej notacji. Może być to dowolna notacja formalna, najczęściej jednak jest to odpowiedni do zastosowań język programowania, co dodatkowo stwarza możliwość częściowego sprawdzenia poprawności ramy za pomocą procesu kompilacji.

Rama może istnieć zatem w wielu postaciach (reprezentacjach), począwszy od formalnego zapisu w notacji wysokiego poziomu, poprzez zapis w języku programowania, aż po przekształconą w procesie kompilacji postać ładowalną, gotową do dynamicznej konsolidacji.

2.2 Kryteria systematyzacji ram

Bazując na przedstawionych przykładach ram, proponujemy następujące kryteria ich systematyzacji:

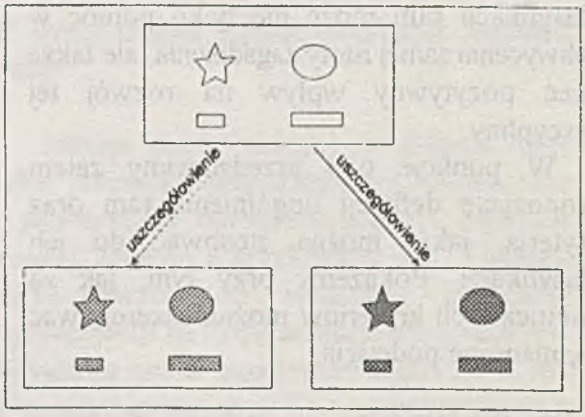
Stopień abstrakcji i komplementarne względem niego pojęcie **stopnia konkretności**. Ramy mogą zawierać w sobie mniej lub więcej elementów abstrakcyjnych, umożliwiających modyfikację i redefinicję ich cech funkcjonalnych. Elementy te określają uniwersalność ramy, jej zdolność dostosowania do zmieniających się zastosowań. I tak np. komórki w PCN są bardziej konkretne niż szablony, ponieważ specyfikują ściśle kod do wykonania w że prezentowanym schemacie interakcji. Właściwość sprawiająca, że do szablonów kod ten można podać jako parametr, czyni je bardziej abstrakcyjnymi i uniwersalnymi.

Zakres funkcjonalny ram. Ramy o dużym zakresie funkcjonalnym, jak np. *ToolTalk* realizują szeroki zestaw funkcji i usług (w tym przypadku protokołów komunikacji). Ramy mniejsze, np. szablony w PCN, realizują węższy ich zestaw, choć jednocześnie mogą cechować się wysokim stopniem abstrakcji (jak jest w tym wypadku).

Sposób reprezentacji ramy określa stopień jej przekształcenia, które może przebiegać od zapisów w formalizmach wysokiego poziomu, poprzez teksty w konkretnym języku programowania aż po postać skompilowaną, dostępną bezpośrednio do dynamicznego łączenia (konsolidacji).

Typ przekształcenia. Zarysowują się dwa zasadnicze sposoby przekształcenia ramy w system użytkowy (czyli jej wykorzystania):

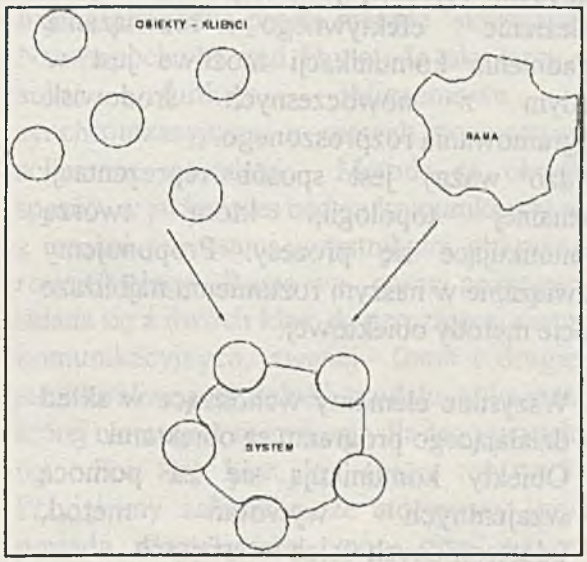
1. **Uszczegółowienie** (ang. *refinement*), polegające na ukonkretnieniu zawartych w ramie abstrakcji, np. kodu, który ma być wykonany w węzłach szablonu w PCN albo metod abstrakcyjnych w Choices. Stosując uszczegółowienie na wiele sposobów można utworzyć klasę podobnych systemów konkretnych (patrz rys. 1).



Rysunek 1. Uszczegółowienie.

2. **Uzupełnienie**, polegające na dołączeniu do ramy obiektów zewnętrznych, korzystających z jej usług i funkcji. Wyobraźmy sobie pakiet oprogramowania o nazwie *DesktopPublishingSystem*,

zawierający m.in. edytory tekstu i grafiki, konwertery formatu dokumentów, przeglądarki, moduły drukarki i naświetlarki, itp.). Programy te, będące odrębnymi aplikacjami, komunikują się ze sobą za pomocą zestawu protokołów, dostarczanych i realizowanych przez *ToolTalk*. Są one w zasadzie samodzielnymi klientami usług ramy, pełniącej w tym wypadku rolę koordynatora. Rama i klienci są samodzielnymi jednostkami, które nawzajem się uzupełniają .



Rysunek 2. Uzupełnienie.

Typ relacji między ramą a systemem konkretnym, utworzonym z jej pomocą. W zależności od typu przekształcenia może to być relacja jednego z dwóch rodzajów:

1. **jest-typu**, oznaczana $\overset{\text{is-a}}{\sim}$ dla systemów utworzonych w procesie uszczegóławiania, np.

$$SunChoices \overset{\text{is-a}}{\sim} Choices,$$
$$EncoreChoices \overset{\text{is-a}}{\sim} Choices$$

2. **jest-częścią**, oznaczana $\overset{\text{is-part-of}}{\rightarrow}$ oraz odwrotna do niej relacja posiada, oznaczana $\overset{\text{has}}{\leftarrow}$, np.

$$ToolTalk \overset{\text{is-part-of}}{\rightarrow} DesktopPublishing\ System \equiv DesktopPublishing\ System \overset{\text{has}}{\leftarrow} ToolTalk$$

3 Metoda konstrukcji ram w językach obiektowych

Badaliśmy możliwości stworzenia i zastosowania ram enkapsulujących schematy interakcji między procesami w systemach rozproszonych. Stosując terminologię poprzedniego paragrafu można powiedzieć, że ramy takie charakteryzują się niewielkim zakresem funkcjonalnym. Powinny one jednak cechować się wystarczająco wysokim stopniem abstrakcji i elastyczności, aby mogły być efektywnie i wielokrotnie używane w różnych zastosowaniach. Podjęliśmy próbę opracowania metody konstrukcji takich ram przy użyciu mechanizmów dostępnych w językach obiektowo zorientowanych. Stosując różne jej warianty, stworzyliśmy ramy dla szeregu schematów interakcji, takich jak strumień, pierścień czy gwiazda (zwana także farmą procesorów). Każda z nich została zastosowana w kilku niedużych programach rozproszonych (zaimplementowano m.in. różne rodzaje sortowania, generacji liczb pierwszych, obliczanie fraktala Mandelbrota, itp.). Zarówno ramy, jak i wspomniane programy użytkowe zostały napisane w obiektowo zorientowanym języku programowania równoległego i rozproszonego SR (*Synchronizing Resources*) oraz uruchomione i przetestowane na sieci stacji roboczych SUN.

Opracowaliśmy metodę konstrukcji aplikacji rozproszonych z użyciem ram w klasie języków imperatywnych stosunkowo niewielkim kosztem implementacyjnym, tj. bez tworzenia notacji wysokiego poziomu i jej kompilatora. Zdecydowaliśmy pozostać na średnim poziomie abstrakcji - poziomie

języków programowania ogólnego użytku. Ramy tworzone w takich językach są bardziej elastyczne i łatwiejsze do konfiguracji lub zmiany. Wybraliśmy języki obiektowo zorientowane, aby móc wykorzystać mechanizmy, których dostarczają ich kompilatory. Mechanizmy te to wielokrotne dziedziczenie lub import (dla języków nie posiadających własności dziedziczenia kodu). Ten drugi mechanizm jest dostępny między innymi w takich językach jak SR czy DisCo . Polega on na możliwości zadeklarowania klasy importującej inną klasę, przy czym klasa importowana nie wchodzi w ten sposób w relację dziedziczności z importującą. Obiekt utworzony z klasy importującej uzyskuje w ten sposób dostęp do interfejsu klasy importowanej i może dzięki temu utworzyć obiekty tej klasy by potem korzystać z ich usług (tj. wywoływać ich metody).

W poniższym paragrafie opiszemy jak konstruować i korzystać z obiektowo zorientowanych ram enkaspulujących schematy interakcji między procesami w programach współbieżnych i rozproszonych. Przedstawimy koncepcję obiektów sklejających, które w proponowanym schemacie konstrukcyjnym pełnią rolę elementów pośrednich (spoiwa) między wykorzystywaną przez programistę ramą a specyficznym dla tworzonej aplikacji algorytmem obliczenia.

3.1 Obiekty sklejające

Konstruktor ram enkaspulujących schematy interakcji między procesami ma do rozwiązania dwa zasadnicze problemy: budowę struktury komunikacyjnej odpowiedniej dla implementowanego schematu interakcji oraz opracowanie sposobu łączenia dwóch oddzielnych i z zasady niezależnych elementów, z których programista składa aplikację: ramy i kodu specyficznego dla obliczenia.

Rozwiązanie problemu pierwszego jest w wysokim stopniu zależne od właściwości języka, w którym konstruujemy ramę. SR, który wykorzystywaliśmy, oferuje bogaty wybór mechanizmów komunikacji i synchronizacji między procesami. Daje to konstruktorowi ramy swobodę przy wyborze najbardziej odpowiednich z nich. My skorzystaliśmy z dwóch: zdalnego wywołania procedury (RPC) i przekazywania komunikatów. Należy jednak zdawać sobie sprawę, że nie wszystkie języki programowania współbieżnego i rozproszonego są tak wygodne pod tym względem. Jesteśmy jednak przekonani, że znalezienie efektywnego rozwiązania zagadnienia komunikacji możliwe jest w każdym z nowoczesnych środowisk programowania rozproszonego.

Bardzo ważny jest sposób reprezentacji wirtualnej topologii, którą tworzą komunikujące się procesy. Proponujemy rozwiązanie w naszym rozumieniu najbliższe istocie metody obiektowej:

- Wszystkie elementy wchodzące w skład działającego programu są obiektami.
- Obiekty komunikują się za pomocą wzajemnych wywołań metod, udostępnionych w ich interfejsach.
- Proces obliczeniowy, wykonywany współbieżnie, jest reprezentowany przez wątek (lekki proces), powstający na skutek wywołania metody obiektu i wykonujący się w jego przestrzeni adresowej.
- Protokół komunikacyjny jest zdefiniowany przez określony zestaw metod zawartych w interfejsie obiektów i odpowiada implementowanemu schematowi interakcji.

Rozwiązanie problemu drugiego powinno być naszym zdaniem niezależne od wybranego języka, pod warunkiem, że jest on obiektowo zorientowany.

Proponujemy zatem wykorzystanie

dziedziczenia lub alternatywnego względem niego importu jako mechanizmu konstrukcji elementów pośrednich między ramą a kodem źródłowym właściwym dla implementowanego przez użytkownika-programistę algorytmu. Elementy takie nazwaliśmy obiektami sklejającymi (ang. *glue objects*). Zawierają one zarówno możliwy do równoległego wykonania kod obliczenia, specyfikowany przez użytkownika, jak i interfejs komunikacyjny. Obiekty takie są używane do konstrukcji styku algorytmu obliczeniowego z ramą. Użytkownik definiuje w nich najczęściej jedną prostą metodę "sklejającą". Nazwa pochodzi stąd, iż metoda taka łączy w sobie funkcje obliczeniowe z synchronizacyjnymi w ramach stosowanego schematu interakcji. Metoda ta określa sposób, w jaki proces będzie komunikował się z innymi procesami-uczestnikami obliczenia równoległego. Rama wg. naszej propozycji składa się z dwóch klas: dostarczającej metod komunikacyjnych, zwanej Comm i drugiej, strukturalizującej całość modelu obliczenia, której nazwa jest specyficzna dla tego modelu, np. Pipe, Ring, Mesh, itp. (patrz też rys.3). Przyjęliśmy założenie, że stosowany język posiada silny system typów oraz nie ma własności metaprogramowania, tzn. typy wszystkich wyrażeń muszą być określone na etapie kompilacji, a kod nie może być parametrem procedury. Cechy takie posiada większość imperatywnych języków programowania rozproszonego, m.in. SR. Nakłada to dodatkowe ograniczenia na proponowaną metodę. Wspomnimy o nich poniżej. Istota proponowanej metody pozostaje jednak bez zmian. Sposób jej wykorzystania obejmuje następujące etapy:

1. **Dekompozycja algorytmu przez programistę**, polegająca na wydzieleniu z algorytmu równoległego części, mogącej wykonywać się jako równoległy proces. Otrzymany w ten sposób fragment kodu źródłowego można zapisać jako metodę compute() klasy UserCode. Oba wymienione identyfikatory nie są obligatoryjne, raz wybrane muszą jednak być konsekwentnie stosowane w następnych etapach.
2. **Zdefiniowanie klasy** Glue, **będącej wzorcem dla obiektów sklejających**. Ta nazwa jest obowiązkowa, stanowi bowiem dla kompilatora informację identyfikującą jednoznacznie klasę sklejającą. Definicja klasy Glue musi zawierać specyfikację metody sklejającej, właściwej dla realizowanego modelu obliczenia. Metoda ta powinna odwoływać się do wyżej wspomnianej funkcji compute() oraz metod komunikacyjnych, dostarczanych przez ramę w specjalnej klasie o ustalonym identyfikatorze Comm. Oto przykład takiej metody dla obliczenia w pierścieniu (zapis w abstrakcyjnym języku obiektowo zorientowanym):

```
method compute_in_ring(data)
{
send_right(compute(data) + receive_left());
}
```

gdzie send_right() i receive_left() są metodami komunikacyjnymi jednokierunkowego pierścienia, dostarczanymi przez klasę Comm. Dlatego Glue musi dziedziczyć lub importować Comm. Ramę reprezentuje klasa dziedziczącą lub importującą Glue, o nazwie równoznacznej z implementowaną topologią (np. Pipe, Ring, Mesh). Utworzenie jej wystąpienia (instancji) spowoduje powstanie wirtualnej topologii komunikacyjnej, składającej się z obiektów typu Comm i Glue. Hierarchia klas obiektów tworzących aplikację wg. opisywanej metody wraz z typem ich powiązań jest zilustrowana na rys. 3.
3. **Zapis programu głównego (aplikacji)**, z utworzeniem ramy i zbudowaniem modelu obliczenia za pomocą odpowiedniego równoległego wywołania obiektów

sklejających tworzących ramę.

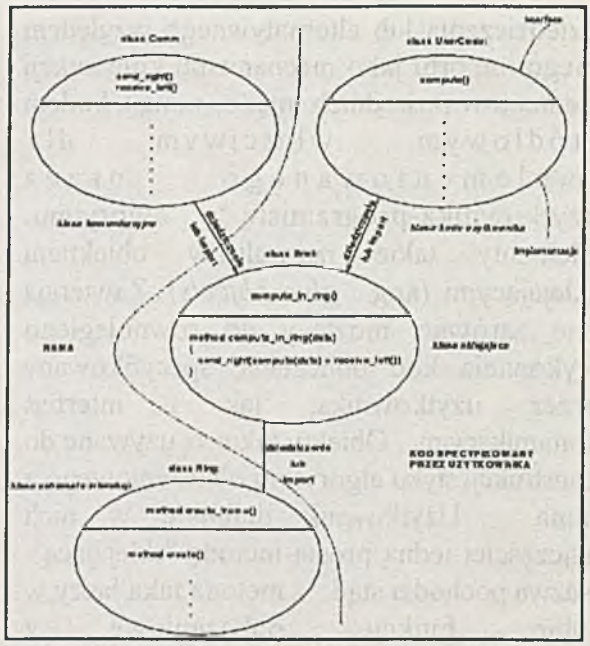
4. **Kompilacji**, która dzięki odpowiednim adnotacjom o wzajemnych relacjach wymienionych klas połączy je mechanizmami dziedziczności i importu, umożliwiając wykonanie programu. Inaczej mówiąc, kod sekwencyjny specyfikowany przez użytkownika ramy uzyska dostęp do funkcji komunikacyjnych, dostarczanych przez ramę i zostanie w momencie jej utworzenia rozproszony w topologii wirtualnej realizowanego przez nią schematu interakcji.

Opiszemy teraz różnice, jakie wynikają z budowy ramy w oparciu o mechanizm dziedziczenia w porównaniu z importem.

3.2 Wykorzystanie mechanizmu dziedziczenia

Rama, reprezentowana przez obiekt, w zależności od dostępnego mechanizmu może dziedziczyć lub importować klasę sklejającą. W przypadku dziedziczenia w tworzonej aplikacji występuje w zasadzie tylko jeden obiekt - rama, raz utworzona, do której metod sklejających, odziedziczonych z klasy sklejającej aplikacja bezpośrednio się odwołuje. Na przykład rama Ring:

```
class Ring: {
  inherit Glue
.
/* szczegoły implementacji sa nieistotne dla uzytkownika ramy
*/
 /* i zalezne od wybranego jezyka*/
.
}
```



Rysunek 3. Hierarchia klas stosowanych do budowy obiektowo zorientowanych aplikacji rozproszonych w schemacie pierścienia z użyciem ramy.

Przykładowa aplikacja ze strukturą pierścieniową korzystająca z takiej ramy ma postać:

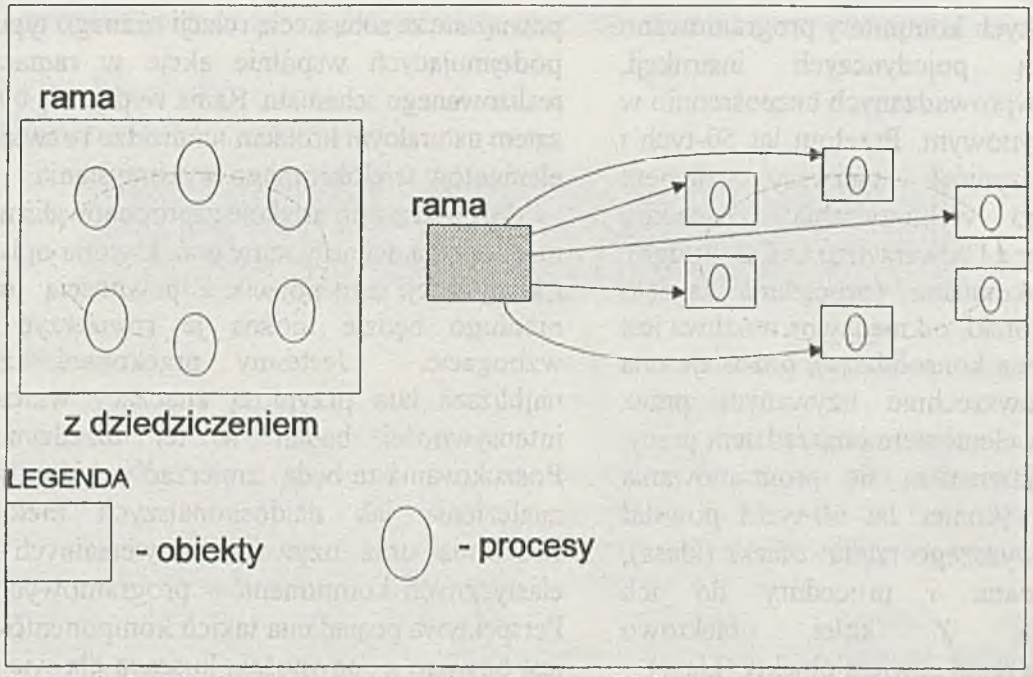
```
main()   {
     table Data[N];

     R := create Ring;            /* utworzenie ramy */
     do in parallel for i := 1 to N  /* zasadnicze obliczenie
                                        rownolegle */
        R.compute_in_ring(Data[i]);   /* metoda sklejajaca,
                  dziedziczona z klasy sklejajacej */
     R.destroy();    /* zniszczenie obiektu-ramy */
   }
```

Rama jest zatem jedynym obiektem, który programista tworzy w aplikacji. Wszelkie inne obiekty tworzone są automatycznie przez ramę. Wywołania zawarte w aplikacji dotyczą wyłącznie metod obiektu-ramy (patrz rys. 4).

3.3 Wykorzystanie mechanizmu importu

W przypadku wykorzystywania mechanizmu importu do udostępniania kodu



Rysunek 4. Różnice w wewnętrznej strukturze obiektu reprezentującego ramę w zależności od stosowanego mechanizmu udostępniania kodu.

klasy sklejającej dla ramy, tworzy ona z tej klasy osobne obiekty, reprezentujące wchodzące w interakcję procesy (patrz rys. 4). Rama może definiować metody dostępu do tych obiektów (a tym samym do ich metod sklejających), jak w przykładzie:

```
class Ring: {
  import Glue

  table_of_objects glues[N];

method create_frame(int k) /* tworzy obiekty sklejajace */
      {
        for i := 1 to N
          glues[i] = create Glue;
      }
      method GlueCode(int i, ptr p) /* udostepnia ich metody
                                       sklejajace */
      {
        glues[i].compute_in_ring(*p);
      }
    }
```

Aplikacja z taką ramą wygląda następująco:

```
main()   {
table Data[N];

R := create Ring(); /* utworzenie obiektu
reprezentujacego rame */
R.create_frame(N);   /* utworzenie N biektow
obiczeniowych */
do in parallel for i := 1 to N /* zasadnicze obliczenie*/
R.GlueCode(i, &Data[i]);   /* z wywolaniem obiektow
sklejajacych */
R.destroy();
}
```

W przypadku korzystania z mechanizmu dziedziczenia rama korzysta jedynie z kodu klasy sklejającej. Używając importu, ramę budujemy z obiektów sklejających w sensie dosłownym, to znaczy obiekt reprezentujący ramę jest odpowiedzialny za ich utworzenie i zainicjowanie w nich obliczeń (wywołanie metod sklejających).

4 Podsumowanie

W historii informatyki można wyróżnić kolejne etapy rozwoju idei wielokrotnego wykorzystania oprogramowania. W latach

40-tych i 50-tych komputery programowano za pomocą pojedynczych instrukcji, początkowo wprowadzanych bezpośrednio w zapisie maszynowym. Przełom lat 50-tych i 60-tych przyniósł pierwszy element wielokrotnego wykorzystania: procedurę (prof. Wheeler z Uniwersytetu w Cambridge). Ciągle udoskonalana (procedury zaczęto łączyć w biblioteki, od niedawna możliwa jest ich dynamiczna konsolidacja), pozostaje ona do dziś powszechnie używanym przez programistów elementem i narzędziem pracy. Wraz z pojawieniem się programowania obiektowego (koniec lat 60-tych) powstał komponent wyższego rzędu: obiekt (klasa), grupujący dane i procedury do ich przetwarzania. Z kolei obiektowo zorientowana rama zawiera obiekty (klasy) powiązane ze sobą siecią relacji różnego typu, podejmujących wspólnie akcje w ramach realizowanego schematu. Rama wydaje się być zatem naturalnym krokiem na drodze rozwoju elementów wielokrotnego wykorzystania.

W powyższym artykule zaproponowaliśmy m.in. ogólną definicję ramy oraz kryteria opisu i klasyfikacji tego pojęcia. Z pewnością już niedługo będzie można je rozszerzyć i wzbogacić. Jesteśmy przekonani, że najbliższe lata przyniosą znaczący wzrost intensywności badań w tej dziedzinie. Poszukiwania te będą zmierzać w kierunku znalezienia jak najdoskonalszych metod tworzenia oraz używania uniwersalnych i elastycznych komponentów programowych. Perspektywa posiadania takich komponentów jest bowiem z pewnością kusząca dla wielu producentów oprogramowania.

Maciej Piasecki
Politechnika Wrocławska

Automatyczne modelowanie semantyki zdań twierdzących języka polskiego.

1. Szkic tematyki

Jednym z najbardziej frapujących zagadnień sztucznej inteligencji a w szczególności jej szczególnego podobszaru: analizy języka naturalnego *(ang. natural language processing)* jest automatyczne tłumaczenie tekstów *(ang. machine translation)* - problem marzenie. Tym ciekawsze, iż przestaje być powoli jedynie ciekawostką badań naukowych a staje się koniecznym do rozwiązania problemem praktycznym. Wystarczy spojrzeć na niezwykle szeroko prowadzone prace badawcze w krajach Wspólnoty Europejskiej. Charakterystyczne jest duże zaiteresowanie tą problematyką państw w których język angielski nie jest językiem urzędowym.[1]

Lata rozwoju systemów automatycznego tłumaczenia tekstów doprowadziły do idei systemów trzeciej generacji - tłumaczących w oparciu o zasadę przenoszenia znaczenia pomiędzy wypowiedziami językowymi: źródłową i generowaną docelową. Wyłania się tu problem: jak formalnie zapisać 'znaczenie' zdań, wypowiedzi językowych.

Już na początku lat pięćdziesiątych, po pierwszych nieudanych próbach stworzenia systemów tłumaczących, zrozumiano kluczowe znaczenie lingwistyki dla dalszych prac. Doprowadziło to zresztą do jej wspaniałego rozwoju, który zmienił jej oblicze jako gałęzi nauki - powstała lingwistyka komputerowa *(ang. computantional linguistics)* ukierunkowana na zastosowania informatyczne.

Dlatego też w każdej pracy z dziedziny przetwarzania języka naturalnego granica pomiędzy 'informatyką' a 'lingwistyką' jest płynna a wykreowany efekt końcowy wymaga spójnego omówienia.

Zbudowanie pełnego systemu tłumaczącego, jest porywającym zadaniem ale znacznie przerastającym ramy pracy magisterskiej dlatego też niezbędne okazało się dokonanie selekcji zagadnień: ograniczyłem się jedynie do zdań twierdzących (z pewnego wycinka języka polskiego) a także, nie realizując całości systemu, jedynie do kluczowego mechanizmu analizy semantycznej tekstu. Celem moim było rozpoznanie 'struktury znaczeniowej' zdania (zamodelowanie jego 'znaczenia') i zapisanie jej w postaci formalnego języka logiki intensjonalnej.

Całość zagadnień związanych z tym lingwistyczno-informatycznym problemem najłatwiej zobrazować rozbijając je na trzy płaszczyzny:

- podstawowa teoria lingwistyczna,
- praktyczne rozwinięcie pod kątem zastosowania informatycznego,
- implementacja teorii w postaci systemu informatycznego (wraz z całym niezbędnym otoczeniem).

2. Podstawowa teoria lingwistyczna.

Teoria lingwistyczna użyta jako podstawa do systemu analizy semantycznej zdań języka naturalnego musi spełniać dwa podstawowe

[1] Po raz pierwszy może informatyka znalazła się tak blisko pojęcia "kultury narodowej"

odpowiednio szerokiego wycinka języka oraz posiadać mechanizmy pozwalające modelować, strukturalizować 'znaczenie' zdań (lub nawet wypowiedzi, tekstów). Dobrze jest gdy obydwa te aspekty są ze sobą spójnie powiązane - ideałem by było pod pojęciem gramatyki połączyć zarówno syntaktykę jak i semantykę języka.

Próba stworzenia takiej teorii została podjęta przez Richarda Montague pod koniec lat siedemdziesiątych i zaowocowała gramatyką nazwaną PTQ (*ang. The Proper Treatment of Quatification in Ordinary English).*

Stanowi ona połączenie cech charakterystycznych dla gramatyki transformacyjnej (np. budowa reguł produkcji) oraz gramatyki kategorialnej (np. konstrukcja systemu kategorii syntaktycznych). Jednak najważniejszą jej nowością jest mechanizm wiążący reguły syntaktyczne z regułami generacji reprezentacji semantycznej - zapisywanej w postaci formuł logiki intensjonalnej.

Zależność reguł syntaktycznych i reguł translacji (reguł semantycznych) jest wyrażana przez najważniejszą zasadę tej gramatyki - zasadę kompozycyjności (*ang the principle of compositionality):*

1. Dla każdej kategorii syntaktycznej istnieje odpowiadający jej typ logiki intensjonalnej.
2. Każde wyrażenie podstawowe, dowolnej kategorii syntaktycznej ('wyraz') posiada odpowiadające mu wyrażenie logiki intensjonalnej - spełniające pod względem typu warunek (1).
3. Każda reguła syntaktyczna ma dokładnie jedną, odpowiadającą jej regułę semantyczną: ich argumenty spełniają pod względem typu warunek (1), w efekcie działania obydwu reguł otrzymujemy wyrażenia spełniające co do typu warunek (1). Ponadto działanie obydwu reguł można opisać jako funkcję na argumentach wejściowych.

Typ gramatyk spełniających powyższą zasadę uzyskał w ostatnich latach miano gramatyki kompozycyjnej.

Bardzo istotnym narzędziem dla PTQ jest logika intensjonalna (dalej będę używał skrótu LI). Powstała ona na bazie logiki standartowej (predykatów i klauzul) rozszerzonej o elementy logiki modalnej, temporalnej, operator lambda λ oraz dwa nowe operatory intensji $\hat{}$ i ekstensji $\check{}$. Dwa ostanie operatory (pojęcia) zostały wprowadzone przez samego Montague a ich inspiracją była teoria możliwych światów (logika możliwych światów) - zdobywająca sobie w momencie powstania PTQ dużą popularność w badaniach nad językiem.

Dla dowolnej formuły LI α znaczenie jej, oznaczane jako $\|\alpha\|^{M,w,t,g}$, zależy od modelu M (zbioru aksjomatów i tautologii), fukcji waluacji zmiennych g oraz pary iloczynu kartezjańskiego $\langle w,t \rangle \in W \times T$, gdzie W jest zbiorem możliwych światów, a T zbiorem momentów czasu (dowolnym zbiorem nieprzeliczalnym ze zdefiniowaną przechodnią relacją porządkującą).

W myśl tak określonego znaczenia formuł IL możemy przybliżyć działanie operatorów intensji i ekstensji:
intensja: jeżeli $\alpha \in ME_a$ wtedy $\|\hat{}\alpha\|^{M,w,t,g}$ jest tą funkcją h o dziedzinie W×T taką, że dla wszystkich $\langle w',t' \rangle \in W \times Th(\langle w',t' \rangle) = \|\alpha\|^{M,w',t',g}(\in ME_{\langle s,a \rangle})$.
(gdzie a jest dowolnym typem IL, ME_a zbiorem wszystkich poprawnych wyrażeń typu a, <s,a> oznacza typ derywowany od a - typ intensja z typu a, s stanowi swoisty modyfikator typu)
ekstensja: jeżeli $\alpha \in ME_{\langle s,a \rangle}$ wtedy $\|\check{}\alpha\|^{M,w',t',g} = \|\hat{}\alpha\|^{M,w,t,g}(\langle w,t \rangle)(\in ME_a)$
(t oznacza standartowy typ logiczny ('boolowski') ze zbiorem wartości {"0", "1"})

Na bazie oryginalnego PTQ generującego wycinek języka angielskiego zbudowałem analogiczną gramatykę dla wycinka języka polskiego. Brak miejsca na pełną jej

prezentację, ograniczę się jedynie do zaprezentowania prostego przykładu.
Rozpatrzmy zdanie *Jan mówi.*
Powstaje ono z wyrażeń podstawowych *Jan* (kategorii T - term - głownie imiona własne) i *mówić* (kategori IV - czasownik nieprzechodni) za pomocą reguły S4:

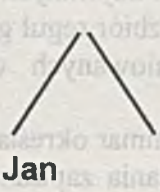
S4. Jeżeli $\alpha \in P_T$ i $\delta \in P_{IV}$ wtedy $F_4(\alpha,\delta) \in P_t$ gdzie $F_4(\alpha,\delta)=\alpha,\delta'$ i δ' powstaje jako rezultat zastąpienia wystąpienia pierwszego czasownika (kategorii B_{IV}, B_{TV}, $B_{IV|t}$, lub $B_{IV|IV}$) w każdej frazie IV, będącej oczkiem łańcucha fraz IV jakim jest δ, przez jego trzecią osobę liczby pojedynczej czasu teraźniejszego, rodzaju:

- męskoosobowego jeżeli w α występuje choć jedno B_{CN}, lub B_T ,
- rodzaju pierwszego B_{CN}, lub B_T w α (rodzaj jest to tylko istotny z formalnego punktu widzenia - nie wpływa na formę czasownika).

(gdzie: P_X oznacza zbiór wszystkich poprawnych fraz kategorii X, B_X zbiór wszystkich wyrażeń podstawowych kategorii X, F_n jest funkcją na 'stringach' identyfikowaną indeksem n, t oznacza kategorię zdanie;
unikając rozbudowanej definicji, można pojęcie **łańcucha fraz IV** nieformalnie opisać jako frazę czasownikową współrzędnie złożoną z innych fraz czasownikową, **oczko** - to fraza IV nie posiadająca już tej właściwości - fraza składowa łańcucha) której odpowiada reguła translacyjna (semantyczna) T4:

T4. Jeżeli $\alpha \in P_T$ i $\delta \in P_{IV}$ oraz $\alpha \Rightarrow \alpha'$ i $\delta \Rightarrow \delta'$ to wtedy $F_4(\alpha,\delta) \Rightarrow \alpha'(^\wedge \delta')$.

Zastosowanie reguły S4 można zobrazować przez proste drzewo derywacji:



Indeks skojarzony z węzłem sygnalizuje użytą regułę syntaktyczną.

Dzięki zasadzie kompozycyjności możemy potraktować numery reguł syntaktycznych bezpośrednio jako numery, skojarzonych z nimi, homomorficzną zależnością, reguł translacji (semantycznych). W dalszej kolejności dokonujemy translacji wyrażeń podstawowych w liściach drzewa do odpowiadających im formuł logiki intesjonalnej (na podstawie zdefiniowanego przez gramatykę słownika) - i jesteśmy już w stanie rekurencyjnie wyliczyć wartość reprezentacji semantycznej zdania *Jan mówi.* Formalny zapis tego procesu jest prezentowany poniżej:

1) **Jan** $\lambda \Rightarrow P[P\{j\}]$ translacja wyrażenie podstawowe,
2) **mówić** $\Rightarrow$ **mówić'** translacja wyrażenie podstawowe,
3) $\lambda P[P\{j\}](^\wedge$**mówić'**) z 1) i 2) przez zastosowanie T4,
4) $^\wedge$**mówić'**{j} podstawienie wartości do operatora lambda
5) $^{\vee\wedge}$**mówić'**(j) rozwinięcie oznaczenia {} do $^\vee$,
6) **mówić'**(j) uproszczenie operatorów ekstensji i intensji

(j oznacza stałą ze zbioru A - zbioru istnień, bytów realnych bądź abstrakcyjnych dowolnego z możliwych światów; zbiór A jest elementem modelu LI)

Otrzymany efekt nie zaskakuje, ale też dowodzi skuteczności gramatyki kompozycyjnej.

3. 'Implementacja' teorii lingwistycznej pod kątem zastosowań informatycznych

Nie chcąc wyważać raz otwartych drzwi postanowiłem wykorzystać doświadczenia już istniejących systemów informatycznych pracujących w oparciu o gramatykę kompozycyjną. Niewątpliwie najciekawszym

z nich jest system automatycznego tłumaczenia tekstu Rosetta opracowany przez zespół prof. Jana Landsbergen'a w Philips Research Laboratories w Eidhoven.

Profesor Landsbergen opierając się na gramatyce PTQ sformułował swoją własną gramatykę nazwaną M-Grammar czyniąc ją bazą dla swoich rozwiązań. Dlaczego nie zostało zastosowane PTQ w swojej czystej postaci?

PTQ, dla informatyka, posiada dwa podstawowe mankamenty: reguły syntaktyczne operują na ciągach symboli ('stringach') - dość kłopotliwej strukturze danych, oraz PTQ to gramatyka o charakterze generatywnym - zbudowanie analizatora ('parsera') w oparciu o nią nie jest zbyt jednoznaczne.

Obydwie powyższe niedogodności zostały ominięte w konstrukcji M-Grammar:

- operuje ona nie na ciągach symboli ale specjalnej strukturze typu drzewa nazwanej S-Tree (drzewo jest dobrze zadomowioną strukturą danych w informatyce),
- zostały sformułowane warunki jakie musi spełniać gramatyka gwarantujące łatwość implementacji na jej podstawie analizatora - warunki spełnione przez M-Grammar. Między innymi oprócz respektowania zasady kompozycyjności każda reguła syntaktyczna jest 'zobowiązana' do posiadania swojej analitycznej wersji.

Niestety profesor J. Landsbergen mając na uwadze główny swój cel: automatyczne tłumaczenie tekstu dokonał w swojej gramatyce kilku poważnych odstępstw od pierwowzoru PTQ. Zrezygnował chociażby z cech charakterystycznych dla gramatyki kategorialnej - co burzy homomorfizm kategorii syntaktycznych i typów LI. W systemie Rosetta nośnikiem znaczenia w procesie tłumaczenia pomiędzy jednym językiem a drugim nie są formuły LI, podane w sposób jawny, a jedynie drzewa derywacyjne otrzymane w wyniku zastosowania reguł gramatyki M-Grammar.

Musiałem przywrócić osłabione 'semantyczne' cechy PTQ - w rezultacie powstała moja własna modyfikacja gramnatyki M-Grammar, którą nazwałem PM-Grammar. Zachowuje ona całą siłę 'polskiego' PTQ a także nie traci nic z efektywności rozwiązań profesora J. Landsbergena.

Gramatyka PM-Grammar operuje również na strukturach S-Tree:

$N[r_1/t_1,...,r_n/t_n]$ $(n \geq 0)$ gdzie:

- $N=C\{a_1:v_1,...,a_k:v_k\}$ $(k \geq 0)$ jest węzłem drzewa oraz:
  - C to dowolny symbol (np. reprezentujący kategorię syntaktyczną) skojarzony z węzłem N,
  - a_i (dla $k \geq i \geq 0$) tworzą zestaw atrybutów danego węzła - każdy z określonym zbiorem dozwolonych wartości,
  - v_i (dla $k \geq i \geq 0$) wartości nadane poszczególnym atrybutom,
- lista par r_j/t_j (dla $k \geq i \geq 0$) reprezentuje dalsze gałęzie drzewa - poddrzewa:
  - r_j określa relację w jakiej pozostaje powiązana z nią gałąź w stosunku do pozostałych z danego węzła,
  - t_j reprezentuje gałąź odchodzącą od węzła N - drzewo (podrzewo) S-tree.

Węzeł o pustej liście gałęzi jest nazywany węzłem terminalnym.

Gramatykę PM-Grammar definiujemy w dwóch fazach. W pierwszym rzędzie określamy strukturę drzew S-Tree poprzez opisanie węzłów wraz listami atrybutów i zbiorami dozwolonych wartości oraz zdefiniowanie zbioru używanych relacji. Następnie konstruujemy zbiór reguł gramatyki operujących na zdefiniowanych wcześniej strukturach S-Tree.

Gramatyka PM-Grammar określa również sposób przejścia od zdania zapisanego jako

ciąg znaków do odpowiadającej mu struktury S-Tree. Narzuca konieczność zdefiniowania gramatyki wstępnej, której jedynym zadaniem jest nadanie każdemu poprawnemu zdaniu wejściowemu przynajmniej jednej struktury S-Tree. Nie wszystkie wytworzone przez gramatykę wstępną struktury muszą zostać zaakceptowane przez gramatykę PM-Grammar jako poprawne - warunkiem jest aby dla zdania poprawnego w myśl PM-Grammar, gramatyka wstępna nadała przynajmniej jedną, akceptowaną dalej strukturę S-Tree.

Rozpatrzmy zastosowanie PM-Grammar do analizy poprzedniego przykładu.

Musimy określić niezbędne elementy struktury S-Tree:

Kategoria:**t**

atrybut	wartość
typ	pojedyncze, złożone
tryb	ozn
negacja	tak, nie

Kategoria: **IV**

atrybut	wartość
podstawa	wszystkie wyrazy reprezentujące leksemy znajdujące się w słowniku 'polskiego' PTQ dla danej kategorii - wszystkie wyrazy w formie morfologicznej podstawowej
typ	szeregowa, orzekająca, czasowa, dopełniona, prosta
czas	teraź, przy, prze
tryb	ozn
negacja	tak, nie
forma	bezokolicznik, orzekacz, przysłownik, zapowiadzacz
rodzaj	m1, m2, m3, f, n1, p1, p2, p3
liczba	poj
osoba	1,2,3

Kategoria T

atrybut	wartości
podstawa	wszystkie wyrazy reprezentujące leksemy znajdujące się w słowniku 'polskiego' PTQ dla danej kategorii - wszystkie wyrazy w formie morfologicznej podstawowej
typ	szeregowa, rzeczownikowa, prosta
przypadek	miano, dopeł, celow, biern, miejs, narzę, wołacz
rodzaj	m1, m2, m3, f, n1, n2, p1, p2, p3
liczba	poj
osoba	1,2,3

Zbiór relacji składa się z trzech elementów {lewy, pom, prawy} - ograniczony rozmiar PM-Grammar nie wymagał silniejszego wykorzystania tego mechanizmu.

Zainspirowany świetną pracą S. Szpakowicza [19] zapisałem gramatykę wstępną w formacie gramatyki DCG z kontekstem. Dodatkową zaletą tego formalizmu jest zapis zbliżony do PROLOG'u.

ZDANIE
= t(2, złożone, tryb, negacja)

t(nr, złożone, tryb, negacja)
= t(nr, pojedyncze, tryb, negacja)

t(nr, pojedyncze, tryb, negacja)
= T(nr, _, szeregowa, mian, liczba, osoba, rodzaj){lewy} IV(_, szeregowa, czas, tryb, negacja, forma, rodzaj, liczba, osoba){prawy} _ALT(forma, orzecznik, przysłownik, zapowiadacz)
(_ ALT - nie zawodzi tylko wtedy gdy wartość pierwszego argumentu jest elementem listy podanej jako drugi argument)

T(nr, podstawa, szeregowa, przypadek, poj, osoba, rodzaj)
=T(nr, podstawa, rzeczownikowa, przypadek, poj, osoba, rodzaj)

T(_, podstawa, rzeczownikowa, przypadek, liczba, osoba, rodzaj)
= T(_, podstawa, prosta, przypadek, liczba, 3, rodzaj)

T(_, podstawa, prosta, przypadek, liczba, 3, rodzaj)
= #sym _SŁOWNIK(sym, podstawa, T, przypadek, rodzaj, liczba, 3, _)

IV (podstawa, szeregowa, czas, tryb, negacja, forma, rodzaj, liczba, osoba)
= IV(podstawa, orzekajaca, czas, tryb, negacja, forma, rodzaj, liczba, osoba)
(... pomijam tutaj podobną listę reguł produkcji jak w przypadku kategorii T)

IV (podstawa, prosta, _, _, tak, forma, rodzaj, liczba, osoba)
= #sym _SŁOWNIK(sym, podstawa, IV, _, rodzaj, liczba, osoba, forma)
(# oznacza zdjęcie pobranie symbolu z listy wejściowej, predykat _SŁOWNIK sprawdza wystąpienie danego symbolu w słowniku)

Stosując powyższe reguły do zdania *Jan mówi* otrzymujemy następującą, reprezentującą je, strukturę drzewa S-Tree:

t{typ: pojedyncze, tryb: ozn, negacja: tak}

lewy / \ prawy

T{podstawa: **Jan**, typ: prosta ,przypadek:miano , rodzaj:f ,liczba:poj , osoba:3 }

IV{podstawa:**mówić**, typ:prosta , czas:teraź, tryb: ozn , negocjacja:tak, forma:orzekacz , rodzaj:f, liczba:poj , osoba:3

Gybyśmy przepisali liście drzewa w kolejności od lewego do prawego, a następnie uwzględniając zawartą w nich informację morfologiczną, rozwinęli wartość atrybutu **podstawa** do pełnej postaci (wyrazu) to otrzymalibyśmy: *Jan mówi.*

Aby dokonać dalszej analizy rozważanego zdania musimy zdefiniować następującą regułę (w wersji generatywnej i analitycznej):

MS4. (schemat reguły) $<P_4,I_4,A_4>$ gdzie

P_4={Q: Q jest zbiorem ścieżek w drzewie S-tree}

$I_4(<t_1,t_2>)$=(def.){Q: takich że jeśli t_1=T{podstawa=pd_1, typ=tp_1, przypadek=p_1, liczba=l_1, osoba=o_1, rodzaj=r_1}[] oraz t_2=IV{podstawa=pd_2, typ=tp_2, czas=cz_2, tryb=ozn, negacja=tak, forma=fm_2, rodzaj=r_2, liczba=l_2, osoba=o_2}[] to wtedy Q jest zbiorem ścieżek wyznaczających w drzewie t_2 łańcuch fraz IV oraz Q jest zdefiniowany następująco:

- gdy tp_2≠szeregowa to Q={p: ścieżki do pierwszego w t_2 drzewa terminalnego kategorii IV, TV, IV|t lub IV||IV i wszystkich węzłów pomiędzy nim a wierzchołkiem drzewa - gdy nie istnieje poszukiwane drzewo terminalne to Q jest zbiorem pustym}

(Pierwszy element jest wyznaczony przez najmniejszą ścieżkę - wyznaczoną przez znaną z matematyki operację porównywania ciągów numerycznych;

[] oznacza dowolne drzewo S-Tree)

- tp_2=szeregowa (przypadek dla nas nieistotny)

$A_4(Q,<t_1,t_2>)$=t{typ=pojedyncze, tryb=ozn, negacja=ng_2 } [lewy/t_1, prawy/(t_2'=)IV{podstawa=pd_2, typ=tp_2, czas=cz_2, tryb=tb_2, negacja=ng_2, forma=fm_2, rodzaj=r_1, liczba=l_1, osoba=o_1}[]] gdzie t_2' powstaje z t_2 przez zamianę dla każdego p∈Q, w każdym drzewie t_2.p wartości atrybutów forma, rodzaj, liczba, osoba na (odpowiednio do kolejności wymieniania): orzekacz, r_1, l_1, 3.

MS4'. (schemat reguły analitycznej) $<P_4, I_4', A_4'>$ gdzie:

$I_4'(v)$={Q: takie że jeśli v=t{typ=pojedyncze, tryb=ozn, negacja=tak} [lewy/ (t_1=) T{podstawa=pd_1, typ=tp_1, przypadek=p_1, liczba=l_1, osoba=o_1, rodzaj=r_1}[], prawy/ (t_2=) IV{podstawa=pd_2, typ=tp_2, czas=cz2, tryb=ozn, negacja=tak, forma=fm_2, rodzaj=r_1, liczba=l_1, osoba=o_1}[]] to wtedy Q jest zbiorem ścieżek wyznaczających w drzewie t_2 łańcuch fraz IV},

A4'(Q,v)=$<t_1,t_2'>$ gdzie t_2' powstaje z t_2 przez zmianę dla każdego p∈Q atrybutów drzew t_2.p do atrybutów formy podstawowej.

W wyniku zastosowania MS4' powstaje para drzew terminalnych - co kończy proces analizy. Jego efektem jest powstałe drzewo derywacji D-Tree:

MS4'

T{podstawa: **Jan**, typ: prosta
,przypadek:mianq , rodzaj:f
,liczba:poj , osoba:3 }

IV{podstawa:**mówić**, typ:prosta
, czas:teraź. tryb: ozn
, negocjacja:tak, forma:orzekacz
, rodzaj:f, liczba:poj, osoba:3

Bardzo przypomina drzewo derywacji z 'polskiego' PTQ - inne są tylko oznaczenia reguł i liście. Dlatego też nie musimy wprowadzać żadnych zmian w homomorfizmie reguł syntaktycznych i semantycznych oraz w samej konstrukcji reguł semantycznych, jedynie słownik translacji wyrażeń podstawowych musi zostać sformułowany jako relacja pomiędzy drzewami terminalnymi a formułami logiki intensjonalnej.

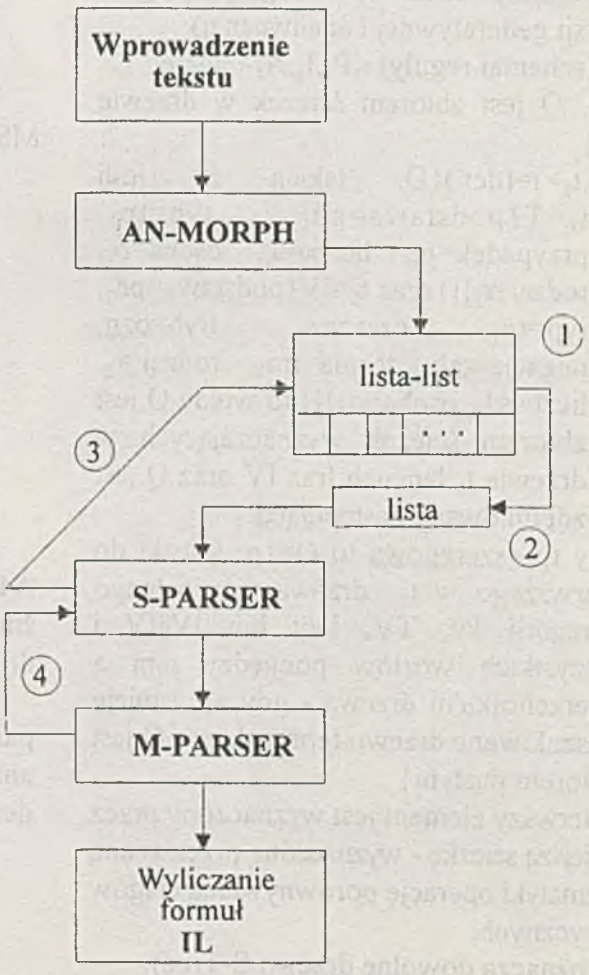
Efekt translacji zdania *Jan mówi* do reprezentacji semantycznej jest taki sam, jak otrzymany przez zastosowanie reguł 'polskiego' PTQ.

4. Realizacja programistyczna systemu

Dysponując odpowiednio sformułowaną teorię lingwistyczną jedynym co pozostaje to zrealizować zaprojektowane rozwiązania w postaci systemu informatycznego. I mimo, iż powyższe twierdzenie brzmi może nieco ironicznie to jednak jest w znacznym stopniu prawdziwe. Odpowiedni sposób sformułowania teorii lingwistycznej i trafność proponowanych w niej rozwiązań, rysunek 1.

Pierwszy moduł jest odpowiedzialny za komunikację z użytkownikiem i wprowadzenie danych - ciągu znaków reprezentującego zdanie.

Moduł AN-MORPH - to analizator morfologiczny. Jego zadaniem jest podział wprowadzonego ciągu znaków na poszczególne jednostki: wyrazy, akceptowane znaki interpunkcyjne. Następnie analizator próbuje przypisać każdemu z wprowadzonych wyrazów przynajmniej jeden pokrywający się z nim wyraz morfologiczny (niefomalnie: leksem wyrażony w odpowiedniej formie morfologicznej). Nieodnalezienie żadnego takiego wyrazu morfologicznego - oznacza nie rozpoznanie danej jednostki tekstu przez system, analizator morflogiczny kończy swoją pracę z komunikatem o błędzie. W przypadku pomyślnego zakończenia pracy zostaje



rys. 1

wygenerowana lista list ①. Każda z list składowych zawiera uzgodnione z danym wyrazem wyrazy morfologiczne, zapisane w postaci terminalnych drzew S-Tree (zawierających pełną informację morfologiczną). Analizator morfologiczny pracuje w oparciu o algorytm wykorzystujący analizę tematów i tablice wzorców odmiany. Został skonstruowany w oparciu o świetną pracę dr Janusza S. Bienia [17]. Szybkie przeszukiwanie słownika tematów wykorzystuje metodę podwójnej listy inwersyjnej opracowaną w Zakładzie Systemów Informacyjnych Politechniki Wrocławskiej. Cały moduł analizatora został napisany w C.

Z listy list tworzymy kolejno wszystkie możliwe ciągi drzew terminalnych (lista ②) - kolejne możliwe warianty zdania. Stanowią one dane wejściowe modułu S-PARSER'a.

S-PARSER - to moduł wstępnego analizatora syntaktycznego, działający w oparciu o zaprezentowaną wcześniej gramatykę wstępną. Próbuje dla każdego dostarczonego ciągu terminalnych drzew S-Tree (lista ②), zbudować przynajmniej jedno pełne drzewo S-Tree. Gdy to się nie powiedzie następuje nawrót po kolejny ciąg drzew terminalnych.

S-PARSER został napisany w PROLOG'u - już sam charakter formalizmu gramatyki 'wymuszał' przyjęcie takiego rozwiązania. Rzeczywiście przejście od gramatyki do programu było zadaniem trywialnym. Problem sprawiły jedynie otwarte pętle rekurencji dla fraz złożonych współrzędnie. Doprowadzały do nieskończonych pętli nawrotów, szczególnie w przypadku gdy wejściowy ciąg drzew terminalnych reprezentował zdanie niepoprawne. Skutecznym rozwiązaniem okazało się użycie górnych ograniczeń ilości wywołań niebezpiecznych pętli. S-PARSER był również źródłem bardzo poważnych kłopotów natury technicznej - na Politechnice Wrocławskiej nie ma żadnej innej implementacji PROLOG'u pod DOS'a, niż stary i bardzo zawodny Turbo Prolog 2.0. Komplikacje okazały się na tyle poważne, że zostałem zmuszony do ustanowienia komunikacji pomiędzy poszczególnymi modułami przez dysk i do ograniczenia obecności PROLOG'u w systemie.

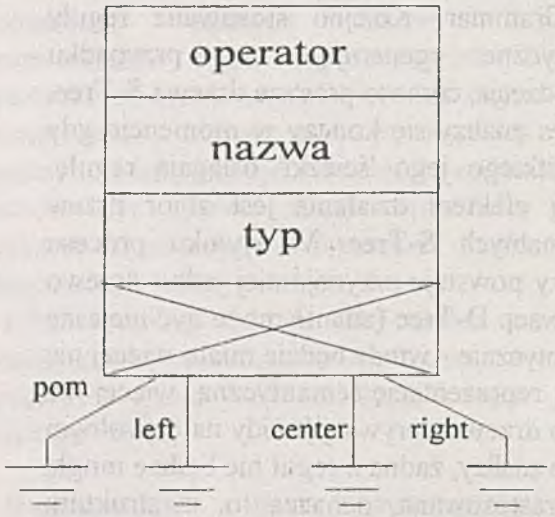
Po nadaniu przez S-PARSER struktury S-Tree wprowadzonemu ciągowi drzew terminalnych (czyli w myśl gramatyki wstępnej zdanie jest poprawne), na tak przygotowanej strukturze, rozpoczyna pracę M-PARSER działający w oparciu o gramatykę PM-Grammar. Kolejno stosowane reguły analityczne generują, w przypadku powodzenia, coraz to prostsze drzewa S-Tree. Proces analizy się kończy w momencie gdy wszystkiego jego 'ścieżki' osiągają regułę, której efektem działania jest zbiór drzew terminalnych S-Tree. W wyniku procesu analizy powstaje przynajmniej jedno drzewo derywacji D-Tree (zdanie może być niejasne semantycznie - wtedy będzie miało więcej niż jedną reprezentację semantyczną, więcej niż jedno drzewo derywacji). Gdy na dowolnym etapie analizy, żadna z reguł nie będzie mogła być zastosowana, oznacza to, iż struktura wejściowa S-Tree reprezentuje zdanie nieakceptowane przez gramatykę PM-Grammar (niepoprawne w mysl gramatyki PM-Grammar).

Strzałki ③ i ④ na rysunku 1 symbolizują nawroty wykonywane przez system w wypadku pomyślnego lub też niepomyślnego zakończenia pracy danego modułu. Dowolne zdanie wprowadzone do systemu może być obarczone dwoma rodzajami niejsności: syntaktyczną (gramatyka wstępna jest w stanie przypisać więcej niż jedną akceptowaną przez PM-Grammar strukturę) oraz semantyczną - wspomnianą wcześniej. Co więcej niejasność syntaktyczna przeważnie pociąga za sobą semantyczną.

Ostatnim etapem pracy systemu jest wyliczenie, na podstawie drzewa derywacji D-Tree i reguł semantycznych, końcowej postaci formuły LI. W rzeczywistości moduł M-PARSER'a przekazuje do dalszej obróbki

nie pojedyncze drzewo ale strukturę opisującą wszystkie wygenerowane drzewa. Zamiast węzłów pojawiają się listy możliwych węzłów. Moduł wyliczający po kolei generuje wszystkie możliwe drzewa i wylicza ich wartość.

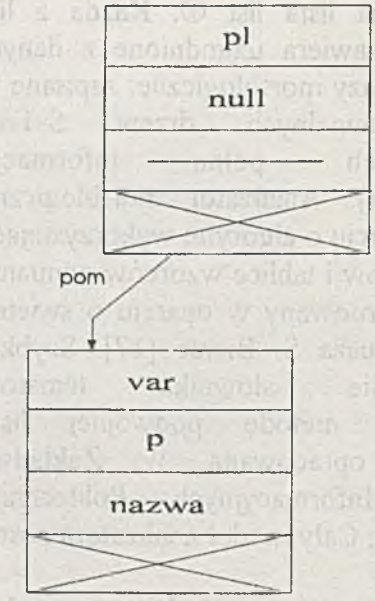
Sam proces wyliczenia przebiega w typowy, rekurencyjny sposób - nie typowe są tylko wyrażenia LI dla jakich on przebiega.



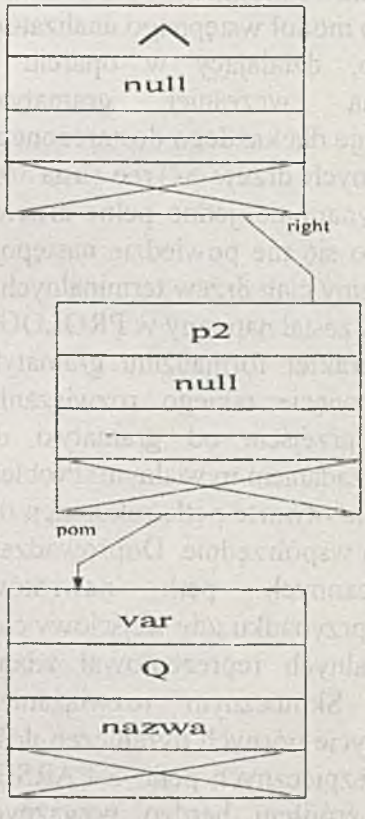
rys. 2

Pierwszy z poważnych problemów to złożoność typów LI (np. intensja z intensji z predykatu, zmienna typu predykat trzymiejscowy) oraz skomplikowane operatory: predykat dwu, trzy miejscowy, operator lambda o argumentach: zmienna określonego typu, wyrażenie zawierające tą zmienną, podstawiona wartość. Aby rozwiązać rodzące się trudności przyjąłem strukturę danych przedstawioną na rysunku 2.

Przykładem zastosowania tej struktury mogą być definicje dwóch zmiennych: P typu predykat jednomiejscowy (rys. 3) oraz Q typu intensja z predykatu dwumiejscowego (rys. 4)



rys. 3



rys. 4

5. Perspektywy dalszych prac badawczych

Celem stworzonego systemu jest jedynie eksperyment z modelowaniem semantyki zdań języka naturalnego. Mam jednak nadzieję, że jego użyteczność nie zamyka się w ciasnych ramach tego tematu. Może być traktowany jako pierwszy krok w kierunku systemu automatycznego tłumaczenia tekstu nowej generacji, tłumaczącego na zasadzie przeniesienia znaczenia. Takie ambicje ma system *Rosetta*, który zresztą po wielu latach rozwoju zbliżył się do momentu możliwości jego praktycznego wykorzystania. Pominąwszy ostani moduł mój system stanowi 'pierwszą połowę' sytemu *Rosetta* - pozostaje 'tylko' druga generacyjna część.

Zarysowane w mojej pracy rozwiązania mogą się również stać impulsem w kierunku rozwiązań w dziedzinie komunikacji pomiędzy człowiekiem a komputerem w języku naturalnym, a nawet w kierunku systemów inteligentnego przetwarzania informacji w języku naturalnym.

Jednym z najpoważniejszych wyłaniających się problemów teoretycznych jest interpretacja tak skomplikowanego języka formalnego jakim jest logika intensjonalna. Należy tutaj zwrócić uwagę, iż gramatyka kompozycyjna jedynie 'strukturalizuje' semantykę języka naturalnego do poziomu leksemów - powstaje pytanie jak zejść 'poniżej'? Może techniki konceptualne - nie dające ciągle poważnych rezultatów w lingwistyce, przy tak użytkowym zastosowaniu, jakim jest system informatyczny, dadzą w połączeniu z gramatyką kompozycyjną obiecujące rezultaty? Moja praca również pozostawia otwartą kwestię tekstu - powiązanego zespołu zdań.

Bibliografia

[1] Lyons John "Wstęp do językoznawstwa", (tytuł oryginału "Introduction to Theoretical Linguistics"), PWN, Warszawa, 1975.

[2] Lyons John "Semantyka" (tytuł oryginału "Semantics"), PWN, Warszawa, 1983.

[3] Montague Richard "English as a Formal Language" zawarte w B. Visentini et al , eds., "Linguaggi nella societa e nella technica", Milan: Edizioni di Comunità, str. 189-224. przedrukowane w [29]

[4] Montague Richard "The Proper Treatment of Quantification in Ordinary English", zawarte w Hintikka J., Moravcsik J., Suppes P. "Approaches to Natural Language", Dordrecht: D. Reidel, str 221-242, przedrukowane w [29]

[5] Dowty D. R., Wall, R. E., Peters S. "Introduction to Montague Semantics", Dordrecht: D. Reidel, 1981

[6] Dowty D. R. "Montague Grammar and Word Meaning", Dordrecht: D. Reidel

[7] "Machine translation, Theoretical and methodological issueus" edycja zbiorcza pod redakcją Sergei Nirenburg, University Press, Cambrige 1987, w serii wyd. "Studies in Natural Language Processing".

[8] "A Survey of Machine Translation" edycja zbiorcza pod redakcją J. Slocum, University Press, Cambrige 1989, w serii "Studies in Natural Language Processing".

[9] Landsbergen Jan, "Montague Grammar and Machine Translation", artykuł w zbiorczym wydaniu "Linguistic Theory and Computer Applications" Academic Press Limited, 1987.

[10] Landsbergen Jan, "The Rosetta Project".

[11] Landsbergen Jan, Odijk Jan, Schenk Andrè, "The Power of Compositional Translation", artykuł w "Literary and Linguistic Computing", Vol. 4, No. 3,

1989, Oxford University Press.
[12] Landsbergen Jan, Appelo Lisette, Fellinger Carel, "Subgrammars, Rule Classes and Control in the Rosetta Translation System", materiały na "3rd Conference ACL, European Chapter", kwiecień 1987
[13] Landsbergen Jan, "Adaptation of Montague Grammar to The Requirements of Parsing" artykuł w zbiorczej publikacji "Formal Methods in the Study of Language" część 2, pod redakcą Groenendijk J.A.G., Janssen T.M.V., Stokhof M.B.J., MC Track 136, Mathematical Centre, Amsterdam, 1981, strony 399-420.
[14] Landsbergen Jan, "Isomorphic Grammars and their Use in the Rosetta Translation System", w zbiorczej publikacji "Machine Translation Today", pod redakcją King M., Edinburg: Edinburg University Press, 1985.
[15] van Munster Elly "The treatment of Scope and Negation in Rosetta" materiały na "12th International Conference on Computational Linguistics", Budapeszt, 22-27 VIII 1988.
[16] Lehrberger John "Machine Translation - Linguistic Characteristics of MT Systems and General Methodologi of Evaluation", 1988
[17] Bień Janusz S. "Koncepcja słownikowej informacji morfologicznej i jej komputerowej weryfikacji", Wydawnictwo Uniwersytetu Warszawskiego, Warszawa 1991.
[18] Szymanowska Irena "Algorytmy analizy morfologicznej" Centrum Informacji Naukowej i Ekonomicznej, Warszawa, 1978.
[19] Szpakowicz Stanisław "Formalny opis składiowy zdań polskich", Wydawnictwo Uniwersytetu Warszawskiego, Warszawa, 1983
[20] Harasymowicz Ryszard "Analiza syntaktyczna języka polskiego", praca magisterska Politechnika Wrocławska, Zakład Systemów Informacyjnych, 1992.
[21] "Man-Machine Interfaces and Natural Language Processing" niepublikowane materiały pomocnicze na kierunku Computer Science, na Technical University of Bristol.
[22] Baldwin J. F., Martin T. P., Pilsworth B. W. "FRIL - Programming Manual", Fril System Ltd., Bristol, 1988
[23] Baldwin J. F., Martin T. P., Pilsworth B. W. "FRIL - Applications Manual", Fril System Ltd., Bristol, 1988
[24] Bąk Piotr "Gramatyka języka polskiego", Wiedza Powszechna, Warszawa, 1978
[25] Bieńkowski Kosma, Sobecki Janusz "Turbo Prolog 2.0 Kompendium", Komputerowa Oficyna Wydawnicza "HELP", Warszawa, 1991
[26] "Turbo Prolog Reference Guide" Borland International, 1988.
[27] Jodłowski Stanisław, Taszycki Witold "Słownik ortograficzny" Zakład Narodowy im. Ossolińkich - Wydawnictwo, Wrocław, 1981
[28] Brady J.M "Informatyka teoretyczna w ujęciu programistycznym" (tytuł oryginału "The Theory of Computer Science A Programming Approach"), Wydawnictwa Naukowo-Techniczne, Warszawa, 1983
[29] Thomason R. H. "Formal Philosophy: Selected Papers of Richard Montague", New Haven, Yale University Press, 1974

Byrys Czerniejewski
Student IV roku Akademii Górniczo - Hutniczej w Krakowie

Moduł automatycznego tłumacza pytań jako interfejs do bazy danych

1. Wstęp

Powszechnym trendem w technice oprogramowania jest dążenie do zwiększenia przyjazności powstających programów dla użytkownika. Dotyczy to zarówno systemów operacyjnych, jak i programów pod nimi uruchamianych. Celem takiego działania jest oczywiście ułatwienie użytkowania systemów komputerowych (a więc komputerów wraz z ich oprogramowaniem) i w efekcie zwiększenie rynku ich zbytu. Dlatego też wspomniany trend zaważył przede wszystkim na kierunku rozwoju oprogramowania dla mikrokomputerów, a następnie dla stacji roboczych, gdyż tu możliwości poszerzenia rynku były największe.

Stopień przyjazności oprogramowania zależy od sposobu w jaki oprogramowanie to komunikuje się z użytkownikiem. Potencjalny nabywca preferuje programy z którymi można porozumiewać się w sposób maksymalnie zbliżony do naturalnego sposobu komunikacji międzyludzkiej, czyli z użyciem języka naturalnego (Natural Language - NL),[1] lub gestu. O ile prace nad analizą języków naturalnych nie przyniosły dotąd zadowalających rezultatów, język gestów udało się uprościć do jednego gestu: gestu wskazywania. Wykorzystano przy tym możliwość wskazywania na słowa z pewnego ograniczonego słownika, co jednak nie ma nic wspólnego z analizą językową. Jeśli zwrócimy dodatkowo uwagę na konieczność wprowadzania tekstu z klawiatury przy braku sprawnych systemów rozpoznających mowę i wynalezienie urządzeń wskazujących (mysz, joystick, trackball) jasne staną się powody wielkiej popularności systemów z graficznym interfejsem użytkownika (Graphic User Interface - GUI), przy stosunkowo małym zainteresowaniu systemami z interfejsem naturalnojęzykowym. Interfejsy graficzne posiadają jednak jedną poważną wadę: umożliwiają przekazanie systemowi jedynie najprostszych poleceń i informacji.[2]

W niniejszym artykule przedstawiono próbę pokonania trudności związanych z analizą języka naturalnego na przykładzie interfejsu w języku polskim służącego do odpytywania baz danych, którego konstrukcja jest tematem pracy dyplomowej autora, powstającej w Katedrze Informatyki AGH w Krakowie.

2. Systemy wyszukiwania informacji

Typową dziedziną zastosowania naturalnojęzykowej komunikacji z komputerem są systemy wyszukiwania informacji (data retrieval systems), czyli systemy złożone z bazy danych, języka zapytań (Query Language - QL) umożliwiającego dostęp do informacji zawartych w bazie i interfejsu użytkownika. Ponieważ języki zapytań są podzbiorami języków manipulacji danymi (Data

[1] "język naturalny - język utworzony w procesie ewolucji biologicznej lub społecznej (...)" - za Słownikiem informatyki polsko - angielsko - rosyjskim pod red. Hanny Jezierskiej, WNT, Warszawa (wyd. 3.) 1989

[2] por.: Elżbieta Dobryjanowicz, Podstawy przetwarzania języka naturalnego, Wybrane metody analizy składniowej, Akademicka Oficyna Wydawnicza RM, Warszawa 1992, str. 11

Manipulation Languages - DML), które są elementami systemów zarządzania bazami danych, możemy uznać systemy wyszukiwania informacji za podzbiory systemów zarządzania bazami danych (Database Management System - DBMS). Możliwe jest jednak istnienie niezależnego (standalone) programu, bądź programu osadzonego wykonywanego pod nadzorem DBMS jako sekwencja instrukcji języka wewnętrznego DBMS zapisanych w pliku komend (command file, script), pełniącego rolę systemu wyszukiwania, w którym inne operacje na danych (dodawanie, kasowanie, etc.) nie są dozwolone. Systemem wyszukiwania z dostępem w języku naturalnym będziemy nazywać taki system wyszukiwania informacji, który umożliwia użytkownikowi zadawanie pytań w języku naturalnym. Chodzi o to by uwolnić użytkownika od konieczności uczenia się języków formalnych, gdyż najczęściej nie ma on na to ani czasu, ani ochoty (por. [Cic-83], str. 36).[3]

2.1. Problem unifikacji wiedzy

Początkowo system wyszukiwania informacji w języku naturalnym "składał się z dwóch zasadniczych modułów: modułu analizy syntaktyczno-semantycznej i modułu wyszukiwania. (...) Obecnie jednak przeważa pogląd, że poszczególne zagadnienia systemu pozostają w ścisłym związku, a zatem nie należy ustalać ich niezależnie" ([Cic-83], str. 39). Rozwiązanie takie podyktowane jest faktem konieczności stosowania do analizy języków naturalnych nie tylko informacji syntaktycznych i semantycznych, lecz również wiedzy o obiektach występujących w świecie rzeczywistym i ich własnościach. Chodzi tu oczywiście przede wszystkim o te obiekty, o które użytkownik systemu może zapytać, czyli o te które reprezentowane są w bazie danych.

Okazuje się więc, że wśród danych leksykalnych potrzebnych do analizy wypowiedzi (pytania) użytkownika potrzebne są nieraz informacje zawarte w bazie danych, do której ta wypowiedź jest skierowana. Dlatego też zaproponowano, by traktować bazę jako swego rodzaju wiedzę o rzeczywistości, z dodatkiem informacji leksykalnych. Proces analizy wypowiedzi zaczął przeplatać się z procesem wyszukiwania informacji..

Takie podejście znacznie usprawniło działanie systemów wyszukiwania informacji z użyciem języka naturalnego. Niestety tego typu systemy odznaczają się jedną wadą: są ściśle zależne od bazy danych i żaden z ich mechanizmów nie nadaje się do wykorzystania w innych systemach. Sprawę pogarsza fakt, iż do reprezentacji wiedzy wykorzystuje się tyle różnych formalizmów, ile jest metod analizy języków naturalnych.

Tymczasem szybki rozwój zunifikowanych relacyjnych systemów obsługi baz danych (Relational DBMS - RDBMS) i prostych metod ich odpytywania, takich jak język zapytań poprzez przykład (Query by Example - QBE) odsunął systemy naturalnojęzykowe jeszcze dalej w cień. Zaistniała więc potrzeba konstrukcji przenośnych modułów analizy języka, które mogłyby współpracować z dowolnymi bazami danych.

2.2. Systemy przenośne

W celu zapewnienia przenośności konieczne było wyróżnienie w systemach informacji leksykalnych, informacji o dziedzinie dialogu i informacji o bazie danych. Dostosowywanie systemu do nowej bazy danych wymagało dostarczenia odpowiednich informacji o jej strukturze i ew. informacji dziedzinowych. W powstałym w SRI Interantional systemie TEAM przewidziano nawet interaktywny interfejs umożliwiający dokonanie tego typu adaptacji zarządcy bazy danych, który nie jest obeznany z metodami

[3] por. także: Elżbieta Dobryjanowicz, op. cit., str. 12

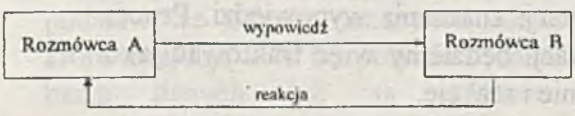
przetwarzania języka naturalnego (por. [Gro-82], [MAP-82]). W systemach takich często baza danych była tworzona od podstaw z wykorzystaniem systemu. Rozwiązanie to pozwalało systemowi gromadzić pełną wiedzę o strukturze i zawartości bazy danych. Systemy takie mogły też mieć zdolność uczenia się (por. [OPS-91], [PS-92]).

3. Program tłumaczący pytania QT

Tłumacz Pytań (Query Translator - QT) został pomyślany jako przenośny interfejs do baz danych, do których dostęp możliwy jest z użyciem strukturalnego języka kwerend (Structured Query Language - SQL). Celem autora było stworzenie programu ułatwiającego użytkownikowi dostęp do informacji zawartych w już istniejących bazach danych. Dlatego też użyto języka SQL, co umożliwia skorzystanie z mechanizmów wyszukiwania informacji zawartych w popularnych systemach obsługi baz danych. Rola programu QT ogranicza się więc jedynie do tłumaczenia sformułowanych w języku polskim pytań użytkownika na kwerendy SQL. Dlatego też program QT należy zaliczyć do programów automatycznego tłumaczenia (Machine Translation - MT). Takie podejście wymaga przyjęcia odmiennego niż zazwyczaj modelu komunikacji użytkownika z systemem.

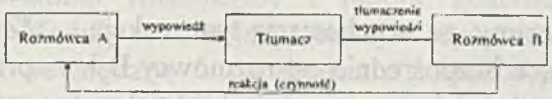
3.1. Schemat komunikacyjny

Dla lepszego zobrazowania różnic w modelach komunikacyjnych przyjmijmy za punkt wyjścia schemat komunikacji międzyludzkiej. W najprostszym przypadku dialogu dwóch osób schemat ten wygląda następująco:



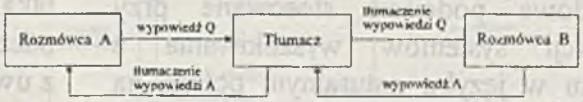
(rys. 1.)

Reakcją rozmówcy B może być również wypowiedź (np. odpowiedź na pytanie rozmówcy A). Powyższy schemat obrazuje jedynie dialog pomiędzy rozmówcami używającymi tego samego języka. W przeciwnym przypadku w schemacie musi pojawić się tłumacz (będący de facto trzecim rozmówcą):



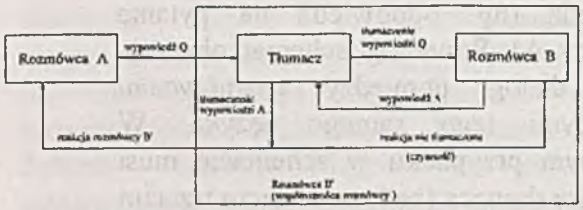
(rys. 2.)

Oczywiście, jeśli reakcją rozmówcy B jest jakaś wypowiedź, to skierowana ona zostaje również do tłumacza, który tłumaczy ją na język zrozumiały dla rozmówcy A. Ponadto tłumacz może dopytywać się o znaczenie poszczególnych słów w wypowiedziach rozmówców. Staje się on wtedy trzecim, pełnoprawnym rozmówcą i powyższy schemat rozpada się na dwa schematy dialogu z rys. 1.:



(rys. 3.)

Z punktu widzenia rozmówcy A najwygodniej jednak jest, by tłumacz był "przezroczysty", tzn. by jego obecność była niezauważalna tak, by dialog z rozmówcą B był w jak największym stopniu podobny do sytuacji w której obaj rozmówcy porozumiewają się w tym samym języku. Rozmówca A skłonny jest więc uznać za swego współrozmówcę system złożony faktycznie z tłumacza i rozmówcy B:



(rys. 4.)

Zauważmy, że w schemacie tym reakcje pochodzące bezpośrednio od rozmówcy B i reakcje tłumacza są nierozróżnialne. Zauważmy również, że tłumacz może w tym przypadku tylko przetłumaczyć wypowiedź rozmówcy A i przekazać ją (jej tłumaczenie) rozmówcy B, lub nawiązać dialog z rozmówcą A, w celu pełniejszego zrozumienia jego wypowiedzi. Inne reakcje tłumacza są niedozwolone.

Przedstawione powyżej schematy zachowują swą ważność, gdy drugim rozmówcą (rozmówcą B, lub B') jest system komputerowy (komputer wraz z oprogramowaniem). Rozmówca A zwany jest wtedy użytkownikiem (systemu). Standardowe podejście stosowane przy konstrukcji systemów wyszukiwania z dostępem w języku naturalnym polega na stosowaniu schematu komunikacji przedstawionego na rys. 1. Podejście prezentowane w niniejszej pracy odpowiada zaś schematowi z rys. 4. Rozmówcą B jest tu baza danych, a moduł QT pełni rolę tłumacza. Oczywiście z punktu widzenia użytkownika jego współrozmówcą jest system komputerowy traktowany jako całość (rozmówca B'). Przy konstrukcji programu przyjęto, że dane będące odpowiedziami bazy na zadane przez użytkownika pytanie, są zrozumiałe same przez się i nie wymagają komentarza. W praktyce można więc było zrezygnować z tłumaczenia odpowiedzi. Stąd też program QT zajmuje się jedynie tłumaczeniem pytań. Rzeczywiste tłumaczenie możliwe jest jednak tylko przy jednoczesnym zrozumieniu tłumaczonej wypowiedzi. Problemowi zrozumienia poświęcimy tu trochę więcej uwagi.

3.2. Rozumienie wypowiedzi

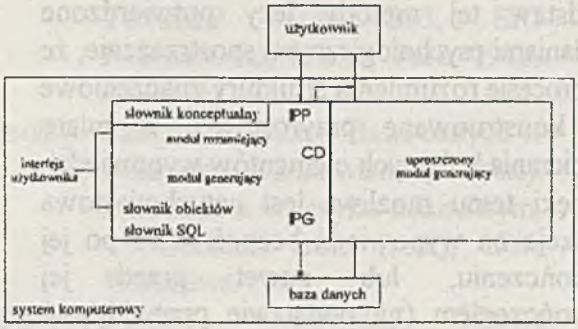
Tłumacz Pytań QT został określony również jako mechanizm rozumienia tekstu. Należy więc zastanowić się, co oznacza rozumienie jakiejkolwiek wypowiedzi. "Znaczenie pojęcia rozumienie jest przedmiotem badań (i sporów) filozofii, psychologii, lingwistyki. W dziedzinie sztucznej inteligencji przyjmuje się, że rozumienie przez system wypowiedzi użytkownika jest określone przez reakcję systemu na tę wypowiedź." - ([BCR-82], str. 70.). Przyjmijmy więc następującą, prostą definicję: rozumienie wypowiedzi jest to proces wykrywania intencji nadawcy (wypowiedzi), w wyniku którego odbiorca (człowiek lub mechanizm) może racjonalnie zareagować na tę wypowiedź (por. [Lub-90], str. 10.). Można tu mieć wątpliwości, jaką reakcję możemy uznać za racjonalną. Zakładamy więc, że reakcja racjonalna to taka, która jest realizacją przez odbiorcę odczytanej z wypowiedzi intencji jej nadawcy, z uwzględnieniem celów własnych odbiorcy. Oczywiście tego typu reakcja podlega dodatkowo pewnym obiektywnym ograniczeniom zewnętrznym, które mogą np. uniemożliwić wykonanie pewnych czynności. Jak wynika z zaproponowanej definicji, reakcja nie jest elementem procesu rozumienia, lecz ogniwem procesu komunikacji. Ogniwem, dzięki któremu możemy wnioskować o poprawności procesu rozumienia. Zadaniem procesu rozumienia będzie zaś wytworzenie wewnętrznej reprezentacji znaczenia wypowiedzi. Proces komunikacji będziemy więc traktować jako rozumienie i reakcję.

3.3. Reprezentacja znaczenia wypowiedzi w module QT

Do reprezentacji wewnętrznej znaczenia wypowiedzi użytkownika zastosowano struktury pojęciowe zaczerpnięte z teorii zależności pojęciowych (Conceptual Dependency - CD) opracowanej przez Rogera C. Schanka na uniwersytecie w Yale (Stany Zjednoczone), w latach 1969-1981 ([Sch-75], [Sch-81]). Struktury te, zwane strukturami pojęciowymi (konceptualnymi) służą do przedstawienia związków jakie zachodzą pomiędzy pewnymi pojęciami ogólnymi, takimi jak czynność, sprawca czynności, obiekt na którym czynność jest wykonywana, odbiorca (lub punkt docelowy) czynności, źródło pochodzenia obiektu (lub punkt początkowy czynności), narzędzie przy pomocy którego wykonywana jest czynność, itp. Czynności mogą być także reprezentowane w strukturach konceptualnych jako czynności nieokreślone powodujące pewne zmiany w relacjach pomiędzy sprawcą czynności i jej obiektem, etc. Dlatego też w teorii CD mówi się o aktach konceptualnych, będących aktywnościami odpowiadającymi wykonywaniu pewnych czynności. W teorii tej zakłada się istnienie jedenastu aktów podstawowych, z których daje się zbudować struktury przedstawiające każdą możliwą relację i aktywność rzeczywistą (w tym dowolną działalność człowieka). Do analizy wypowiedzi nie jest jednakże konieczne rozbijanie jej na struktury zawierające akty elementarne jeśli struktury będące w rzeczywistości bardziej złożone są wystarczające do jej jednoznacznej reprezentacji, a więc i zrozumienia (por. [Sch-77], str. 16 i [Lub-90], str. 37). Korzystając z tej możliwości w programie QT za podstawowe zostały uznane te zależności, które znajdują swe bezpośrednie odbicie w bazie danych lub są możliwe do wywnioskowania na podstawie danych z bazy i pewnej wiedzy o rzeczywistości (np. wiek można określić jako różnicę pomiędzy rokiem bieżącym i rokiem urodzenia). W niniejszym artykule zadowolimy się tym stwierdzeniem, odsyłając zainteresowanych do prac R. C. Schanka i jego współpracowników.

3.4. Budowa Tłumacza Pytań

Podstawowymi składowymi programu QT są dwa współpracujące ze sobą moduły: moduł rozumiejący i moduł generujący. Moduł rozumiejący buduje reprezentację wewnętrzną znaczenia wypowiedzi użytkownika. Moduł generujący zajmuje się zaś generacją odpowiedniej wypowiedzi w języku SQL. Oprócz tego istnieje szczątkowy moduł generowania odpowiedzi mający za zadanie jedynie zastąpienie generowanych przez system zarządzania bazami danych komunikatów o błędach, analogicznymi komunikatami w języku polskim. Schemat systemu przedstawiono na rys. 5.



(rys. 5.)

3.4.1. Moduł rozumiejący

Zadaniem modułu rozumiejącego jest interpretacja możliwie dowolnego w kontekście ustalonej bazy danych pytania użytkownika, przedstawienie jego znaczenia w postaci struktur konceptualnych za pomocą zewnętrznego słownika konceptualnego i przekazanie sterowania do modułu generującego. W przypadku, gdy moduł nie

jest w stanie wygenerować odpowiedniej struktury (z powodu sprzeczności wypowiedzi, wystąpieniu w niej pojęć nieznanych lub braku niezbędnych pojęć) nawiązuje on dialog z użytkownikiem, prosząc o uzupełnienie lub zmianę ostatniej wypowiedzi. Niektóre brakujące elementy struktury konceptualnej są wypełniane przez moduł rozumiejący wartościami domyślnymi (defaults). Zakres rozumienia programu ograniczony jest ściśle do zawartości bazy danych. Ograniczenie to narzucane jest przez słownik konceptualny, pozwalający rozpoznać tylko te słowa i frazy języka polskiego, które mają zdeterminowane znaczenie w kontekście wykorzystywanej bazy danych. Dzięki takiemu rozwiązaniu można uniknąć sytuacji, w której baza nie jest w stanie odpowiedzieć na poprawnie zrozumiane pytanie. W programie zastosowano metodę analizy (parsingu) języka opracowaną przez zespół twórcy teorii Zależności Pojęciowych Rogera C.Schanka, zwaną Zintegrowaną Analizą Przyrostową (Integrated Partial Parsing - IPP, [Sch-80]). U podstaw tej metody leży potwierdzone badaniami psychologicznymi spostrzeżenie, że w procesie rozumienia struktury znaczeniowe są konstruowane przyrostowo, w miarę odbierania kolejnych elementów wypowiedzi. Dzięki temu możliwa jest natychmiastowa reakcja na wypowiedź bezpośrednio po jej zakończeniu, lub nawet przed jej zakończeniem (na podstawie przewidywań dotyczących ostatnich elementów wypowiedzi). IPP jest metodą zorientowaną semantycznie, w której składnia odgrywa jedynie rolę pomocniczą, przy rozpoznawaniu znaczeń elementów wypowiedzi. Umożliwia to pominięcie stosowanego tradycyjnie przy przetwarzaniu języka etapu konstrukcji schematu syntaktycznego wypowiedzi. Własność ta sprawia, że Zintegrowana Analiza Przyrostowa jest praktycznie niezależna od języka i doskonale nadaje się do analizy języków o wolnym szyku zdania, których przykładem jest język polski. Program może również zostać przystosowany do analizy pytań w innych językach. Wymaga to jedynie wymiany słownika konceptualnego. Połączenie takich słowników umożliwia programowi przetwarzanie wypowiedzi w kilku językach równocześnie.

3.4.2. Moduł generujący

W znanej literaturze brak jest wzmianek o metodach generacji wypowiedzi w językach formalnych wprost z reprezentacji konceptualnej. Zaistniała więc konieczność opracowania odpowiednich mechanizmów. Moduł generujący musi umieć wygenerować poprawną składniowo kwerendę SQL (instrukcja SELECT), na podstawie struktury konceptualnej, będącej reprezentacją znaczenia (konceptualizacją) pytania postawionego przez użytkownika. Podstawowym problemem okazał się fakt, że struktury konceptualne stanowiące dane wejściowe dla generatora nie są uporządkowane liniowo i ich analiza musi uwzględniać dynamiczną zmianę kolejności przeglądania elementów, w zależności od wartości elementów już zanalizowanych. Z tego względu do generacji nie dało się zastosować standardowych metod parsingu zakładających liniowe uporządkowanie danych wejściowych. Dlatego też generator wyposażono w interpreter metajęzyka służącego do sterowania kolejnością przeglądania struktur konceptualnych. Instrukcje metajęzyka zapisane są w zewnętrznych słownikach wraz z informacjami dotyczącymi składni języka SQL. Informacje o składni przedstawiono również za pomocą instrukcji metajęzyka. Moduł wykorzystuje do generacji dwa słowniki: słownik operacji możliwych do wykonania z użyciem SQL (słownik fraz komendy SELECT), oraz słownik obiektów i relacji świata rzeczywistego, o których informacje zawarte są bezpośrednio w bazie, lub które daje się otrzymać z wykorzystaniem

informacji z bazy i zmiennych systemowych. Dzięki rozdzieleniu informacji specyficznych dla SQL i informacji zależnych od bazy, adaptacja programu do współpracy z inną bazą danych wymaga jedynie wymiany słownika obiektów. Umieszczenie fraz SQL w zewnętrznym słowniku umożliwia zaś wykorzystanie programu do współpracy z systemem zarządzania bazami danych wykorzystującym inny język kwerend (np. dBase). W takim przypadku adaptacja programu wymaga wymiany obu słowników.

4. Podsumowanie

Z uwagi na odstępstwa od standardu stosowane w niektórych wersjach SQL, jak i z powodów technicznych program QT musi być związany z konkretnym systemem obsługi baz danych. Przedstawiona metoda tłumaczenia wydaje się jednak dostatecznie uniwersalna, by można było ją stosować z dowolną bazą danych, obsługiwaną przez dowolny system zarządzania bazami danych, zawierającą informacje z dowolnej dziedziny. Opracowana metoda generacji umożliwia nawet generowanie zdań z wąskiego podzbioru języka polskiego. Zgodnie z założeniami wymaga to tylko wymiany słowników, z których korzysta moduł generujący. W takiej konfiguracji program dokonuje (deterministycznej) parafrazy zdań w języku polskim. Łatwo zauważyć, że opracowanie odpowiednich słowników dla języków obcych pozwoliłoby na dokonywanie z użyciem programu QT automatycznych tłumaczeń pomiędzy dwoma językami naturalnymi. Jak już bowiem wspomniano nic nie stoi na przeszkodzie, by językiem wejściowym nie był język polski. Jednakże podstawowym celem prac, jak to już wspomniano, było stworzenie programu ułatwiającego użytkownikowi dostęp do informacji zawartych w bazie danych. Cel ten został osiągnięty. Dalsze ułatwienie polegać może na zastosowaniu modułu rozpoznawania mowy. Warto tu zauważyć, że w tym przypadku danymi wejściowymi dla Tłumacza Pytań mogłyby być wypowiedzi zapisane w formie fonetycznej. Adekwatność przedstawionej metody do automatycznego tłumaczenia tekstów i budowa analizatora mowy są jednak odrębnymi tematami badawczymi.

5. Literatura

[BCR-82] Leonard Bolc, Małgorzata Cichy, Ludmiła Różańska, Przetwarzanie języka naturalnego, WNT, W-wa 1982

[Gro-82] Barbara Grosz, TEAM: A Transportable Natural-Language Interface System, Technical Note 263R, SRI International, Menlo Park 1982

[MAP-83] Paul Martin, Douglas Appelt, Fernando Pereira, Transportability and Generality in a Natural Language Interface System, Technical Note 293, SRI International, Menlo Park 1983, str. 35-84

[Cic-83] Małgorzata Cichy, Baza danych w systemach wyszukiwania informacji z dostępem w języku naturalnym, w: Leonard Bolc (red.), UW, Warszawa 1983

[OPS-91] Michał Ostrowski, Hanna Popowska, Jerzy Solak, Sprawozdanie z realizacji zadania: Opracowanie wersji 1.0 systemu DABINAL, uczącego się komunikacji z bazami wiedzy w języku naturalnym, wykonanego na podstawie umowy nr 734/K5.21 pomiędzy KBN a IINTE, Instytut Informacji Naukowej, Technicznej i Ekonomicznej, Warszawa 1991

[PS-92] Hanna Popowska, Jerzy Solak, DABINAL - uczący się system

komunikacji w języku naturalnym zbazami danych, w: Leonard Bolc, Jacek Zaremba, Wprowadzenie do uczenia się maszyn, Akademicka Oficyna Wydawnicza RM, Warszawa 1992, str. 93-110

[Sch-75] Roger C.Schank, Conceptual Information Processing, seria: Fundamental Studies in Computer Science, tom 3., North-Holland, Amsterdam, etc. 1975 (wyd. 2., 1984)

[Sch-77] Roger C. Schank, Robert P. Abelson, Scripts, Plans, Goals and Understanding; An Inquiry into Human Knowledge Structures, Lawrence Erlbaum Associates, Hillsdale 1977

[Sch-80] Roger C.Schank, Michael Lebowitz, Lawrence Birnbaum, An Integrated Understander, American Journal of Computational Linguistics, Vol. 6, No. 1, January - March 1980

[Sch-81] Roger C.Schank, Christopher K.Riesbeck (red.), Inside Computer Understanding: Five Programs Plus Miniatures, Lawrence Erlbaum Associates, Hillsdale 1981

Tadeusz Wilusz
Akademia Ekonomiczna w Krakowie

Komputeryzacja, czy automatyzacja systemu informacyjnego przedsiębiorstwa?

1. Wprowadzenie

Wprowadzanie metod i środków informatyki do praktyki gospodarczej najczęściej określa się skrótowo jako "komputeryzację przedsiębiorstwa". Na przestrzeni ostatnich kilku lat proces ten w Polsce uległ istotnej intensyfikacji. Można więc postawić tezę, że stanowi on jeden z ważniejszych elementów procesu likwidacji "luki technologicznej" pomiędzy krajami rozwiniętymi o utrwalonej i ustabilizowanej gospodarce rynkowej, a państwami wychodzącymi ze scentralizowanej gospodarki socjalistycznej. Najważniejszym rysem charakterystyki tych przemian jest ich rewolucyjny charakter. Pod tym sformułowaniem kryje się konstatacja faktu, że wdrażane rozwiązania pomijają szereg pośrednich faz rozwojowych, przez które przechodziły kraje, gdzie analogiczne procesy przebiegały i przebiegają w sposób ciągły, który w związku z tym można określić przymiotnikiem "ewolucyjny".

W każdej dziedzinie możliwość wymiany technologii w sposób rewolucyjny, to olbrzymia szansa zgarnięcia tzw. "premii za zacofanie", czyli sięgnięcia po sprawdzone i ustabilizowane rozwiązania bez ponoszenia ryzyka i kosztów prac badawczo-rozwojowych. Ma to kapitalne znaczenie przy silnym ograniczeniu dostępnych środków inwestycyjnych, które jest typowym i powszechnym ograniczeniem, z racji skali przemian, z jakim mamy do czynienia w procesach związanych ze zmianami ustrojowymi gospodarki.

Deklarując chęć szybkiej integracji z krajami o gospodarce rynkowej jesteśmy w pewnym sensie zmuszeni do działań w przyspieszonym tempie aby nadrobić wieloletnie zapóźnienia, ale powstaje pytanie, czy tym samym, w świetle powyższych uwag, jesteśmy "skazani" na sukces? Odpowiedź brzmi niestety "nie". Jak to zwykle bywa, sytuacje oferujące duże szanse wnoszą równie duże, a czasami znacznie większe od szans, zagrożenia. W rozważanym kontekście zagrożenie ma identyczne źródła jak szanse. Sięgamy po rozwiązania sprawdzone, ale powstałe w innych realiach we wszystkich możliwych aspektach: językowo-terminologicznych, prawnych, organizacyjnych, czyli jednym słowem kulturowych. Powstaje zatem problem ich wdrożenia i akurat w tym, tak ważnym dla końcowego efektu, procesie doświadczenia autorów wdrażanych rozwiązań i wypracowane przez nich metody muszą zostać poddane procesowi adaptacji w celu dostosowania do istniejących realiów. Reasumując, zagrożenie, o którym mowa można ująć jako niebezpieczeństwo zwielokrotnionych kosztów procesu wdrożenia z racji braku, czy też opóźnień we wprowadzaniu, właściwych rozwiązań formalno-prawnych i organizacyjnych w otoczeniu procesu wdrażania. Jeśli kosztów tych można by uniknąć, podejmując wyprzedzająco określone działania, a zostały one poniesione, to mamy do czynienia z klasycznym przykładem formalnie nie zawinionego marnotrawstwa.

Ten, z konieczności skrótowy i niepełny,

szkic sytuacyjny prowadzi nieuchronnie do postawienia następującego problemu: jak zminimalizować prawdopodobieństwo wystąpienia marnotrawstwa środków, czyli jakie działania winny być podjęte, aby wykorzystać istniejące szanse i uniknąć tych z potencjalnych zagrożeń, które są do uniknięcia. Takie sformułowanie problemu dobrze oddaje dyskusyjny charakter możliwych do przyjęcia rozwiązań i uzasadnia potrzebę wymiany istniejących w kraju, sprawdzonych w praktyce doświadczeń oraz potrzebę bieżących dyskusji nad stworzeniem warunków dla przyjmowania optymalnych, w kontekście istniejących realiów, strategii realizacyjnych procesu wdrażania.

Jeśli przyjmiemy, że sformułowanie "stwarzanie warunków" obejmuje również działania zmierzające do usunięcia źródeł potencjalnych zagrożeń, to niniejsze wystąpienie należy traktować właśnie jako głos w dyskusji na identyfikacją źródeł potencjalnego marnotrawstwa.

Głos ten dotyczy jednej przyczyny i dwóch źródeł zagrożeń. Wspólną przyczyną istnienia obydwu źródeł jest nader częsta tendencja do bagatelizowania ich znaczenia. A są nimi: uwarunkowania organizacyjne procesu wdrażania oraz brak wykształconej i jednoznacznie pojmowanej terminologii polskojęzycznej, który uniemożliwia właściwą komunikację pomiędzy decydentami, użytkownikami systemu oraz zespołami projektowo-wdrożeniowymi.

Warto w tym miejscu podkreślić, że zwłaszcza to drugie źródło zagrożeń jest wyjątkowo często świadomie pomijane, jako zupełnie nieistotne, a tymczasem, po głębszej analizie okazuje się być pierwotną przyczyną wielu kłopotów o różnorodnej naturze. Przy tym nie jest intencją autora wywoływanie dyskusji nad potrzebą wprowadzania nowej terminologii, a wręcz odwrotnie, wskazanie na pilną potrzebę klaryfikacji pojęć, które zadomowiły się w praktyce komunikacji w języku polskim, ale których zakres znaczeniowy jest nader często dość dowolnie, często niestety błędnie przyjmowany.

2. Komputeryzacja a automatyzacja

Słownik języka polskiego[1] pod hasłem *automatyzacja* stanowi, że jest to "*wprowadzanie do produkcji, transportu, pracy biurowej, itp. środków technicznych w celu samoczynnego przebiegu sterowania, regulowania i kontrolowania różnych procesów i operacji*". Według tego samego źródła termin *komputeryzacja* oznacza *"wprowadzanie nowoczesnych metod przetwarzania danych przy użyciu komputerów, np. przy projektowaniu, sterowaniu maszyn roboczych, w administracji dużych przedsiębiorstw"*.

Jest rzeczą oczywistą, że na obydwie definicje różnie zareagują przedstawiciele różnych profesji. Informatyk się żachnie i nie zostawi na żadnej z nich suchej nitki dowodząc bez trudu, że są niepełne, niekonsekwente, nieadekwatne, archaiczne (np. z uwagi na datę publikacji cytowanego źródła) i w ogóle do zmiany. Ale trudno oczekiwać takiej samej reakcji od np. głównego księgowego przedsiębiorstwa z wieloletnim stażem. Dla niego obie definicje jawią się jako do przyjęcia i jest skłonny je zaakceptować. Tych dwóch ludzi uczestnicząc w tym samym przedsięwzięciu musi się ze sobą komunikować, ale nader trudno będzie się im porozumieć. A nieporozumienia zwykle kosztują. I to stanowi zagrożenie dla prawidłowości procesu wdrożeniowego, obojętnie, czy nazywać go komputeryzacją, czy automatyzacją. Przyjrzyjmy się tej przyczynie nieco bliżej.

Uprzedzając zarzut, o tendencyjnym doborze definicji zgódźmy się traktować je

[1] "Słownik języka polskiego" pod red. prof dr Mieczysława Szymczaka, tom pierwszy A-K, PWN, Warszawa 1978.

jako przykłady potocznego rozumienia znaczenia obydwu terminów. Ten sposób rozumienia oddają następujące stwierdzenia:

1. Komputeryzacja i automatyzacja są terminami semantycznie rozłącznymi,
2. Automatyzacja raczej odnosi się do procesów produkcyjnych i kojarzy się z potrzebą bardzo różnorodnego oprzyrządowania,
3. Komputeryzacja kojarzy się z pozaprodukcyjną działalnością człowieka (prace administracyjno-biurowe, projektowanie, obliczenia itp.),
4. Komputeryzacja stwarza możliwość dużych oszczędności nakładów pracy ludzkiej a tym samym zmniejszenie kosztów poprzez redukcję zatrudnienia.
5. Automatyzacja jest problemem trudnym i skomplikowanym, który winien być powierzony wysokospecjalizowanym fachowcom,
6. Główny problem komputeryzacji to nakłady na sprzęt i opanowanie pracy z komputerem.

Lista ta jest oczywiście zarówno niepełna jak i, w sensie sformułowań, prawdopodobnie dyskusyjna, ale pozwala zaryzykować tezę, że takie rozumienie komputeryzacji i automatyzacji jest efektem głównie świadomości przeciętnego człowieka, kształtowanej prze różnorodne źródła informacji, doświadczenie życiowe i wyobraźnię. Innymi słowy, gdyby przytoczone definicje zastąpić w cytowanym źródle takimi, które przez specjalistów od automatyzacji i informatyki zostałyby uznane za prawidłowe, to w rozpatrywanym aspekcie ... niewiele by to zmieniło. Ten jednostkowy przykład można uogólnić jako problem terminologii procesu komunikacji w wielobranżowych zespołach opracowujących i wdrażających systemy informatyczne. Niezwykle ważnym elementem wstępnego przygotowania takich przedsięwzięć jest troska o jednoznaczność języka komunikacji w takich zespołach, a tu problemy natury terminologicznej są wyjątkowo trudne, jako, że wiele terminów fachowych ma również znaczenie potoczne, często na tyle różne, że wypaczające cały proces komunikacji.

Aby domknąć przykład z automatyzacja i komputeryzacją, zaryzykujmy stwierdzenie, że elementem wstępnego przygotowania (i chyba ze strony informatyków) winno być wyjaśnienie, że komputeryzację należy rozumieć jako automatyzację procesu przetwarzania danych. Już ten krótki suplement do ich potocznego rozumienia , po pierwsze porządkuje relację pomiędzy nimi, a po drugie eliminuje wiele typowych, odruchowych oczekiwań przeciętnego użytkownika w stosunku do końcowych efektów.

Oczekiwania te, których źródłem pierwotnym jest między innymi właśnie opisana sytuacja, a których skutki są różnorodne ale generalnie negatywne, to najczęściej:

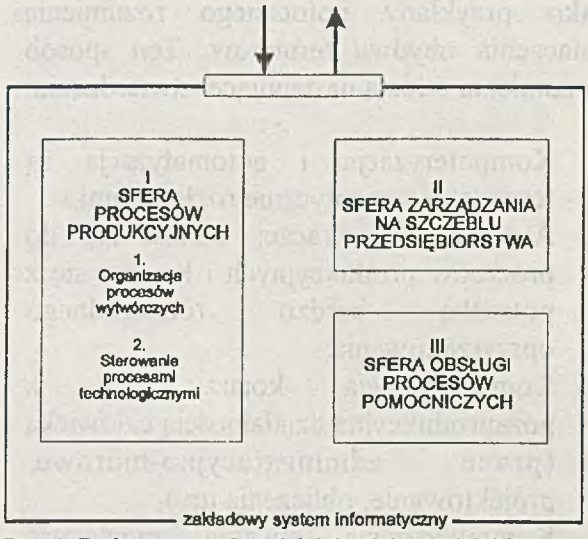
1. Traktowanie procesu komputeryzacji jako łatwego przedsięwzięcia, polegającego głównie na zakupach sprzętowo-programowych;
2. Przekonanie, że głównym wymiernym efektem komputeryzacji będzie widoczny spadek zatrudnienia;
3. Różnorodne pomysły dotyczące tzw. "oszczędności" finansowo-czasowych (np. nie projektujemy docelowego rozwiązania tylko komputeryzujemy kadry, finanse, sprzedaż itd.);
4. Traktowanie automatyzacji procesów produkcyjnych jako dziedziny zupełnie odrębnej od sfery zarządzania, których jedynym punktem wspólnym jest ewentualnie sprawozdawczość;
5. Przekonanie, że skomputeryzowany system zarządzania przedsiębiorstwem, będzie świetnie funkcjonował w dotychczasowej formule organizacyjnej.

Podsumowując ten punkt można stwierdzić, że termin "*komputeryzacja*" jest znacznie powszechniej używany w odniesieniu do procesu wdrażania metod i środków informatyki do praktyki gospodarczej aniżeli termin "*automatyzacja procesów przetwarzania danych*", i nie ma powodu, aby upierać się przy tezie o konieczności jego zmiany, pod warunkiem, że świadomi jesteśmy potencjalnych zagrożeń wynikających z niewłaściwego, potocznego rozumienia, co za tym określeniem faktycznie się kryje, i we właściwym momencie stosowne podejmiemy działania pozwalające te zagrożenia wyeliminować.

3. Zintegrowane systemy wytwarzania (CIM) jako docelowy model funkcjonowania przedsiębiorstwa produkcyjnego

Jak już wskazywano w poprzednim punkcie, jednym z zagrożeń procesu komputeryzacji przedsiębiorstwa wynikającym, przynajmniej częściowo, również z powodów terminologicznych, jest częsta praktyka niewłaściwej organizacji całego procesu. W szczególności nader często za terminem "komputeryzacja przedsiębiorstwa" kryje się wyłącznie automatyzacja bardzo wycinkowej sfery działalności, bez, chociażby szkicu, wizji rozwiązania docelowego. Tymczasem truizmem jest stwierdzenie, że trudno oczekiwać, aby suma optymalnych rozwiązań problemów, z których każdy jest rozpatrywany bezkontekstowo, dała w efekcie optymalną całość. Popełnianie tego błędu, jest tym bardziej bezzasadne, że w tej mierze istnieją zarówno doświadczenie jak i sprawdzona metodologia działań.

Treść określenia Computer Integrated Manufacturing (CIM) można zdefiniować z jednej strony jako wizję, a z drugiej jako rozwiązanie docelowe, do którego zmierzają przedsiębiorstwa wytwórcze krajów rozwiniętych. Wizja ta zakłada integrację wszystkich procesów informacyjnych przedsiębiorstwa (por. rys. 1.) proponując jednolity sposób podejścia do tak diametralnie różnych kategorii informacji jak np. przyjmowanie zamówień (sfera zarządzania) i sterowanie robotem przemysłowym (sfera wytwórcza).



Rys. 1. Podstawowe sfery działalności przedsiębiorstwa produkcyjnego.

Zatem, jeśli uzgodnimy dalsze uściślenie terminu "*komputeryzacja przedsiębiorstwa*" jako przedsięwzięcia mającego na celu automatyzację wszystkich procesów informacyjnych, to bezkontekstowa komputeryzacja poszczególnych sfer, nie powinna mieć nigdy miejsca.

4. Organizacyjne uwarunkowania procesu komputeryzacji

Zacznijmy od kilku stwierdzeń, które jeśli nawet są powszechnie aprobowane to jednak niezbyt często faktycznie uwzględniane w praktyce. I tak:

1. Komputeryzacja przedsiębiorstwa jest przedsięwzięciem złożonym ingerującym we wszystkie sfery działalności przedsiębiorstwa.
2. Optymalne rozwiązanie musi być ściśle dopasowane do specyfiki danego

przedsiębiorstwa, a więc wzorowanie się na rozwiązaniach już istniejących zawsze będzie mieć ograniczony zakres.

3. Komputeryzacja przedsiębiorstwa nie jest aktem jednorazowym. Jest natomiast procesem rozłożonym w czasie i jest niezwykle ważnym dla szeroko rozumianego efektu końcowego, aby spełnić wszystkie możliwe warunki pozwalające zakładać, że będzie on przebiegał w sposób bliski optymalnemu przy istniejących i znanych ograniczeniach.
4. Pomimo, iż stwierdzenie zawarte w poprzednim punkcie zakłada dochodzenie do rozwiązania docelowego etapami, to należy wyraźnie podkreślić, iż najważniejszym etapem, niekiedy decydującym o powodzeniu i prawidłowej realizacji całego przedsięwzięcia, jest etap pierwszy.
5. Istnieje pewien poziom nakładów (zarówno w sferze rzeczowej jak i organizacyjnej),- specyficzny dla każdego tego typu przedsięwzięcia - które muszą być poniesione, jeżeli chcemy utrzymać prawdopodobieństwo osiągnięcia zakładanych celów na rozsądnym poziomie.

Na główny grzech pierworodny kłopotów z komputeryzacją już wskazywaliśmy, ale ponieważ jest to punkt wyjściowy do uwag w tym punkcie, to powtórzmy, że jest nim

brak sprecyzowanej wizji rozwiązania docelowego opartego na analizie potrzeb przedsiębiorstwa traktowanego jako całość.

W kontekście typowego dążenia do skrócenia czasu trwania komputeryzacji prowadzi to w praktyce do sytuacji, w której zamiast komputeryzacji przedsiębiorstwa mamy do czynienia z wieloma procesami komputeryzacji poszczególnych funkcji przedsiębiorstwa, które nie będąc od samego początku podporządkowane wspólnej koncepcji mogą w efekcie nie tylko nie przynieść spodziewanych efektów, ale stworzyć nowe problemy, których rozwiązanie wymagać będzie praktycznie powrotu do punktu wyjściowego. Zatem przyjęcie wizji rozwiązania docelowego warunkuje prawidłowość realizacji całego procesu projektowania i wdrażania zaprojektowanych rozwiązań .

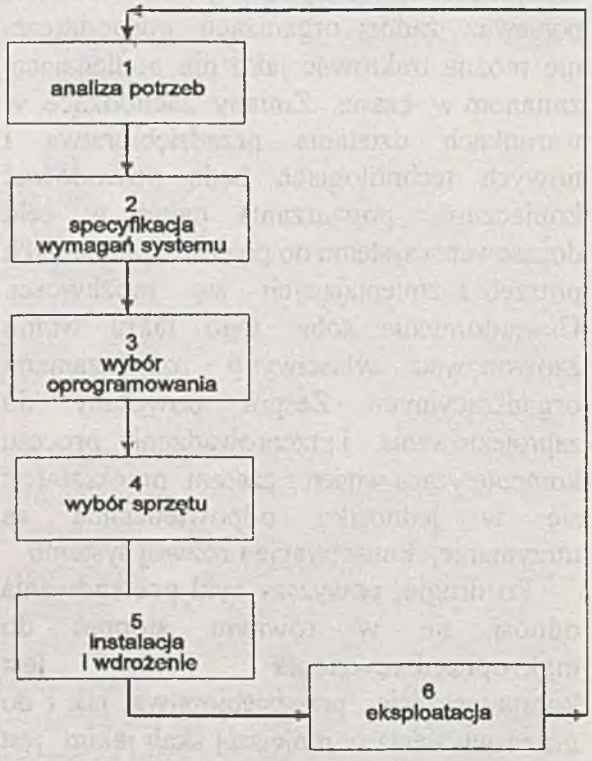
Wiedza o prawidłowościach cyklu projektowania i wdrażania systemów informatycznych, który w najbardziej zagregowanej postaci pokazano na rys 2. daje asumpt do kolejnych wniosków.

Po pierwsze, przedstawienie powyższego toku postępowania w postaci nieskończonego cyklu jest celowe, ponieważ żadnej organizacji gospodarczej nie można traktować jako nie podlegającej zmianom w czasie. Zmiany zachodzące w warunkach działania przedsiębiorstwa i nowych technologiach będą powodować konieczność powtarzania cyklu w celu dopasowania systemu do poziomu aktualnych potrzeb i zmieniających się możliwości. Uświadomienie sobie tego faktu winno zaowocować właściwymi rozwiązaniami organizacyjnymi. Zespół powołany do zaprojektowania i przeprowadzenia procesu komputeryzacji winien z czasem przekształcić się w jednostkę odpowiedzialną za utrzymanie, konserwację i rozwój systemu.

Po drugie, powyższy cykl postępowania odnosi się w równym stopniu do makroprzedsięwzięcia jakim jest komputeryzacja przedsiębiorstwa, jak i do przedsięwzięcia w mniejszej skali jakim jest wdrożenie określonego systemu aplikacyjnego. Innymi słowy, sprawą niezwykłej wagi jest unifikacja metodologii prowadzenia prac na wszystkich szczeblach struktury organizacyjnej z czytelnie określoną hierarchią i definicją zarówno wzajemnych współzależności, jak i posiadanej w ramach ogólnego

przedsięwzięcia autonomii.

Kolejnym elementem, na który należy zwrócić uwagę jest relacja sprzęt - oprogramowanie. Z rysunku jasno wynika, i jest to kolejny raz pozorna oczywistość, że kryterium nadrzędnym musi być oprogramowanie. Jeśli przyjąć, że przedsiębiorstwo zamierza potrzebne oprogramowanie opracować w całości we własnym zakresie, to w tym i tylko w tym przypadku mamy do czynienia z sytuacją, która w praktyce dominuje. Chodzi o to, że decyzje o wyborze sprzętu z reguły są nie uwarunkowane analizą dostępnego oprogramowania na tenże sprzęt pod kątem specyfiki potrzeb przedsiębiorstwa.



Rys. 2. Cykl opracowania i wdrożenia systemu informatycznego.

Dodatkowo teza o korzyści z unifikacji sprzętu prowadzi do typowych, i niestety w ogólnym rozrachunku kosztownych, wynaturzeń w postaci kurczowego trzymania się określonej kategorii sprzętu we wszystkich dziedzinach aplikacji. Tymczasem prawda jest taka, że trudności techniczne doprowadzenia do współpracy sprzętu o zróżnicowanej architekturze logicznej mają wiele zadowalających rozwiązań i nie stanowią już większego problemu, podczas gdy wytworzenie odpowiedniej klasy oprogramowania użytkowego i adekwatna do potrzeb konkretnej aplikacji klasa sprzętu mogą stanowić istotny problem, warunkujący osiągnięcie zakładanych efektów całego przedsięwzięcia.

Na marginesie tego problemu warto zauważyć, że wysoki poziom walki konkurencyjnej o światowe rynki firm zarówno sprzętowych, jak software'owych doprowadził do sytuacji, w której te same efekty można osiągnąć stosując niekiedy diametralnie różne rozwiązania w sensie wyboru rodzaju sprzętu i oprogramowania. Wybór jest wielokryterialny i bardzo często decyzje są podejmowane w oparciu o trzecio, a nawet czwartorzędne kryteria. Powstaje w tym miejscu obawa, czy dokonaliśmy najlepszego możliwego wyboru. Otóż doświadczenie innych znowu uczy, że o efektach w pełnym rozrachunku decyduje poziom świadomości użytkownika w zakresie swoich potrzeb, wizja docelowego modelu przedsiębiorstwa oraz konsekwencja w dochodzeniu do tegoż modelu docelowego. Okazuje się, że dwa podobne, w sensie profilu swojej działalności, przedsiębiorstwa mogą w oparciu o różny sprzęt i oprogramowanie stworzyć rozwiązania, które są funkcjonalnie bardzo bliskie sobie. Co więcej, potrzeba kooperacji pomiędzy różnymi przedsiębiorstwami musiała zaowocować rozwiązaniami umożliwiającymi współdziałanie systemów informacyjnych bazujących na różnych rozwiązaniach sprzętowo-programowych (heterogeniczne sieci komputerowe). Istniejące rozwiązania w tym zakresie są znane jako tzw. standardy międzynarodowe

spośród których największe znaczenie dla opracowania koncepcji komputeryzacji przedsiębiorstwa ma przyjęty przez Międzynarodową Organizację Normalizacyjną ISO (International Standard Organization) model otwartego systemu połączeń międzysieciowych OSI (Open Systems Interconnection) mający rangę komputerowego esperanto pozwalającego rozciąć gordyjski węzeł współpracy systemów o diametralnie rożnej architekturze logicznej.

5. Zakończenie

Jeśli przedstawiony sposób widzenia wybranych, w dość arbitralny zresztą sposób, aspektów związanych technologii procesów informacyjnych w polskich przedsiębiorstwach znajdzie tyluż zwolenników co przeciwników, to dalsza dyskusja staje się nieuchronna, a tym samym cel tego opracowania - osiągnięty. Ważną do spełnienia rolę w tej dyskusji mają do spełnienia przedsiębiorstwa, które mogą podzielić się swoimi doświadczeniami potwierdzającymi, bądź negującymi postawione i pominięte tezy. Powstaje tylko problem, kto winien być organizatorem takiego forum dyskusyjnego, jaką ono powinno przyjąć formę organizacyjną, bo teza o potrzebie dyskusji i wymianie doświadczeń jest zbyt oczywista, by ją uzasadniać.

Polskie Towarzystwo Informatyczne nasuwa się jako odruchowa odpowiedź na pytanie "kto?". Doceniając w pełni dotychczasowy wkład PTI w rozwój polskiej sceny informatycznej i niewątpliwe sukcesy należy jednakże zwrócić uwagę, jest ono jednak postrzegane jako organizacja branżowa, której prace w związku z tym będą miały duże szanse dotrzeć do niewłaściwego adresata, bądź, w najlepszym przypadku, do zbyt wąskiego grona zainteresowanych odbiorców. Zatem postulat poszerzenia platformy o inne stowarzyszenia branżowe, niewątpliwie w tym kontekście wart jest rozważenia.

Stanisław Kowalik
Politechnika Śląska

Wykorzystanie teorii zbiorów rozmytych do podejmowania decyzji

1. WSTĘP

Informacje dotyczące otaczającego nas świata często są nieprecyzyjne, niepełne lub niepewne. Powstała niedawno teoria zbiorów rozmytych stwarza możliwość opisu formalnego tej informacji nieprecyzyjnej [2], [4], [5], [6], [7], [8], [9], [10]. Te matematyczne metody teorii zbiorów rozmytych pozwalają pełniej i w sposób bardziej naturalny opisać zjawiska świata rzeczywistego. Sytuacje niepewne w podejmowaniu decyzji objawiają się tutaj poprzez niejednoznaczność i nieprecyzyjność opisu warunków, w jakich ma być podjęta decyzja. Innymi słowy opis otaczającego nas świata jest niedokładny, a my musimy podejmować pewne decyzje. Teoria zbiorów rozmytych jak na razie nie jest jednoznaczną strukturą matematyczną. "Jest to raczej rodzina teorii o różnym stopniu ogólności i różnych specyficznych możliwościach zastosowań" [2]. My będziemy wzorowali się na klasycznej teorii zbiorów rozmytych opracowanej przez L.A. Zadeha [10]. Po raz pierwszy w 1965 roku Zadeh w swojej pracy [10] określił pojęcie rozmytości (fuzziness) oraz sformułował podstawowe pojęcia dotyczące zbiorów rozmytych (fuzzy sets). Na tych pojęciach opierają się reguły tzw. logiki rozmytej. W 1970 roku wspólnie z Bellmanem opublikował prace [1] o podejmowaniu decyzji w warunkach rozmytych. W następnych latach teoria zbiorów rozmytych szybko się rozbudowała i zrobiła błyskawiczną karierę. W 1973 roku Zadeh w swojej pracy [11] podaje podstawowe pojęcia i reguły logiki rozmytej. Warto w tym miejscu wspomnieć, ze na wiele lat przed wprowadzeniem pojęcia zbioru rozmytego, polski matematyk Jan Łukasiewicz stworzył podstawy logiki wielowartościowej. W związku z rozwojem nowej teorii, prace Łukasiewicza budzą ponowne zainteresowanie. Teoria zbiorów rozmytych wzoruje się na klasycznej teorii zbiorów z uwzględnieniem tzw. funkcji przynależności elementu do zbioru. W związku z tym wprowadza się specjalny zapis zbiorów rozmytych oraz określa się działania na tych zbiorach. Wprowadza się też pojęcie liczb rozmytych oraz odpowiednie operatory arytmetyczne. Podejmowanie decyzji w rozmytych warunkach otoczenia polega na odpowiednim wnioskowaniu z przesłanek o charakterze rozmytym. Prezentowana teoria znalazła już szereg zastosowań w różnych dziedzinach jak: topologia, teoria miary struktur algebraicznych, sterowanie, mechanika, budowa modeli sieciowych, stabilność, organizacja, wnioskowanie podejmowanie decyzji, diagnozowanie medyczne.

2. ZBIORY ROZMYTE

W klasycznej teorii zbiorów każdy zbiór posiada jednoznacznie określone granice oddzielające elementy należące do niego od nienależących. Jeżeli mamy zbiór A i element x, to możemy stwierdzić, czy element x należy do zbioru A, czy też nie należy. Oznaczmy przez X przestrzeń wszystkich rozpatrywanych elementów x. W tej przestrzeni jest określona funkcja charakterystyczna zbioru A [2], [4], [6], [8]. Ta funkcja zdefiniowana jest następująco

$$\forall x \in X, \chi_A(x) = \begin{cases} 1 \text{ dla } x \in A, \\ 0 \text{ dla } x \notin A. \end{cases} \quad (1)$$

Funkcja charakterystyczna χ_A odwzorowuje przestrzeń X w zbiór {0,1}. W wielu przypadkach, gdy operuje się pojęciami charakteryzowanymi w sposób nieprecyzyjny występują trudności w określeniu przynależności elementu do danego zbioru. Funkcja charakterystyczna określona wzorem (1) jest wtedy niewygodna i mocno ograniczająca zastosowanie jej do sytuacji niedokładnie określonych. Zadeh w swej pracy [10] wprowadził pojęcie zbioru rozmytego. Zbiór rozmyty posiada taka własność, że jego funkcja określająca przynależność elementu do zbioru, odwzorowuje przestrzeń X w odcinek [0,1]. Jest to więc rozszerzenie przeciwdziedziny funkcji charakterystycznej określonej wzorem (1) na odcinek [0,1].

Definicja 1 [3].

Zbiorem rozmytym A określonym na przestrzeni X jest zbiór uporządkowanych par:

$$A = \{ (x, \mu_A(x)) \text{ dla } x \in A \}, \quad (2)$$

gdzie μ_A jest funkcją przynależności zbioru A.

Przykład 1.

Niech X oznacza zbiór cyfr {0,1,2,3,4,5,6,7,8,9},
A - oznacza cyfrę średnią, B - oznacza cyfrę dużą.
Funkcje przynależności zbiorów rozmytych A i B mogą być następujące:

$\mu_A(0)=0$, $\mu_A(1)=0$, $\mu_A(2)=0.4$, $\mu_A(3)=0.7$,
$\mu_A(4)=1$, $\mu_A(5)=1$, $\mu_A(6)=0.8$, $\mu_A(7)=0.5$,
$\mu_A(8)=0$, $\mu_A(9)=0$;
$\mu_B(0)=0$, $\mu_B(1)=0$, $\mu_B(2)=0$, $\mu_B(3)=0$,
$\mu_B(4)=0$, $\mu_B(5)=0.3$, $\mu_B(6)=0.6$, $\mu_B(7)=0.8$,
$\mu_B(8)=1$, $\mu_B(9)=1$.

Przykład 2.

Niech X = R jest przestrzenią liczb rzeczywistych, a A zbiorem liczb dużo większych od 100. Funkcję przynależności zbioru A możemy określić następująco

$$\mu_A(x) = \begin{cases} 0 & \text{dla } x \le 100, \\ \dfrac{1}{1+100(x-100)^{-1}} & \text{dla } x > 100. \end{cases}$$

Definicja 2.

Nośnikiem zbioru A nazywamy zbiór U elementów przestrzeni X, dla których $\mu_A(x)>0$. Oznaczamy go przez supp A (ang. support)

$$\text{supp } A = \{ x : \mu_A(x) > 0 \}. \quad (3)$$

Definicja 3.

Wysokością zbioru A jest kres górny funkcji $\mu_A(x)$, tzn.

$$\sup_{x \in X} \mu_A(x). \quad (4)$$

Definicja 4.

Zbiór rozmyty nazywamy znormalizowanym, gdy jego wysokość jest równa 1.

W celu uproszczenia zapisu zbioru rozmytego L.A. Zadech wprowadził specjalną notację do teorii zbiorów rozmytych [2], [10]. Zbiory rozmyte, których nośniki są nieprzeliczalne zapisujemy w postaci

$$A = \int_U \mu_A(x)/x. \quad (5)$$

Jeżeli nośnik jest przeliczalny, to zbiór rozmyty zapisujemy w postaci

$$A = \sum_i \mu_A(x_i)/x_i. \quad (6)$$

Jeśli elementy nośnika U nie są liczbami, to wzór (6) modyfikujemy do postaci

$$A = \sum_i \mu_A(x_i)x_i. \qquad (7)$$

Tak wiec zbiory A i B przedstawione w przykładzie 1 można zapisać w postaci

A = 0.4/2 + 0.7/3 + 1/4 + 1/5 + 0.8/6 + 0.5/7,
B = 0.3/5 + 0.6/6 + 0.8/7 + 1/8 + 1/9.

Należy zwrócić uwagę na to, że w takim zapisie biorą udział jedyne elementy należące do nośnika rozpatrywanego zbioru.

W teorii zbiorów rozmytych wprowadza się też pojęcia zawierania się i równości zbiorów.

Definicja 5.

Zbiór rozmyty A zawiera się w zbiorze rozmytym B ($A \subset B$) wtedy i tylko wtedy, gdy

$$\forall_{x \in X} \mu_A(x) \le \mu_B(x) \qquad (8)$$

Definicja 6.

Zbiór rozmyty A jest równy zbiorowi rozmytemu B ($A = B$) wtedy i tylko wtedy, gdy

$$\forall_{x \in X} \mu_A(x) = \mu_B(x) \qquad (9)$$

3. OPERATORY ROZMYTE

3.1. Suma

Definicja 7.

Sumą zbiorów rozmytych A i B nazywamy zbiór $A \cup B$ określony przez funkcję przynależności (10)

$$\forall_{x \in X} \mu_{A \cup B}(x)=\max(\mu_A(x), \mu_B(x))=(\mu_A(x) \vee \mu_B(x)$$

3.2. Iloczyn (przecięcie)

Definicja 8.

Iloczynem zbiorów rozmytych A i B nazywamy zbiór $A \cap B$ określony przez funkcję przynależności (11)

$$\forall_{x \in X} \mu_{A \cap B}(x)=\min(\mu_A(x), \mu_B(x))=\mu_A(x) \wedge \mu_B(x)$$

3.3. Dopełnienie (uzupełnienie)

Definicja 9.

Uzupełnieniem zbioru rozmytego A nazywamy zbiór A' określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A'}(x) = 1 - \mu_A(x) \qquad (12)$$

3.4. Suma ograniczona

Definicja 10.

Sumą ograniczoną zbiorów rozmytych A i B nazywamy zbiór $A \oplus B$ określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A \oplus B}(x)=\min(\mu_A(x) + \mu_B(x), 1) \qquad (13)$$

3.5. Różnica ograniczona

Definicja 11.

Różnicą ograniczoną zbiorów rozmytych A i B nazywamy zbiór $A \ominus B$ określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A \ominus B}(x) = \max(\mu_A(x) - \mu_B(x), 0) \qquad (14)$$

3.6. Iloczyn ograniczony

Definicja 12.

Iloczynem ograniczonym zbiorów rozmytych A i B nazywamy zbiór $A \odot B$ określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A \odot B}(x)=\max(0, \mu_A(x) + \mu_B(x) - 1) \qquad (15)$$

3.7. Suma algebraiczna

Definicja 13.

Sumą algebraiczną zbiorów rozmytych A i B nazywamy zbiór A+B określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A+B}(x)=\mu_A(x)+\mu_B(x)-\mu_A(x)\mu_B(x) \quad (16)$$

3.8. Iloczyn algebraiczny

Definicja 14.

Iloczynem algebraicznym zbiorów rozmytych A i B nazywamy zbiór AB określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{AB}(x)=\mu_A(x)\mu_B(x) \quad (17)$$

3.9. Suma drastyczna

Definicja 15.

Sumą drastyczną zbiorów rozmytych A i B nazywamy zbiór A∨B określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A \vee B}(x)= \begin{cases} 1 & \text{dla } \mu_A(x)>0 \text{ i } \mu_B(x)>0 \\ \mu_A(x) & \text{dla } \mu_B(x)=0 \\ \mu_B(x) & \text{dla } \mu_A(x)=0 \end{cases} \quad (18)$$

3.10. Iloczyn drastyczny

Definicja 16.

Iloczynem drastycznym zbiorów rozmytych A i B nazywamy zbiór A∧B określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A \wedge B}(x)= \begin{cases} 0 & \text{dla } \mu_A(x)<1 \text{ i } \mu_B(x)<1 \\ \mu_A(x) & \text{dla } \mu_B(x)=1 \\ \mu_B(x) & \text{dla } \mu_A(x)=1 \end{cases} \quad (19)$$

3.11. Mnożenie przez liczbę rzeczywistą

W przypadku, gdy liczba rzeczywista dodatnia przemnożona przez wysokość zbioru A nie jest większa od 1, możemy zdefiniować mnożenie zbioru przez liczbę.

Definicja 17.

Iloczynem zbioru rozmytego A przez liczbę rzeczywistą α nazywamy zbiór αA określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{\alpha A}(x)=\alpha \mu_A(x) \quad (20)$$

3.12. Potęgowanie algebraiczne zbioru rozmytego

Dla liczby rzeczywistej $\alpha>0$ możemy określić potęgę zbioru rozmytego A w myśl następującej definicji.

Definicja 18.

Potęgą zbioru rozmytego A nazywamy zbiór αA określony przez funkcję przynależności

$$\forall_{x \in X} \mu_{A^\alpha}(x) = [\mu_A(x)]^\alpha \quad (21)$$

3.13. Iloczyn kartezjański

Definicja 19.

Iloczynem kartezjańskim zbioru rozmytego A z przestrzeni X i zbioru rozmytego B z przestrzeni Y nazywamy zbiór AxB określony przez funkcję przynależności

$$\forall_{x \in X} \forall_{y \in Y} \mu_{A \times B}(x,y) = \min(\mu_A(x), \mu_B(y)) \quad (22)$$

3.14. Obcięcie zbioru rozmytego

Definicja 20.

α-obcięciem zbioru rozmytego A nazywamy zbiór A_α określony następująco

$$A_\alpha = \{x \in X: \mu_A(x) \geq \alpha\} \qquad \alpha \in [0, 1] \quad (23)$$

3.15. Koncentracja

Jako szczególne przypadki potęgowania, które często występują w zastosowaniach, rozpatruje się operacje koncentracji i rozpraszania.

Definicja 21.
Koncentracją zbioru rozmytego A nazywamy zbiór CON(A) określony następująco

$$CON(A) = A^2 \quad (24)$$

Funkcja przynależności tego zbioru wyraża się wzorem

$$\mu_{CON(A)}(x) = \mu^2_A(x) \quad (25)$$

3.16. Rozpraszanie

Definicja 22.
Rozproszeniem zbioru rozmytego A nazywamy zbiór DIL(A) określony następująco

$$DIL(A) = A^{1/2}. \quad (26)$$

Funkcja przynależności tego zbioru wyraża się wzorem

$$\mu_{DIL(A)}(x) = \mu_A^{1/2}(x) \quad (27)$$

3.17. Intensyfikacja kontrastu

Definicja 23.
Zbiorem rozmytym o zintensyfikowanym kontraście nazywamy zbiór INT(A) określony następująco

INT(A) = CON(A) dla $\mu_A(x)<0.5$ lub
DIL(A) dla $\mu_A(x)\geq 0.5$ (28)

Funkcja przynależności tego zbioru wyraża się wzorem

$$\mu_{DIL(A)}(x) = \begin{cases} \mu_A^2(x) & \text{dla } \mu_A(x)<0.5 \\ \mu_A^{1/2}(x) & \text{dla } \mu_A(x)\geq 0.5 \end{cases} \quad (29)$$

3.18. Zmniejszenie kontrastu

Definicja 24.
Zbiorem rozmytym o zmniejszonym kontraście nazywamy zbiór BLR(A) określony następująco

BRL(A) = DIL(A) dla $\mu_A(x)<0.5$ lub
CON(A) dla $\mu_A(x)\geq 0.5$ (30)

Funkcja przynależności tego zbioru wyraża się wzorem

$$\mu_{DIL(A)}(x) = \begin{cases} \mu_A^{1/2}(x) & \text{dla } \mu_A(x)<0.5 \\ \mu_A^2(x) & \text{dla } \mu_A(x)\geq 0.5 \end{cases} \quad (31)$$

4. PODEJMOWANIE DECYZJI W OTOCZENIU ROZMYTYM

4.1. Określenie otoczenia rozmytego

Podstawowym pojęciem takiego podejścia do podejmowania decyzji, zaproponowanego przez Bellmana i Zadeha [1] jest pojęcie otoczenia rozmytego.

Definicja 25 [8].
Otoczeniem rozmytym problemu podejmowania decyzji nazywamy następującą czwórkę uporządkowaną

(X,G,C,D) (32)

gdzie X={x} jest zbiorem możliwych decyzji, G - celem rozmytym, C - ograniczeniem rozmytym, D - decyzją rozmytą.

Elementy zbioru X mogą być dowolne np. rodzaj materiału, wielkość inwestycji, rodzaj zakupu, strategie rozwoju ekonomicznego itp. - wszystko co podlega wyborowi. Mamy tu do

czynienia z sytuacją, gdzie na możliwe decyzje są nałożone pewne ograniczenia. A wiec nie wszystkie decyzje są dopuszczalne. Szukać będziemy najlepszej decyzji spośród dopuszczalnych.

Podamy teraz definicje celu rozmytego i ograniczenia rozmytego. Następnie określimy decyzje rozmytą.

Definicja 26.

Cel rozmyty określa się jako zbiór rozmyty $G \subset X$ o funkcji przynależności $\mu_G(x)$. Przykładowo, niech X=R będzie zbiorem liczb rzeczywistych. Celem rozmytym może być osiągnięcie liczby dużo większej od 100. zbiór G można określić przy pomocy funkcji przynależności $\mu_G(x)$ np.

$$\mu_A(x) = \begin{cases} 0 & \text{dla } x \le 100, \\ \dfrac{1}{1+100(X-100)^{-1}} & \text{dla } x > 100. \end{cases}$$

Definicja 27.

Ograniczenie rozmyte definiuje się jako zbiór rozmyty $C \subset X$ o funkcji przynależności $\mu_C(x)$.

I tak na przykład, gdy X=R jest zbiorem liczb rzeczywistych, to ograniczeniem może być warunek, że liczba powinna być około 200. Funkcja przynależności zbioru C może być przedstawiona wzorem

$$\mu_C(x) = \begin{cases} 0 & \text{dla } x<150 \text{ lub } x>250, \\ 0.02x-3 & \text{dla } 150 \le x \le 200, \\ 0.02x+5 & \text{dla } 200 \le x \le 250, \end{cases} \quad (34)$$

Jak widać definicje celu rozmytego i ograniczenia rozmytego są właściwie identyczne. Występuje tu ścisła analogia miedzy tymi pojęciami. W teorii zbiorów rozmytych traktuje się je jednakowo.

4.2. Decyzja rozmyta

Decyzja ma być taka, aby osiągnąć pożądany cel oraz spełnić ograniczenia. Decyzja rozmyta D jest więc pewną agregacją zbiorów rozmytych G i C. Ta agregacja zbiorów G i C może być dokonana na różne sposoby. Podstawową i powszechnie stosowaną decyzją rozmytą jest decyzja rozmyta typu minimum [7]. Opiera się ona na zasadzie, aby osiągnąć cel G i jednocześnie spełnić ograniczenie C. Odpowiada to agregacji zbiorów G i C w sensie iloczynu (przecięcia) zbiorów rozmytych.

Definicja 28 [7].

Decyzję rozmytą typu minimum określa się jako

$$D = G \cap C. \quad (35)$$

Funkcja przynależności decyzji rozmytej D jest następująca

$$\mu_D(x) = \min_x (\mu_G(x), \mu_C(x)) \quad (36)$$

W literaturze tę definicję nazywa się "pesymistyczną" jako, że bazuje na mniejszych wartościach funkcji $\mu_G(x)$, i $\mu_C(x)$ [3], [4], [8]. Dla celu G i ograniczenia C przedstawionych wzorami (33) i (34) decyzja D będzie jak na rysunku 1. Możliwe dopuszczalne decyzje są z przedziału (150,250).

Można też tworzyć decyzje rozmyte wykonując inne działania na zbiorach G i C np.

a) decyzja rozmyta typu iloczyn algebraiczny

$$D = G \bullet C, \quad (37)$$

b) decyzja rozmyta typu kombinacja wypukła o funkcji przynależności

$$\mu_D(x) = r \bullet \mu_G(x) + (1-r) \bullet \mu_G(x), \; r \in [0,1], \quad (38)$$

c) decyzja rozmyta typu maksimum

$$D = G \cup C. \qquad (39)$$

Ostatnia wymieniona decyzja nazywana jest "optymistyczną" [3], [4], [8].

4.3. Decyzja optymalna

Otrzymana decyzja rozmyta jako iloczyn celu i ograniczenia daje pewną wskazówkę jaka decyzja nierozmyta jest najlepsza. Będziemy wybierać taką decyzję nierozmytą, której stopień przynależności w decyzji rozmytej jest największy.

Definicja 29.
Decyzją optymalną nazywamy takie x_{opt}, że

$$\mu_D(x_{opt}) = \sup_{x \in X} \mu_D(x) \qquad (40)$$

Należy zwrócić uwagę, że decyzja x_{opt}, nie zawsze musi być jednoznaczna, a nawet może nie istnieć w przypadku, gdy zbiory G i C są rozłączne.

Na rysunku 1 decyzją optymalną jest

$$x_{opt} \cong 222.47.$$

Podamy teraz przykład związany z praca i bezpieczeństwem górników.

Przykład 3.
W kopalni należy wywiercić otwór w górotworze w odległości około 500 metrów od wyznaczonego punktu. Ponieważ własności górotworu w tym rejonie nie są całkowicie znane, należy wziąć pod uwagę to, aby bezpieczeństwo górników było możliwie duże. Cel G "około 500 metrów" scharakteryzowano funkcją przynależności (41)

$$\mu_G(x) = \begin{cases} 0.02x-8 & \text{dla } 400<x<450, \\ 1 & \text{dla } 450 \le x \le 550, \\ -0.02x+12 & \text{dla } 550<x<600, \\ 0 & \text{dla } x \le 400 \text{ lub } x \ge 600. \end{cases}$$

Bezpieczeństwo pracy górników na interesującym nas kierunku od wyznaczonego punktu określono w sposób przybliżony przy pomocy funkcji

$$\mu_C(x) = \begin{cases} 1 & \text{dla } 0 \le x \le 400, \\ 1.25 \cdot 10(x-600)+0.5 & \text{dla } 400<x<600, \\ 0.5 & \text{dla } x \ge 600. \end{cases} \qquad (42)$$

Liczba 1 oznacza tu bezpieczeństwo równe 100%. Funkcje $\mu_G(x)$ i $\mu_C(x)$ pokazane są na rysunku 2. Możliwymi decyzjami do podjęcia są $x \in [400,600]$. Decyzja optymalna x_{opt}, maksymalizuje funkcję przynależności $\mu_D(x)$.

$$\mu_D(x_{opt}) = \sup_{x \in [400,600]} \mu_D(x) = \sup_{x \in [400,600]} \min[\mu_D(x), \mu_C(x) \qquad (43)$$

Maksimum dla funkcji $\mu_D(x)$ jest osiągnięte dla x_{opt}=440.83. Należy więc wywiercić otwór w odległości 440.83 metrów od wyznaczonego punktu.

5. UWAGI KOŃCOWE

Teoria zbiorów rozmytych zyskuje sobie obecnie coraz większe uznanie. Znajduje zastosowanie w coraz więcej dziedzinach, o których była mowa w rozdziale 1. Ponieważ w górnictwie wiele wielkości czy cech górotworu nie jest dokładnie znanych lub wiele związków nie da się dokładnie opisać wydaje się, ze teoria zbiorów rozmytych mogłaby tez być tu zastosowana. Podamy teraz dziedziny, w których ta teoria mogłaby usprawnić proces podejmowania decyzji w górnictwie:

a) przy ustalaniu rejonów szczególnie niebezpiecznych w kopalni, gdy znajomość górotworu jest tylko przybliżona,
b) przy prognozowaniu wstrząsów i tąpań, gdy opis zjawiska powstawania wstrząsów jest niedokładny,
c) przy organizacji pracy z uwzględnieniem

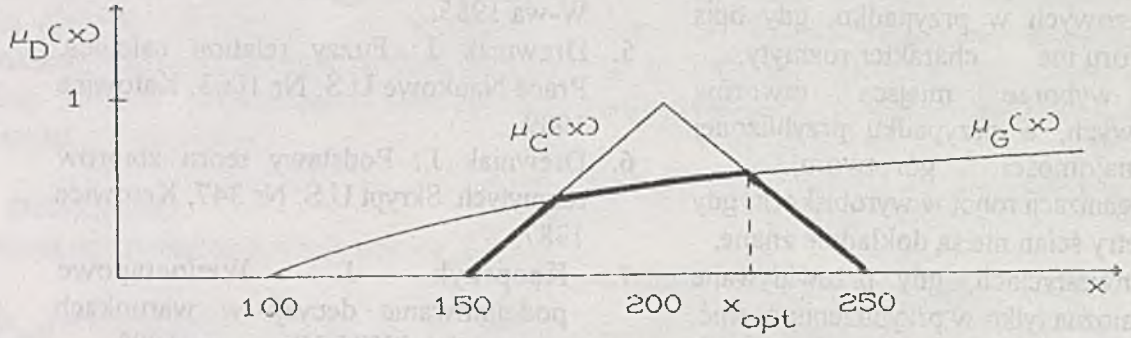
możliwie dużego bezpieczeństwa pracy,

d) przy wyborze miejsca wiercenia otworów badawczych pod szyb, w przypadku niedokładnego oszacowania warunków naturalnych złoża,
e) przy wyborze sposobu organizacji głębienia szybu ze względu na rozmyty opis warunków hydrogeologicznych,
f) przy planowaniu drążenia wyrobisk korytarzowych w przypadku, gdy opis górotworu ma charakter rozmyty,
g przy wyborze miejsca otworów strzałowych, w przypadku przybliżonej tylko znajomości górotworu,
h) przy organizacji robot w wyrobiskach, gdy parametry ścian nie są dokładnie znane,
i) przy inwestycjach, gdy przewidywane efekty można tylko w przybliżeniu ocenić,
j) przy projektowaniu kopalń, gdy wskaźnik oceny ekonomicznej efektywności inwestycji trudno jest ocenić. Trudno przewidzieć, ile będzie on wynosił po zrealizowaniu inwestycji.

Przytoczyliśmy tutaj niektóre tylko zagadnienia z zakresu górnictwa, w których można by wykorzystać teorie zbiorów rozmytych w celu polepszenia procesu podejmowania decyzji. Tych obszarów zastosowań może być więcej, ponieważ sytuacji niejasnych i nieprecyzyjnie opisanych może być dużo.

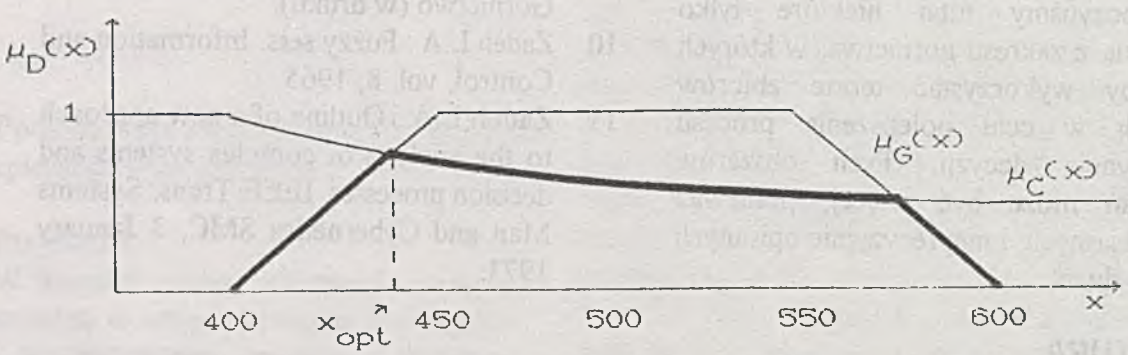
LITERATURA

1. Bellman R.E., Zadeh L.A.: Decision - making in fuzzy enviroment. Man. Sc.17, N.4, 1970.
2. Bolc L., Borodziewicz W., Wojcik M.: Podstawy przetwarzania informacji niepewnej i niepełnej. PWN, Warszawa 1991. 3. Czogała E., Pedrycz W.: Elementy i metody teorii zbiorów rozmytych. Skrypt Politechniki Śląskiej Nr 989, Automatyka, Gliwice 1980.
4. Czogała E., Pedrycz W.: Elementy i metody teorii zbiorów rozmytych. PWN, W-wa 1985.
5. Drewniak J.: Fuzzy relation calculus. Prace Naukowe U.Ś. Nr 1063, Katowice 1989.
6. Drewniak J.: Podstawy teorii zbiorów rozmytych. Skrypt U.Ś. Nr 347, Katowice 1987.
7. Kacprzyk J.: Wieloetapowe podejmowanie decyzji w warunkach rozmytości. PWN, Warszawa 1983.
8. Kacprzyk J.: Zbiory rozmyte w analizie systemowej. PWN, Warszawa 1983.
9. Kowalik S.: Wykorzystanie teorii zbiorów rozmytych do podejmowania decyzji. Zeszyty Naukowe Pol. Śl., Górnictwo (w druku).
10. Zadeh L.A.: Fuzzy sets. Information and Control, vol. 8, 1965.
11. Zadeh L.A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. Systems Man and Cybernetics SMC, 3 January 1973.



Rys. 1. Wyznaczanie decyzji rozmytej i optymalnej

Fig. 1. Calculating of fuzzy decision and optimal decision



Rys. 2. Decyzja rozmyta i optymalna dla wiercenia otworu w górotworze (przykład 5)

Fig. 2. Fuzzy decision and optimal decision for perforation in rock mass (example 5)

Jacek Stochlak
ICL Warszawa

Zarządzanie zmianami w środowisku gospodarowania lat 90-tych

Wprowadzenie

Dekada lat 90-tych będzie okresem szybko zmieniających się wymagań rynku, okresem krótko trwających szans odniesienia sukcesu oraz konieczności natychmiastowego podejmowania decyzji w zarządzaniu przedsiębiorstwami.

Dla organizacji gospodarczych z dobrze zdefiniowaną wizją rozwoju i skutecznym, posiadającym pełne informacje szczeblem zarządzania, lata 90-te oferują szersze i bardziej dostępne rynki zbytu. Dostępność tych rynków oznacza jednak za sobą wzrost konkurencji. Aby uzyskać przewagę konkurencyjną, przedsiębiorstwa muszą coraz bardziej ściśle współpracować ze swoimi dostawcami, klientami i innymi organizacjami uczestniczącymi w procesie gospodarowania.

Systemy informatyczne, wspomagające procesy gospodarowania, stały się życiodajną krwią dla większości organizacji gospodarczych. Systemy te rozwijały się jednak tak szybko, że wiele organizacji i przedsiębiorstw pozostało bez jasno zdefiniownej strategii ich rozwoju i integracji. Odizolowane wyspy informacyjne są bardzo często spotykaną rzeczywistością.

Zatrzymajmy się na chwilę próbując odpowiedzieć na pytanie jakie jest miejsce systemów informatycznych w świecie zdominowanym przez konkurecję, świecie szybkiego postępu i rozwoju technologicznego oraz świecie olbrzymich szans sukcesu.

Nowoczesne przedsiębiorstwa wykorzystujące różnorodne systemy informatyczne do wspomagania procesu gospodarowania, napotykają na trzy podstawowe bariery:

- Trudności w integracji różnorodnych systemów informatycznych.
- Trudności w przekazywaniu i współdzieleniu danych pomiędzy systemami.
- Brak możliwości przenoszenia aplikacji pomiędzy różnymi platformami systemowymi.

Przedsiębiorstwa dostrzegając te ograniczenia, próbują je usunąć poprzez:

- zakup systemów komputerowych różnych wielkości od jednego dostawcy, oczekując, że będą one ze sobą współdziałały,
- zatrudnienie pracowników o specjalistycznych umiejętnościach, celem opracowania interfejsów niezbędnych do współpracy ze sobą różnych systemów,
- zakup systemów zgodnych ze standardami międzynarodowymi i przemysłowymi, które zapewniają ich wzajemną współpracę,

Coraz większego znaczenia nabierają w takiej sytuacji standardy Systemów Otwartych. Większość specjalistów i konsultantów jest zgodna co do tego, że opracowanie spójnego systemu informatycznego obejmującego wszystkie aspekty zarządzania nowoczesnego przedsiębiorstwa jest niemożliwa bez standardów. Standardy są konieczne na każdym poziomie do zapewnienia integracji wszystkich elementów systemu, od bazy danych, usług sieciowych, oprogramowania operacyjnego do oprogramowania

narzędziowego czy też poczty elektronicznej.

Zbudowanie nowoczesnego systemu informatycznego, odpowiadającego wymaganiom współczesnego środowiska ekonomicznego, nie jest jednak zadaniem prostym. ICL rozumiejąc skalę i złożoność tego procesu opracował metodologię, która ma na celu pomoc przy tworzeniu, implementacji i wdrażaniu systemów informatycznych oraz ich integracji z procesami gospodarowania przedsiębiorstw. Jest nią **OPEN***framework*.

Czym jest **OPEN***framework* ?

OPEN*framework* jest prosta i pewną metodą integracji systemów informatycznych w świecie Systemów Otwartych.

Prostą, dzięki strukturze dopasowanej do metod **OPEN***framework*. Pewną, dzięki wiedzy i doświadczeniu w integracji systemów informatycznych wykorzystujących najnowsze rozwiązania wielu producentów. To metoda integracji systemów, ponieważ osiąganie sukcesu w działalności gospodarowania wymaga pełnej integracji wykorzystywanych procesów gospodarowania ze wspomagającymi je systemami informatycznymi. I wreszcie, dotyczy ona **świata Systemów Otwartych**, gdyż właśnie w tym środowisku działają metody **OPEN***framework*.

OPEN*framework* określa architekturę oraz zapewnia wiedzę i umiejętności umożliwiające przedsiębiorstwom osiąganie maksymalnych korzyści z posługiwania się technologią informatyczną. Zachęcając jednocześnie do poszukiwania i wdrażania koniecznych zmian w swojej strategii i procesach gospodarowania, strukturze organizacyjnej oraz miejscu i roli systemów informatycznych w przedsiębiorstwie.

Artykuł ten jest ogólną prezentacją architektury **OPEN***framework* oraz określonych jej metod wykorzystywanych do opracowywania architektury gospodarowania i architektury systemów informatycznych przedsiębiorstw, działających w dynamicznie zmieniającym się środowisku gospodarowania końca lat 90-tych.

Przesiębiorstwo

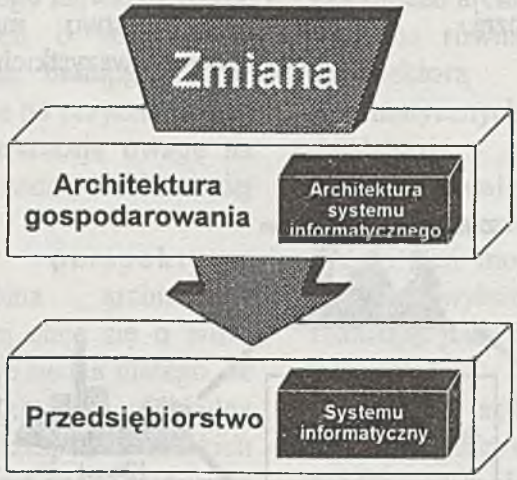
Rys. 1 przedstawia ogólny model zmian zachodzących we współczesnym przedsiębiorstwie.

W terminologii **OPEN***framework*, pod pojęciem **architektura gospodarowania** są rozumiane zasady gospodarowania przedsiębiorstwa. Obejmują one również architekturę systemów informatycznych wspomagających tą działalność. Jak przedstawiono na Rys. 1, celem wprowadzenia tego modelu jest ułatwienie przedsiębiorstwom zarządzanie zmianami, będącymi charakterystyczną cechą współczesnego środowiska gospodarowania.

Poprawnie zdefiniowana architektura gospodarowania jest istotnym czynnikiem pomagającym kadrze kierowniczej w określeniu zadań dla systemów informatycznych przedsiębiorstwa. Co więcej, powinna być ona również istotną pomocą we wskazaniu w jaki sposób przedsiębiorstwo może zwiększyć swą podatność na zmiany. Zwrot ten w terminologii **OPEN***framework* oznacza elastyczność i skuteczność wykorzystywanych zasad gospodarowania.

Architektura zasad gospodarowania przedsiębiorstwa jest jego unikalnym zasobem. Nie jest ona czymś, co można kupić od producenta czy innego dostawcy. Jasno zdefiniowana architektura może zapewnić przedsiębiorstwu stabilną i pewną podstawę do zarządzania zmianami w wielu obszarach jego działalności. Obejmują one:

- Kontrolę projektowanych zmian w procesach gospodarowania
- Szybką reakcję na zmiany w środowisku gospodarowania



Rys. 1 Zarządzanie zmianami

- Wprowadzanie integracji czynnika ludzkiego i technologii informatycznych
- Zachęcanie do stopniowego lecz stałego wprowadzania zmian
- Zmianę zakresu gospodarowania
- Kierowanie wyborem najbardziej właściwej technologii
- Zapewnienie możliwości współistnienia systemów odmiennych generacji
- Wykorzystywanie nawet najmniejszych szans odniesienia sukcesu

Architektura zasad gospodarowania wypracowana metodami **OPEN*framework***, zaczyna się analizą zasobów ludzkich danego przedsiębiorstwa oraz drogi na jakiej wykorzystują systemy informatyczne. Jest to szczególnie istotne w przypadku rozproszenia całego środowiska gospodarowania. Środowiska takie charakteryzują najczęściej odmienne wymagania dla poszczególnych obszarów działania.

Architektura systemu informatycznego umożliwia przedsiębiorstwu zbudowanie systemu informatycznego najlepiej dopasowanego do wykorzystywanych zasad gospodarowania i podatnego na zmiany nieuniknione w nadchodzących latach. Zbudowanie takiego systemu jest jednak procesem stopniowym. Tylko niewiele przedsiębiorstw może pozwolić sobie na rozpoczęcie od zera. Architektura **OPEN*framework*** uwzględnia ten fakt i pozwala na rozpoczęcie tego procesu od punktu w którym przedsiębiorstwo aktualnie się znajduje. Każde przedsiębiorstwo, które wykorzysta architekturę **OPEN*framework*** do zbudowania własnego systemu informatycznego odkryje, że jest ona przygotowana do uwzględnienia obecnie wykorzystywanych procedur i technologii.

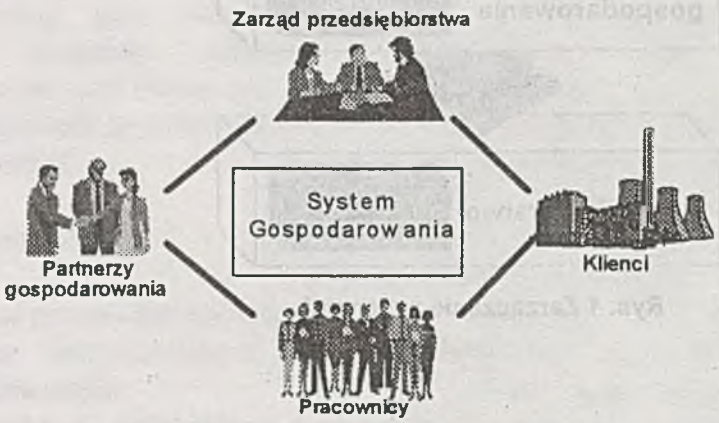
Architektura OPEN*framework*

Architektura **OPEN*framework*** obejmuje kilka podstawowych pojęć. Są nimi: **perspektywa gospodarowania i perspektywa informatyczna, atrybuty jakości** oraz architektoniczne **elementy gospodarowania i elementy informatyczne**. Dzięki nim można osiągnąć lepsze zrozumienie oczekiwań i

potrzeb środowiska ekonomicznego, procesów i zasad gospodarowania oraz roli i miejsca systemów informatycznych w prowadzonej działalności gospodarczej.

Perspektywy gospodarowania

Sposób w jaki funkcjonuje daneprzedsiębiorstwo musi uwzględniać opinie i zadania wszystkich osób, które mają



Rys. 2 Perspektywy

udział w jego sukcesie. Perspektywy gospodarowania przedstawione na rys. 2, reprezentują role i wzjemne powiązania pomiędzy tymi osobami, a także identyfikują kluczowe obszary dotyczące przedsiębiorstwa i środowiska ekonomicznego.

Dyrekcja przedsiębiorstwa jest odpowiedzialna za zarządzanie operacyjne oraz opracowanie strategii gospodarowania przedsiębiorstwa. Tu właśnie podejmowane są decyzje inwestycyjne w odniesieniu do nowych produktów i usług, a także rozbudowy i usprawnienia systemów informatycznych oraz innych zasobów przedsiębiorstwa. Najważniejsze jest jednak to, że pełną odpowiedzialność za wszystkie podejmowane w przedsiębiorstwie działania ponosi jego dyrektor.

Klienci to grupa osób obsługiwanych bezpośrednio lub pośrednio przez przedsiębiorstwo. Niezależnie od prowadzonej działalności, o tym czy odniesie ono sukces czy też poniesie porażkę decyduje ocena i stopień spełnienia oczekiwań klientów. Dlatego też, identyfikacja klientów oraz zrozumienie ich potrzeb jest decydującym czynnikiem odniesienia sukcesu.

Partnerzy gospodarowania przedsiębiorstwa to te elementy środowiska ekonomicznego z którymi zawierane są różnego rodzaju krótko lub długoterminowe powiązania kooperacyjne. Powiązania te mogą mieć czasami bardzo głęboki i złożony charakter. Na przykład, dotyczyć one mogą wspólnych prac badawczo-rozwojowych nad nowymi produktami i usługami rynkowymi. Niezależnie jednak od tego, czy są one złożone czy proste, maja istotny wpływ na pozycję i kondycję finansową przedsiębiorstwa. Partnerzy mają określone wymagania w stosunku do przedsiębiorstwa, ono zaś pewne oczekiwania w stosunku do

swoich dostawców i kooperantów.

Pracownicy pracują w przedsiębiorstwie. To oni stanowią jeden z jego najważniejszych elementów, decydujących o sukcesie w procesie gospodarowania, budując w nim jednocześnie swe nadzieje na przyszłość. Oni też w dużym stopniu zwracają uwagę na środowisko pracy oraz zadowolenie z niej płynące.

Wykorzystując perspektywy gospodarowania, ogólna architektura działalności gospodarczej staje się o wiele bardziej przejrzysta. Dzieje się tak dlatego, że podejście to pozwala na dokładną identyfikację wszystkich zespołów ludzkich uczestniczących w procesie gospodarowania wraz z ich zadaniami i oczekiwaniami odnośnie wykonywanych zadań.

Perspektywy informatyczne

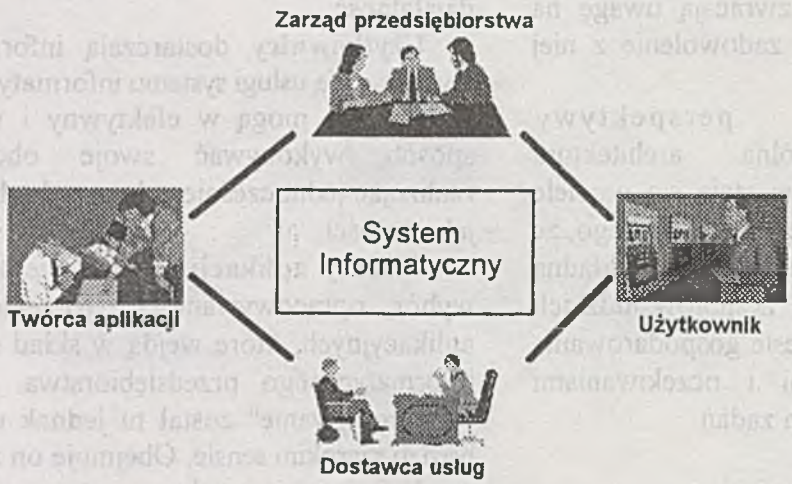
Zespoły pracownicze związane z systemami informatycznymi przedsiębiorstwa opisują perspektywy informatyczne architektury **OPEN***framework* (Rys. 3). Taki sposób klasyfikacji pracowników ma na celu ułatwienie przedsiębiorstwu ocenę efektywności i pełności posiadanych systemów informatycznych.

Do zadań dyrekcji przedsiębiorstwa należy określenie w jaki sposób systemy informatyczne mają wspomagać prowadzoną działalność gospodarczą, przyczyniać się do poprawy efektywności i wydajności pracy oraz zapewniać realizację celów i misji przedsiębiorstwa. Dyrekcja przedsiębiorstwa wskazuje i zleca opracowanie usług aplikacyjnych koniecznych do zaspokojenia potrzeb i wykorzystania wszystkich szans odniesienia sukcesu w procesie gospodarowania. Ponadto decyduje ona o strategii wykorzystania informatyki w przedsiębiorstwie oraz nakreśla zasady pracy twórców aplikacji, dostawców usług i samych użytkowników systemu. Dba także o to, aby systemy te były przygotowane i podatne na zmiany konieczne w przyszłości. Obszarem działania dyrekcji przedsiębiorstwa jest zasadniczo architektura gospodarowania lecz jest ona również mostem łączącym ją z architekturą techniczną systemów informatycznych wspomagajacych tą działalność.

Użytkownicy dostarczają informacje i wykorzystują usługi systemu informatycznego. Dzięki nim mogą w efektywny i wydajny sposób wykonywać swoje obowiązki, realizując jednocześnie cele przedsiębiorstwa jako całości.

Twórcy aplikacji są odpowiedzialni za wybór, opracowywanie i tworzenie usług aplikacyjnych, które wejdą w skład systemu informatycznego przedsiębiorstwa. Termin "opracowywanie" został tu jednak użyty w bardzo szerokim sensie. Obejmuje on również między innymi, zakupy poszczególnych elementów składowych rozwiązania oraz prowadzenie koniecznych szkoleń użytkowników systemu.

Dostawcy usług planują i zarządzają zasobami informatycznymi przedsiębiorstwa. Są odpowiedzialni za zapewnienie koniecznych atrybutów jakości systemu informatycznego (dostępności, użyteczności, wydajności, bezpieczeństwa oraz podatności na zmiany) przy zachowaniu najniższych kosztów jego posiadania. Dostawcy usług zapewniają konieczny sprzęt i oprogramowanie systemowe, prowdzą konieczne szkolenia oraz udostępniają dokumentację systemu. Ich zadaniem jest optymalizacja niezbędnych nakładów inwestycyjnych w obszarze systemów informatycznych.



Rys. 3 Perspektywy informatyczne

Atrybuty jakości

Metodologia **OPEN*framework*** wprowadza 5 atrybutów jakości (Rys. 4) architektury gospodarowania i architektury systemu informatycznego przedsiębiorstwa.
Relacje istniejące pomiędzy uczestnikami procesów gospodarowania, uwidocznionych w perspektywach gospodarowania i prespektywach informatycznych, określają zadania poszczególnych grup wyrażone w kategoriach atrybutów jakości. Atrybuty jakości **OPEN*framework*** ukazują ich oczekiwania w stosunku do elementów gospodarowania i elementów informatycznych systemu.

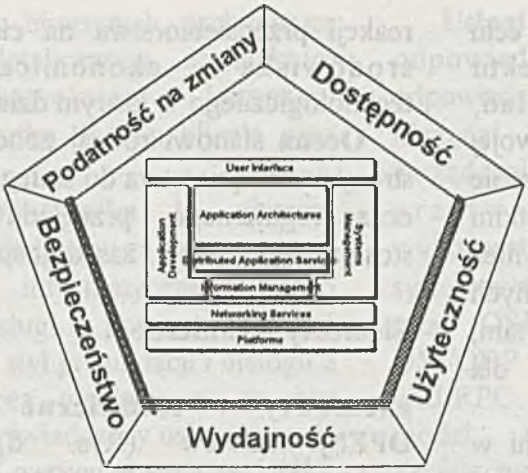
Dostępność to miara jakości odpowiadająca na pytanie czy usługi systemowe są wszędzie tam gdzie są potrzebne oraz czy spełniają oczekiwania użytkowników. Ponadto, atrybut ten określa niezawodność i pewność świadczonych usług.

Użyteczność to miara jakości określająca jak efektywnie użytkownicy mogą wykonywać stawiane im zadania w danym środowisku i miejscu pracy. Ponadto, atrybut ten mówi o skuteczności i efektywności przedsiębiorstwa, jakości jego produktów i usług oraz troski i dbałości o jego klientów.

Wydajność to atrybut systemu mówiący o tym, czy czas reakcji systemu i jego przepustowość spełnia oczekiwania działalności gospodarczej. Ponadto, dotyczy on rentowności gospodarowania, poziomu inwestycji, wprowadzanych innowacji, zadowolenia pracowników oraz poziomu świadczonych usług.

Bezpieczeństwo to miara jakości określająca jego odporność na próbę niedozwolonego dostępu do informacji i usług systemowych. Ponadto dotyczy ona ochrony finansowej, stopnia ryzyka oraz rzetelności i integralności przedsiębiorstwa.

Podatność na zmiany to przygotowanie systemu gospodarowania do dalszego rozwoju i wprowadzania koniecznych zmian w



Rys. 4 Atrybuty jakości

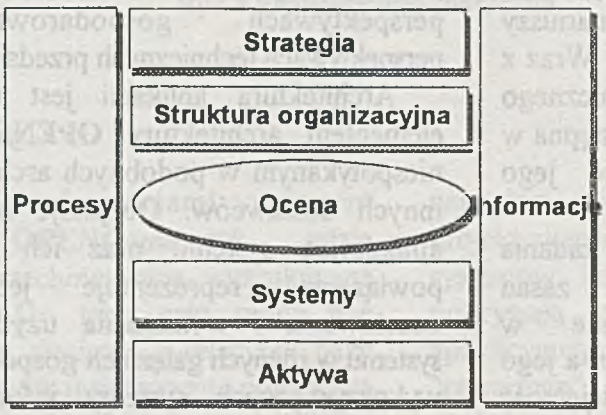
odpowiedzi na pojawiajace się nowe wymagania i oczekiwania rynku. Jak również, możliwość ponownego wykorzystania i adaptacji posiadanych elementów systemu. Atrybut ten odnosi się do takich obszarów działalności jak wielokrotne wykorzystywanie posiadanych aktywów i zasobów, wprowadzanie automatyzacji pracy oraz wdrażaniu koncepcji stale uczącego się przedsiębiorstwa.

Elementy gospodarowania

Elementy gospodarowania przedsiębiorstwa (Rys. 5) reprezentują te jego aspekty, które stanowią o jego obecnej sytuacji i pozycji na rynku. Tworzą one obszar bezpośrednio dostępny dyrekcji przedsiębiorstwa, oddziaływający jednak na pozostałych uczestników prowadzonej działalności, uwidocznionych w perspektywach gospodarowania.

Strategia określa produkty i usługi, które będą oferowane przez przedsiębiorstwo, zakres i obszary planowanej działalności, terytoria na których zamierza prowadzić działalność oraz wszystkie inne zamierzenia przedsiębiorstwa na przyszłość.

Struktura organizacyjna określa w jaki sposób zasoby przedsiębiorstwa są



Rys.5 Elementy gospodarowania

wykorzystywane i zarządzane w celu osiągnięcia najlepszego efektu gospodarowania. Na przykład, przedsiębiorstwa muszą uwzględnić w swojej strukturze organizacyjnej, powierzenie wymaganego stopnia autonomii wszystkim tym, którzy go potrzebują, jak również łączenia w jedną spójną całość odrębnych jednostek organizacyjnych wszędzie tam, gdzie połączenie takie jest użyteczne dla prowadzonej działalności.

Systemy są narzędziami dostępnymi w przedsiębiorstwie, których zadaniem jest najbardziej efektywne i skuteczne prowadzenie procesów gospodarowania. Wprawdzie systemy informatyczne zajmują szczególnie ważne miejsce w metodologii **OPEN***framework* jednak nie zapomina ona o wszystkich innych systemach odgrywających istotną rolę i przyczyniających się do sukcesu gospodarowania przedsiębiorstwa.

Aktywami są wszystkie zasoby, umiejętności, kompetencje, infrastruktura oraz wiedza istniejąca w przedsiębiorstwie. Powinny być one we właściwy sposób wykorzystywane i odnawiane w ramach programów rozwoju, tak aby były zawsze dostępne i dobrze przygotowane do swoich zadań.

Informacje posiadane przez przedsiębiorstwo świadczą o jego bogactwie zarówno w obszarze zrozumienia istniejącej sytuacji rynkowej jak również umiejętności modelownia alternatywnych scenariuszy zachowania się rynku w przyszłości. Wraz z rozwojem środowiska ekonomicznego bazującego o wiedzy, informacja dostępna w przedsiębiorstwie będzie stanowić jego największe bogactwo i siłę.

Procesy określają w jaki sposób zadania wynikające z przyjętych zasad gospodarowania są wykonywane w przedsiębiorstwie oraz pomiędzy nim a jego partnerami koperacyjnymi. Umiejętność łagodnej i efektywnej zmiany obowiązujących procesów, jest głównym czynnikiem szybkiej reakcji przedsiębiorstwa na ciągłe zmiany środowiska ekonomicznego i technologicznego w którym działa.

Ocena stanowi rodzaj zobowiązania ze strony przedsiębiorstwa do stałego uczenia się oraz regularnego przeglądu i analizy stosowanych metod i zasad gospodarowania.

Elementy techniczne

Elementy techniczne metody **OPEN***framework* (Rys. 6) określają strukturalny model systemu informatycznego przedsiębiorstwa, ze szczególnym uwzględnieniem granic i intefejsów występujących pomiędzy jego poszczególnymi składnikami. Wartość tego modelu strukturalnego leży w jego podatności na zmiany, która wywodzi się z bardzo dokładnego określenia funkcji jego elementów oraz precyzyjnego zdefiniowania intefejsów i zasad współpracy pomiędzy nimi.

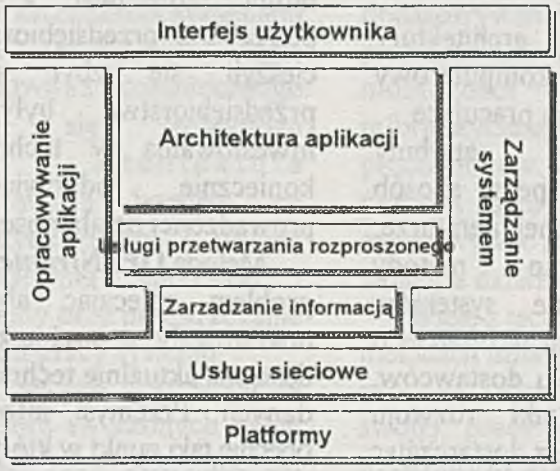
Architektura **OPEN***framework* definiuje silny i spójny szkielet oraz stabilną i pewną podstawę budowy systemu informatycznego przedsiębiorstwa. Opisuje ona wkład każdego elementu technicznego do całości systemu oraz pozwala regulować wzajemne współdziałanie pomiędzy nimi. Ukazuje ona również, w jaki sposób technologie informatyczne mogą być wykorzystywane do realizacji potrzeb i oczekiwań różnych grup pracowniczych występujących w perspektywach gospodarowania i perspektywach technicznych przedsiębiorstwa.

Architektura aplikacji jest unikalnym elementem architektury **OPEN***framework* niespotykanym w podobnych architekturach innych dostawców. Obejmuje ona usługi aplikacyjne systemu oraz ich wzajemne powiązania, reprezentuje jednocześnie oczekiwania i wymagania użytkowników systemu w różnych gałęziach gospodarowania wykorzystujących systemy informatyczne. Architektura aplikacji definiowana przez ICL obejmuje między innymi: architekturę

automatyzacji prac biurowych, architekturę obsługi handlu detalicznego, architekturę systemu informatycznego korporacji przemysłowej i banku obsługi klienta oraz wiele innych.

Interfejs użytkownika to element architektury odpowiedzialny za udostępnianie usług systemu informatycznego jego użytkownikom. Usługi te powinny posiadać spójny i przejrzysty styl prezentacji i dialogu z użytkownikiem, bez względu na to czy program użytkowy świadczący usługi pracuje na komputerze PC, serwerze systemu UNIX czy też systemie komputerowym klasy mainframe.

Usługi przetwarzania rozproszonego są odpowiedzialne za zapewnienie odpowiedniego środowiska systemowego oraz narzędzi umożliwiających tworzenie i zarządzanie systemów obiektowego przetwarzania rozproszonego wykorzystującego różne platformy systemowe. Podstawowymi standardami są tutaj OSI/TP (*Transaction Processing*), OSI/ODP (*Open Distributed Processing*) i OSI/RPC (*Remote Procedure Call*) oraz model OMA (*Object Management Architecture*) i DTP (*Distributed Transaction Processing*).



Rys.6 Elementy informatyczne

Zarządzanie informacjami to element architektury **OPEN***framework* gdzie informacja jest przechowywana, wyszukiwana i wymieniana. Do tego celu mogą być wykorzystywane sieciowe i relacyjne bazy danych, bazy danych zorientowane obiektowo oraz wszystkie typy i rodzaje systemu plików.

Opracowywanie aplikacji obejmuje narzędzia i metodologie tworzenia, projektowania, implementacji i pielęgnacji systemów informatycznych. Narzędzia te pokrywają pełny okres życia systemu aplikacyjnego i są skupione wokół "otwartego" słownika danych będącego centralnym zbiornikiem wszystkich informacji o systemie.

Zarządzanie systemem dotyczy zarządzania złożonymi systemami informatycznymi zbudowanymi na bazie platform i aplikacji pochodzących od wielu dostawców. Ten element architektury **OPEN*framework*** obejmuje sprzęt, usługi systemowe oraz infrastrukturę komunikacyjną specjalnie opracowaną w celu efektywnej i skutecznej kontroli oraz sterownia pracą systemu.

Usługi sieciowe są odpowiedzialne za zapewnienie możliwości współpracy oraz wymiany informacji pomiędzy aplokacjami pracującymi na różnych platformach systemowych oraz stacjach roboczych. Współpracujące ze sobą systemy mogą być węzłami rozległych sieci WAN i sieci lokalnych LAN wykorzystując praktycznie wszystkie dostępne obecnie protokoły komunikacyjne.

Platformy systemowe architektury **OPEN*framework*** to sprzęt komputerowy oraz systemy operacyjne na nim pracujące.

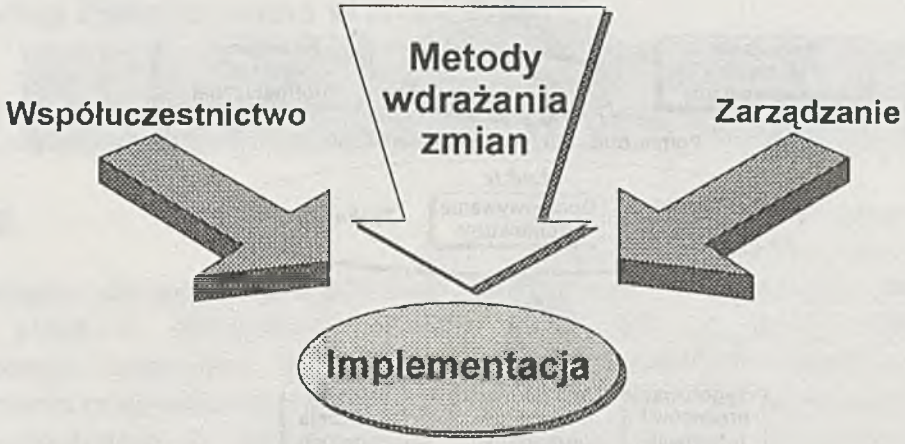
Wszystkie elementy i atrybuty **OPEN*framework*** zostały w pełny sposób opisane w powszechnie dostępnej literaturze. Zawiera ona omówienie metody **OPEN*framework*** w świetle systemów informatycznych zbudowanych w oparciu o produkty pochodzące od wielu dostawców, wskazując trendy i kierunki rozwoju technologii informatycznych oraz dostarczając porad i wskazówek implementacji systemów. Książki te stanowią istotną część wiedzy **OPEN*framework***.

Metodologia OPEN*framework*

Bazowa architektura **OPEN*framework*** musi być dostosowana do konkretnych wymogów każdego przedsiębiorstwa. Proces ten jest określany mianem specjalizacji osiąganej za pomocą metod **OPEN*framework***. Mówiąc prościej, specjalizacja powstaje poprzez połączenie potrzeb procesów gospodarowania przedsiębiorstwa z możliwościami dostępnej technologii informatycznej oraz bazowej wiedzy stojącej za architekturą **OPEN*framework***. Połączenie to ma na celu opracowanie architektury systemu informatycznego przedsiębiorstwa spełniającego aktualne i przyszłe wymagania prowadzonej działalności gospodarczej.

Tradycyjne podejście do opracowywania systemów informatycznych nie ułatwiało za nie odpowiedzialnym szybko i skutecznie reagować na zachodzące zmiany w środowisku ekonomicznym. W przeszłości departamenty odpowiadające za funkcjonowanie systemów informatycznych były zwykle odsunięte od procesów zarządzania przedsiębiorstwa. Wybór technologii był wówczas najczęściej podyktowany tym co poszczególni dostawcy mogli zaoferować a nie rzeczywistymi potrzebami przedsiębiorstwa. W rezultacie, cieszyli się zbyt silną pozycją, a przedsiębiorstwa były zmuszone do inwestowania w technologie, które nie koniecznie odpowiadały potrzebom prowadzonej działalności.

Metoda **OPEN*framework*** rozwiązuje ten problem zalecając aby przedsiębiorstwa inwestowały w architekturę systemu a nie dostępną aktualnie technologię przetwarzania danych. Przemysł informatyczny osiągnął obecnie taki punkt w którym standardy zostały powszechnie zaakceptowane, a tym samym rozwiązania mogą być budowane ze standardowych podzespołów pochodzących od różnych dostawców. W tych nowych rozwiązaniach jest również miejsce na dotychczas wykorzystywane systemy tak aby nie zmarnować dotychczas poniesionych nakładów inwestycyjnych. Oznacza to, że architektura systemu może być wykorzystywana jako przewodnik w zarządzaniu zmianami wynikającymi z ciągłych turbulencji środowiska ekonomicznego oraz osiągnięcia maksymalnych korzyści z posiadanego



Rys.7 Metodologia OPEN*framework*

systemu informatycznego.

Podejście oferowane przez **OPEN*framework*** zapewnia pewną i solidną podstawę tworzenia i zarządzania systemami informatycznymi w środowisku ciągle zmieniającego się środowiska ekonomicznego oraz rozwijającej się technologii informatycznej. Archtektura **OPEN*framework*** pozwla bowiem na wprowadzanie kolejnych zmian. Z drugiej strony jednak, zapewnia stabilność i przestrzeganie dyscypliny oraz przestrzeganie przyjętych zasad i reguł pracy systemu.

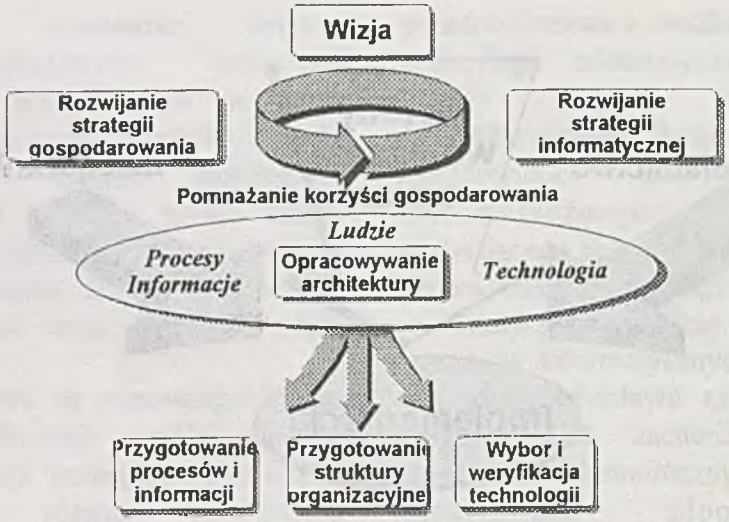
OPEN*framework* traktuje architekturę systemu jako prekursora tworzonych usług aplikacyjnych, nie zaś jako oderwane od rzeczywistości pojęcie teoretyczne. Zgodnie z podejściem **OPEN*framework***, każde przedsiębiorstwo ma odrębny i niepowtarzalny charakter. Dlatego też, sensownym wydaje się fakt, że jego zasady gospodarowania oraz wspomagający je system informatyczny powinny być traktowane jako coś unikalnego. Praktycznie każde przedsiębiorstwo może zastosować metodę **OPEN*framework*** do opracowania unikalnej architektury własnych procesów gospodarowania i systemu informatycznego.

Istotne jest również to, że metoda **OPEN*framework*** dotyczy całości przedsiębiorstwa a nie tylko jego systemu informatycznego. Jedną z jej zalet jest ciągłe dopasowywanie zasad gospodarowania przedsiębiorstwa i jego zasobów do możliwości architektury systemu informatycznego oraz wiedzy i umiejętności pracowników. Zawiera ona zbiór metod, za pomocą których przedsiębiorstwa mogą opracowywać swoją wizję na przyszłość oraz strategie działania konieczne do jej realizacji.

Wiele z technik wykorzystywanych w tych metodach istnieje już od dawna. Obejmują one procesy, które wykorzystują na co dzień w swojej pracy menadżerowie, architekci i integratorzy systemów. Wartość tych metod podkreśla fakt, że stosują one sprawdzone techniki, które zostały opracowane w taki sposób aby obejmowały wszystkie aspekty danego zadania, począwszy od opracowania początkowej wizji przedsiębiorstwa, a skończywszy na wyborze technologii implementacji architektury systemu informatycznego (Rys. 7).

Metody wdrażania zmian

Metdy polecane przez **OPEN*framework*** koncentrują się na pomocy przedsiębiorstwu w



Rys.8 Metody zarządzania zmianami OPEN *framework*

identyfikacji kluczowych zmian w zasadach gospodarowania i środowisku ekonomicznym na które musi zareagować. Wykorzystując tą informację, przedsiębiorstwo może opracować własną architekturę gospodarowania, która będzie stanowiła stabilną podstawę dla opracowania koniecznych zmian w wykorzystywanych procesach gospodarowania (Rys. 8).

Współuczestnictwo

Metody wykorzystwane w przedsiębiostwie do zarządzania zmianami są wykorzystywane przez różne zespoły pracowników w różnym czasie. W zależności od rodzju zadań, które muszą zostać wykonane, może się okazać, że zaistnieje potrzeba utworzenia grup roboczych obejmujących pracowników o różnej wiedzy i umiejętnościach. Może się również zdarzyć, że konieczne będzie skorzystanie z doświadczeń partnerów gospodarowania a nawet klientów.

Zarządzanie

Kolejnym aspektem zarządzania zmianami w przedsiębiorstwie jest sposób jego zarządzania. Może się okazać, że efektywne zastosowanie i wykorzystanie metod oferowanych przez **OPEN**framework będzie wymagało zmian w stylu jego zarządzania.

Implementacja

Jak wynika z powyżej przedstawionego rysunku, metody reagowania na zmieniające się oczekiwania rynku oraz środowiska gospodarowania. Powinny one doprowadzić przedsiębiostwa do punktu w którym będą one gotowe do wyboru odpowiednich narzędzi zarządzania konieczych do implementacji zakładanych strategii oraz procesów gospodarowania.

Tak więc na etapie implementacji, przedsiębiorstwa powinny posiadać dokładnie opracowaną architekturę gospodarowania oraz gotowość uruchomienia opracowanych procesów gospodarowania. Ponadto powinny one sformułować plany zmian postaw i metod pracy swoich pracowników wykorzystujących te procesy oraz przeprowadzić wdrożenie koniecznych systemów informatycznych je wspomagających.

Jerzy Skrzypek
Akademia Ekonomiczna w Krakowie

Rola i miejsce modeli symulacyjnych w procesie strategicznego zarządzania firmą

1.Wstęp

Spróbujmy zastanowić się w jaki sposób można poprawić efektywność procesu strategicznego zarządzania firmą poprzez wykorzystanie możliwości stwarzanych przez modele symulacyjne. Ze względu na istotę procesu strategicznego zarządzania firmą należy podkreślić, że rozważania dotyczą tle ustrukturalizowanych problemów decyzyjnych, to znaczy takich, w przypadku których trudno wskazać gotowe algorytmy obliczeniowe. W tej sytuacji, symulacyjny model komputerowy postawiony do dyspozycji decydenta, powinien pozwalać rozpoznać, zrozumieć i sformułować problem a następnie rozwiązać go przy pomocy narzędzi analitycznych dostarczanych przez model. W konsekwencji, autor niniejszego artykułu pragnie się skoncentrować na ocenie przydatności modeli symulacyjnych w trakcie konstrukcji strategicznego planu firmy, jego wdrożenia oraz kontroli wykonania. Jednym z podstawowych czynników branych pod uwagę przy ocenie użyteczności modelowania symulacyjnego w planowaniu strategicznym będzie możliwość wspomagania wdrożenia wybranej strategii.

2. Fazy zarządzania strategicznego versus fazy budowy modelu symulacyjnego

Zarządzanie strategiczne związane jest z wyborem strategii działania firmy, jej wdrożeniem oraz kontrolą realizacji, przebiegającym w fazach przedstawionych w Tablicy 1.

Tablica 1. Fazy procesu zarządzania strategicznego

Nazwa fazy	Stosowana metodologia
Identyfikacja problemu, który wymaga podjęcia działania	Analiza strategiczna (otoczenie, zasoby, cele działania)
Diagnoza sytuacji	Ocena pozycji strategicznej firmy
Zaprojektowanie niezbędnych działań	Wybór strategii (generowanie wariantów, ocena wariantów, wybór wariantu do realizacji)
Wdrożenie strategii	Wdrożenie strategii
Kontrola wykonania	Obserwacja realizacji wybranej strategii oraz wprowadzanie korekt

Źródło: opracowanie własne.

Z punktu widzenia konstruktora modelu symulacyjnego, stawiającego sobie za cel wspomaganie procesu strategicznego zarządzania firmą, warto wyróżnić następujące etapy zarządzania strategicznego:

- konstrukcja strategicznego planu przedsięwzięcia (obejmująca analizę strategiczną oraz wybór strategii);
- wdrożenie wybranej strategii;
- kontrola wykonania założeń planu strategicznego oraz ewentualne korekty.

Wspomniane wyżej fazy skonfrontujemy z fazami konstrukcji modeli sumulacyjnych (tablica 2).

Tablica 2. Fazy konstrukcji modeli symulacyjnych

Fazy i zadania	Opis fazy lub zadania
Faza "A"	Opis badanego obiektu wraz z analizą jakościową
Zadanie "A1"	Definicja celów modelowania
Zadanie "A2"	Tworzenie koncepcji modelu
Zadanie "A3"	Konstrukcja modelu graficznego
Zadanie "A4"	Analiza jakościowa modelu graficznego
Zadanie "A5"	Ustalenie wartości początkowych zmiennych oraz parametrów (kalibracja)
Zadanie "A6"	Konstrukcja modelu matematycznego
Zadanie "A7"	Zapisanie modelu komputerowego (kodowanie)
Faza "B"	Testowanie wiarygodności modelu
Zadanie "B1"	Przeprowadzenie testów weryfikacyjnych (verification)
Zadanie "B2"	Testowanie zgodności zachowania modelu z zachowaniem modelowanego obiektu (validation)
Zadanie "B3"	Testowanie zgodności struktur modelu ze strukturą modelowanego obiektu (legitimation)
Faza "C"	Analiza ilościowa modelu przy wykorzystaniu symulacji komputerowej
Zadanie "C1"	Analiza i synteza modelu na drodze intuicyjnych eksperymentów symulacyjnych
Zadanie "C2"	Zastosowanie algorytmów sterowania optymalnego
Zadanie "C3"	Stosowanie heurystycznych algorytmów optymalizacji

Źródło: Skrzypek, J.(1993).

Porównanie zawartości tablicy 1 oraz tablicy 2 jasno wyznacza zadania konstruktora modelu w poszczególnych fazach konstrukcji

strategii działania firmy.

3. Konstrukcja strategicznego planu przedsięwzięcia

3.1. Analiza strategiczna

Analiza strategiczna obejmuje:

- ocenę sygnałów płynących z otoczenia firmy (czynniki ekonomiczne, polityczne, technologiczne, etyczne i społeczne);
- ocenę zasobów będących w dyspozycji firmy (zasoby fizyczne, ludzkie, finansowe, niematerialne);
- identyfikację celu działania i oczekiwań właścicieli firmy.

Z punktu widzenia konstruktora modelu symulacyjnego analiza strategiczna przedsięwzięcia dostarcza informacji umożliwiających wykonanie zadań związanych z fazą "A" (opis badanego obiektu wraz z analizą jakościową) oraz fazą "B" (testowanie wiarygodności modelu).

Szczególną uwagę należy zwrócić na fakt, iż analiza zasobów oraz otoczenia firmy pozwala skonstruować warunki ograniczające modelu oraz dobrać jego parametry, natomiast cele i oczekiwania właścicieli narzucają postać funkcji kryterium oceny wyników eksperymentów z modelem.

Skonstruowany w ten sposób model powinien zostać poddany procedurze testowania jego wiarygodności[1], bowiem tylko model wiarygodny może stanowić narzędzie użyteczne w następnych fazach procesu zarządzania strategicznego.

3.2. Wybór strategii

3.2.1. Generowanie wariantów

W oparciu o strategie bazowe określane są alternatywne kierunki oraz metody realizacji strategii. Zadanie te wykonywane są zwykle na drodze generowania wariantów planu strategicznego. W rezultacie otrzymujemy serię założeń do kolejnych eksperymentów symulacyjnych.

Kolejny krok stanowi analiza ilościowa modelu przy wykorzystaniu symulacji komputerowej, polegająca na przeprowadzeniu serii eksperymentów symulacyjnych (w tym optymalizacji komputerowej). Wyniki tych eksperymentów pozwalają na wstępną selekcję wariantów, a do dalszych etapów postępowania zostają zakwalifikowane tylko te z nich, które zapewniają rozwiązania dopuszczalne[2].

3.2.2. Ocena i wybór wariantu

Spośród wariantów, które przejdą wstępną selekcję, należy wybrać do realizacji wariant optymalny. Podstawą oceny wariantów są wyniki eksperymentów symulacyjnych dla wybranych scenariuszy.

Wybrany wariant powinien zapewniać:

- wewnętrzną zgodność planu strategicznego,
- dostosowanie planu strategicznego do warunków panujących w otoczeniu;
- optymalną realizację oczekiwań właścicieli..

Jeśli konstruktor zbudował model, który uznany został za wiarygodny, to istnieje duża szansa, że otrzymane w jego wyniku warianty będą zapewniać:

- osiągnięcie wytyczonych celów;
- wykorzystanie możliwości oraz uniknięcie zagrożeń stwarzanych przez otoczenie firmy;

[1] Szczegółowy opis problemu można znaleźć w pracy (Skrzypek,1993).

[2] czyli te, które spełniają warunki ograniczające

- dostosowanie do zasobów stojących w dyspozycji firmy.

4. Wdrożenie

Odrębny problem stanowi wdrożenie strategii wybranej do realizacji. Do dalszych rozważań przyjmiemy założenie, że strategia wybrana do realizacji została przygotowana przy użyciu wiarygodnego modelu symulacyjnego, co stanowi warunek konieczny powodzenia procesu jego wdrażania.

W fazie wdrażania strategii musimy sobie zadać jednak następujące pytania:

- czy kadra menedżerska, mająca za zadanie wdrożenie strategii rozumie jej założenia?
- czy istnieje zgodność pomiędzy założeniami strategii a oczekiwaniami kadry menedżerskiej?
- czy istnieje zgodność pomiędzy założeniami strategii a oczekiwaniami pracowników firmy?
- czy kadra menedżerska posiada umiejętności niezbędne do skutecznego wdrożenia strategii?

Rozwiązanie powyższych problemów wymaga przekonania do proponowanych rozwiązań zarówno kadry menedżerskiej jak i załogi. Rola modelu symulacyjnego polega tu zwykle na:

- skutecznej pomocy w prezentacji założeń strategii oraz skutków jej wprowadzenia,
- umożliwieniu zdobycia umiejętności niezbędnych do skutecznego wdrożenia strategii.

Zdaniem autora niniejszego artykułu modele symulacyjne znajdują szczególne zastosowanie w przypadku ostatniego z wymienionych wyżej problemów[3]. Nieprzypadkowo centra szkoleniowe wielkich firm opierają swą działalność na wszelkiego typu modelach symulacyjnych i grach symulacyjnych.

5. Kontrola wykonania

Nawet sprawne wdrożenie planu strategicznego nie zwalnia kadry menedżerskiej od nieustannej kontroli jego wykonania. Trudno niedoceniać tu użyteczności modelu symulacyjnego, która nabiera szczególnego znaczenia w przypadku wystąpienia konieczności modyfikacji planu strategicznego, a więc powrotu do etapu generowania nowych wariantów.

6. Przykład ilustracyjny

6.1. Uwagi wstępne

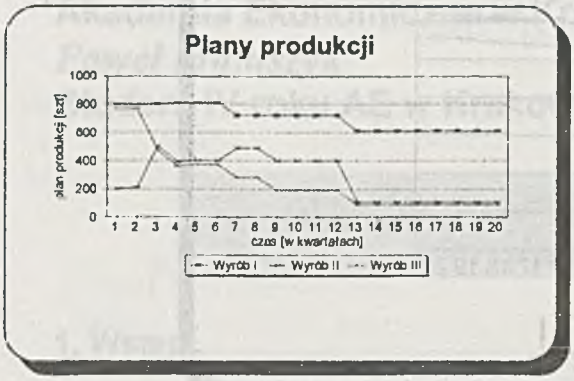
W celu ilustracji problemów omawianych w niniejszej pracy autor pragnie przytoczyć opis zastosowania symulacyjnego modelu JOTES, do konstrukcji planu strategicznego firmy.

Model JOTES został skonstruowany zgodnie z zaleceniami przedstawionymi w Tabeli 2 oraz przeszedł pomyślnie serię testów sprawdzających jego wiarygodność[4].
Następnie został on wykorzystany do skonstruowania strategicznego planu pewnej firmy. Zadanie to wykonano przeprowadzając serię eksperymentów optymalizacyjnych. Przebieg jednego z tych eksperymentów zostanie opisany w dalszej części niniejszego opracowania.

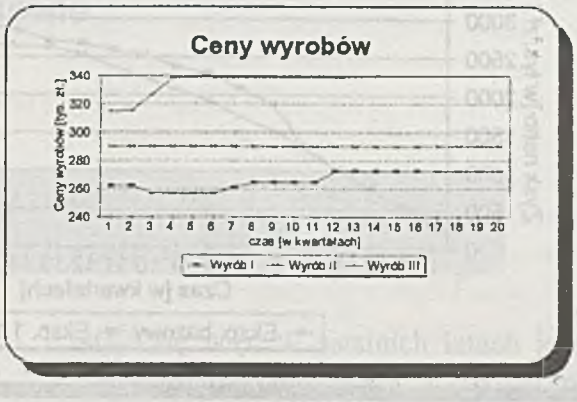
---

[3] Problematyka zastosowań modeli symulacyjnch do kształcenia menedżerów została omówiona w pracy (Skrzypek, Szubra,1992)

[4] Szczegóły związane z konstrukcją modelu JOTES można znaleźć w pracy (Skrzypek,1988)



Rys.1 Optymalne plany produkcji



Rys.2 Optymalne ceny wyrobów

6.2. Założenia eksperymentu

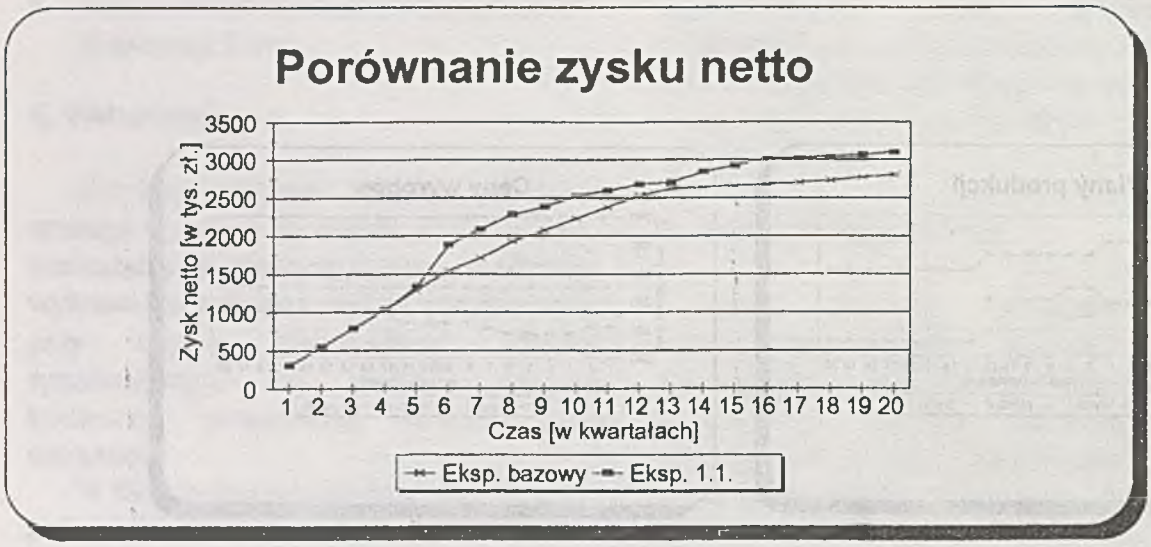
Eksperyment przeprowadzono na modelu JOTES, stosując zasadę optymalizacji wieloetapowej[5], ze zmiennym okresem podejmowania decyzji, który w pierwszym roku wynosił trzy miesiące, w drugim i trzecim sześć miesięcy, a w ostatnich dwóch latach dwanaście miesięcy, przy stałym, dwunastomiesięcznym horyzoncie planowania. Optymalizacji poddano sześć zmiennych decyzyjnych: a to plany produkcji trzech wyrobów (DIPRO1 - DIPRO3) oraz planowane ceny tych wyrobów (DCENT1 - DCENT3). Okres optymalizacji wynosił sześćdziesiąt miesięcy, a funkcja kryterium optymalizacji stanowiła wartość zakumulowanego zysku netto firmy. Cel eksperymentu określono jako optymalizację planów produkcji i cen wyrobów gotowych w długim okresie czasu (60 miesięcy), przy zastosowaniu algorytmu optymalizacji wieloetapowej.

6.3. Przebieg eksperymentu

Optymalne wartości zmiennych decyzyjnych osiągnięto po wykonaniu 412 iteracji. Wyniki eksperymentu przedstawiono na wykresach 1, 2 oraz 3.

Wnioski: w rezultacie zastosowania algorytmu optymalizacji wieloetapowej ze zmiennym okresem podejmowania decyzji, osiągnięto wzrost wartości zakumulowanego zysku netto o około 44,4% w skali 60 miesięcy, w stosunku do wyników uzyskanych w eksperymencie bazowym (optymalizacja jednoetapowa - por. wykres 3.). Poprawa wartości funkcji kryterium wystąpiła w tym przypadku w konsekwencji zastosowania o wiele krótszego (niż w poprzednich eksperymentach) okresu podejmowania decyzji. Umożliwiło to bardziej elastyczne dostosowywanie decyzji do zmiennej sytuacji w otoczeniu przedsiębiorstwa.

[5] Idea optymalizacji wieloetapowej wyjaśniona jest w pracy (Skrzypek,1993).



Rys.3.Porównanie zysku netto w dwóch eksperymentach.

7. Uwagi końcowe

Na koniec warto podkreślić, że w niniejszej pracy zaprezentowano rozważania dotyczące sposobów wykorzystania modeli symulacyjnych w procesie strategicznego zarządzania firmą. Autor próbował określić w jakich fazach tworzenia i wdrażania planu strategicznego firmy warto korzystać z modeli symulacyjnych. Rozważania ilustrowane są przykładem zastosowania modelu symulacyjnego firmy w procesie generowania i oceny wariantu strategii. Niestety ze względu na brak miejsca opis modelu JOTES oraz stosowanej metodologii eksperymentów optymalizacyjnych został znacznie ograniczony.

Literatura:

Olzacka, B., Pałczyńska-Gościniak,R.,(1994), Jak oceniać firmę?, *ODDK Gdańsk*;

Porter, M.,E., (1992), Strategia konkurencji, *PWE Warszawa;*

Skrzypek, J.(1988), Optymalizacja w modelach Dynamiki Systemowej, *Prace Naukowe AE we Wrocławiu;*

Skrzypek, J., Szubra, M., (1992), Symulacyjne gry decyzyjne jako narzędzie aktywnego kształcenia menedżerów, *Informatyka nr 9;*

Skrzypek, J., (1993), Metodologiczne aspekty modelowania procesów gospodarczych (w ujęciu Dynamiki Systemowej), *Materiały V Jubileuszowej Wyższej Górskiej Międzynarodowej Szkoły PTI,* PTI Katowice;

Wąsik, B., (1983), Elementy dynamiki systemowej dla ekonomistów, *AE Kraków*

E. Jacek Kwarciak
Akademia Górniczo - Hutnicza w Krakowie
Jerzy T. Skrzypek
Akademia Ekonomiczna w Krakowie
Paweł Matuszyk
Student IV roku AE w Krakowie

Symulacyjny model procesu oceny ryzyka kredytowego (ORKA)

1. Wstęp

Istnieją dziedziny w których swobodne eksperymentowanie na funkcjonującym organiźmie może być niebezpieczne, bardzo kosztowne lub po prostu niemożliwe. Jedną z tych dziedzin jest niewątpliwie sfera zarządzania procesem udzielania kredytów oraz kontrola przebiegu ich spłaty. Zdobywanie umiejętności oceny ryzyka związanego z udzieleniem kredytu poprzez bezpośrednie doświadczenie może bowiem doprowadzić do niepowodzeń o trudnych do przewidzenia konsekwencjach finansowych.

Należy wziąć przecież pod uwagę fakt, że sytuacja pracownika działu kredytowego w banku jest często dużo trudniejsza niż, na przykład, inżyniera, który zwykle ma do czynienia z obiektywnymi prawami. Kredytowiec natomiast, często napotyka na niejednolitą i niekompletną teorię, a nie istnieje przecież laboratorium, w którym mógłby wykonywać eksperymenty na badanym obiekcie. Ponadto struktura procesów gospodarczych jest zwykle zmienna z czasie, co wymaga odpowiednich, bieżąco dokonywanych, korekt podejmowanych decyzji. Kolejny problem stwarzają dane empiryczne, często niekompletne, natomiast obarczone trudnymi do oszacowania błędami pomiaru.

Znalezienie racjonalnego sposobu zarządzania procesem udzielania kredytów staje się więc w ostatnich latach jednym z najważniejszych problemów rozpatrywanych na gruncie teorii bankowości. Sposób taki powinien zapewnić przede wszystkim zgodność osiąganych rezultatów z uprzednio wyznaczonymi celami. W każdym przypadku oznacza to konieczność zdefiniowania celów działania kredytodawcy, a następnie przełożenia ich na konkretne decyzje. Należy przy tym brać pod uwagę aktualną sytuację ekonomiczną obiektu, stopień jego samodzielności a także przewidywane zmiany w otoczeniu. Jest rzeczą oczywistą, że wykorzystanie w tym celu wyłącznie możliwości "nieuzbrojonego" umysłu ludzkiego nie może przynieść pożądanych rezultatów. Poleganie na intuicji i doświadczeniu decydentów pozwala zwykle na ocenienie a w konsekwencji i wybór spośród najwyżej kilku wariantów postępowania, nie zapewniając przy tym uzyskania rozwiązania optymalnego. Tym bardziej, że w sytuacjach kryzysowych - zgodnie z prawem Rudina - *gdy trzeba decydować się na wybór między alternatywami większość decydentów wybiera zwykle rozwiązania najgorsze.*

Jedną z metod, które pozwalają rozwiązywać wspomniane problemy jest budowa i aplikacja modeli symulacyjnych. Metoda ta jest szczególnie skuteczna w przypadku, gdy wykorzystuje się ją do oceny skutków wielu potencjalnych wariantów

podejmowanych decyzji. Pozwala ona także na konstrukcję symulacyjnych gier decyzyjnych, które stanowią znakomite narzędzie aktywnego kształcenia i oceny decyzji podejmowanych przez pracowników.

W związku z powyższym, autorzy niniejszego artykułu podjęli próbę konstrukcji serii modeli symulacyjnych, mających za zadanie wspomaganie procesu szkolenia i oceny pracowników banku, ze szczególnym uwzględnieniem problemów oceny ryzyka kredytowego. W konsekwencji niniejsza praca dzieli się na dwie zasadnicze części. Pierwsza z nich zawiera scenariusz symulacyjnej gry decyzyjnej ORKA, (będącej w fazie realizacji komputerowej), stawiającej sobie za zadanie wspomaganie procesu szkolenia pracowników działów kredytowych banków w zakresie oceny ryzyka kredytowego. Zrealizowanie powyższego zadania, umożliwi - w drugiej fazie - przygotowanie **komputerowego systemu wspomagania procesu podejmowania decyzji kredytowych.**

Ocena płynności finansowej firmy jest kluczowym problemem przy szacowaniu szans spłaty kredytu zaciągniętego przez przedsiębiorstwo. Rodzi to konieczność doskonalenia umiejętności sporządzania przez pracowników banku zestawień przepływów pieniężnych. Druga część opracowania poświęcona została więc prezentacji trzech modeli (PIRANIA, DELFIN, AUTOMAT), które mogą przyczynić się do lepszego przyswajania metodyki sporządzania zestawień przepływów pieniężnych.

Model "DELFIN" stanowi narzędzie dydaktyczne, wspomagające proces edukacji w zakresie sporządzania zestawień przepływów pieniężnych. Z kolei model "PIRANIA" jest przeznaczony do testowania stopnia umiejętności sporządzania tego typu zestawień przez pracowników działów kredytowych. Przy okazji powstał również model "AUTOMAT", który pozwala na automatyczne - oczywiście po wprowadzeniu danych wejściowych - sporządzanie sprawozdań z przepływu środków pieniężnych.

2. Założenia scenariusza gry symulacyjnej ORKA

2.1. Elementy gry decyzyjnej

Proponowana przez nas gra decyzyjna "ORKA" składać się będzie z czterech, nierozerwalnie powiązanych między sobą, elementów:

ZESTAWU MODELI SYMULACYJNYCH
OPISÓW OCENIANYCH FIRM
OPISÓW SYTUACJI DECYZYJNYCH
ZBIORU ZACHOWAŃ UCZESTNIKÓW GRY

2.2 Zestaw modeli symulacyjnych

Elementem niewidocznym dla uczestnika gry będzie zestaw modeli matematycznych opisujących typowe firmy i ich otoczenie. Autorzy scenariusza mają tu doświadczenia (uprzednio zrealizowane gry KRAEK oraz TEES). Modele matematyczne zostaną odpowiednio sparametryzowane, tak, że jeden model będzie mógł służyć jako punkt wyjścia dla wielu modeli symulacyjnych, różniących się od siebie parametrami.

Przewidujemy stopniową budowę modeli dwóch rodzajów firm:

a) firmy duże (przedsiębiorstwa państwowe, spółki akcyjne, spółki z o.o.);
b) firmy reprezentujące "small business". Podstawowym zadaniem spełnianym przez modele będzie dostarczanie reakcji na działania uczestników gry (pracowników działów kredytowych).

2.3 Opis ocenianych firm

2.3.1 Informacje przygotowane przez przedsiębiorstwo

Uczestnik powinien każdorazowo otrzymać wniosek kredytowy oraz opis ocenianej firmy, w formie typowego planu działania firmy (biznes plan).

Uczestnik otrzymywałby również:

- Bilanse firmy za trzy lata;
- Rachunek wyników firmy za trzy lata;
- Zestawienie przepływów pieniężnych firmy za trzy lata;
- Informacje o sytuacji rynkowej w formie komunikatów, typowych raportów oraz analiz.

2.4. Opis sytuacji decyzyjnych

Przewidujemy cztery podstawowe warianty sytuacji decyzyjnych dla dwóch różnych sposobów oceny zachowań uczestników gry:

A. PORÓWNANIE

WARIANT I:

Grupa graczy otrzymuje serię planów działania (np. 10) różnego typu firm (w formie powyżej opisanej) wraz z propozycjami warunków udzielania kredytu, zgłoszonymi przez kredytobiorcę. Na tej podstawie zostaje podjęta decyzja o udzieleniu (bądź nie) kredytu. Następnie zostaje uruchomiony model symulacyjny firm na czas odpowiadający okresowi spłaty kredytu.

WARIANT II;

Różni się od poprzedniego wariantu możliwością ingerencji pracownika działu kredytowego w bieżącą działalność firmy (instrumenty poprawiające płynność finansową firmy oraz renegocjacje kredytu).

B. KONKURENCJA

WARIANT III:

Jest to rodzaj "przetargu" o kredyt. Pracownicy działów kredytowych, dzielą się na grupy reprezentujące konkurujące ze sobą banki. Grupy te po analizie wniosków kredytowych proponują warunki na jakich zostanie udzielony kredyt. Kredytobiorca wybierze bank, który zaproponuje najkorzystniejsze warunki.

WARIANT IV:

Grupy konkurujących ze sobą banków mogą w czasie symulacji dokonywać ingerencji w zachowanie przedsiębiorstwa jak w WARIANCIE II.

3. Program "DELFIN"

3.1. Charakterystyka programu "DELFIN"

Przedstawiony poniżej system "DELFIN", może przyczynić się do lepszego przyswajania metodyki sporządzania zestawień przepływów pieniężnych. Autorzy systemu postawili sobie bowiem za cel stworzenie narzędzia dydaktycznego, które będzie mogło wspomagać proces edukacji jako swego rodzaju podręcznik sterowany.

"DELFIN" jest programem umożliwiającym interakcyjną edukację w zakresie sporządzania zestawień przepływów pieniężnych (cash flow) w przedsiębiorstwach. Dostarcza on gotowych zestawów danych finansowych (za trzy kolejne lata) w postaci:

- BILANSU w wersji urzędowej (arkusz F02);
- RACHUNKU WYNIKÓW w wersji urzędowej (arkusz F01);
- BILANSU w wersji publikacyjnej; RACHUNKU WYNIKÓW w wersji publikacyjnej.

Zadaniem użytkownika jest wypełnienie arkusza zawierającego znormalizowany wzorzec zestawienia CASH FLOW (obowiązujący w Banku Przemysłowo-Handlowym) na podstawie wyżej wymienionych dokumentów.

Program udostępnia rozbudowany system

pomocy kontekstowej dostarczający informacji dotyczących tródła danych niezbędnych do wypełnienia dowolnie wskazanej pozycji w wypełnianym arkuszu (podawany jest sposób jej naliczania). Po wypełnieniu całego zestawienia CASH FLOW możliwe jest zweryfikowanie jego poprawności. Procedura sprawdzająca porównuje wprowadzone wartości z wartościami wzorcowymi (wyliczonymi automatycznie przez system na podstawie zadanych reguł), a w przypadku wykrycia błędnie wypełnionej pozycji wyświetlany jest komunikat o błędzie wraz z informacją o poprawnym sposobie wyliczenia danej kategorii. Jeżeli arkusz wypełniony jest poprawnie na ekranie pojawia się stosowna informacja.

Dla ułatwienia obsługi program zawiera procedury umożliwiające wykorzystanie zarówno myszki (do poruszania się po arkuszach jak i do przenoszenia danych z dokumentów finansowych do CASH FLOW) jak i (w przypadku jej braku) klawiatury.

4. Uwagi końcowe

Autorzy niniejszych programów mają nadzieję, że pomogą one wszystkim użytkownikom opanować umiejętność sporządzania zestawień przepływów pieniężnych. W przygotowaniu znajdują się już kolejne wersje programu:

- PIRANIA (testowanie umiejętności sporządzania zestawień przepływów pieniężnych),
- AUTOMAT (automatyczne sporządzanie zestawień przepływów pieniężnych),

Literatura:

Olzacka, B., Pałczyńska-Gościniak,R.,(1994), Jak oceniać firmę?, *ODDK Gdańsk*;

Skrzypek, J., (1993), Metodologiczne aspekty modelowania procesów gospodarczych (w ujęciu Dynamiki Systemowej), *Materiały V Jubileuszowej Wyższej Górskiej Międzynarodowej Szkoły PTI*, PTI Katowice;

Skrzypek, J., Szubra, M., (1992), Symulacyjne gry decyzyjne jako narzędzie aktywnego kształcenia menedżerów, *Informatyka nr 9;*

Wąsik, B., (1988), Komputerowa gra kierownicza KRAEK-1, *Skrypt AE Kraków.*

Marek Miłosz
Politechnika Lubelska

Dyskretna symulacja systemów informatycznych

1. Wstęp

O sprawności i efektywności systemów informatycznych (SI), a zwłaszcza tak skomplikowanych jak lokalne sieci komputerowe, czy też przetwarzanie w architekturze klient-serwer, decyduje wiele różnorodnych czynników. Są to zarówno parametry techniczne poszczególnych elementów systemu (serwera, stacji roboczych, kart sieciowych, osprzętu) i jego topologia, jak też realizowane w systemie algorytmy wymiany informacji. Integralnym elementem SI jest także użytkownik i jego oprogramowanie aplikacyjne. Połączenie dużej liczby czynników natury technicznej, informacyjnej i psychologicznej utrudnia rozwiązanie problemu projektowania i oceny skuteczności SI [4].

Problemy takie powstają na każdym z tych etapów życia SI, na których są ustalane jego parametry, począwszy od projektowania, poprzez strojenie, eksploatację i rozbudowę [3]. Praktycznie każdy system posiada "wąskie gardła", które decydują o jego efektywności. Są to jednocześnie miejsca, które powinny być w pierwszej kolejności rozpatrywane przy rozwiązywaniu problemu podniesienia sprawności systemu.

Ocena efektywności istniejącego SI może być dokonywana przy pomocy monitorowania jego pracy w normalnych (lub też ekstremalnych) warunkach. Taki pasywny (bądź aktywny) eksperyment nie jest możliwy, z wielu względów, dla nowoprojektowanych, a nawet tylko rozbudowywanych systemów.

Jedną z technik, umożliwiających ocenę a priori wydajności i efektywności SI, jest budowa i badanie ich modeli symulacyjnych. Natura procesów zachodzących w SI (dyskretność i losowość) implikuje ich dyskretną i stochastyczną symulację.

2. Problemy oceny wydajności systemów informatycznych.

Rys. 1. Szczegółowość analizy efektywności systemów informatycznych

Efektywność SI posiada różne miary, które zależą od celów, stopnia szczegółowości i horyzontu czasowego badań.

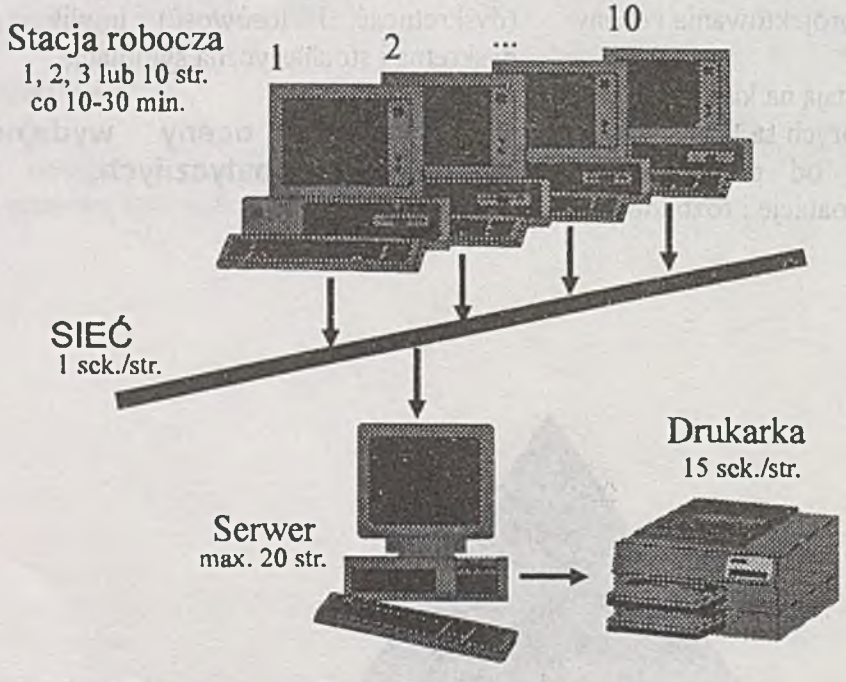
Analiza niezawodności systemów (rys.1) zakłada badania odporności SI na uszkodzenia i awarie jego elementów. Na tym poziomie określa się współczynnik gotowości, stopę błędów, średnie i całkowite czasy sprawności i niesprawności SI. W analizie tej operuje się bardzo długim horyzontem czasowym (tygodniami i latami) i wysokim poziomem agregacji zdarzeń.

Najbardziej naturalnym dla użytkownika jest poziom badania efektywności SI, związany z analizą wykorzystania zasobów (przeważenie współdzielonych). Mieszczą się tu takie pojęcia jak: czas reakcji systemu, czas realizacji transakcji lub całego zadania, przepustowość ogólna systemu, współczynnik wieloprogramowości itp. Miary powyższe wyznacza się dla średnich czasów pracy SI (np. godziny, dni) i średniego poziomu agregacji zdarzeń.

Najbardziej szczegółowe i związane ze sprzętem dane uzyskuje się podczas analizy pracy SI na najniższym poziomie - realizacji konkretnych protokołów i algorytmów przesyłania danych. Uzyskiwane są następujące parametry: obciążenie poszczególnych elementów SI, przepustowość chwilowa, ilość błędów, kolizji czy też przesyłanych paczek, średnie czasy potwierdzenia itp.

Symulacyjne stochastyczne modele SI powinny się charakteryzować modułowością budowy, możliwością symulacji procesów równoległych, różnych algorytmów i strategii przydziału zasobów systemu. Wymaganiom tym odpowiadają modele budowane w języku GPSS/PC [2].

3. Przykład: drukowanie wlokalnejsieci mikrokomputerowej.



Rys. 2. Schemat współpracy terminali z drukarką sieciową.

W lokalnej sieci komputerowej pracuje jednocześnie 10 stacji roboczych (terminali) - rys. 2. Każda z nich generuje średnio (rozkład równomierny) co 20±10 minut zadanie,

W lokalnej sieci komputerowej pracuje jednocześnie 10 stacji roboczych (terminali) - rys. 2. Każda z nich generuje średnio (rozkład równomierny) co 20±10 minut zadanie, polegające na drukowaniu na drukarce sieciowej pewnej liczby stron. Liczba stron do wydruku jest różna. Zaobserwowano, że użytkownicy drukują z równym prawdopodobieństwem po 1, 2, 3 lub 10 stron w ramach pojedynczego zadania. Zadania przesyłane są po lokalnej sieci

```
10 *       Przykład: współpraca stacje robocze - drukarka
20 *
30 *   1 j.cz.modelowego = 0.1 sek
40 *                EXPONENTIAL DISTRIBUTION
50 *
60 XPDIS   FUNCTION    RN3,C24          ;Exponential distribution functi
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
70 *
80 L_STRON  FUNCTION    RN1,D4           ;liczba stron wydruku
0.25,1/.5,2/.75,3/1,10
90 *
100 PRZES_STR FVARIABLE  P1#10#FN$XPDIS     ;czas przesylania P1 stron
110 BUFOR   STORAGE     20               ;pojemnosc bufora serwera druku (str)
120 CZAS_R  QTABLE      KOL,0,500,30     ;histogram czasu oczekiwania
130 *
140        GENERATE     ,,,10            ;liczba stacji w sieci
150 POCZ   ADVANCE      12000,6000       ;generowanie zadan-wydrukow
160        ASSIGN       1,FN$L_STRON     ,P1=liczba stron wydruku w zadaniu
170        SPLIT        1,POCZ           ;ponowne generowanie zadan
180        QUEUE        KOL              ,wyslanie zadania
190        SEIZE        SIEC             ;zajecie sieci
200        ADVANCE      V$PRZES_STR      ;czas przesylania zadania - P1 str.
210        ENTER        BUFOR,P1         ;zajecie bufora serwera druku
220        RELEASE .    SIEC             ;zwolnienie sieci - koniec przesyl
230        SEIZE        DRUKARKA         ;rozpoczecie druku
240 POCZ_DR ADVANCE     150,FN$XPDIS     ;czas druku 1 strony
250        LEAVE        BUFOR,1          ;zwolnienie 1 str. z bufora
260        LOOP         1,POCZ_DR        ;drukuj nastepna strone
270        RELEASE      DRUKARKA         ;zwolnienie drukarki
280        DEPART       KOL              ;koniec realizacji zadania
290        TERMINATE    1                ,zakonczenie zadania
```

Wydruk 1. Model współpracy terminali z drukarka sieciową

transmisji danych do komputera zarządzającego drukowaniem - serwera wydruku. Średnia szybkość transmisji jest proporcjonalna do liczby przesłanych stron i

wynosi 1 sek. na 1 stronę wydruku. Szybkość transmisji całego zadania ma charakter losowy o rozkładzie wykładniczym.

Zadania zgromadzone są w serwerze wydruku (pojemność bufora: 20 stron) i w miarę możliwości drukowane na drukarce. Czas wydruku pojedynczej strony jest losowy o rozkładzie wykładniczym i wartości średniej równej 15 sekund (4 strony /min).
Należy zbadać efektywność wykorzystania drukarki i parametry czasu oczekiwania na wydruk.

Model symulacyjny (wydruk 1) jest oczywiście bardzo uproszczonym modelem działania sieci komputerowej. Jednak dla postawionych w zadaniu celów badań - analizy stopnia wykorzystania drukarki - może okazać się wystarczająco dokładnym.

Rezultaty badań symulacyjnych wskazują na średnie (52%) obciążenie drukarki przy stosunkowo długim (rys. 3) czasie oczekiwania na wydruk (średnio: 132 sek, ale maksymalnie nawet 700 sek.). Zmieniając liczbę stacji roboczych w modelu systemu (wydruk 1, linia nr 140) można wyznaczyć ich liczbę, przekroczenie której znacznie pogarsza efektywność systemu (rys. 4).

4. Symulator sieci Novell Netware.

Opracowany w Katedrze Informatyki Politechniki Lubelskiej [1], symulator sieci Novell Netware modeluje pracę lokalnej sieci na poziomie protokołów wymiany danych. Odwzorowuje on trzecią - sieciową warstwę modelu OSI, realizując protokoły NCP, IPX i SPX.

Symulator (program w języku GPSS/PC, ok. 3000 linii, ponad 200 bloków) składa się z następujących segmentów - modułów:

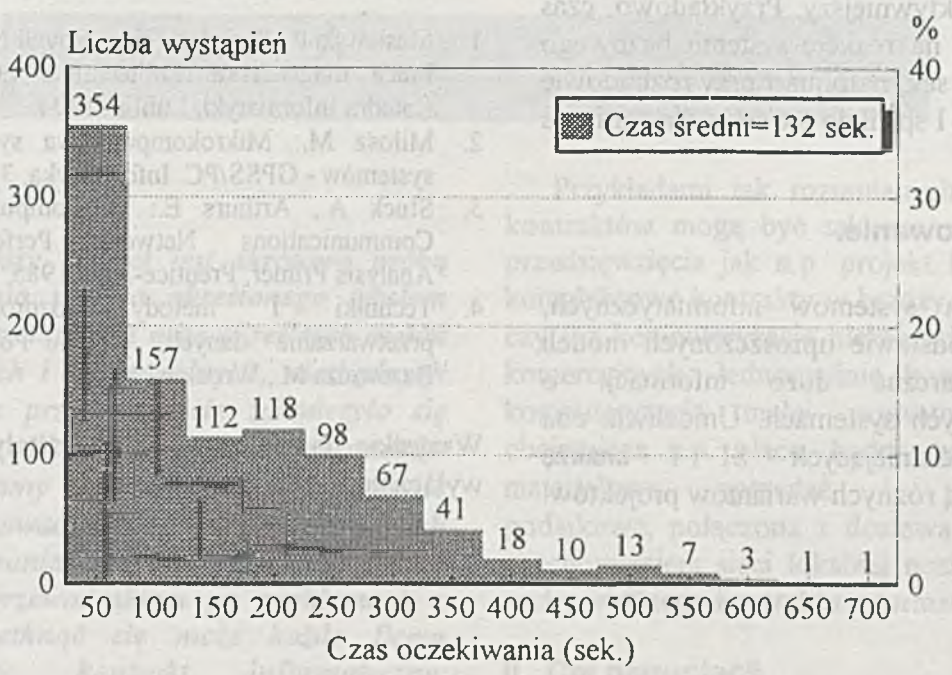
1) UŻYTKOWNIK - generuje zadania, dokonuje jego podziału na paczki;
2) STACJA - realizuje algorytmy rywalizacji o dostęp do kanału transferu danych i odbioru paczek, z różnymi algorytmami potwierdzenia, zbierania itp.
3) SERWER - modeluje procesy zachodzące w serwerze sieciowym.

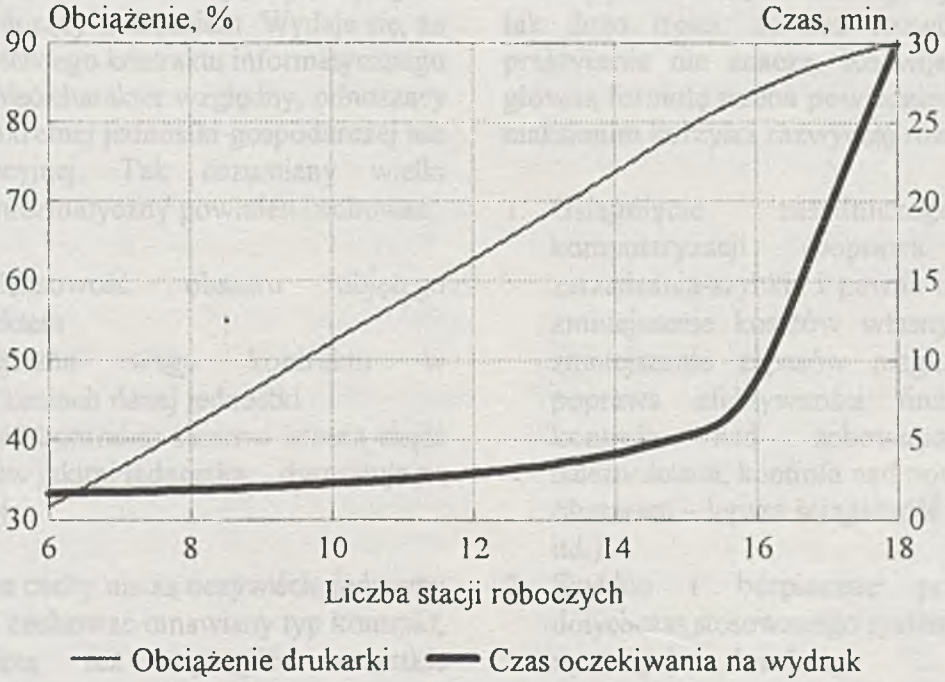
Parametry wejściowe symulatora opisują: konfigurację sieci (liczba stacji), parametry jej elementów (czas dostępu do dysku i pamięci, szybkość transmisji, wielkość pamięci serwera, liczba i wielkość buforów itp.), parametry medium transmisyjnego (prędkość propagacji) i nastrojenie protokołów (wielkość przesyłanych paczek). Krok czasu modelowego odpowiada jednej mikrosekundzie.

Rezulatatami modelowania sieci Novell Netware są szczegółowe dane, dotyczące ruchu paczek w sieci (liczba paczek - sumaryczna i z podziałem na stacje, liczba zadań, ilość transmitowanej informacji i szybkość przesyłu, liczba kolizji, błędów itp), obciążenia elementów systemu (kanału transmisji, kontrolera dysku twardego serwera, buforów itp.). Możliwa jest także rejestracja danych stastycznych, opisujących czas realizacji zadania, reakcji systemu itp.

Symulator sieci zweryfikowano przy pomocy aktywnego eksperymentu, w czasie którego stacje robocze realizowały ściśle określone zadania, a cała sieć była monitorowana przy pomocy LANanalysera. Model symulacyjny odwzorowuje rzeczywistość z błędem mniejszym niż 10%. Symulacja 2 minut pracy prostej sieci (5 mikrokomputerów) na mikrokomputerze typu 486 DX 33MHz wymagała ponad dwóch godzin obliczeń.

Symulator sieci Novell Netware wykorzystano do analizy dwóch wariantów rozbudowy sieci w biurze maklerskim: 1) wymiana dysku twardego na nowy (parametry starego: 26 msek., 2,5Mb/s, nowego: 10 msek, 7,5 Mb/s); 2) zwiększenie z 4 do 8MB pamięci operacyjnej w serwerze. Rezultaty


Liczba wystąpień
%
Czas średni=132 sek.
400
300
200
100
0
40
30
20
10
0
354
157
112
118
98
67
41
18
10
13
7
3
1
1
50 100 150 200 250 300 350 400 450 500 550 600 650 700
Czas oczekiwania (sek.)


Obciążenie, %
Czas, min.
90
80
70
60
50
40
30
30
25
20
15
10
5
0
6
8
10
12
14
16
18
Liczba stacji roboczych
Obciążenie drukarki
Czas oczekiwania na wydruk

obliczeń wskazują, że pierwszy wariant jest of ok. 60% efektywniejszy. Przykładowo, czas oczekiwania na reakcję systemu bazowego wynosił 1,.6 sek, natomiast przy rozbudowie w/g wariantu 1 spadł do 0,6 sek a dla wariantu 2 - 0,8 sek.

5. Podsumowanie.

Symulacja systemów informatycznych, nawet na podstawie uproszczonych modeli może dostarczać dużo informacji o projektowanych systemach. Umożliwia ona badanie nieistniejących SI i analizę porównawczą różnych wariantów projektów.

Literatura

1. Masłowski P.: Symulator sieci Novell Netware. Praca magisterska. Politechnika Lubelska, Katedra Informatyki, Lublin, 1994.
2. Miłosz M.: Mikrokomputerowa symulacja systemów - GPSS/PC. Informatyka, 3, 1993.
3. Stuck A., Arthurs E.: A Computer and Communications Network Performance Analysis Primer. Prentice-Hall, 1985.
4. Techniki i metody rozproszonego przetwarzania danych, cz. I. Pod. red. Bazewicza M., Wrocław, 1986.

Wszystkie nazwy zastrzeżone zostały użyte wyłącznie w celu identyfikacji

Marek Ujejski
Bank Śląski S.A. w Katowicach

Negocjacje wielkich kontraktów informatycznych

I. Wstęp

Niniejszy referat jest skrótową próbą przybliżenia pojęcia określonego tytułem mojego wystąpienia oraz określenia metod formalnych i nieformalnych, niezbędnych aby dane przedsięwzięcie zakończyło się sukcesem. Określone przez organizatorów Szkoły ramy czasowe pozwalają jedynie zasygnalizować pewne problemy i metody ich rozwiązywania,co czyni niniejszą pracę jedynie przewodnikiem po problemach z jakimi zetknąć się może każda firma negocjując kontrakt informatyczny, niekoniecznie wielki.

Jednoznaczna definicja wielkiego kontraktu informatycznego nie istnieje. To co dla jednej firmy jest wielkim i strategicznym przedsięwzięciem dla innych jest jedynie niezbyt znaczącym zadaniem. Wydaje się, że definicja wielkiego kontraktu informatycznego powinna mieć charakter względny, odnoszący się do konkretnej jednostki gospodarczej lub administracyjnej. Tak rozumiany wielki kontrakt informatyczny powinien cechować:

1. Kompleksowość obszaru objętego kontraktem
2. Strategiczna waga kontraktu w zamierzeniach danej jednostki
3. Wartość kontraktu stanowi istotną część środków jakimi jednostka dysponuje na rozwój

Podane cechy nie są oczywiście jedynymi jakie musi cechować omawiany typ kontrakt, nie muszą też wystąpić wszystkie jednocześnie.

Przykładami tak rozumianych wielkich kontraktów mogą być tak znane w kraju przedsięwzięcia jak n.p. projekt POLTAX, kompleksowe kontrakty w branży górniczej, czy też komputeryzacje niektórych banków komercyjnych. Jednocześnie kompleksowa komputeryzacja małej prywatnej firmy obejmująca n.p. płace, kadry, gospodarkę materiałową, sprzedaż i rozliczenia podatkowe, połączona z dostawą sprzętu i uruchomieniem sieci lokalnej nosi wszelkie cechy wielkiego kontraktu informatycznego.

II. Cel negocjacji

Cel negocjacji wielkiego kontraktu informatycznego jest niezwykle prosty:

Osiągnąć maksimum korzyści przy minimalnych kosztach

Niestety jasne i zwięzłe formuły kryją w sobie tak dużo treści, że bez rozwinięcia nic praktycznie nie znaczą. Rozwijając zatem główną formułę trzeba powiedzieć, że przez maksimum korzyści zazwyczaj rozumiemy:

1. Osiągnięcie zasadniczego celu komputeryzacji (poprawa precyzji zarządzania-szybkie i pewne informacje, zmniejszenie kosztów własnych - n.p. zmniejszenie zapasów magazynowych, poprawa efektywności finansowej - kontrola nad zobowiązaniami i należnościami, kontrola nad powierzonym obszarem - lepsza ściągalność podatków itd.)
2. Szybkie i bezpieczne przejście z dotychczas stosowanego systemu (także ręcznego) na docelowy
3. Podniesienie wydajności pracy przez

wyzwolenie ukrytych rezerw, widocznych dopiero po wprowadzeniu systemu informatycznego
4. Zintegrowanie załogi wokół wspólnego zadania. Często takim zadaniem jest właśnie kompleksowa komputeryzacja

Mówiąc o minimalizacji kosztów zazwyczaj mamy na myśli:

1. Czas niezbędny do pełnego wdrożenia systemu. Parametr ten jest zazwyczaj szacowany przez obie strony zbyt optymistycznie! Należy zawsze jego ocenie poświęcić szczególną uwagę.
2. Cena kontraktu. Ten parametr zawsze jest doceniany przez obie strony. Ponieważ jest wymierny i ma łatwo weryfikowalny wymiar główne rozmowy negocjacyjne toczą się wokół niego. Często kierownictwo firmy ocenia wyniki negocjacji jedynie na podstawie tego parametru.
3. Ryzyko w wypadku niepowodzenia całego przedsięwzięcia lub jego znaczącej części. Duże opóźnienie w realizacji kontraktu. Upadek lub poważne kłopoty firmy, z którą zamierzamy zawrzeć kontrakt. Możliwe problemy finansowe własnej firmy uniemożliwiające realizację przyszłych płatności.

Autor namawia do równorzędnego traktowania wszystkich wymienionych czynników. Przyszły negocjator powinien starannie zapoznać się ze strategicznymi planami kierownictwa, ocenić stan finansowy firmy, w imieniu której będzie prowadził negocjacje i jej przyszłe możliwości, stan finansowy firm, z którymi zamierza się zawrzeć kontrakt, a na końcu przedyskutować z kierownictwem swoje wątpliwości i wytyczne jakie otrzymał. Dobrym obyczjem powinno być przygotowanie analizy planowanej inwestycji metodą NPV i IRR, niezależnie od tego czy zamierzamy starać się o kredyt bankowy czy też prowadzimy inwestycję ze środków własnych. Często obserwuje się, zwłaszcza w Polsce, że szefowie firm bądź wysokiej rangi członkowie kierownictwa samodzielnie próbują negocjować czy wręcz zawierać kontrakty.

Jest to ciężki grzech negocjatora !
Osoba ta, sama zamyka sobie pole manewru na wypadek zmiany założeń czy kierunku negocjacji. Wysokiej rangi członkowie kierownictwa powinni pozostać w odwodzie i do nich należy słowo ostateczne.

III. Cel renegocjacji

Renegocjacja kontraktu kryje w sobie zazwyczaj smutną prawdę o załamaniu się istotnych elementów już zawartego kontraktu. Powoduje to często konieczność zmiany warunków uprzednio podpisanych a obecnie nieaktualnych lub bardzo uciążliwych dla jednej lub obu stron. Bardzo często jest to wynik kłopotów płatniczych beneficjenta kontraktu, powstałych wskutek pogorszenia się jego kondycji finansowej lub nieuzyskania kolejnej transzy kredytu na dalsze finansowanie przedsięwzięcia. Również nieosiągnięcie zaplanowanych etapów kontraktu przez jego realizatora (zazwyczaj kłopoty z implementacją oprogramowania) są przyczyną konieczności renegocjacji wcześniej ustalonych warunków.

Ogólnie można stwierdzić, że przyczyny konieczności renegocjacji mogą leżeć zarówno po jednej jak i obu stronach kontraktowych. Podobnie prowadzeniem renegocjacji może być zainteresowana jedna lub obie strony kontraktu. Nie zawsze jest to gra o sumie zerowej (korzyści i straty mogą nie układać się symetrycznie).

Przystępując do renegocjacji trzeba mieć precyzyjnie określony cel, to jest wiedzieć co w wyniku rozmów i ew. działań chcemy osiągnąć. Takim celem może być:

- uzyskanie korzystniejszych warunków

płatności
- zmiana zakresu kontraktu
- zmiana terminów określonych w kontrakcie
- uzyskanie moratorium na sankcje prawne określone w kontrakcie
- kombinacje wymienionych

IV. Debiut negocjacji

Prowadzenie negocjacji bardzo przypomina partię szachów. Wiele zależy od pierwszego ruchu, zgrabnej kombinatoryki analizy wariantów w zaawansowanych częściach negocjacji oraz zgrabnego i finezyjnego zakończenia. Mamy więc tu zarówno debiut, grę środkową, jak i końcówkę, jaką jest podpisanie kontraktu. To uzasadnia tytuł tego i następnych rozdziałów.

IV.1. Wymagania użytkownika

Przystępując do negocjacji kontraktu trzeba dysponować kompletem materiałów określających podstawowe wymagania użytkownika. W przeciwnym razie nie będziemy w stanie drugiej stronie postawić jasno naszych wymagań.

Dokumentem takim jest zazwyczaj szczegółowy opis wymagań funkcjonalnych stawianych systemowi, sporządzony samodzielnie lub z pomocą specjalistów i tytułowany zazwyczaj "WYMAGANIA UŻYTKOWNIKA" (USER REQUIREMENTS).

Szczególnie ważne jest to w przypadku tak zwanego kontraktu zintegrowanego, obejmującego zarówno dostawę sprzętu jak i uruchomienie aplikacji oraz szkolenie. Brak fundamentalnych założeń ze strony użytkownika może być bezlitośnie wykorzystany przez drugą firmę, zwłaszcza jeśli uruchomienie kontraktu poprzedzają znaczne zaliczki lub transze płatności wyprzedzają fazy realizacyjne.

Autor zna co najmniej dwa przykłady wielkich kontraktów informatycznych zawartych w Polsce bez odpowiedniego określenia wymogów użytkownika. W obu przypadkach spowodowało to znaczne kłopoty dla wszystkich stron kontraktu.

Negocjator przystępując do rozmów ma prawo i obowiązek domagać się takiego dokumentu.

IV.2. Nadzór nad kontraktem i projektem informatyzacji

Dla prawidłowego zrealizowania wielkiego kontraktu informatycznego zaleca się powołanie ciała formalnego złożonego z wysoko umocowanych przedstawicieli kierownictwa firmy (członkowie Zarządu spółki, dyrekcja firmy, dyrektorzy Departamentów itd.) Ciało to nie powiela już występujących w firmie struktur, lecz tworzy nowe, zadaniowo zorientowane na realizację projektu. Wskazane jest powołanie takiego ciała przed rozpoczęciem negocjacji, gdyż ma ono wpływ na ich kształt i przebieg. Daje ono wytyczne, kontroluje i rozlicza Zespół Negocjacyjny.

Ciało takie nosi zazwyczaj zapożyczoną z języka angielskiego nazwę Komitetu Sterującego.

Jedną z ważniejszych decyzji jakie należy podjąć przed rozpoczęciem WKI (wielkiego kontraktu informatycznego) jest decyzja o zatrudnieniu konsultanta lub firmy konsultingowej do poszczególnych etapów prac. Umowa taka może obejmować:

- przygotowanie wspólnie z użytkownikiem dokumentu USER REQUIREMENTS
- przygotowanie wytycznych do negocjacji
- prowadzenie negocjacji
- prowadzenie i nadzór nad projektem

Należy zawsze starannie rozważyć celowość zatrudnienia firmy konsultingowej do poszczególnych etapów lub całości przedsięwzięcia.

Autor zaleca raczej metodę zadaniową

rozliczaną po jego wykonaniu lub w określonym czasie.

IV.3. Zapytanie ofertowe

Punktem zerowym do rozpoczęcia negocjacji jest wystosowanie zapytania ofertowego. Jest on tym czym licytacja w bridge. Po jego konstrukcji druga strona może zorientować się z siłą i znajomością rzeczy partnera. Profesjonalnie przygotowane zapytanie ofertowe eliminuje na wstępie słabe lub niewiarygodne firmy i sygnalizuje konieczność poważnego przygotowania oferty. Zapytanie ofertowe powinno pozwolić na uzyskanie jak największej ilości informacji o rozwiązaniu proponowanym przez firmę i warunkach realizacji ew. kontraktu. Jednocześnie powinno pozostawić margines negocjacyjny dla własnych zobowiązań. Nie zaleca się stawiania sztywnych warunków w kontraktach WKI, gdyż wiele elementów jest tu niezwykle trudnych do określenia. Nawet cena ostateczna kontraktu może być ustalona po przeprowadzeniu całości postępowania ofertowego. Wybór formy przetargu dokonuje zazwyczaj kierownictwo firmy, często po konsultacjach. Przetargi mogą mieć charakter ograniczony (skierowane do określonej grupy firm), nieograniczony (każda firma spełniająca publicznie podane warunki może brać udział),różnie mogą być zdefiniowane fazy kontraktu i warunki przejścia pomiędzy nimi. Niektóre przetargi mają ściśle określony tryb i zasady rozgrywania. Należą do nich n.p. przetargi realizowane według zasad Banku Światowego lub przetargi na rzecz administracji rządowej.

Często spotyka się formę dwustopniową, złożoną z części pisemnej i rozmów indywidualnych z wszystkimi lub częścią oferentów.
Również skala czasu jest trudna do określenia. Przetarg może być rozegrany w ciągu jednego dnia (nie dotyczy to oczywiście WKI !) lub w trakcie wielu lat. Ten ostatni okres jest n.p. typowy dla przetargów organizowanych według reguł Banku Światowego.

Głównym zadaniem negocjatora w tej fazie jest dokładne opanowanie określonych reguł danego przetargu, zrozumienie celu głównego planowanego kontraktu oraz przygotowanie wielu wariantów rozmów negocjacyjnych.

V. Gra środkowa

Autor starał się tutaj przybliżyć zasadnicze problemy głównej fazy negocjacji, określić pewne wymagania jakie trzeba postawić negocjatorom oraz zasady których powinno się przestrzegać w tej fazie.

V.1 Własna negocjacja kontraktu

Jeśli decydujemy się na prowadzenie negocjacji własnymi siłami powinniśmy być przekonani że podołamy temu zadaniu. Zadanie takie można powierzyć jedynie osobie, która ma za sobą już praktyczny udział w tego typu rozmowach. Jeżeli wyznaczamy zespół w jego składzie musi być co najmniej jedna osoba z takim doświadczeniem. Pozostałe osoby mogą brać udział z ograniczonym prawem głosu, co powinna regulować stosowna instrukcja omówiona przez głównego negocjatora. Dobrze jest określić role członków zespołu przydzielając im konkretne zadania. Kierujemy się przy podziale zadań doświadczeniem zawodowym poszczególnych osób.
I tak specjalista od oprogramowania systemowego nie powinien zabierać głosu w sprawach finansowych kontraktu, a sprzętowiec dyskutować cechy funkcjonalne aplikacji. Są to sprawy, o których w ferworze dyskusji często się zapomina i każdy członek zespołu za punkt honoru stawia sobie przedstawienie swojego stanowiska.
Tego błędu nie popełniają doświadczone firmy. Rozdanie ról jest tam dobrze widoczne.

W skład zespołu negocjatorów nie

powinny wchodzić osoby nerwowe, zapalczywe, czy też cechujące się gadulstwem. Brak umiejętności koncentracji w długim okresie również jest przeciwskazaniem. Niezbędna jest również niezła kondycja fizyczna i zdolność odbywania długotrwałych (i często męczących) podróży.

Bardzo ważna jest znajomość języka, w którym prowadzone będą negocjacje. Dążyć należy do prowadzenia negocjacji w języku własnym lub obcym dla obu stron.
Najczęściej jednak językiem standardowym w kontraktach międzynarodowych jest język angielski.

Negocjacje może prowadzić też wyznaczona osoba (lub zespół) z możliwością konsultacji z swoimi specjalistami. Taki tryb pracy często jednak przedłuża tę fazę negocjacji i stwarza niebezpieczeństwo nieporozumień.

V.1.1. Pierwsze spotkanie

W trakcie pierwszego spotkania zespół negocjacyjny powinien wywrzeć mocne wrażenie lub uśpić czujność drugiej strony. Zależnie od tak określonego zamiaru powinien być przygotowany scenariusz tego spotkania. Autor nie zaleca jednak przesadnego ukrywania lub wyolbrzymiania walorów własnego zespołu, gdyż może być to źródłem nieufności w dalszych rozmowach po ujawnieniu rzeczywistej siły stron. Strona negocjująca powinna znać dobrze ofertę drugiej strony, w szczególności mieć już wcześniej przeanalizowane i przedyskutowane rozwiązania, które są przedmiotem oferty. Wybór miejsca i czasu spotkania powinien być zgodny z ogólnie przyjętymi zasadami i nie powinien narzucać drugiej stronie nadmiernie uciążliwych warunków. Także meble powinny być wygodne i podkreślć równoprawność stron (Sławny problem okrągłego stołu w rozmowach amerykańsko - wietnamskich w Genewie omal nie przekreślił możliwości porozumienia !)

W trakcie pierwszego spotkania powinna nastąpić obustronna prezentacja zespołów negocjacyjnych, sprawdzenie pełnomocnictw do prowadzenia rozmów, uzgodniony ostatecznie język roboczy oraz miejsce i ew.terminy dalszych spotkań. Jest to również dobry moment na sprecyzowanie reguł, które nie były do tej pory jasno ustalone (n.p. nie wynikały z zapytania ofertowego). Czasem już na tym etapie ustala się prawo, według którego dany kontrakt będzie rozpatrywany. Może to mieć duże znaczenie w przypadku pojawienia się sporu pomiędzy stronami. Dążyć należy do przyjęcia prawa polskiego i własnej jurysdykcji.

W razie jakichkolwiek wątpliwości lepiej jest poprosić o czas na odpowiedź niż pochopnie przyjmować rozwiązania, których dokładnie nie przeanalizowaliśmy.

V.1.2.Analiza merytoryczna ofert (y)

Zazwyczaj po fazie ustaleń wstępnych następuje faza, w której ocenia się walory użytkowe proponowanego rozwiązania.
Składa się ona zazwyczaj z następujących elementów:

- prezentacja teoretyczna
- test użytkownika
- wizyta referencyjna

Uczestnictwo w tej fazie zespołu negocjacyjnego jest niezmiernie pożyteczne i pozwoli na wyrobienie sobie głębszego poglądu na przedmiot kontraktu. Często tę fazę kończy się przygotowaniem tzw. Listu Intencyjnego. Dokument ten precyzuje wolę doprowadzenia do fazy ostatecznej jaką jest zawarcie kontraktu. Nie ma on jednak charakteru obligatoryjnego dla wszystkich stron.
Przygotowanie i przekazanie Listu Intencyjnego powinno być czynnością starannie przemyślaną. Nie należy Listu Intencyjnego wręczać zbyt szybko i

pochopnie, zwłaszcza gdy przebieg testów i negocjacji raczej nie wskazuje na wybór danej firmy.

V.1.3.Negocjacja kontraktu

Po pomyślnym zakończeniu testów kwalifikacyjnych, odbyciu wizyt referencyjnych nadchodzi czas na sformułowanie kontraktu.

Kształt kontraktu proponuje strona oferująca

Na tym etapie często wraca sprawa języka w jakim ma być sporządzony kontrakt, prawa właściwego dla danego kontraktu, sądów, które mają rozstrzygać ew. spory. Autor może tu powtórzyć wcześniejsze zalecenia:

- kontrakt w języku polskim
- prawo polskie lub neutralne (najlepiej niemieckie lub austriackie)
- jurysdykcja polskich sądów arbitrażowych lub powszechnych

Wystrzegać należy się przyjmowania prawa angielskiego i sądów angielskich, gdyż są one oparte o zawiłą procedurę precedensową.
Przedmiotem oceny i negocjacji powinny być:

- proponowane terminy poszczególnych etapów (lepiej przyjmować krótkie, łatwo weryfikowalne etapy niż godzić się na kontrakt jednoetapowy)
- kryteria akceptacyjne etapów i całości kontraktu. Precyzyjne ich zdefiniowanie umożliwi w przyszłości egzekwowanie własnych wymagań
- cena kontraktu. Powinno żądać się przedstawienia kalkulacji przedwykonawczej na poszczególne elementy systemu. Utrudni to drugiej stronie nadmierne windowanie cen. Kontrakt powinien przewidywać ustalenie ceny ostatecznej po zakończeniu prac zaakceptowanych odpowiednimi testami. Spotyka się często ustalenie sztywnej ceny kontraktu, ale praktyka dowodzi, że zazwyczaj obraca się to na niekorzyść beneficjenta.
- warunki płatności. Bardzo istotny punkt kontraktu, zwłaszcza jeśli kontrakt ma być finansowany środkami obcymi, n.p. kredytem bankowym. Określenie transz płatności powinno korelować z uzgodnionymi transzami kredytu bankowego, bądź też pokrywać z przewidywaną dyspozycją środków własnych przeznaczonych na rozwój. Często pokutuje przekonanie o celowości odsuwania płatności na odleglejsze terminy. Nie zawsze tak być musi. Przewidywane zwiększenie obciążenia firmy w późniejszym okresie powinno być wskazaniem do wcześniejszego regulowania należności.
- zabezpieczenia prawne i finansowe. Dobrze skonstruowany kontrakt powinien przewidywać możliwość wycofania się z kontraktu w wypadku niespełnienia w określonym czasie założonych etapów wraz z odpowiednią refundacją wyłożonych środków. Dodatkową gwarancją prawidłowego biegu kontraktu może być złożenie określonego depozytu przez wykonawcę kontraktu na rachunku bankowym, bądź wystawienie weksla, który może być zrealizowany w określonych okolicznościach. Trzeba jednak powiedzieć wyraźnie, że tak silne zabezpieczenie może być zrealizowane jedynie przez znaczącego i potężnego beneficjenta kontraktu. Takim zabezpieczeniem są asekurowane niektóre kontrakty rządowe.
- zabezpieczenia techniczne. Prawidłowe zabezpieczenie techniczne dotyczy przede wszyskim oprogramowania. Kontrakt powinien obligować oferenta do złożenia w depozycie (obustronnie uzgodnionym) źródeł programów i zapewnić ich aktualizację. Również dokumentacja projektowa powinna być objęta taką

klauzulą. Kontrakt powinien precyzować sposób wykonywania obsługi oprogramowania i sposób rozliczania jej kosztów. Istotne elementy zabezpieczenia technicznego powinny być ustalone i przedstawione przez specjalistów z tej dziedziny i biorących udział w negocjacjach.

- uzyskanie rabatu. Ponieważ każda oferta zawiera w sobie margines cenowy (nieraz znaczny) należy dążyć do jego maksymalnego obniżenia. Uzyskać to można poprzez podkreślenie w rozmowach cech szczególnych kontraktu, takich jak jego wielkość, czas trwania, pionierską rolę na rynku. Zwłaszcza pierwszy kontrakt z daną firmą daje dobre podstawy do żądania dużego upustu. Warto tu przypomnieć, że dawniej Centrale Handlu Zagranicznego stawiały warunek uzyskania rabatu jako obligatoryjny do możliwości podpisania kontraktu. Warto podkreślić rolę Komitetu Sterującego w kontroli przebiegu negocjacji i określaniu wytycznych do ich prowadzenia. Komitet Sterujący powinien czuwać już na tym etapie nad prawidłowym kształtem przyszłego kontraktu. Jeśli nie powołano komitetu Sterującego rolę nadzoru powinien pełnić wyznaczony członek ścisłego kierownictwa danej jednostki. Kształt kontraktu krystalizuje się zwykle w trakcie wielu spotkań roboczych. Ustalenia z tych spotkań powinny być rejestrowane w formie pisemnej i po obustronnej akceptacji przedstawiane w formie raportów właściwemu ciału nadzorującemu. Notowanie ustaleń jest bardzo ważne, zwłaszcza gdy negocjacje toczą się w jezyku obcym.

W tej fazie negocjacji należy dokładnie rejestrować wszelkie niezgodności ,aby nie wróciły niespodziewanie przed samym podpisaniem kontraktu.

Spośród znanych technik negocjacyjnych zdecydowanie właściwą jest przy negocjacji nowego kontraktu technika kooperatywna. Pozwala ona wykorzystać mechanizmy pozytywne porozumienia i współpracy. Oczywiście ma ona swoje ograniczenia. W sytuacjach gdzie trudno znaleźć porozumienie lepiej jest zawiesić na pewien czas rozmowy niż próbować forsować swoje stanowisko. Dobrym rozwiązaniem może być zintensyfikowanie rozmów z inną firmą. Będzie to sygnałem, że rozmowy dotychczasowe nie w pełni nas satysfakcjonują i poszukujemy innego rozwiązania. Nie należy jednak z tego rozwiązania korzystać pochopnie, gdyż może ono zepsuć dotychczasowe pozytywne rezultaty, zwłaszcza gdy już udało się uzyskać ponadstandardowe korzyści.

Autor zachęca do studiowania nielicznej wprawdzie, ale dostępnej na rynku polskim literatury w zakresie technik socjologicznych stosowanych w negocjacjach. Godne zalecenia jest również uczestnictwo w organizowanych też w Polsce treningach w zakresie prowadzenia negocjacji.

W prowadzeniu negocjacji obowiązują pewne zasady których nie wolno łamać. Można tu przytoczyć najważniejsze:

- oddzielenia ludzi od spraw które reprezentują
- nie stawiania bezczelnych propozycji i warunków
- zasadę zaufania ograniczonego i proporcjonalnego do dotychczasowych doświadczeń z tą firmą
- nieuleganie emocjom
- wyważonej skromności (odnosi się do osób prowadzących rozmowy)
- szacunku dla partnera

Mimo oczywistości tych zasad autorowi znane są przypadki ich brutalnego łamania, nawet przez doświadczonych negocjatorów.

Ostatecznie wynegocjowany kształt

kontraktu powinien zostać zaparafowany przez obie strony i przedstawiony do akceptacji właściwemu ciału.

V.1.4. Kontrakt negocjowany przez obcą firmę

Podstawowe zalecenia w wypadku prowadzenia negocjacji przez inną firmę (n.p. konsultingową) ograniczają się do:

- precyzyjnego określenia celu negocjacji wraz z ramowym określeniem parametrów kontraktu (cena, etapy, zasady rozliczania)
- zastrzeżenia prawa udziału własnego w negocjacjach
- określenia zasad uzgadniania stanowisk z wynajętą firmą
- określeniem warunków ekstremalnych (na co nie wolno się zgodzić w żadnym przypadku)

Tak prowadzone negocjacje również powinny podlegać okresowej kontroli Komitetu Sterującego.

VI. Końcówka - podpisanie kontraktu

Podpisanie kontraktu ma zazwyczaj charakter uroczysty. Podkreśla dobrą atmosferę zapoczątkowanej współpracy. Kontrakt podpisują zazwyczaj członkowie kierownictwa wysokiego szczebla zgodnie z posiadanymi pełnomocnictwami do reprezentowania danej jednostki. Osoby podpisujące powinny być wcześniej uzgodnione, a przed podpisaniem powinno być przedstawione uwierzytelnione pełnomocnictwo do działania wskazanej osoby w tym zakresie. W wypadku firm polskich wystarczy wyciąg z Rejestru Handlowego. Szczególnie starannie należy sprawdzić prawo do reprezentowania firmy jednoosobowo.
Teksty kontraktów zazwyczaj sporządzone w dwóch lub czterech jednobrzmiących egzemplarzach (sprawdzić !) są parafowane na każdej stronie przez głównego negocjatora każdej ze stron.Jest to gwarancją dla osoby podpisującej że egzemplarze zawierają treść rzeczywiście uzgodnioną.

W wypadku wątpliwości należy je niezwłocznie wyjaśniać, tak aby podpisać rzeczywiście uzgodniony dokument. Nie należy przyjmować uwag typu: "to błąd formalny, jutro wymienimy strony". W takim przypadku porawkę należy nanieść ręcznie wraz z podpisem i datą poprawki.Poprawki dokonujemy przez pojedyncze przekreślenie i wpisanie właściwej treści.

Przy okazji podpisania kontraktu jest w zwyczaju wręczenie przez firmę która uzyskała kontrakt drobnych upominków drugiej stronie. Wartość tych upominków nie może być znacząca w stosunku do zwyczajów przyjętych w kraju oferenta. Oczywiście nigdy i w żadnym wypadku nie wolno przyjąć pieniędzy lub innych środków mających cechy pieniądza (czeki i.t.p.) W tym miejscu autor stwierdza że w Polsce zasada ta niestety przez wiele osób nie jest rygorystycznie przestrzegana. Obniża to ich wartość jako negocjatorów przyszłych kontraktów, gdyż wiele informacji prędzej czy później przedostanie się do opinii publicznej. Stwierdzić też trzeba że wiele osób i instytucji prowadzi negocjacje i realizuje kontrakty w sposób wzorowo czysty i prawidłowy, co już skutkuje pozytywnymi opiniami w środowisku.

Finałem jest podpisanie kontraktu przez upoważnione osoby i zazwyczaj wspólny obiad fundowany przez realizatora kontraktu. Osoby które prowadziły rozmowy kontraktowe biorą w nim udział. Tematy rozmów nie powinny być związane z przebiegiem negocjacji, mogą natomiast dotyczyć planowanej przyszłości i współpracy obu stron. W ten sposób uroczystym akcentem zamyka się pewien etap (b.ważny) współpracy stron kontraktu.

Edward Kolbusz , Edward Kram
Uiwersytet Szczeciński

Problemy metodyczne wdrażania SIZ

Wdrażaniem nazywamy sekwencję działań, których celem jest wprowadzenie w sposób adaptowalny do praktycznej działalności podmiotu gospodarczego, modelu systemu informatycznego zarządzania /SIZ/ - wybranego jako efekt procesu projektowania. Z definicji tej wynika, że wdrażanie jest jednym z wielu rodzajów działań, podejmowanych w procesach tworzenia /realizacji/ SIZ, czyli, jak się to niekiedy nazywa jedną z faz cyklu życia systemu. W niniejszym tekście przedstawia się podstawowe tezy wykładu na temat problemów metodycznych wdrażania SIZ.

1. Wdrażanie jako faza procesu realizacji SIZ.

Rozważając proces tworzenia SIZ w kategoriach teorii realizacji, wymienić należy następujące jego główne fazy:

1. identyfikacja potrzeb
2. projektowanie koncepcyjne
3. projektowanie szczegółowe
4. URUCHAMIANIE OPROGRAMOWANIA
5. WDROŻENIE PILOTOWE
6. ROZPOWSZECHNIANIE
7. eksploatacja
8. doskonalenie

Dużymi literami oznaczono fazy tworzące proces wdrażania. Poszczególne fazy wyodrębniono ze względu na: cele /zadania/, rodzaje zużywanych zasobów oraz mierniki oceny. W każdej z tych faz działania organizowane powinny być w sposób zapewniający skuteczność. W teorii organizacji działanie takie nazywa się cyklem organizacyjnym, określającym pożądaną kolejność działań przy realizacji celu stopniowalnego. Synteza wielu opisanych w teorii modeli tego cyklu, pozwala na sprowadzenie go do trzech następujących czynności:

- przygotowanie działania,
- wykonanie zadania,
- kontrola.

Jest to punkt wyjścia do kształtowania cyklu organizacyjnego dla konkretnych przedsięwzięć. W zależności od rodzaju i skali podejmowanych działań oraz rodzaju i sposobu oddziaływania otoczenia powinna być kształtowana struktura cyklu organizacyjnego. Stwierdzimy zatem, że sformułowany adekwatnie do celów cykl działania zorganizowanego będzie podstawową metodą działania w procesach realizacji SIZ, zarówno w sensie ogólnym, jak również w poszczególnych fazach tego procesu. Stąd też proces wdrażania SIZ można przedstawić w postaci spirali, której głównymi ogniwami będą etapy tego procesu, zaś w każdym ogniwie wyodrębnimy etapy cyklu działania zorganizowanego. W tym sensie proces wdrażania jest procesem cyklicznym i iteracyjnym.

2. Struktura procesu wdrażania.

Na wstępie stwierdzimy, że przedstawiony poniżej sposób strukturalizacji procesu wdrażania nie jest jedynym możliwym. Można bowiem sformułować wiele innych, być może w konkretnych warunkach równie skutecznych. Nie należy się również upierać

przy stosowanej terminologii - ta bowiem jest problemem drugorzędnym. Należy również podkreślić, że w zależności od skali działań (o czym będzie mowa w zakończeniu) struktura procesu może być wielopoziomowa, wyróżnić w niej można zarówno działania sekwencyjne, jak i współbieżne, zaś granice między elementami tego, procesu mogą mieć charakter rozmyty. Proponujemy zatem niekoniecznie gotową metodę postępowania, a raczej pewien sposób /kryterium, wzorzec/ porządkowania i kształtowania realnie istiejących procesów wdrażania. Wyróżnimy zatem w procesie wdrażania trzy główne fazy:

- uruchamianie oprogramowania,
- wdrożenie pilotowe,
- rozpowszechnianie.

Uruchamianie oprogramowania.

Oprogramowanie systemu jest elementem struktury informacyjnej SIZ, i upraszczając sprawę można określić, że składa się ono z oprogramowania użytkownika i oprogramowania systemowego. Ogół czynności związanych z uruchamianiem oprogramowania można przyrównać do procesu tworzenia prototypu. Z tej uproszczonej analogii wynikają dwa bardzo istotne stwierdzenia, a mianowicie:

- celem tego procesu jest złożenie całości /systemu programów/ z przygotowanych uprzednio części /wytestowanych programów/ i sprawdzenie czy otrzymana całość funkcjonuje zgodnie z założeniami
- otrzymana w wyniku tych procesów biblioteka programów, jako prototyp, ma prawo zawierać jeszcze pewne niedopasowania, w związku z czym w organizacji procesu muszą funkcjonować układy wykrywające niedomagania i istnieć działania zmierzające do ich usunięcia.

Rozpatrzmy problem w konwencji cyklu działania zorganizowanego.

Przygotowanie działania. Należy tu w szczególności wymienić dwie grupy czynności: planowanie działań oraz pozyskiwanie i rozmieszczanie zasobów. Ważnym zadaniem w przygotowaniu realnego harmonogramu jest opracowanie listy zdarzeń oraz ułożenie między nimi zależności logicznych i czasowych, czyli ustalenie listy czynności. Takimi kluczowymi zdarzeniami na przykład mogą być:

- uruchomienie programów zakładających bazę danych,
- uruchomienie programów realizujących poszczególne jednostki przetwarzania /podsystemy, moduły, funkcje itp/
- uruchomienie oprogramowania sieci.

Opracowana lista /sieć/ czynności powinna zawierać tak istotne elementy harmonogramu jak: czas i terminy, zasoby (oprogramowanie, dokumentacja, kadry itp), planowany rezultat wraz z kryterium jego oceny, wykonawcy. Bardzo ważnym rodzajem zasobu w tej fazie wdrażania są dane do testowania programowania.

Wykonanie zadania. Na problem ten spojrzeć można z dwóch punktów widzenia:

- wykonanie zadania jako, ciąg czynności związany z realizacją harmonogramu. Mieści się tu cała problematyka rzeczywistego pozyskiwania zasobów i ich rozmieszczenia, kierowania realizacją i oceną itd.
- wykonanie zadania jako ciąg czynności wykonywanych bezpośrednio na oprogramowaniu, a mających na celu badanie dopasowania tego oprogramowania do warunków modelowych.

Z pierwszego punktu widzenia obserwuje się procesy sfery regulacji, z drugiego punktu widzenia wykonanie zadania polega na

działaniach na strukturach informacyjnej i funkcjonalnej SIZ. Oprogramowanie jako element struktury informacyjnej SIZ spełnia funkcję integrowania elemntów struktury funkcjonalnej systemu, przyczynia się w sposób decydujący do tego, że ta ostatnia "żyje". Dokonując wszelkich dopuszczalnych powiązań między elementami struktury informacyjnej i funkcjonalnej SIZ, na różnych poziomach hierarchii, ogólną procedurę uruchamiania oprogramowania można zdefiniować, jak następuje:

- uruchamianie poszczególnych programów:
- uruchamianie oprogramowania elementarnych jednostek funkcjonalno - technologicznych,
- składanie oprogramowania SIZ na składnikach struktury informacyjnej i funkcjonalnej.

Kontrola Ocena w obszarze tej problematyki sprowadza się do badania:

- sprawności otrzymanego produktu,
- sprawności procesu,
- przydatności dokumentacji dla dalszych faz procesu realizacji.

Wdrażanie pilotowe.

Celem wdrażania pilotowego jest wytestowanie SIZ na danych i w warunkach rzeczywistych. Jest to faza w pewnym sensie "krytyczna", następuje w niej bowiem konfrontacja modelu systemu z rzeczywistością - próba wmontowania tego modelu do praktycznej działalności obiektu gospodarczego.

Ze względu na rozmiar prac, zakres zmian dokonywanych w obiekcie oraz poziom innowacyjności rozwiązań systemowych, wdrażanie pilotowe może być interpretowane w dwojaki sposób:

- jako działanie innowacyjne w rozumieniu teorii innowacji, wraz z wszystkimi wynikającymi stąd konsekwencjami,
- jako imitacja, co dotyczy z reguły modeli systemów wcześniej już sprawdzonych w innych obiektach, co pozwala na wykorzystanie doświadczeń tych obiektów. W tym przypadku skala przedsięwzięć jest odpowiednio mniejsza, podobnie jak i ryzyko.

Przygotowanie działania. Wymienimy tu cztery następujące grupy działań:

- wybór metody wdrażania,
- planowanie działań i zasobów,
- pozyskanie i rozmieszczenie zasobów,
- opracowanie kryteriów oceny.

Metoda działania warunkuje treść, przestrzeń i czas pozostałych działań przygotowawczych. Z teoretycznego punktu widzenia istnieje możliwość zastosowania jednej z trzech metod:

- wdrażanie kolejnymi składnikam struktury funkcjonalnej SIZ we wszystkich uwzględnianych elementach struktury organizacyjnej obiektu gospodarczego,
- wdrażanie SIZ w wybranym składniku struktury organizacyjnej obiektu gospodarczego,
- wdrażanie kolejnymi składnikami struktury funkcjonalnej SIZ w wybranych skadnikach struktury organizacyjnej obiektu.

Dalsze przedsięwzięcia dotyczą planowania działań i zasobów, a również sposobów ich pozyskiwania i rozmieszczania. Główne zasady w tej fazie to w szczególności:

- sprzęt techniczny,
- kadry specjalistyczne,
- dokumentacja systemu,
- biblioteka programów,
- środki finansowe.

W ujęciu czynnościowym, plan - harmonogram powinien definiować czynności do wykonania oraz zdarzenia towarzyszące tym czynnościom, opisane za pomocą takich czynników, jak: czas, terminy, wykonawcy. Zdarzeniami kluczowymi w tej fazie wdrażania są:

- organizacyjne przygotowanie obiektu do przyjęcia SIZ,
- komputerowa instalacja SIZ.

Wykonanie działania polega na osiągnięciu powyższych zdarzeń kluczowych, a w szczególności:

- w zakresie organizacyjnego przygotowania obiektu:
  - wdrożenie nowych rozwiązań organizacyjnych i w systemie zarządzania,
  - wdrożenie niektórych elementów struktury informacyjnej SIZ, a w szczególnosci bazy normatywnej i wejść systemu,
  - przygotowanie załogi w zakresie obsługi i stosowania SIZ /szkolenie, instruktaż, działania motywujące itd/;
- w zakresie instalacji SIZ na komputerze:
  - założenie baz danych, a w tym przygotowanie danych do zakładania baz danych, konwersja zbiorów wraz z kontrolą i weryfikacją, wdrożenie procedur aktualizacji itd.,
  - wdrożenie cykli obliczeniowych;
- wdrożenie składników struktury funkcjonalnej, m.in. poprzez wprowadzenie do praktycznego stosowania wyjść systemu.

Z przedstawionego zakresu zadań dla fazy wdrażania pilotowego wynika, że niektóre z nich mogą być wykonywane przed rozpoczęciem tej fazy, szczególnie niektóre zadania z zakresu organizacyjnego przygotowania obiektu.

Rozpowszechnianie SIZ.

Wdrożenie pilotowe SIZ powinno dać odpowiedź na pytanie, czy rozwiązania projektowe i oprogramowanie systemu spełniają założone wymagania, i czy SIZ został zaakceptowany przez system gospodarczy. W przypadku odpowiedzi pozytywnych można przystąpić do jego rozpowszechniania, co może mieć dwojakie znaczenie:

- jako ciąg działań o charakterze handlowym /na przykład sprzedaż i wdrażanie gotowego oprogramowania/,
- jako ciąg działań mających na celu poszerzenie zakresu funkcjonowania SIZ w znaczeniu przestrzennym.

Sytuacja wyjściowa dla tej fazy jest odmienna niż w fazie poprzedniej, istnieje bowiem już przekonanie o poprawności i akceptacji systemu. W związku z tym zakres podejmowanych działań jest podobny, ma jednak inny charakter.

Przygotowanie działań koncentruje się na planowaniu i harmonagramowaniu zdarzeń oraz na pozyskiwaniu i rozmieszczaniu zasobów.

Wykonanie zadania w zakresie przygotowania organizacyjnego polega na podjęciu identycznych czynności jak w fazie wdrażania pilotowego, natomiast instalacja systemu na komputerze polega przede wszystkim na rozszerzeniu:

- zawartości informacyjnej bazy danych,
- grona użytkowników wyjść systemu.

Po wykonaniu fazy rozpowszechniania, SIZ można uznać jako wdrożony i przekazany do eksploatacji.

3. Niektóre warunki skutecznego wdrażania.

W ślad za dyrektywami prakseologicznej teorii organizacji określimy skuteczność wdrażania jako takie działanie, które prowadzi do osiągnięcia założonych celów, to znaczy do wprowadzenia do praktycznej działalności obiektu gospodarczego elementów SIZ opisanych w jego projekcie. Aby proces wdrażania SIZ mógł być uznany za skuteczny, musi on nadto umożliwiać osiągnięcie następujących celów:

- wdrożony SIZ realizuje wszystkie założone w projekcie cele w sposób ekonomiczny i racjonalny,
- wdrożony SIZ jest akceptowany przez podsystem społeczny obiektu gospodarczego.

Jakie uwarunkowania determinują skuteczność procesów wdrażania ?
Oto niektóre z nich.
Rozeznanie potrzeby jest warunkiem a priori wszelkiego skutecznego działania. Rzecz w tym, że zamiar zaspokojenia potrzeby jest punktem wyjścia dla określenia celów i sposobów działania.
Drugim ważnym warunkiem jest **przygotowanie działań,** będące etapem cyklu organizacyjnego, i którego rolę w metodologii wdrażania SIZ, już podkreślono. Przygotowanie procesow wdrożeniowych powinno zapewnić sprawność działania, m.in. poprzez zdefiniowanie dobrego planu działania. Plan taki charakteryzuje się w szczególności celowością, racjonalnością, operatywnością, elastycznością i wewnętrzą zgodnością. Nie wchodząc w dalsze szczegóły, trzeba zwrócić uwagę na dwa istotne zagadnienia. Czynności przygotowawcze są warunkiem koniecznym skutecznego i racjonalnego wdrażania SIZ, jednakże z drugiej strony nie można ich fetyszyzować, tzn. tolerować ich niewspółmierności w stosunku do zakresu działań. W praktyce często można spotkać działania charakteryzujące się jednym z dwóch skrajnych podejść:

- improwizacją jako działaniem bez uprzedniego wystarczającego przygotowania,
- nadmierną szczegółowością planów i harmonogramow, dążeniem do uczynienia procesu wdrażania procesem zdeterminowanym.

Obydwa wymienione wyżej skrajne przypadki są oczywiście szkodliwe dla skuteczności procesów wdrażania SIZ.
Z problematyką przygotowania procesów wdrożeniowych wiąże się ciągle kwestia skali zmian, jakie niesie sobą SIZ do obiektu gospodarczego. Istnieje bowiem ścisła zależność pomiędzy **skalą zmian**, a zakresem prac przygotowawczych i stopniem skomplikowania samych procesów wdrożeniowych. Z teoretycznego punktu widzenia wymienić można następujące zakresy zmian wprowadzane przez SIZ do obiektu:

- SIZ wprowadzający zmiany bezpośrednio w systemie zarządzania obiektem gospodarczym,
- SIZ wprowadzający zmiany w systemie informacyjnym obiektu,
- SIZ wprowadzający zmiany w metodach i technikach przetwarzania danych, tj. w systemie przetwarzania danych obiektu.

W tym kontekście uwzględnić należy również sposób zaspokojenia potrzeb obiektu. Można tu wymienić trzy podstawowe podejścia:

- zakup gotowego oprogramowania,
- modyfikacja istniejących rozwiązań,
- tworzenie systemu od podstaw, w pełnym cyklu realizacyjnym.

W załączonej tablicy przedstawiono przestrzeń kombinatoryczną obrazującą wyliczenie teoretycznych modeli struktur procesów wdrażania SIZ. Rozwinięcie tej problematyki przekracza możliwości niniejszej publikacji. Na zakończenie wymienić należy warunek przygotowania otoczenia społecznego .

W praktyce sprowadza się to przeważnie do przeprowadzania szkoleń, traktowanych zresztą w sposób formalny.

Problematyka wpływu **otoczenia społecznego** na wdrażanie SIZ wymaga odrębnego omówienia.

Skala zmian Sposób zaspokojenia potrzeb	Zmiany w systemie przetwarzania danych	Zmiany w systemie informacyjnym	Zmiany w systemie zarządzania
Gotowe oprogramowanie			
Modyfikacja istniejących rozwiązań			
Tworzenie nowego systemu			

Ewa Kolasińska
ZWiK Szczecin

Oczekiwania użytkowników wdrażających systemy komputerowe

1. Wstęp.

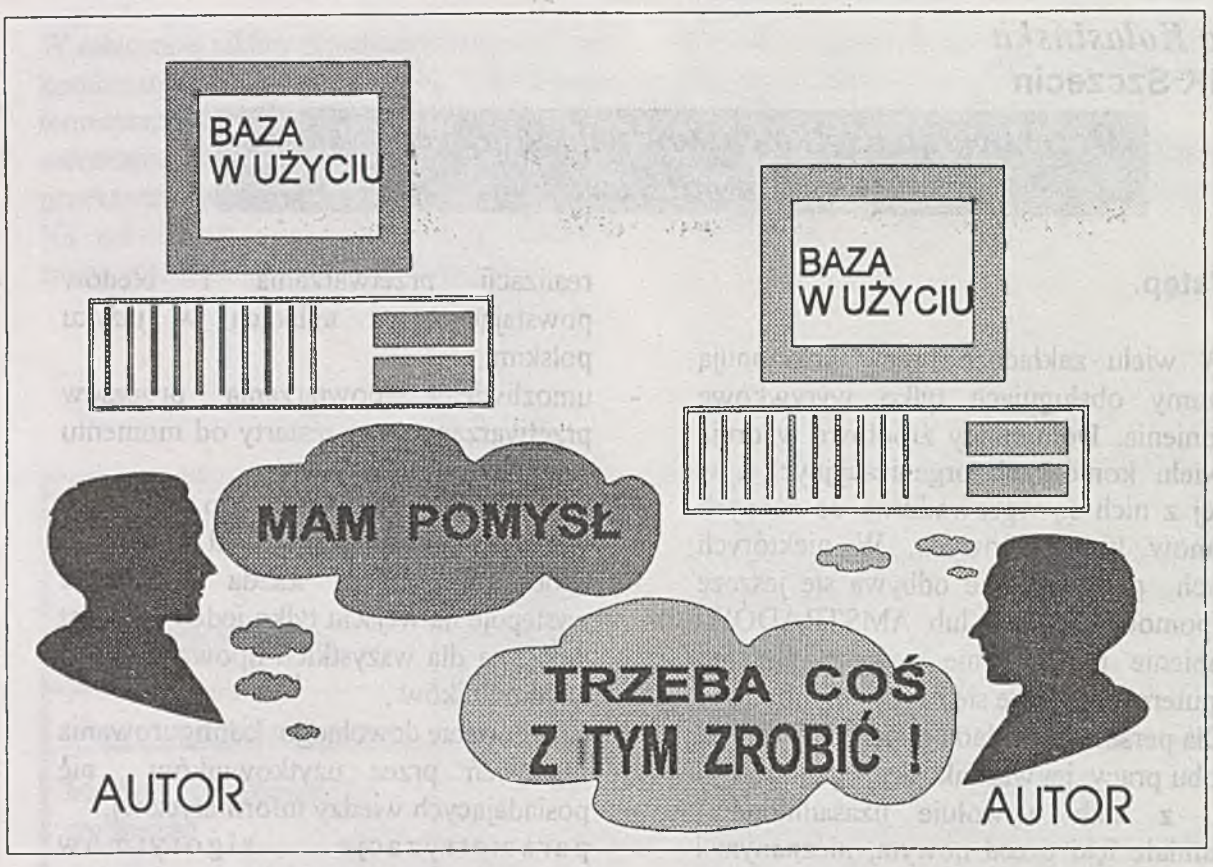
W wielu zakładach pracy funkcjonują programy obsługujące tylko wyrywkowe zagadnienia. Dokumenty źródłowe wędrują po wielu komórkach organizacyjnych i w każdej z nich są wprowadzane do różnych systemów komputerowych. W niektórych firmach przetwarzanie odbywa się jeszcze przy pomocy ASCOT lub AMSTRADÓW. Zakupienie i wdrożenie nowego systemu komputerowego staje się tu nieuniknione.

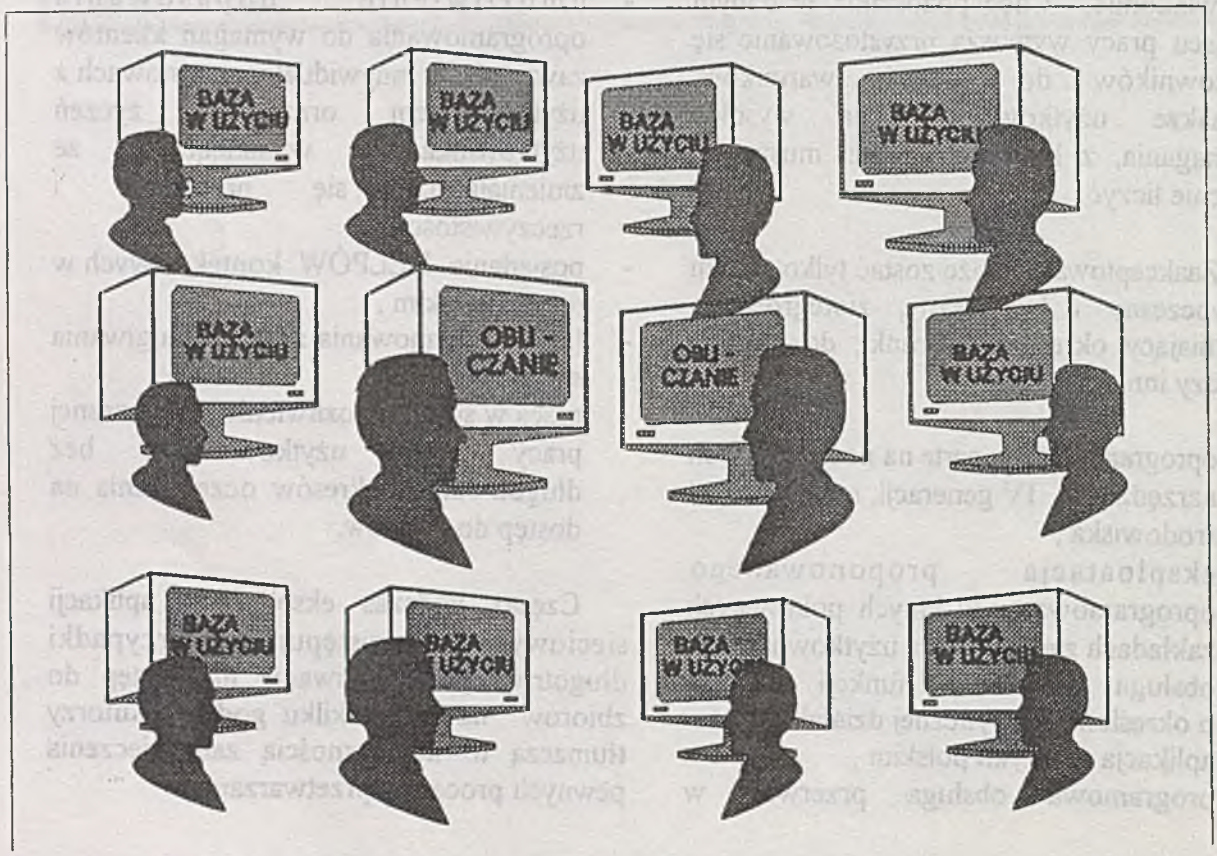
Dla personelu zakładu oznacza to zmianę sposobu pracy, jej warunków i organizacji. U wielu z nich wywołuje uzasadnione i zrozumiałe lęki przed nowym, nieznanym i często nieakceptowanym świadomie lub nieświadomie. Chęć pozostania w danym miejscu pracy wymusza przystosowanie się pracowników do nowych warunków. Jednakże użytkownicy mają wysokie wymagania, z którymi oferenci muszą się obecnie liczyć.

Zaakceptowany może zostać tylko system nowoczesny i bezpieczny, zintegrowany, spełniający określone warunki, do których między innymi należą:

- oprogramowanie oparte na nowoczesnych narzędziach IV generacji, niezależne od środowiska ,
- eksploatacja proponowanego oprogramowania w innych pokrewnych zakładach zadowalająca użytkowników ,
- obsługa wszystkich funkcji zakładu o określonej specyficznej działalności,
- aplikacja w języku polskim ,
- programowa obsługa przerwań w realizacji przetwarzania i błędów powstających w aplikacji w języku polskim ,
- umożliwienie powtarzania procesów przetwarzania oraz restarty od momentu przerwania ,
- umożliwienie ustalenia kto i kiedy wprowadzał lub modyfikował dane,
- takich w których każda informacja występuje na wejściu tylko jeden raz i jest dostępna dla wszystkich upoważnionych użytkowników ,
- umożliwienie dowolnego konfigurowania zestawień przez użytkowników nie posiadających wiedzy informatycznej,
- parametryzacja algorytmów obliczeniowych i sposobów rozwiązań,
- umożliwienie dopasowania oprogramowania do wymagań klientów zawartych w indywidualnych umowach z użytkownikiem oraz do życzeń użytkownika wynikających ze zmieniających się przepisów i rzeczywistości,
- posiadanie HELPÓW kontekstowych w języku polskim ,
- łatwość opanowania zasad posługiwania się aplikacją ,
- praca w sieci i umożliwienie jednoczesnej pracy wielu użytkownikom, bez długotrwałych okresów oczekiwania na dostęp do zbiorów.

Często podczas eksploatacji aplikacji sieciowych występują przypadki długotrwałego oczekiwania na dostęp do zbiorów nawet do kilku godzin. Autorzy tłumaczą to koniecznością zabezpieczenia pewnych procesów przetwarzania.


BAZA
W UŻYCIU
BAZA
W UŻYCIU
MAM POMYSŁ
TRZEBA COŚ
Z TYM ZROBIĆ !
AUTOR
AUTOR


BAZA
W UŻYCIU
OBU-
CZANIE

Twierdzą, że tak musi być i że nie można dopuścić do baz danych innych użytkowników, przykładowo w trakcie liczenia płac. Nie jest w tym czasie nawet możliwe przeglądanie danych, nie wspominając już o wprowadzaniu, czy drukowaniu. Liczenie płac trwa około dwóch godzin, a więc wszyscy inni użytkownicy tego systemu dwie godziny nie mogą pracować.

Personel Działu Kadr przez dwie godziny nie może wykonywać swoich obowiązków służbowych, ponieważ cała jego praca polega na wykorzystywaniu aplikacji. Jeżeli zaistnieje potrzeba powtórnego przeliczenia płac z jakiegoś powodu to następne dwie godziny wszyscy inni nie mogą pracować. Dział Kadr obsługuje interesantów (pracowników) w czasie rzeczywistym. W opisanej krytycznej sytuacji nie może udzielić pracownikowi odpowiedzi o ilości przysługującego mu jeszcze urlopu i nie może wydać odpowiedniego zaświadczenia. Pracownik musi czekać na odpowiedź lub jest odesłany na następny dzień. Takie sytuacje występują w różnych aplikacjach i po interwencjach użytkowników są przynajmniej częściowo usuwane. Jestem przekonana, że gdyby autorzy popracowali w takich warunkach przez jakiś czas sami doszliby do wniosku, że to trzeba zmienić i znaleźliby odpowiedni sposób.

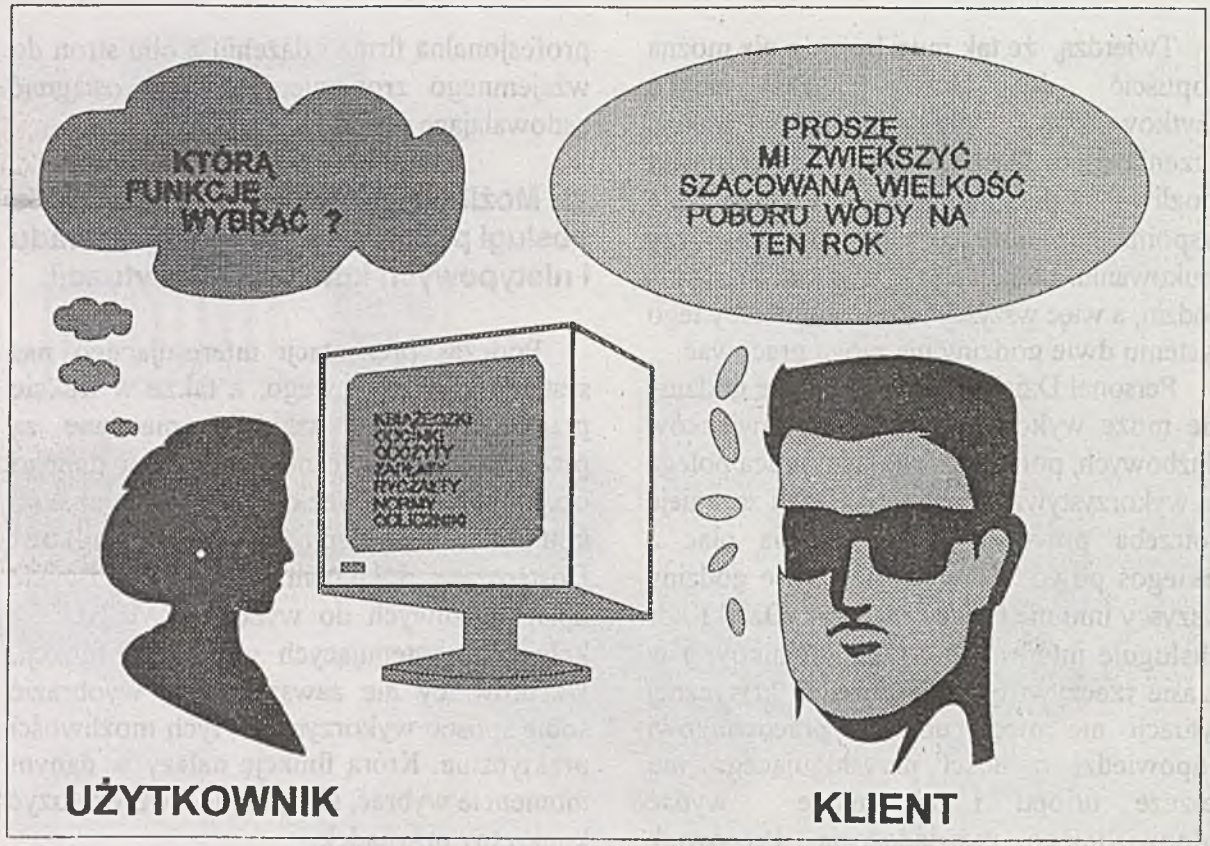
Bardzo trudno znaleźć w Polsce firmę komputerową oferującą systemy spełniające od razu wszystkie wymagane warunki. Zakład pracy średniej wielkości obsługujący kilkadziesiąt tysięcy klientów na podstawie indywidualnych umów, rozrzucony terytorialnie, który charakteryzuje się prowadzeniem specyficznej działalności i posiadający bardzo różnorodny sposób wynagradzania pracowników oraz rozbudowany system księgowy potrzebuje zwykle systemów na zamówienie, gdyż oferowane gotowe nie spełniają stawianych wymagań. Podczas współpracy z profesjonalną firmą i dążeniu z obu stron do wzajemnego zrozumienia można osiągnąć zadowalające rezultaty.

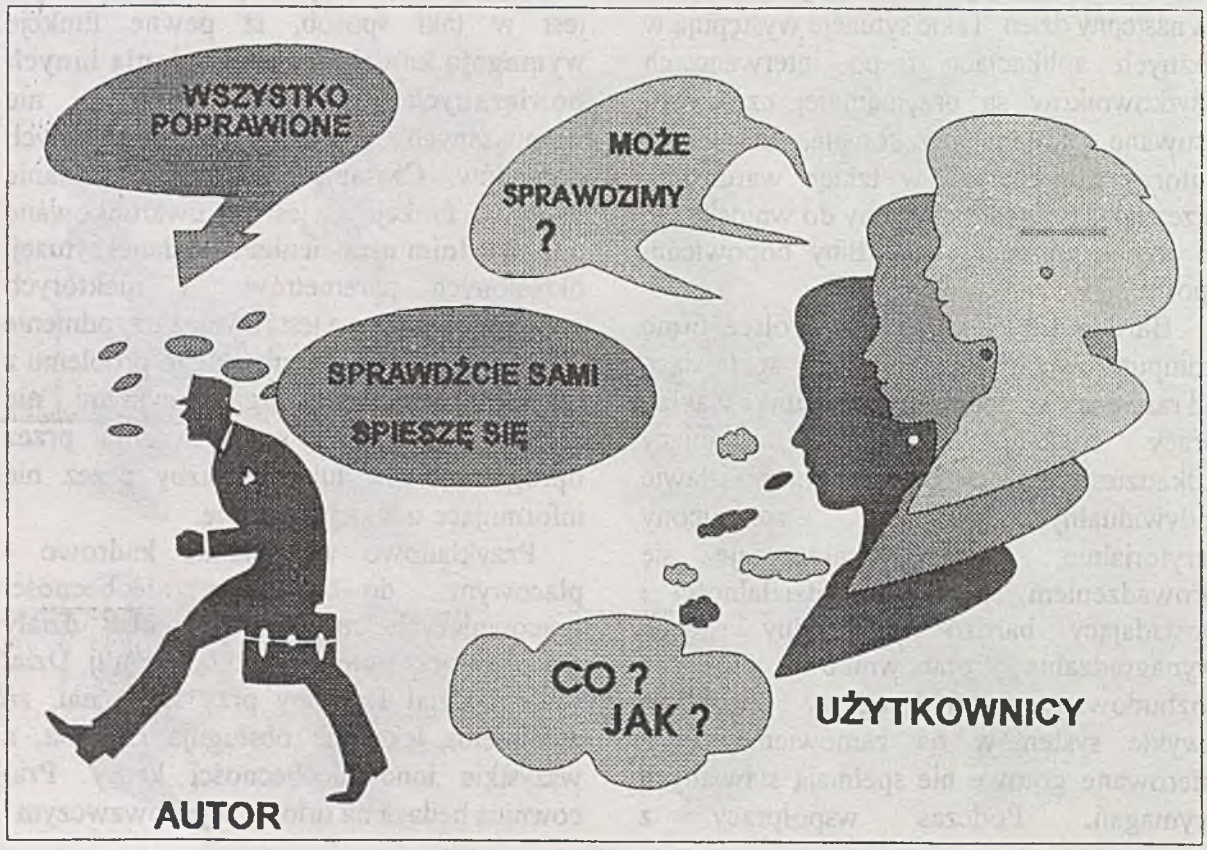
2. Możliwości aplikacji w aspekcie obsługi podstawowych funkcji zakładu i nietypowych konkretnych sytuacji.

Podczas prezentacji interesującego nas systemu komputerowego, a także w trakcie przeprowadzanych szkoleń omawiane są przeważnie teoretyczne możliwości danego oprogramowania bez odniesienia do obsługi konkretnie występujących przypadków. Dostarczana dokumentacja zawiera zwykle opisy możliwych do wyboru "MENU" i kolejno występujących w nich funkcji. Użytkownicy nie zawsze mogą wyobrazić sobie sposób wykorzystania tych możliwości praktycznie. Którą funkcję należy w danym momencie wybrać, aby prawidłowo obsłużyć konkretny przypadek.

Bardzo często aplikacja zorganizowana jest w taki sposób, iż pewne funkcje **wymagają koniecznego wywołania innych powiązanych** z wybranymi, a nie wymuszanych automatycznie z różnych powodów. Czasami prawidłowe działanie danej funkcji jest uwarunkowane **odpowiednim ustawieniem** dla danej sytuacji określonych parametrów. W niektórych sytuacjach konieczne jest również uzgodnienie sposobu rozwiązania konkretnego problemu z innymi komórkami organizacyjnymi nie zawsze wymuszane automatycznie przez oprogramowanie lub chociażby przez nie informujące o takiej potrzebie.

Przykładowo w systemie kadrowo - płacowym do zbioru nieobecności pracowniczych mają dostęp oba działy pracujące przy pomocy tego systemu tj. Dział Kadr i Dział Rachuby przy założeniu, że zwolnienia lekarskie obsługuje rachuba, a wszystkie inne nieobecności kadry. Pracownica będąca na urlopie wychowawczym


KTÓRĄ FUNKCJĘ WYBRAĆ ?
PROSZĘ MI ZWIĘKSZYĆ SZACOWANĄ WIELKOŚĆ POBORU WODY NA TEN ROK
UŻYTKOWNIK
KLIENT


WSZYSTKO POPRAWIONE
MOŻE SPRAWDZIMY ?
SPRAWDŹCIE SAMI SPIESZĘ SIĘ
CO ? JAK ?
AUTOR
UŻYTKOWNICY

nabywa prawo do urlopu macierzyńskiego płatnego. Jest to konkretna sytuacja, do której obsługi nie jest przewidziana specjalna funkcja. Nie można z rozwijanych "MENU" wybrać obsługi takiego przypadku. Dokumentacja będąca w posiadaniu użytkownika nie dostarcza potrzebnego instruktażu.

Pracownica zgłasza się do Działu rachuby, ponieważ istotna dla niej jest należąca się jej zapłata. Personel tego działu wprowadza do systemu komputerowego informację w taki sposób, jaki wydaje mu się właściwy.

Sytuacja taka winna być zgłoszona do Działu Kadr i przez jego pracowników wprowadzona do systemu, ponieważ to oni są upoważnieni do kontrolowania nieobecności.

Ze względu na brak instrukcji na ten temat sposób postępowania powinien być przedyskutowany z autorem, aby uzyskać prawidłowe zestawienia dotyczące stanów obecności w danym dniu i sprawozdania o przeciętnym zatrudnieniu.

Jednakże pracownicy obsługujący komputerowy system zwykle **bardzo niechętnie zwracają się o pomoc w trakcie powstawania trudności. Wolą "poradzić" sobie sami**. Gubią się w gąszczu "MENU" i często postępują nieprawidłowo otrzymując w rezultacie błędne wyniki. Następnie **bardzo się denerwują, tracą wiarę w prawidłowe działanie oprogramowania, przeżywają stresy i obdarzają niechęcią dany system oraz komputery**. W kolejnym etapie wzywają autora i proponują zmianę programu, a po zrozumieniu błędów w swoim postępowaniu domagają się **przeprowadzenia dokładnego szkolenia i dostarczenia szczegółowej instrukcji** obsługi wszystkich mogących wystąpić przypadków wraz z przykładami.

W ramach szkolenia użytkownicy oczekują omówienia postępowania w konkretnych, występujących praktycznie sytuacjach **rutynowych dla danego zakładu, a także nietypowych**. Specyficzne sytuacje mogą nie być obsługiwane automatycznie przez funkcje aplikacji, ale musi istnieć sposób na wprowadzenie ich do systemu komputerowego ze względu na konieczność otrzymywania prawidłowych wyników w końcowych zestawieniach i sprawozdaniach. Nie jest wskazane poprawianie ręczne zestawień otrzymywanych w postaci wydruków komputerowych. Jednakże jeśli pewne rodzaje nietypowych sytuacji występują w dużych ilościach oczekuje się, iż system będzie obsługiwał je automatycznie .

3. Uciążliwości sprawdzania poprawności działania aplikacji oraz kompletności danych w okresie wdrożenia, a także podczas modyfikacji oprogramowania w eksploatacji.

Sprawdzanie poprawności działania aplikacji i kompletności baz danych jest bardzo uciążliwe i często wykonywane chaotycznie. Od firmy komputerowej prowadzącej wdrożenie oczekuje się :

- usystematyzowania oraz sformalizowania prac sprawdzających,
- propozycji harmonogramu i sposobów ich przeprowadzania,
- pomocy przy ustalaniu prawidłowości wprowadzanych danych i otrzymywanych wyników.

Po wdrożeniu danego systemu i przyjęciu go do eksploatacji często występują nowe sytuacje wymuszające modyfikację parametrów, a także oprogramowania. Użytkownicy spodziewają się zazwyczaj, że autor podejmie i zrealizuje prace modyfikacyjne.

Jednocześnie oczekuje się od autorów przetestowania oprogramowania w nowej postaci i dostarczenie prawidłowo działającej nowej wersji aplikacji oraz przeprowadzenia szkolenia w zakresie dokonanych zmian.

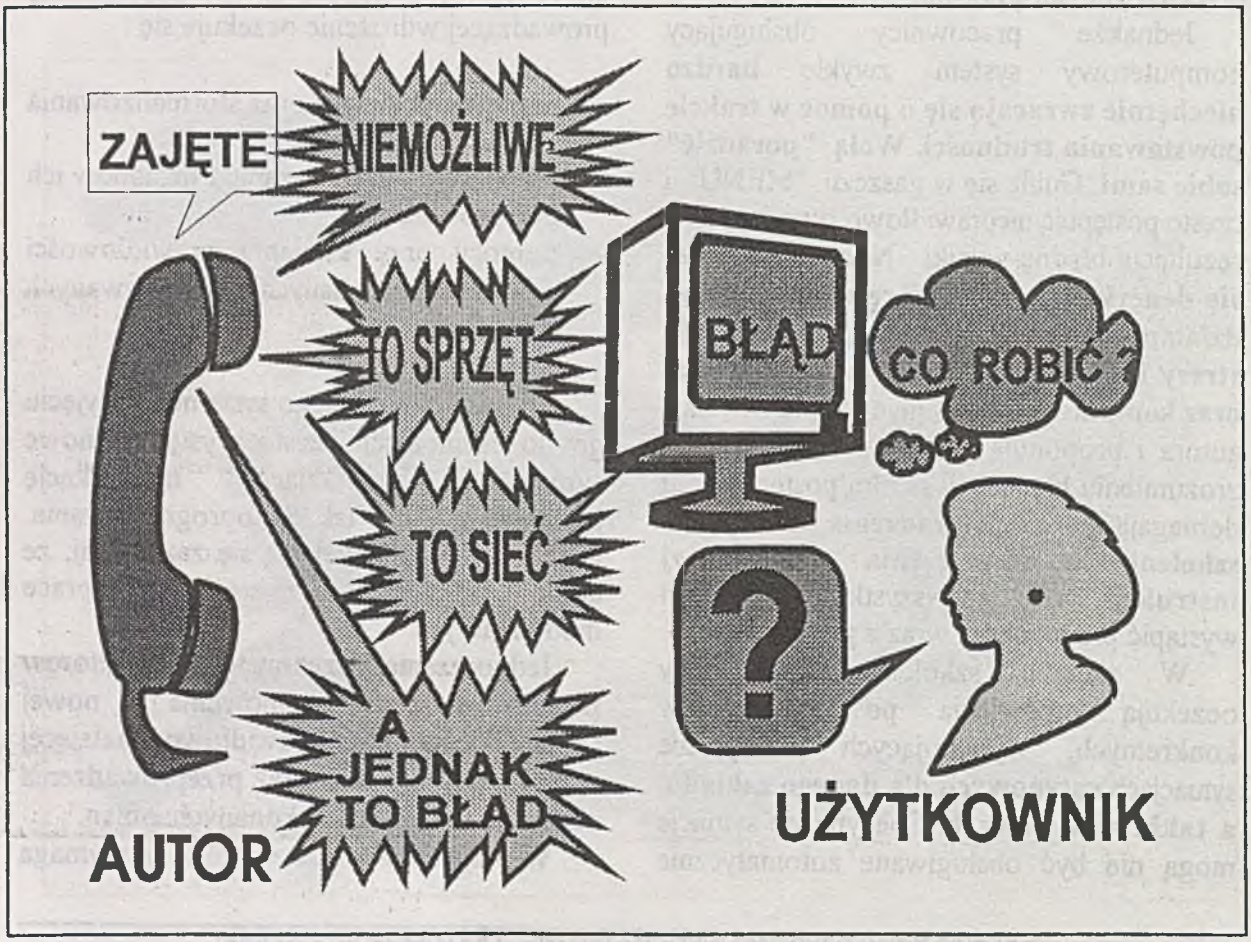
W przypadku zmian w aplikacji wymaga

się **dokonania uzupełnień w dokumentacji o nowe elementy** występujące w obsłudze systemu, jak również w rozumieniu funkcji oraz szczegółowego opisu sposobu sprawdzenia poprawności działania systemu po modyfikacji. Wyszczególnienia: które funkcje należy sprawdzić i w jaki sposób.

W praktyce często pozostawia się ten problem użytkownikowi. **Autorzy rzadko dostarczają wyczerpujących informacji** na temat przeprowadzonej modyfikacji i nie zawsze dokonują kompletnego testowania działania wszystkich funkcji systemu po każdej z poprawek, a czasami gubią się w swoim oprogramowaniu. U użytkownika pojawiają się **błędy w poprawnie już poprzednio działającym oprogramowaniu** i w pełni już eksploatowanym, co zdecydowanie dezorganizuje mu pracę, powoduje dezorientację i zgłaszanie nowych uwag.

Użytkownicy oczekują, że **zgłaszane przez nich uwagi** do wdrażanego, a później eksploatowanego systemu komputerowego **będą traktowane poważnie** przez autorów i współpracującą firmę. Spodziewają się, iż autorzy po otrzymaniu uwag ustalą z użytkownikiem sposób ich uwzględnienia i niezwłocznie przystąpią do realizacji tego zamierzenia.

Poważne traktowanie użytkownika wyklucza obarczanie go szukaniem **przyczyn nieprawidłowości w działaniu systemu** i sugerowanie ich poza oprogramowaniem. Jednakże zdarza się, że autor jest przekonany o prawidłowym działaniu systemu, gdyż nie znajduje przyczyn zgłaszanych błędów. Wskazane w takiej sytuacji jest sprawdzenie sprzętu, sieci komputerowej i zasilającej oraz danych i procesu ich wprowadzania. Czasami błędy wynikają z konfliktów w gospodarowaniu pamięcią operacyjną.



Niektóre **programy rezydujące** w pamięci powodują takie efekty.
Przykładowo program pozwalający administratorowi sieci NOVELL oglądać pracę na poszczególnych końcówkach XTSERV rezyduje w pamięci końcówki i bez żadnego komunikatu powoduje błędne działanie aplikacji.

Gdy **użytkownik wykluczy wszystkie inne elementy** związane z eksploatowanym przez siebie systemem komputerowym i nadal zauważa nieprawidłowości w jego działaniu **oczekuje**, że (pomimo przekonania o braku podstaw do szukania błędów w oprogramowaniu) **autorzy podejmą działania zaradcze,** gdyż **to ich oprogramowanie błędnie działa** i użytkownik jest w tej sytuacji **bezradny**, a z doświadczenia wynika, że najczęściej nieprawidłowości występują jednak w oprogramowaniu.

4. Zadania i cechy informatyka zakładowego.

Informatyk zakładowy **może pomóc** w rozwiązywaniu problemów powstających w procesie komputeryzacji, **ale tylko wtedy,** gdy zostanie poinformowany o działalności zakładu i potrzebach użytkowników oraz jeśli otrzyma wystarczającą ilość wiadomości na temat aplikacji .

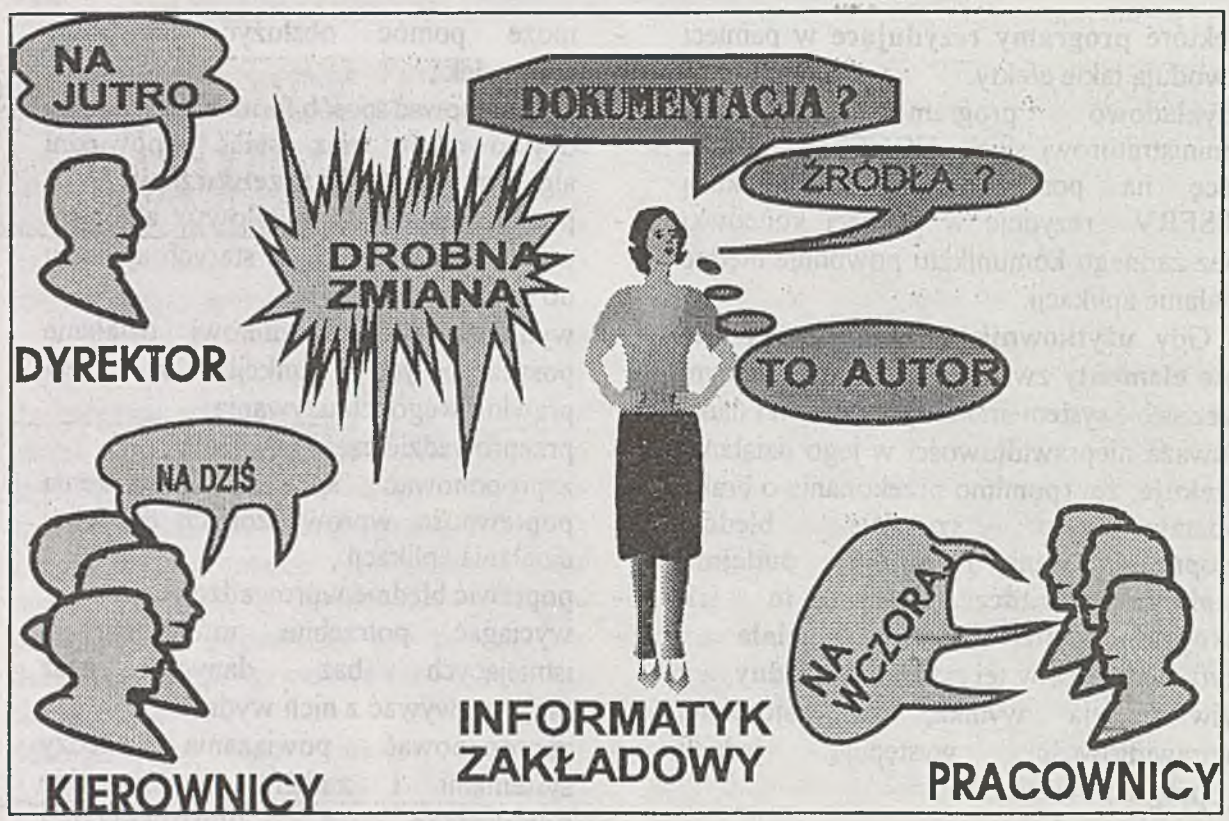
Nieodzowne jest również **dostrzeżenie** przez użytkowników systemu pracujących w poszczególnych komórkach organizacyjnych oraz autorów **potrzeby współpracy** z zakładowym informatykiem oraz **korzyści z niej wynikające dla wszystkich stron.**

Mając doświadczenie przy wdrażaniu systemów komputerowych:

- pomoże wybrać najodpowiedniejsze dla użytkownika oprogramowanie ,
- łatwiej dostrzega zastosowanie możliwości aplikacji do obsługi konkretnych występujących sytuacji ,
- może pomóc obsłużyć nietypowy przypadek ,
- zaproponować sposób formalizacji życzeń użytkowników oraz ustalić odpowiedni algorytm lub sposób przetwarzania ,
- pomóc ustalić prawidłowy algorytm przeniesienia danych ze starych aplikacji do nowego systemu ,
- wytłumaczyć użytkownikowi działanie poszczególnych funkcji w celu prawidłowego ich używania ,
- przeprowadzić część szkolenia ,
- zaproponować sposób sprawdzenia poprawności wprowadzonych danych i działania aplikacji ,
- poprawić błędnie wprowadzone dane ,
- wyciągać potrzebne informacje z istniejących baz danych oraz przygotowywać z nich wydruki ,
- zaproponować powiązania między systemami i zmiany organizacyjne wynikające z wprowadzania zintegrowanych systemów komputerowych .

W pierwszym etapie użytkownicy pragną zapoznać się z nowym systemem i pracować według starych zasad przy pomocy nowego systemu wymuszając na autorach dopasowanie ich oprogramowania do swojej organizacji i umożliwienie wielokrotnego wprowadzania danych przez poszczególne komórki organizacyjne. Dopiero w kolejnym etapie wdrażania można spodziewać się zgody użytkowników na przeprowadzenie zmian organizacyjnych i wyeliminowanie wielokrotnego wprowadzania tych samych danych.

Od informatyka zakładowego oczekuje się, że będzie posiadał wyobraźnię, zdolności dydaktyczne, możliwości przewidywania, łatwość nawiązywania kontaktów oraz usposobienie pogodne i życzliwe.



5. Sprzeczność interesów firmy sprzedającej aplikację i jej użytkownika.

Autorzy oraz firmy zajmujące się opracowywaniem, sprzedażą i wdrażaniem systemów komputerowych **obawiając się naruszenia praw autorskich** oraz wystąpienia błędów w wyniku ingerencji w oprogramowanie dążą do zachowania w tajemnicy przyjętych rozwiązań, struktur baz danych i algorytmów rozliczeniowych.

Użytkownicy domagają się dostarczenia im wszelkich możliwych informacji o działaniu poszczególnych funkcji systemu jedynie w tym celu, aby sprawniej go wdrożyć i zapewnić sobie możliwość pracy przy jego pomocy.

Czasami od informatyków zakładowych wymaga się dokonywania **"drobnych" zmian** w zakupionym oprogramowaniu i bardzo trudno jest wyjaśnić dlaczego jest to w zasadzie niemożliwe. Trudno przecież poprawiać obce programy ze względu na niemożliwość nie popełnienia błędu. Jest to oczywiste, ale tylko w gronie informatyków. Sądzę, że do czasu , kiedy jest możliwość **korzystania z usług autorów są oni jedynymi najodpowiedniejszymi do dokonywania poprawek** w swoim oprogramowaniu. Takie poprawki są obciążone najmniejszą ilością możliwych do powstania błędów.

Jednakże w sytuacji niemożliwości skorzystania z usług autorów wynikającej z **ich braku** może powstać konieczność zmian oprogramowania, które będzie musiał przeprowadzić informatyk potrafiący posługiwać się danym narzędziem programowym i zorientowany w aplikacji.

Po wdrożeniu aplikacji użytkownik często w praktyce nie ma już przeważnie **żadnej innej możliwości wypełniania swoich zadań.** Jest zależny od sprawności swojego

systemu komputerowego. **Musi być samodzielny w obsłudze istniejących w nim funkcji** i potrafić wszystkie występujące praktycznie przypadki do niego wprowadzić, gdyż **to on odpowiada za otrzymywane z systemu wyniki przed zwierzchnikiem i przed klientem.**

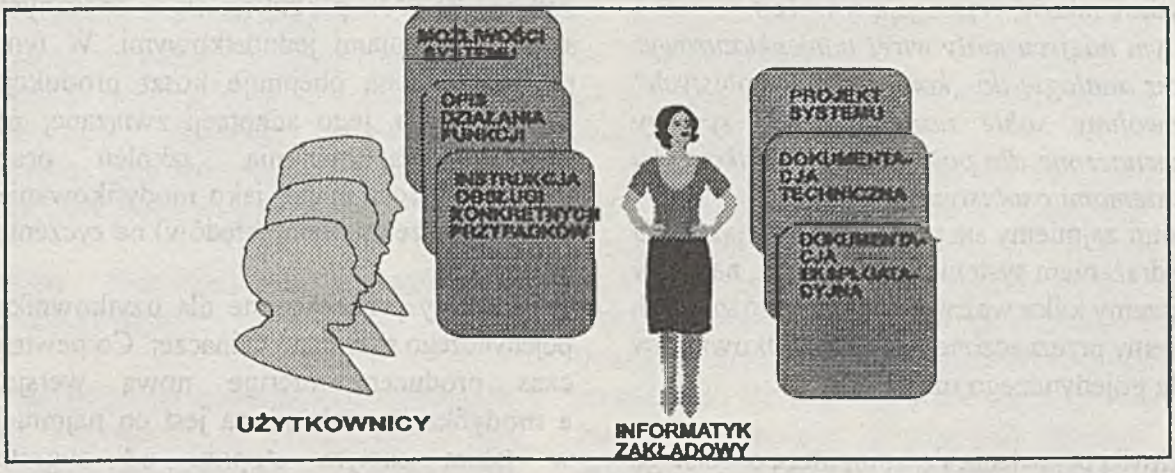
Nie może tłumaczyć się zakupem nowego oprogramowania i problemami z tym związanymi. Niezbędna więc jest mu pełna dokumentacja systemu obejmująca :

- możliwości systemu ,
- opis działania poszczególnych funkcji ,
- instrukcja obsługi konkretnych przypadków ,
- projekt systemu zawierający propozycje zmian organizacyjnych, propozycje kodów i skorowidzów najlepiej wykorzystujących możliwości systemu do spełnienia życzeń użytkowników ,
- dokumentacja techniczna zawierająca struktury baz danych, opis sposobu przetwarzania danych oraz stosowanych algorytmów,
- dokumentacja eksploatacyjna obejmująca informacje potrzebne informatykom zakładowym do sprawnej obsługi systemu, dotycząca między innymi ustawiania parametrów, archiwizacji .

Opracowanie takiej dokumentacji wymaga od przedstawicieli firmy komputerowej i autorów **dłuższego pobytu u użytkownika** w celu skonkretyzowania jego potrzeb i zaobserwowania różnorodności występujących sytuacji, ze względu na to, **iż użytkownik często nie potrafi ich ustalić ani opisać.** Uwzględnianie wymagań użytkownika jest zwykle uciążliwe i pracochłonne. Jednakże powodzenie w przeprowadzeniu wdrożenia zależy od przychylności personelu użytkownika. Ta z kolei kształtuje się pod wpływem złych lub dobrych doświadczeń podczas współpracy z autorami wybranego systemu i samym systemem. Może zaistnieć sytuacja, że zniechęci się, zrezygnuje z aplikacji i powróci do poprzedniego sposobu pracy. Nawet najlepsze oprogramowanie nabiera swojej wartości dopiero wtedy, gdy jest wykorzystywane.

Dokumentacja powinna być dostarczona wraz z systemem .
Nie należy spodziewać się, że będzie ona czytana na zapas. **Stanowi zabezpieczenie i umożliwia doinformowanie w potrzebie.**

Podjęcie trudu zaowocuje szybszym i łagodniejszym przebiegiem procesu wdrażania oraz spowoduje zadowolenie użytkowników, a więc przyniesie pełen sukces autorom.



Bogusław Jackowski, Tomasz Plata-Przechlewski, Marek Ryćko
Gdańsk

Wdrażanie systemów przeznaczonych dla pojedynczego użytkownika: porównanie systemów interakcyjnych i programowalnych

Wstęp

Celem niniejszej publikacji jest próba sformułowania paru pytań natury ogólnej związanych z teraźniejszością i przyszłością systemów przeznaczonych dla pojedynczego użytkownika. Z odpowiedziami jak zawsze jest trudniej, zwłaszcza jeżeli chodzi o przyszłość. Tym niemniej nie oparliśmy się pokusie, by oprócz zdań pytających sformułować parę zdań twierdzących. Nie pretendujemy przy tym do bezstronności, licząc po trosze na sprowokowanie polemiki. A nuż coś się uda wyjaśnić?

System dla pojedynczego użytkownika

Wdrożenie systemu na ogół kojarzy się z ogromnymi systemami, przeznaczonymi dla wielu użytkowników. Nie należy jednak zapominać, że wszelkie systemy, w tym systemy przeznaczone dla pojedynczych użytkowników, wymagają wdrożeń.

W tym miejscu mały wtręt nomenklaturowy: przez analogię do „komputerów osobistych" pozwolimy sobie nazwać krótko systemy przeznaczone dla pojedynczego użytkownika „systemami osobistymi".

Zanim zajmiemy się problemami związanymi z wdrażaniem systemów osobistych, najpierw opiszemy kilka ważniejszych cech różniących systemy przeznaczone dla grup użytkowników i dla pojedynczego użytkownika.

Z chwilą pojawienia się komputera osobistego systemy przeznaczone dla pojedynczego użytkownika zaczęły przeżywać gwałtowny rozwój. Takie giganty jak Microsoft, Borland czy Ashton-Tate zbudowały swą potęgę na oprogramowaniu przeznaczonym dla szerokich rzesz posiadaczy komputerów osobistych.

Systemy takie muszą być tanie. Pojedynczego użytkownika nie stać po prostu za zakup drogiego software'u. Systemy osobiste muszą być sprzedawane w dużych ilościach, aby ich produkcja była opłacalna przy niskiej cenie jednostkowej. W konsekwencji systemy osobiste są w znacznym stopniu zunifikowane, tzn. wszystkim grupom użytkowników oferowany jest w zasadzie ten sam program. Czasem producent oferuje dodatkowo warianty typu *professional, extended* czy im podobne.

Systemy przeznaczone dla grup użytkowników, takie jak systemy bankowe czy systemy kontroli produkcji, są w znacznym stopniu systemami jednostkowymi. W tym przypadku cena obejmuje koszt produkcji jądra systemu, jego adaptacji związanej ze specyfiką zastosowania, szkoleń oraz pielęgnacji, rozumianej jako modyfikowanie systemu (także usuwanie błędów) na życzenie klienta.

Systemy przeznaczone dla użytkownika pojedynczego rozwijane są inaczej. Co pewien czas producent oferuje nową wersję, a modyfikacja uzależniona jest co najmniej w takim samym stopniu od sugestii

dotychczasowych użytkowników, co od sytuacji na rynku czy oferty konkurentów.

Odmienny cykl życia systemu powodowany jest także odmiennym kręgiem odbiorców. Decyzję o zakupie systemów wielodostępnych podejmują (lub przynajmniej powinni podejmować) fachowcy zatrudnieni w różnego rodzaju „centrach komputerowych" czy „komórkach ds komputeryzacji". Są oni także odpowiedzialni za rozwój systemu.

Inaczej wygląda sytuacja pojedynczego użytkownika. Podejmując decyzję o zakupie nie posiada on często dostatecznej wiedzy w tym zakresie. Dobry księgowy nie musi znać się na niuansach programów do księgowości, a jego sugestie co do rozwoju software'u, który użytkuje, nie muszą być --- i przeważnie nie są --- uwzględniane przez producenta.

W szczególnie złej sytuacji jest wtedy użytkownik o nietypowych wymaganiach, gdyż producentowi zwyczajnie nie opłaca się dokonywanie postulowanych modyfikacji. Każdy z nas, użytkowników posługujących się innym językiem niż angielski, na własnej skórze doświadczył traktowania po macoszemu przez amerykańskich potentatów produkujących edytory tekstów.

Skoro producenci systemów przeznaczonych dla pojedynczego użytkownika zdają się lekceważyć potrzeby swoich odbiorców, jak wytłumaczyć sukces ekonomiczny wspomnianych wyżej gigantów?

Zanim podejmiemy próbę wyjaśnienia tego fenomenu, zajmijmy się przez chwilę klasyfikacją systemów „osobistych". Podstawowa linia demarkacyjna to podział na systemy interakcyjne i programowalne.

Systemy interakcyjne a systemy programowalne

Przez system **programowalny** rozumiemy system wyposażony w stosunkowo ubogą bazę operacji podstawowych oraz w narzędzia do definiowania operacji złożonych. Przez **system interakcyjny** rozumiemy system komunikujący się z użytkownikiem za pomocą skończonego, z góry ustalonego, zbioru pytań i odpowiedzi.

Zilustrujmy tę różnicę na przykładzie nieco abstrakcyjnym, za to bardzo prostym. Wyobraźmy sobie program realizujący dodawanie liczb naturalnych. System programowalny mógłby być wyposażony w jedną jedyną podstawową operację *dodaj_1* oraz możliwość definiowania operacji złożonych z operacji już zdefiniowanych. Na przykład operacja *dodaj_2* mogłaby być zdefiniowana jako dwukrotne wykonanie operacji *dodaj_1*, operacja *dodaj_3* --- jako złożenie operacji *dodaj_1* i *dodaj_2*, itd. System interakcyjny byłby natomiast wyposażony w operacje *dodaj_1*, *dodaj_2*, *dodaj_3*, aż do --- powiedzmy --- *dodaj_100*; wersja „professional" mogłaby np. mieć jeszcze operacje *dodaj_1000 i dodaj_10000*.

Przykłady systemów interakcyjnych to: system operacyjny Windows, programy graficzne takie jak CorelDRAW! czy Fontographer; przykłady systemów programowalnych to: system operacyjny LINUX, programy graficzne takie jak AutoCad czy METAFONT, system składu tekstów TeX. W tej grupie mieszczą się też klasyczne języki programowania, takie jak Pascal czy C. Systemy składu tekstów w rodzaju PageMakera czy arkusze kalkulacyjne są formami pośrednimi, tzn. systemami interakcyjnymi z elementami programowalności. Do tego tematu jeszcze powrócimy.

Powody, dla których znaczna większość ludzi preferuje systemy interakcyjne, mają przypuszczalnie głębokie podłoże psychologiczne i są — jak się wydaje --- warte głębszej analizy. Nie wdając się wszakże w dywagacje psychologiczne poprzestaniemy na stwierdzeniu faktu: systemy oferujące gotowy zestaw „najbardziej użytecznych operacji" lepiej się sprzedają niż systemy oferujące

użytkownikowi możliwości definiowania operacji według własnych potrzeb, ale za to wymagające inwencji intelektualnej.

I tu wracamy do problemu wdrażania. Otóż współczesne systemy interakcyjne można by określić mianem „samowdrażalnych". Producenci stosunkowo szybko zorientowali się w sytuacji i systemy osobiste zaczęły nabierać ogłady. Bycie programem *user friendly* stało się swoistym kanonem *savoir vivre'u* wśród systemów osobistych. Bieda w tym, że potoczne rozumienie przyjazności dla użytkownika sprowadza się przekonania, że program musi mieć okna, komputer --- mysz, a użytkownik --- co najmniej jeden palec. Tak więc producentom systemów osobistych udało się przekonać swoich klientów, że odwieczne marzenie ludzkości --- być fachowcem niczego się nie ucząc --- jest realizowalne.

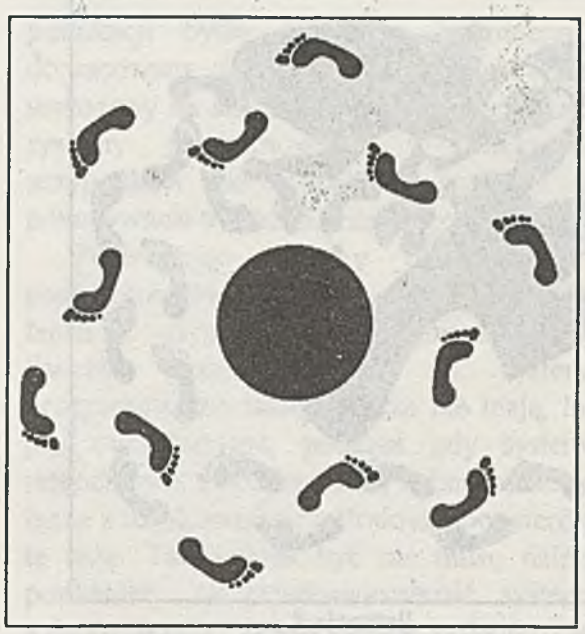
Niestety, wdrażanie systemu nie kończy się na rozpoczęciu pracy. W miarę upływu czasu z reguły okazuje się, że repertuar operacji przewidziany przez producenta nie wystarcza. Ale trudno, oprogramowanie zostało kupione, a użytkownikowi pozostaje czekać na kolejną wersję oprogramowania i modlić się o kompatybilność z wersją poprzednią.

Ponownie zatem należy zapytać, dlaczego użytkownicy wciąż się dają łapać na lep „przyjazności dla użytkownika"? Naszym zdaniem sprzymierzeńcem producentów interakcyjnego oprogramowania jest samo życie. Na szczęście dla nich --- i dla użytkowników --- wiele zadań, jakie dziś przychodzi rozwiązywać ze wspomaganiem komputera (jeżeli nie wręcz większość), to zadania o nieprecyzyjnie sformułowanych celach (p. ilustracje 1--4). Cóż to wpraktyce oznacza?

Weźmy na przykład zadanie zaprojektowania okładki. Grafik pracujący ze wspomaganiem komputera nie musi dbać o to by linie które prowadzi były ściśle równoległe, by proporcje między poszczególnymi elementami rysunku wyrażały się ściśle określonymi liczbami, itd. Mógłby --- ale nie musi. Rysunek, który tworzy, ma wyglądać „jakoś", co bynajmniej nie musi oznaczać „byle jak". W tym sensie cel, jaki grafik przed sobą stawia, może być rozmyty. W takich przypadkach systemy interakcyjne, z gotowymi receptami na wiele z góry przewidzianych przypadków, są niezastąpione. Pierwsze przybliżenie uzyskuje się z reguły bardzo łatwo, a parę prób wystarcza by osiągnąć zadowalający rezultat.

Problemy pojawiają się przy modyfikacjach, będących wprawdzie nieodłącznym atrybutem rzeczywistości, tym niemniej stanowiących utrapienie świadczących usługi komputerowe. W przypadku systemów interakcyjnych przeważnie lepiej jest wykonać rzecz od nowa, niż modyfikować dopiero co wykonany projekt. Na szczęście użytkownikowi systemu interakcyjnego nie przysparza to aż tak wielkiego kłopotu, ze względu na wspomnianą wyżej sprawność systemów interakcyjnych w fazie wstępnej.

Innym problemem, podsuwanym złośliwie przez życie, jest kwestia czynności powtarzalnych, ściśle związana z modyfikacjami. Często aby uzyskać ostateczny rezultat, np. wydruk na drukarce, użytkownik musi wykonać wiele kroków, odpowiedzieć systemowi na pytania o „format wyjścia", „rozmiar strony", „rodzaj urządzenia", itp. Jeśli taki ciąg czynności wykonuje się raz czy dwa razy, wszystko jest w porządku. Natomiast jeśli trzeba go powtórzyć kilkanaście lub kilkadziesiąt razy --- a tak się dzieje w przypadku modyfikacji czy generowania wariantów --- można w końcu stracić cierpliwość. Producenci wyposażają swoje systemy w surogaty w rodzaju plików konfiguracyjnych czy rozwiązań typu *hot-keys,które* może nieco uspokajają użytkowników nie rozwiązują sprawy.



Ilustracja 1

Próba zrealizowania precyzyjnie sformułowanego zadania zapomocą systemu *interakcyjnego* żywo przypomina dreptanie w kółko: cel zostaje dość szybko osiągnięty w przybliżeniu, ale dokładne zrealizowanie wytyczonego celu okazuje się prawie niemożliwe, gdyż poprawka w jednym miejscu psuje coś w drugim; po paru nieudanych próbach w końcu przychodzi poprzestać na etapie mniej więcej pierwszego przybliżenia; na szczęście jest ono stosunkowo łatwo osiągalne i w praktyce użytkownik często się nim zadawala.

Ilustracja 2

Systemy *programowalne* dostarczają środków do realizacji skomplikowanych zadań, tyle że wymagają żmudnej drogi - dopiero po wielu próbach i błędach użtkownik realizuje postawione zadanie; sytuacja bywa czasem psychicznie trudna do zniesienia z tego powodu, że przez długi czas stan prac nie pozwala na uzyskanie choćby namiastki wyniku końcowego, gdyż osiągnięcie nawet pierwszego przybliżenia wymaga z reguły sporego wysiłku, co może sprawiać wrażenie braku postępu prac w początkowej fazie realizacji zadania.

Systemy programowalne niewątpliwie popadły ostatnio w niełaskę. Ciąży nad nimi powszechna opinia, że trzeba być „szamanem", by posługiwać się biegle takimi systemami, podczas gdy posługiwanie się systemami interakcyjnymi jest rzekomo prostsze. W naszym przekonaniu biegłe posługiwanie się którymkolwiek z systemów wymaga zainwestowania sporej ilości czasu w naukę. I to nie w naukę samego systemu, a --- przede wszystkim --- w zrozumienie zagadnienia. Przykładowo w przypadku systemu składu tekstów chodziłoby w pierwszym rzędzie o wiedzę drukarską, bardzo obszerną i wcale nietrywialną.

Systemy programowalne mają swoje bezsprzeczne zalety i --- naszym zdaniem --- ich całkowity upadek, na który się zanosi, oznaczałby niepowetowaną szkodę. Do takich zalet należy łatwość modyfikacji. Jeżeli zadanie zostało **dobrze** zaprogramowane, to uzyskanie rozwiązań wariantowych sprowadzać się może do prostej zmiany parametrów liczbowych. Jest to możliwe dzięki powiązaniu logicznemu pomiędzy poszczególnymi elementami.

Na przykład w programowalnych systemach graficznych proporcje pewnych obiektów graficznych, ich wzajemne położenie, itp., mogą zostać wyrażone za pomocą zależności parametrycznych, co pozwala na nieraz bardzo skomplikowane zmiany całości z zachowaniem ustalonych

Ilustracja 3

Ilstracja 4

Inaczej rzecz się ma w przypadku zadań o celach nieprecyzyjnie określonych, "rozmytych"; w takich razach systemy *interakcyjne* są niezwykle wydajne - przy pewnej dozie szczęścia już pierwsze przyblienie może okazać się zadawalające, a zwykle wystarcza kilka prób (nawet na chybił-trafił) by uzyskać wynik, który można uznać za zupełnie dobry.

W przypadku zadań sformułowanych nieprecyzyjnie systemy *programowalne* skazują użytkownika na pracę na próżno, ze względu na konieczność włożenia nadmiernej energii w uzyskanie pierwszego przybliżenia; systemy *programowalne* nie czynią żadnego użytku z informacji o uproszczeniu zadania, wynikającego z nieprecyzyjnego określnia celu.

zasad wzajemnego pozycjonowania obiektów. Ci, którzy korzystali z programu CorelDRAW!, zgodzą się zapewne, że nawet takie proste zadanie jak pochylenie *(skew)* prostokąta z zachowaniem odległości między jego bokami jest niewykonalne w sposób dokładny, chyba żeby pomóc sobie na boku kalkulatorem. Dzieje się tak dlatego, że CorelDRAW! nie pozwala na nakładanie zbyt skomplikowanych więzów logicznych na poszczególne elementy obiektu graficznego. Nakładanie dowolnych więzów jest możliwe jedynie w systemach programowalnych.

Kolejną zaletą jest możliwość oprogramowania czynności powtarzalnych. Pozwala to na usprawnianie linii technologicznej poprzez przyśpieszenie czynności dających się zautomatyzować oraz poprzez eliminowanie błędów popełnianych przez człowieka prowadzącego zbyt długo konwersację z komputerem. Wyobraźmy sobie, że arkusz kalkulacyjny zawiera stale uaktualniane dane, na podstawie których periodycznie chcemy wydrukować plik tych samych zestawień. Bez programowalności użytkownik arkusza jest skazany na parę godzin beznadziejnego „klikania" myszą. W systemie, gdzie takie zadanie można opisać programem, wystarczy wypisanie nazwy tego programu i udanie się do drukarki po wydruki.

Natomiast podstawową wadą, przesądzającą o — oby chwilowej — klęsce rynkowej systemów programowalnych jest to, że początkowy etap prac jest trudny, wymaga konstruowania schematów logicznych, a pierwsze efekty działań pojawiają się stosunkowo późno. Mamy tu przypuszczalnie do czynienia ze swoistą odmianą znanego prawie każdemu „lęku przed pustą kartką" — rozpoczynając jakąkolwiek pracę, np. pisanie publikacji, szybko chciałoby się mieć widoczne efekty pracy; w przypadku

publikacji byłby to tekst niekoniecznie dopracowany, ale przynajmniej w miarę ładnie rozłożony na stronie. Niestety, współczesne systemy programowalne w większości przypadków nie dostarczają środków do pokonywania trudności wstępnego etapu prac.

Porównując systemy interakcyjne i programowalne nie można nie wspomnieć o łączu z użytkownikiem (*user interface*). Zwykle przyjmuje się, że systemy programowalne takiego łącza nie mają, lub jest ono toporne, podczas gdy systemy interakcyjne wyposażone są w bardzo dobre łącze z użytkownikiem. Praktyka potwierdza tę tezę. Tak jednak być nie musi, należy podkreślić, że programowalność systemu a dobre łącze z użytkownikiem to dwie różne rzeczy.

Zakończenie

Wydaje się, że niektóre z naszych spostrzeżeń są nieobce producentom interakcyjnych programów osobistych. Dość dawno zorientowali się oni, że systemy pozbawione elementów programowalności są nieatrakcyjne dla pewnej grupy użytkowników. W związku z tym zaczęli wyposażać swoje produkty w protezy pozwalające na realizowanie niektórych zadań w sposób programowalny.

Skomplikowane arkusze kalkulacyjne typu EXCELL wydają się zmierzać w kierunku połączenia systemu interakcyjnego z programowalnym. Innym przykładem programowalnych systemów interakcyjnych są znane edytory tekstów np. Emacs (Unix) czy Brief (DOS). W tych systemach, w przypadku gdy standardowa konfiguracja okazuje się dla użytkownika niewystarczająca, ma on możliwość rozbudowy systemu, korzystając ze specyficznego, zorientowanego na konkretne zadanie języka.

Czasami element programowalności jest niezamierzony przez producenta. Tak jest np. w przypadku graficznych programów interakcyjnych posiadających możliwość eksportu obiektów w formacie POSTSCRIPT-owym. POSTSCRIPT, jako język programowania, pozwala zaawansowanym użytkownikom korzystać z dobrodziejstw programowalności.

Ani pierwsza, ani tym bardziej druga forma połączenia systemu interakcyjnego z programowalnym nam nie odpowiada. Naszym marzeniem byłby system pozwalający wszystkie operacje wykonywane w trybie interakcyjnym przetłumaczyć na program — być może w nieco egzotycznym języku. Częścią interakcyjną takiego systemu użytkownik mógłby posługiwać się na tym etapie zadania, na którym działanie interakcyjne jest efektywniejsze od programowania. Cyzelowanie w razie potrzeby mógłby przeprowadzać na poziomie programowania.

Powracając do najbliższych nam systemów graficznych byłoby to na przykład naszkicowanie za pomocą myszy obiektu, który potem na podstawie wygenerowanego przez program kodu można by poprawiać już w trybie programowym. W ten sposób i wilk byłby syty, i owca cała: i tłumy wielbicieli klikania myszą, i ostatni Mohikanie podejścia programowalnego mieliby możliwość korzystania z tych samych systemów. Trudno przecenić możliwe do pomyślenia korzyści płynące z połączenia wysiłków tak diametralnie różnych typów umysłowości.

Jaka będzie przyszłość systemów osobistych? Czy nastąpi konwergencja systemów interakcyjnych i programowalnych? Trudno powiedzieć. Zdając sobie sprawę z ogromu problemów kryjących się w tym przypadku za słowem „konwergencja", jesteśmy raczej pesymistami. Jedynie bowiem presja rynku, co w tym wypadku oznacza istotny wzrost wymagań użytkowników, może wymusić na producentach wysiłki w kierunku stworzenia tego typu hybryd, czego byśmy sobie i innym życzyli.

Edward Janota, Andrzej Trojnar
CSBI Ltd.

Dystrybucja i utrzymanie systemów powielarnych

System powielarny

Celem niniejszego tekstu jest przedstawienie doświadczeń w utrzymaniu systemów powielarnych wytworzonych i dystrybuowanych przez firmę CSBI. Pojęcie systemu powielarnego zostało bogato przedstawione w literaturze, na przykład w pracy [1]. W artykule zostaną przedstawione również narzędzia do wspomagania niezbyt wdzięcznych czynności wymienionych w tytule.

Systemy powielarne o których mowa mają następujące cechy:

- około 50 MB (niektóre więcej) tekstów źródłowych,
- skomplikowane struktury danych, ponad 200 tablic (w sensie relacyjnych baz danych) i średnio ponad 500 indeksów,
- rozmiary baz danych u użytkowników zaczynają się od około 300 MB i "rosną" dość szybko,
- systemy działają w różnych środowiskach sprzętowo-programowych,
- w przypadku niektórych systemów liczba instalacji przekracza 50,
- systemy są utrzymywane w różnych wersjach językowych (dotyczy języków naturalnych, a nie programowania),
- systemy te dotyczą zwykle kilku aspektów działalności instytucji w których pracują.

Przykładem takich systemów dystrybuowanych przez CSBI są:

BANKIER - system bankowy (ponad 150 instalacji), wytworzony w CSBI Ltd,

PRO/MIS - system wspomagania zarządzania (FK, Kadry, Płace, Obrót Towarowy, itd., ponad 50 instalacji), wytworzony w CSBI Ltd.

MFG/PRO- produkt amerykańskiej firmy qad.inc, kompleksowy system wspomagania zarządzania włącznie z produkcją (MRP II), obecnie oferowany w Polsce, największy z wymienionych systemów, bardzo duża liczba instalacji na świecie,

Z problemami opisanymi w tekście spotyka się kilka firm w Polsce i wiele firm na świecie. Przypuszczalnie czeka to jeszcze wielu wytwórców oprogramowania. Mamy nadzieję, że parę poniższych uwag przyda się przy organizowaniu procesów utrzymania i dystrybucji oprogramowania. Z kolei użytkownikom i dystrybutorom systemów powielarnych chcielibyśmy wskazać na parę problemów, z których być może nie zdają sobie sprawy.

Utrzymanie

Tworzenie oprogramowania jest działaniem całkiem przyjemnym do czasu pojawienia się postaci, którą nazywamy użytkownikiem. Osoba ta, lub osoby najczęściej nie wiedzą czego właściwe chcą już przy analizie systemu. Przy jego projektowaniu, kodowaniu, testowaniu i instalowaniu nie uczestniczą, a potem okazuje się że chciały zupełnie czegoś innego. Z drugiej strony z punktu widzenia użytkownika, informatyk, to człowiek który na niczym się nie zna i interesują go tylko ładnie wyglądające formatki i narzędzia programowe o trudnych do wymówienia

nazwach.

Te ścierające się, już w trakcie tworzenia systemu, strony muszą borykać się dalej ze sobą jeszcze długo po zainstalowaniu systemu, to jest w fazie życia aplikacji zwanej **utrzymaniem**.

Dochodzi tutaj zwykle trzecia siła w postaci zmieniającego się otoczenia systemu. Otoczenie to kształtują między innymi:

- rząd, który zmienia przepisy jak rękawiczki zwłaszcza w fazie przechodzenia od socjalizmu do kapitalizmu i z powrotem,
- producenci sprzętu komputerowego, którzy bankrutują albo przestają wytwarzać używane przez nas urządzenia,
- rynek, czyli specjaliści od marketingu, którzy sprzedają napisany przez nas program z nazwą wzbogaconą o przyrostek +, lub ++, ale za to jako dwa produkty. Przecież 200 tysięcy linii kodu bardzo łatwo podzielić na dwie części po 100 tysięcy linii każda.

To wszystko dotyczy oczywiście utrzymania każdego systemu, ale utrzymanie systemu powielarnego zaspokajającego potrzeby wielu różnych użytkowników, pracującego w wielu środowiskach jest procesem bardziej złożonym. Wyszczególnijmy najważniejsze naszym zdaniem problemy. Przy czym opieramy się tu raczej o doświadczenie a nie o teorię.

1. **Liczba żądań zmian** jest wprost proporcjonalna do liczby użytkowników (chyba nie trzeba tego twierdzenia udowadniać).
2. **Duża parametryzacja** systemu powielarnego komplikuje jego konstrukcję tym samym komplikując wprowadzanie zmian.
3. **Utrzymywanie kilku kolejnych wersji** systemu. Nie zawsze jest możliwe zainstalowanie nowszej wersji u użytkownika, bo albo nie ma on możliwości technicznych albo po prostu nie chce on kupić nowej wersji.
4. **Utrzymywanie różnych wariantów oprogramowania**. Poprzez wariant rozumiemy tą samą wersję systemu dla innego użytkownika ale nieznacznie różniącą się funkcjonalnie, na przykład jednym modułem. Kłóci się to trochę z pojęciem systemu powielarnego, ale niestety takie przypadki się zdarzają.
5. **Przenoszenie oprogramowania**. Nawet jeżeli system jest napisany przy użyciu narzędzia (język programowania, DBMS) dostępnego w różnych środowiskach, zawsze w praktyce występują problemy związane z:
   a) instalacją (przykład - nowy kompilator języka na komputer Alpha AXP generuje za długi kod wynikowy),
   b) testowaniem (trudno mieć we własnej firmie oprogramowanie NetWare, Sun'a, VAX'a, AS/400 i inne systemy),
   c) obsługą techniczną, niewielu jest programistów systemowych (administratorów) znających wszystkie dostępne środowiska, a nie zawsze administratorzy u użytkownika są odpowiednio przeszkoleni.
6. **Utrzymywanie (aktualizacja) dokumentacji użytkownika**. Niestety, w fazie utrzymania produktu dokumentacja techniczna (tzn. diagramy z analizy, projekt, specyfikacja wymagań) często nie odpowiada rzeczywistości. Zaradzić na to można dobrą metodyką postępowania przy tworzeniu oprogramowania lub może metodami represyjnymi z "ustawą o przywiązaniu programisty do firmy" włącznie. Ale dokumentacja użytkowa jest zwykle czytana przez klientów i może stanowić jeden z warunków poprawnego wdrożenia systemu, więc jej aktualizacja musi nadążać za zmianami oprogramowania.
7. **Reklamacje i informowanie klienta.**

Problem ten związany jest z tzw. technical support. Często nie wiadomo, zwłaszcza w małych firmach, kto powinien się tym zajmować, czy jest to działalność związana z wdrażaniem systemu czy z jego utrzymaniem. Idealnym rozwiązaniem jest, gdy istnieje przeznaczona do tego osoba (komórka organizacyjna). Pracownik takiej komórki, korzystając z pomocy autorów lub bazy wiedzy (rozwiązanie stosowane w dużych zachodnich firmach [3]) nie tylko udziela informacji klientom ale również weryfikuje reklamacje, oddzielając na przykład zgłoszenia o ewidentnych wadach produktu od pytań typu "czy można się zarazić wirusem komputerowym". Idealnym rozwiązaniem przy systemach powielarnych jest uruchamianie dyżurów telefonicznych tak zwanych "hot line", funkcjonujących przez całą dobę.

8. **Kooperacja**. Zdarza się, że system wytwarzany jest w kilku firmach lub kilku oddziałach jednej firmy. Oprócz tego często użytkownicy tworzą własne moduły współpracujące z systemem powielarnym lub korzystające z jego danych. Oczywiście sprzęgi, procedury przekazywania, struktury danych, itp. definiuje się na etapie wytwarzania, lecz w momencie jakiejkolwiek zmiany utrzymanie tak "rozproszonego" systemu wymaga specjalnych działań organizacyjnych i bardzo wydłuża sprawę w czasie.

Dystrybucja

Podczas przeglądania rożnych modeli cyklu tworzenia oprogramowania rzadko spotyka się fazę o nazwie "Dystrybucja". Być może te czynności zawierają się w innych fazach lub są związane z działalnością czysto handlową, niemniej jednak pomiędzy "wyjściem" systemu z produkcji a jego zainstalowaniem należy wykonać parę czynności. Czasami w anglojęzycznej literaturze spotyka się termin **deployment**, chyba najlepiej odpowiadający omawianemu tu zagadnieniu. Nie wdając się w rozważania akademickie, czy dystrybucja systemu zawiera się w cyklu jego życia, czy nie, możemy powiedzieć, że sprawia ona dużo kłopotów właśnie w przypadku systemów powielarnych. A oto klika problemów, które napotykamy przy dystrybucji:

1. **Problemy logistyczne**. Kojarzy się to trochę z działaniami wojskowymi, ale sytuacja wymaga czasami podejścia militarnego. Przy dużej liczbie instalacji ważnych (krytycznych, ang. critical mission) aplikacji należy koniecznie zadbać o właściwe zapewnienie, znowu używając terminologii wojskowej, sił i środków w postaci:

   - zespołów wdrożeniowych,
   - specjalistów od instalacji, zwłaszcza w różnego rodzaju środowiskach sieciowych,
   - transportu i łączności (m.in. zdalne łącza komputerowe)
   - nośników lub przenośnych komputerów.

   Jeżeli ktoś uważa, że przesadzamy to kilka przykładów (niektóre są autentyczne):

   - system płacowy "wiesza się" z błędem dwa dni przed wypłatą u klientów w czterech województwach szczecińskim, suwalskim, jeleniogórskim i krośnieńskim,
   - następuje zmiana sposobu naliczania kredytu w banku, który posiada kilkanaście oddziałów na terenie kraju a wymiana oprogramowania musi być dokonana w trakcie weekendu,
   - w wyniku testów wykryty zostaje błąd polegający na złym naliczaniu pewnych wartości w przypadku zmiany roku, błąd ten znajduje się we

wszystkich instalacjach, a został wykryty w tygodniu pomiędzy Bożym Narodzeniem a Nowym Rokiem.

2. **Problemy kompletowania wersji**. W przypadku dużej liczby klientów i kilku utrzymywanych wersji należy zapamiętać, co który klient posiada. Staje się to jednak zawiłe w przypadku gdy systemy dystrybuowane (sprzedawane) są w podziale na moduły, na różne środowiska, w różnych wersjach językowych, itp. Liczba kombinacji rośnie wtedy bardzo szybko. Pomijając sytuację, że użytkownik końcowy dostanie produkt inny lub niekompletny może się zdarzyć sytuacja, że dostanie coś za co nie zapłacił, co być może go nie zmartwi, ale producenta naraża na straty.

4. **Redystrybucja**. Coraz częściej na naszym rynku systemy sprzedawane są przez firmy nie będące producentami (resellerów). Osoby te reprezentują producenta u użytkownika końcowego i w tym przypadku należy zapewnić specjalne procedury postępowania przy przekazywaniu oprogramowania jak również kontroli produktów wychodzących od redystrybutora. Ważnym jest, aby redystrybutor zapewniał odpowiedni poziom usług wdrożeniowych i serwisowych dla systemu. Oczywiście zdarzają się sytuacje, że pośrednik potrafi lepiej obsłużyć użytkownika niż producent.

5. **Zdalna dystrybucja**. Coraz powszechniejsze w kraju sieci rozległe pozwalają na instalowanie, testowanie i aktualizację produktów bezpośrednio u klienta. Pomijając kwestię porównania kosztów, czasu i jakości transmisji z kosztami wyjazdu pracownika lub przesyłania nośników, należy zwrócić uwagę na różne aspekty sprawy, na przykład:

   - protokołowania transmisji od strony producenta - zdalna poprawka musi być w jakiś sposób zapamiętana i potwierdzona w systemie dystrybucji, niezależnie czy jest to system ręczny czy komputerowy,

   - potwierdzenia transmisji od strony klienta - użytkownik powinien w jakiś sposób potwierdzić, że zmiana (instalacja) została dokonana,

   - zabezpieczenia transmisji i zdalnego dostępu do zasobów klienta, co ma szczególne znaczenie przy systemach bankowych

   Dobrym sposobem na "podpisywanie" raportów są wszelkie systemy e-mail. Powyższe uwagi trącą trochę biurokracją, ale pamiętajmy, że systemy powielarne są kosztowne i zazwyczaj nie są sprzedawane z półki, tylko stanowią przedmiot kontraktu z gwarancjami, umowami serwisowymi i wszelkimi tego prawnymi konsekwencjami.

Propozycje rozwiązań

Istnieje wiele sposobów (metodyk) i narzędzi usprawniających system utrzymania produktów. Systemy takie nazywane bywają czasami Systemami Zarządzania Konfiguracją (ang. Configuration Management System). Przykładem może być IBM'owski pakiet CMVC - Configuration Management Version Control dostępny na Unix'owych stacjach roboczych.

Jednym z bardziej popularnych jest tak zwany SCCS (Source Code Control System) obecny w systemach typu UNIX, pozwalający na utrzymywanie wielu wersji oprogramowania w kodzie źródłowym.

W systemach używających baz danych

pojawia się jednak nowy problem, mianowicie kwestia utrzymania struktur danych i panowania nad ich zmianami. Systemy zarządzania bazami danych pamiętają swoje definicje (metadane) w różnych dedykowanych formatach, dlatego niewiele jest uniwersalnych narzędzi do konserwacji zmian struktur danych.

Dobrym wyjściem jest trzymanie wszystkich definicji używanych w utrzymywanym systemie w osobnej bazie danych, tak zwanym repozytorium. Bazy takie dostarczane są zazwyczaj z dostępnymi na rynku narzędziami CASE. W systemie operacyjnym VMS firmy DEC, takie repozytorium dostarczane jest niezależnie (od narzędzi CASE) pod nazwą CDD - Common Data Dictionary) i umożliwia "pamiętanie" i obsługę wszelkiego rodzaju obiektów używanych w różnych pakietach systemu VMS tj. formatek, definicji tablic baz danych, definicji rekordów plików indeksowo-sekwencyjnych, raportów, itp. W systemie VMS dodatkowo dostarczany jest pakiet pod nazwą MMS - Module Management System pozwalający na konfigurowanie produktów w różnych wersjach.

W momencie, w którym mamy kilka wersji systemu u różnych klientów pożądane byłoby mieć system łączący **utrzymanie z rozprowadzaniem oprogramowania**. Można to uzyskać przez "złożenie" bazy danych klientów z bazą opisującą produkt. Podejście takie przydaje się zwłaszcza w początkowym okresie życia systemu gdzie ilość błędów jest duża lub przy szybko zmieniającym się otoczeniu systemu, o czym już była mowa wcześniej. W poszukiwaniu takiego rozwiązania napotkaliśmy interesujący pakiet ROUNDTABLE amerykańskiej firmy Roundtable Software Inc. przeznaczony do wspomagania zarządzania i dystrybucji systemów napisanych w PROGRESS'ie (nasza firma posługuje się tym narzędziem). Pakiet ten ma następujące funkcje:

- zarządzanie konfiguracją - kontrola wersji programów, dokumentacji i struktur danych,
- obsługa dystrybucji, włącznie z generowaniem programów i skryptów dla systemów operacyjnych pozwalających na automatyczną instalację,
- zarządzanie zadaniami - zadanie w postaci modułu programowego przyporządkowane jest do odpowiedniej grupy użytkowników (programistów, testerów, kierowników) i tylko oni mają odpowiednie prawa do modyfikacji projektu, co pozwala uzyskać kontrolę nad stopniem realizacji zadań.

W końcu jednak zdecydowaliśmy się na własne rozwiązanie.

System PROWERS

Po analizie omawianych zagadnień i możliwości różnych narzędzi zaprojektowaliśmy system o nazwie PROWERS, przeznaczony do aplikacji pisanych w języku PROGRESS 4GL i z użyciem tego samego (PROGRESS) systemu zarządzania bazą danych, chociaż istnieje możliwość dołączenia obsługi baz danych typu Oracle (wersja 6 i 7), Sybase oraz Rdb/VMS. Pozwolimy sobie go pokrótce przedstawić.

1. Analiza problemu

Fragment cyklu życia dowolnej dystrybuowanej i utrzymywanej powielarnej aplikacji przedstawiony jest na rysunku numer 1. Rysunek został sporządzony jako diagram przepływu danych (Data Flow Diagram) w notacji Gane-Sarson. Z punktu widzenia analizy strukturalnej przedstawia on tzw. **aktualny model fizyczny systemu dystrybucji**. System PROWERS został zaprojektowany z użyciem własnej metodyki CSBI i pakietu wspomagającego EasyCASE. Oprogramowanie zostało napisane w języku PROGRESS 4GL.

System utrzymania i dystrybucji oprogramowania składa się z wzajemnie powiązanych procedur (procesów), przepływów danych i składnic danych.

Procedury - odpowiadają różnym obszarom działań niezbędnych do podjęcia w czasie rozwoju i utrzymania systemu. Procedury otrzymują i przesyłają dane za pośrednictwem przepływów danych. Poniżej zamieszczony jest wykaz procedur (procesów) zaznaczonych na diagramie literą P oraz numerem procedury:

P1. Weryfikacja uwag
P2. Rozwój
P3. Utrzymanie
P4. Testowanie
P5. Generowanie instalacji
P6. Kompletacja wersji
P7. Dystrybucja
P8. Instalacja

Procesy S1 - Sprzedaż i S2 - Wdrożenie zostały w opisie pominięte, ponieważ nie są objęte działaniem systemu.

Przepływy - opisują strumienie danych o określonej zawartości przepływające pomiędzy dwoma obiektami. Typowe strumienie danych (informacji) to na przykład propozycje zmian "płynące" od klienta, czy biuletyn przekazywany VAR-om. Przepływy danych (informacji) które będą reprezentowane przez specjalne formularze zostały oznaczone literą F:

F1. Propozycje
F2. Reklamacje
F3. Nowa wersja
F4. Raport z testowania
F5. Dokumentacja
F6. Korekty
F7. Poprawki
F8. Aktualna wersja
F9. Zmiany
F10. Licencja
F11. Wersja dla użytkownika
F12. Raport z instalacji
F13. Poprzednia wersja

Pozostałe przepływy takie jak zamówienia czy umowy nie mające stałej formy zostały tylko zaznaczone. Przepływy są wymienione przy opisie wejść do procedur.

Składnice - reprezentujące miejsca, gdzie dane są przechowywane pomiędzy procesami, które na nich operują. Składnice są dostępne z procesów, które mogą działać na danych w nich zawartych. Składnicą może być teczka z dokumentami, archiwum czy zbiór raportów z rozmów z klientem. Składnice oznaczono literą D. Składnice są wymienione przy opisie wyjść z procedur:

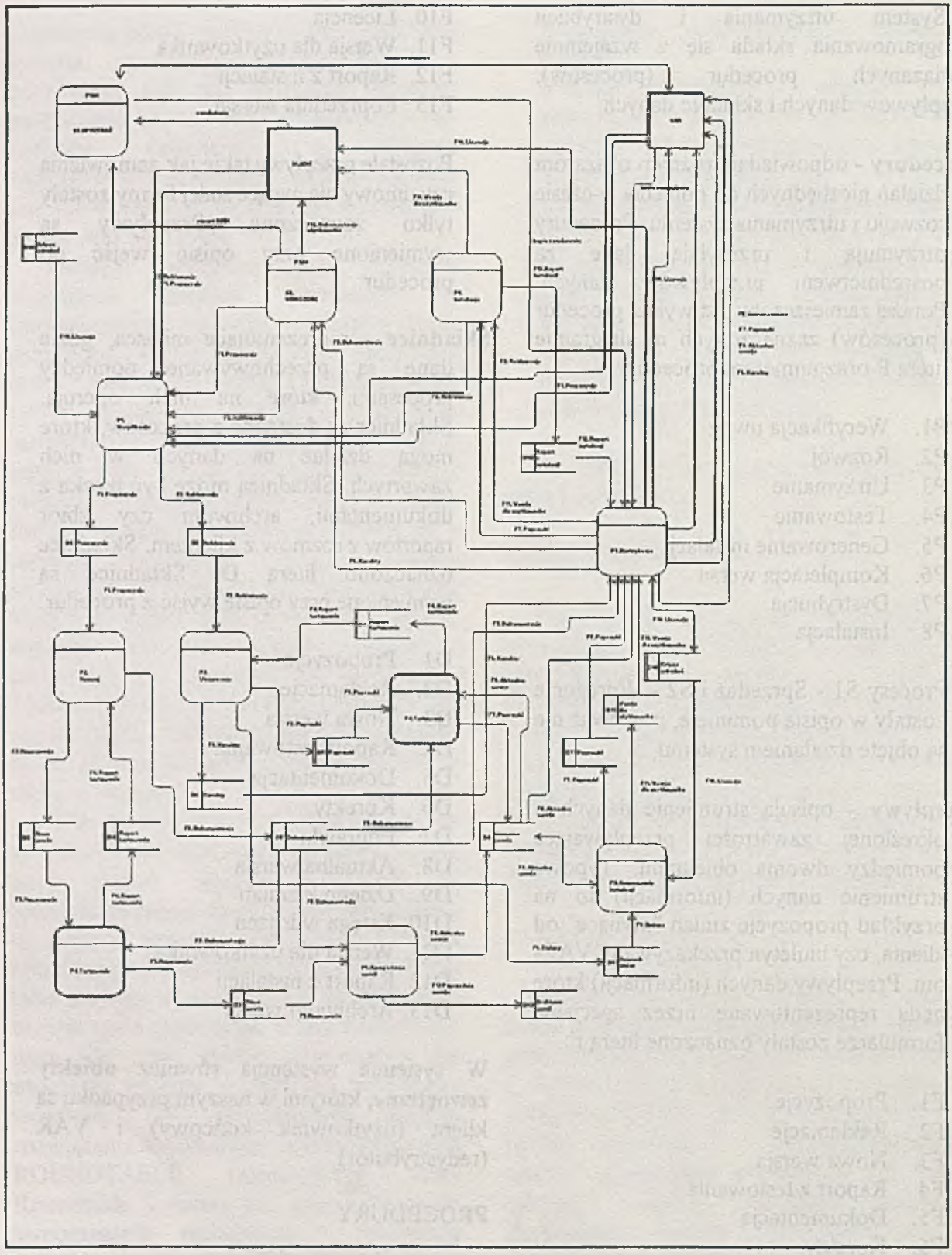
D1. Propozycje
D2. Reklamacje
D3. Nowa wersja
D4. Raport testowania
D5. Dokumentacja
D6. Korekty
D7. Poprawki
D8. Aktualna wersja
D9. Dziennik zmian
D10. Księga wdrożeń
D11. Wersja dla użytkownika
D12. Raport z instalacji
D13. Archiwum wersji

W systemie występują również **obiekty zewnętrzne**, którymi w naszym przypadku są klient (użytkownik końcowy) i VAR (redystrybutor).

PROCEDURY

Procedura P1 - Weryfikacja reklamacji

Procedura służy do weryfikacji dokumentów



Rysunek 1.

(telefonów) spływających od użytkownika systemu zwanego dalej klientem oraz od osób zajmujących się wdrażaniem i sprzedażą. Dokumenty te obejmują reklamacje (zauważone usterki) oraz propozycje zmian. Weryfikacji podlegają również reklamacje i propozycje pochodzące od VAR-ów.

Procedura P2 - Rozwój

Procedura przedstawiona na diagramie dotyczy tylko fragmentu pracy działu rozwoju oprogramowania dotyczącego rozpatrywania propozycji klientów. Propozycje rozpatrywane są na protokołowanych zebraniach zespołu i wchodzą w zakres projektu następnych wersji.

Procedura P3 - Utrzymanie

W tym miejscu dokonywane są poprawki błędów wykryte podczas testowania, instalacji, wdrożenia oraz normalnej eksploatacji.

Procedura P4 - Testowanie

Procedura występuje dwukrotnie na diagramie, jako że testowanie dotyczy zarówno nowej wersji jak i poprawek do wersji aktualnej. Procedura jest wykonywana cyklicznie aż do uzyskania wolnego od błędów raportu testowania. Celem procedury jest wykrycie jak największej liczby błędów a nie wykazanie, że program błędów nie zawiera. Testowanie przeprowadza się zgodnie z planem testów obowiązującym w firmie wzorowanym na normach amerykańskich IEEE, dotyczących tzw. SQAP (Software Quality Assurance Plan).

Procedura P5 - Generowanie instalacji

Procedura służy przygotowaniu wersji dla użytkownika według jego zamówienia zgodnego z licencją. Tutaj też przygotowuje się bieżące poprawki do zainstalowanych produktów w postaci specjalnych zestawów nośników.

Procedura P6 - Kompletacja wersji

Procedura ma na celu przygotowanie poprawnej wersji oprogramowania, opisania jej, nadania numeru wersji oraz zarchiwowania.

Procedura P7 - Dystrybucja

W ramach tej procedury przygotowywane są zestawy dystrybucyjne dla konkretnych odbiorców.

Procedura P8 - Instalacja

Szczegółowy sposób instalacji powinien znajdować się w odpowiedniej dla aplikacji instrukcji, część procedury P8 to specjalne programy komputerowe.

PRZEPŁYWY DANYCH

Niektóre przepływy danych będą miały swoje fizyczne odpowiedniki w postaci ściśle zdefiniowanych dokumentów na przykład propozycje (patrz rysunek nr 2). W przypadku "zmaterializowanych" przepływów danych, takich jak na przykład oprogramowanie na nośniku i dokumentacja, istnieje dokument ściśle specyfikujący zawartość nośnika.

SKŁADNICE DANYCH

Dla przykładu opiszemy dwie składnice.

Składnica D9 - Dziennik zmian

Tutaj znajdują się wszystkie adnotacje o zmianach źródeł, struktur danych, parametrów i dokumentacji.

Składnica D10 - Księga wdrożeń

Tutaj pamięta się, jakie licencjonowane

F1	System PROMIS	Propozycje zmian

Prosimy Państwa o przesłanie nam Waszych uwag odnośnie rozbudowy systemu PROMIS, zwłaszcza w zakresie Państwa potrzeb. Prosimy również, o podzielenie się z nami doświadczeniami w zakresie eksploatacji modułów systemu. Postaramy się uwzględnić Państwa uwagi w kolejnych wersjach PROMISa. Propozycje prosimy kierować na adres CSBI w Katowicach lub w Warszawie, z dopiskiem "PROMIS". Reklamacje wymagające bieżących poprawek prosimy wysyłać na formularzach F2.

Numer licencji:	Data:

Uwagi:

Rysunek 2. Formularz propozycji zmiany.

produkty posiada klient, kiedy były instalowane, przez kogo i z jakim wynikiem.

2. Realizacja

Opisanie w artykule programu jest sprawą dość trudną, ograniczymy się więc do opisania struktury danych i wymienienia głównych funkcji w postaci schematu strukturalnego (STC - Structure Chart) przedstawiającego układ menu programu.

Rysunek nr 3 przedstawia diagram związków obiektów systemu (ERD - Entity Relationship Diagram). Podstawowym obiektem jest WERSJA dowolnego systemu.

Spójrzmy na system "od strony" dystrybucji. W obrębie wersji istnieją tak zwane GENERACJE, powstające w momencie gdy zostanie wykryty błąd uniemożliwiający pracę systemu, lub system nieznacznie się zmieni (bez zmian funkcjonalnych). Poszczególne GENERACJE różnią się przyrostami parametrów, plików (tablic bazy danych) i procedur. Zmiany dokumentacji obrazuje obiekt KOREKTY DOKUMENTACJI. Każda GENERACJA podlega INSTALACJI. INSTALACJA posiada takie atrybuty jak data, liczbę nośników, itp. INSTALACJĘ opisuje RAPORT. W bazie danych RAPORT pamiętany jest jako rekord wskazujący na plik systemu operacyjnego (np. w formacie ASCII lub WordPerfect) z TREŚCIĄ raportu. RAPORT sporządzany jest przez UŻYTKOWNIKA systemu PROWERS (nie mylić z KLIENTEM), który może być wdrożeniowcem, instalatorem, programistą, itp. INSTALACJA związana jest z LICENCJĄ, najważniejszym chyba dokumentem z punktu widzenia KLIENTA. KLIENT może posiadać LICENCJE na różne (wiele) produkty i na różne ŚRODOWISKA (Unix, DOS, NetWare). PRODUKT jednoznacznie wskazuje na MODUŁ systemu i tutaj wchodzimy w sferę utrzymania.

Od "strony" utrzymania pamiętane są wszystkie BAZY danych wraz z tablicami - obiekt PLIKI ich powiązania z MODUŁAMI. Zakładamy, że system składa się z modułów np. Kadry, FK, Kredyty. MODUŁ składa się z DOKUMENTACJI, PARAMETRÓW i PROCEDUR. Zmiany tych części MODUŁU opisują obiekty oznaczone na diagramie słowem Delta. Każdy MODUŁ tworzony jest przez AUTORA, który zawsze jest UŻYTKOWNIKIEM SYSTEMU. W tym miejscu określone są kompetencje osób korzystających z systemu PROWERS. Inne uprawnienia mają twórcy aplikacji a inne osoby wdrażające. Jeszcze inne ma Administrator, który odpowiedzialny jest za generowanie nowych WERSJI lub GENERACJI aplikacji.

System PROWERS pozwala na rejestrację i kontrolę realizacji REKLAMACJI od KLIENTÓW, o ile posiadają oni ważną LICENCJĘ na PRODUKT. System automatycznie kieruje REKLAMACJE do AUTORÓW, korzystając z poczty elektronicznej systemu operacyjnego.

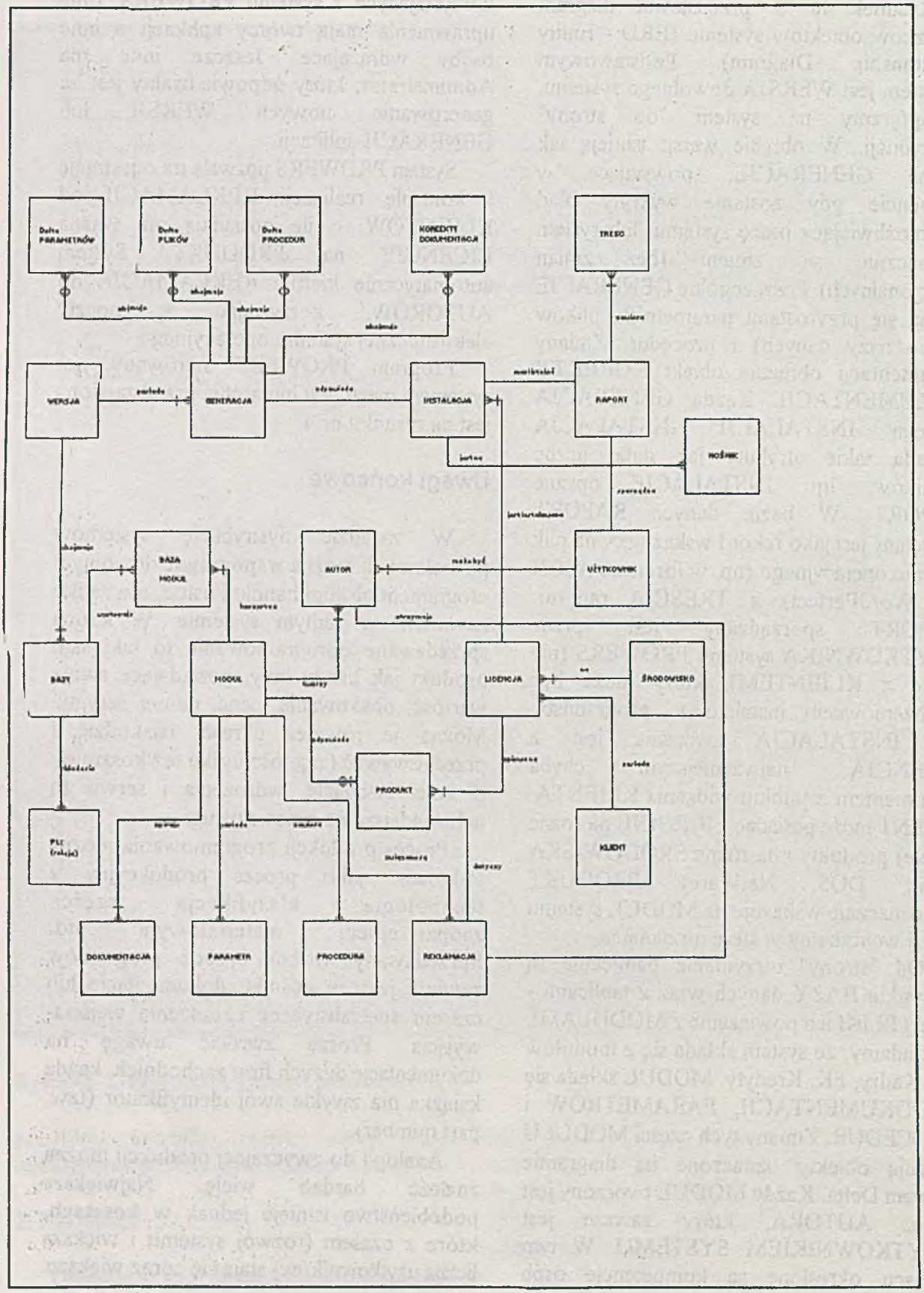
Program PROWERS sterowany jest systemem menu. Ich hierarchia przedstawiona jest na rysunku nr 4.

Uwagi końcowe

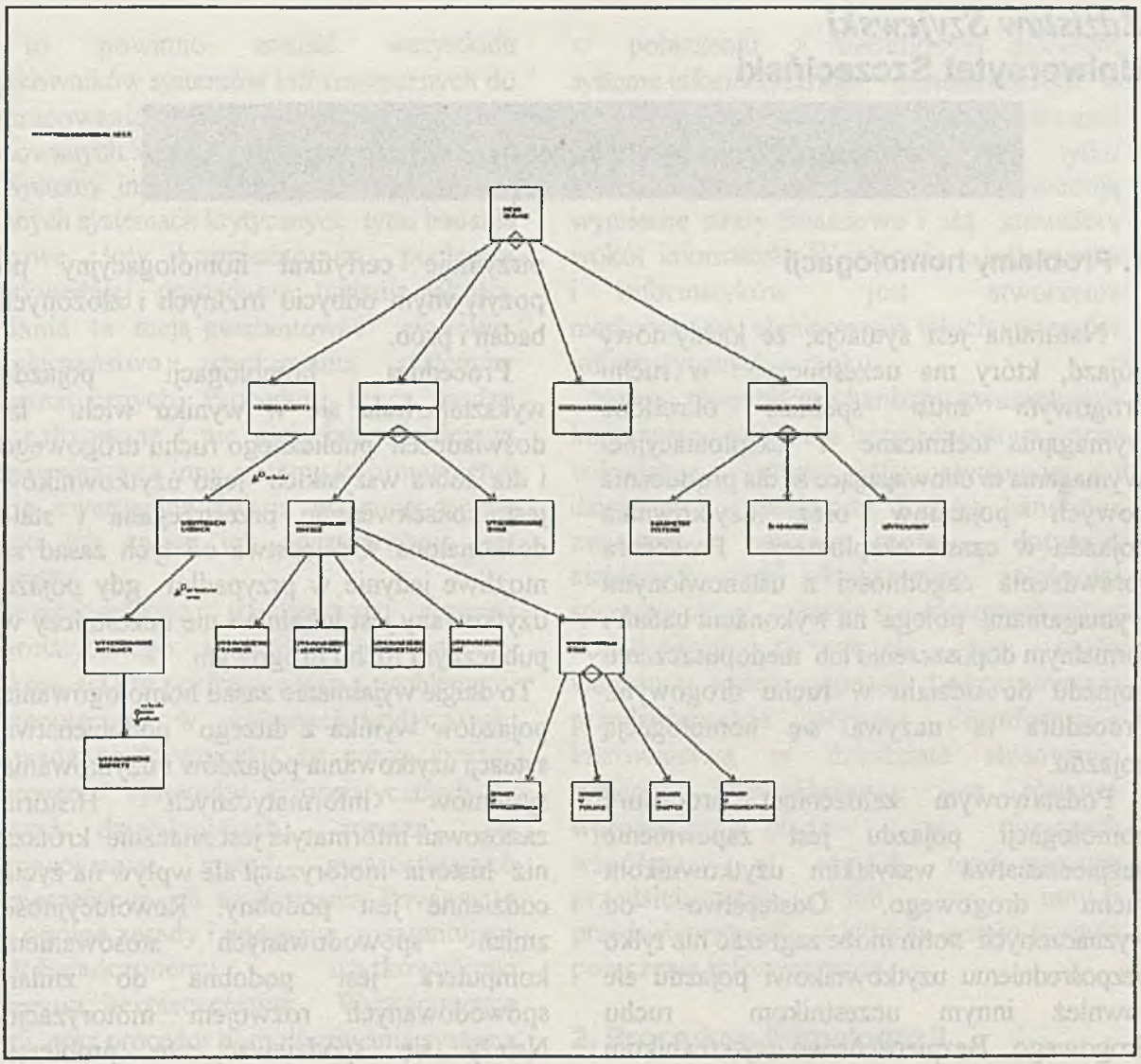
W zasadzie dystrybucję systemów powielarnych można **wspomagać** dowolnym programem obsługi handlu i usług, nierzadko zawartym w samym systemie. W końcu sprzedawane oprogramowanie to taki sam produkt jak każdy inny, posiadające swoją wartość, opakowanie, cenę, numer seryjny. Można je również ukraść, uszkodzić i przechowywać (zajętość dysku też kosztuje). Z kolei instalacje, wdrożenia i serwis to najzwyklejsze na świecie usługi.

Proces produkcji programowania można traktować jako proces produkcyjny z technologią, klasyfikacją części, zaopatrzeniem materiałowym itd. Sprzedawany system oprócz programów zawiera jeszcze nośniki, dokumentację lub czasem specjalistyczne urządzenia wejścia-wyjścia. Proszę zwrócić uwagę na dokumentację dużych firm zachodnich, każda książka ma zwykle swój identyfikator (tzw. part number).

Analogii do zwyczajnej produkcji można znaleźć bardzo wiele. Największe podobieństwo istnieje jednak w **kosztach**, które z czasem (rozwój systemu i większa liczba użytkowników) stają się coraz większe.


Dane PARAMETRÓW
Dane PLIKÓW
Dane PROCEDUR
KONCEPCYJNA DOKUMENTACJA
TREŚĆ
WERSJA
GENERACJA
INSTALACJA
RAPORT
NOŚNIK
BAZA MODUŁ
AUTOR
UŻYTKOWNIK
BAZY
MODUŁ
LICENCJA
ŚRODOWISKO
PRODUKT
PLIK (relacja)
KLIENT
DOKUMENTACJA
PARAMETR
PROCEDURA
REKLAMACJA

Rysunek 3.



Rysunek 4.

Dlatego o problemach utrzymania i dystrybucji należy pamiętać już przy planowaniu i analizie przedsięwzięcia jakim jest system powielarny i dlatego chyba warto stosować oprogramowanie wspomagające.

Literatura.

[1] Łukasik-Makowska B., "Informatyczne systemy powielarne. Standaryzacja. Weryfikacja. Wdrażanie", PWE Warszawa 1992.

[2] Roundtable Software Inc., Dokumentacja systemu ROUNDTABLE

[3] Sperry S.L., "Help is on the way", Byte, July 1993

Zdzisław Szyjewski
Uniwersytet Szczeciński

Homologacja systemów informatycznych

1. Problemy homologacji

Naturalna jest sytuacja, że każdy nowy pojazd, który ma uczestniczyć w ruchu drogowym musi spełniać określone wymagania techniczne i eksploatacyjne. Wymagania te obowiązujące są dla producenta nowych pojazdów oraz użytkownika pojazdu w czasie eksploatacji. Procedura sprawdzenia zgodności z ustanowionymi wymaganiami polega na wykonaniu badań i formalnym dopuszczeniu lub niedopuszczeniu pojazdu do udziału w ruchu drogowym. Procedura ta nazywa się homologacją pojazdu.

Podstawowym założeniem procedury homologacji pojazdu jest zapewnienie bezpieczeństwa wszystkim użytkownikom ruchu drogowego. Odstępstwo od wyznaczonych norm może zagrażać nie tylko bezpośredniemu użytkownikowi pojazdu ale również innym uczestnikom ruchu drogowego. Bezpieczeństwo użytkownikom mają gwarantować zasady ruchu drogowego oraz sprawność techniczna pojazdu. Sprawność techniczna oceniana jest na podstawie zgodności z ogólnymi zasadami (ilość i rodzaj świateł, minimalne wyposażenie dodatkowe itp.) oraz badane są niektóre parametry eksploatacyjne pojazdu (siła hamowania, czystość spalin itp.).

Homologacja nowych pojazdów spoczywa na producencie, który ma obowiązek używać homologowanych części oraz określonych procedur i urządzeń wytwarzania. Wymagania stawiane producentowi nie są definiowane szczegółowo a dotyczą tylko newralgicznych dla produktu końcowego części i procedur produkcji. Nowy model czy typ pojazdu otrzymuje certyfikat homologacyjny po pozytywnym odbyciu trudnych i złożonych badań i prób.

Procedura homologacji pojazdu wykształtowała się w wyniku wielu lat doświadczeń publicznego ruchu drogowego i dla dobra wszystkich jego użytkowników jest konsekwentnie przestrzegana i stale doskonalona. Odstępstwa od tych zasad są możliwe jedynie w przypadku gdy pojazd użytkowany jest lokalnie i nie uczestniczy w publicznym ruchu drogowym.

To długie wyjaśnianie zasad homologowania pojazdów wynika z dużego podobieństwa sytuacji użytkowania pojazdów i użytkowania systemów informatycznych. Historia zastosowań informatyki jest znacznie krótsza niż historia motoryzacji ale wpływ na życie codzienne jest podobny. Rewolucyjność zmian spowodowanych stosowaniem komputera jest podobna do zmian spowodowanych rozwojem motoryzacji. Należy się spodziewać, że problemy porządkowania zasad dopuszczania do użytkowania i organizacja użytkowania w publicznych sieciach informatycznych systemów informatycznych, będą podobne. W celu uniknięcia błędów warto skorzystać z doświadczeń motoryzacji i adoptować niektóre procedury na grunt informatyki.

Ilość komputerów oraz dziedziny życia, w których systemy informatyczne są stosowane powoduje, że należy zająć się porządkowaniem procedur organizacyjnych regulujących dopuszczenie do użytkowania i kontrolę procesu eksploatacji. Możliwe do wystąpienia zagrożenia wynikające z korzystania z niesprawnych systemów informatycznych stają się coraz bardziej realne

i to powinno zmusić wszystkich użytkowników systemów informatycznych do wypracowania zasad podobnych do stosowanych w ruchu drogowym.

Systemy informatyczne stosowane w tak zwanych systemach krytycznych typu badania jądrowe, loty kosmiczne itp. podlegają odpowiedniej procedurze badania jakości. Badania te mają gwarantować wysokie bezpieczeństwo użytkowania systemów informatycznych. Procedury te są bardzo sformalizowane i nie mają bezpośredniego przeniesienia na inne systemy informatyczne, gdzie ewentualna awaria ma mniej groźne skutki lub zasięg ich oddziaływania jest mniejszy.

Bezpieczeństwo użytkowania systemu informatycznego w służbie zdrowia czy bankowości jest porównywalne z problemami występującymi w systemach krytycznych. Prowadzi to do wniosku, że coraz szersze stosowanie systemów informatycznych w różnych działalnościach zmusza do wypracowania metod gwarantujących bezpieczeństwo ich użytkowania. Powinny to być ogólne zasady i wytyczne gwarantujące niedoświadczonemu użytkownikowi minimum bezpieczeństwa. Wypracowanie norm oraz procedur homologowania systemu informatycznego staje się koniecznością.

Systemy informatyczne stosowane są w coraz bardziej newralgicznych działach życia gospodarczego. Wmontowanie systemów informatycznych w procedury działalności gospodarczej powoduje uzależnienie gospodarki od sprawności i jakości systemów informatycznych. W pierwszej kolejności należałoby wypracować procedurę homologowania systemów wykorzystywanych w działalności finansowo-księgowej. Mnogość systemów finansowo-księgowych na rynku, w połączeniu z nienajwyższą jakością niektórych z nich, powoduje poważne zagrożenie bezpieczeństwa informacji. Duże nakłady poniesione na sprzęt komputerowy w połaczeniu z nietrafionym zakupem systemu informatycznego, zastosowanego w newralgicznej dla każdego przedsiębiorstwa działalności księgowej, są nie tylko nietrafioną inwestycją ale powodują wymierne straty finansowe i złą atmosferę wokół informatyki. W interesie użytkownika i informatyków jest stworzenie mechanizmów eliminowania takich systemów informatycznych z rynku.

Należy stworzyć mechanizmy gwarantujące kupującemu minimum bezpieczeństwa przy wdrożeniu systemu informatycznego do działalności gospodarczej. Ryzyko handlowe związane z zakupem może dotyczyć niektórych cech jakościowych, solidności sprzedawcy w zakresie świadczonych usług dodatkowych i serwisu ale nie może wdrożony system zagrażać funkcjonowaniu przedsiębiorstwa poprzez dezinformację kierownictwa w dziedzinie stosowania systemu. Niewykluczone jest również wprowadzania błędów w obszarach współpracy w ramach tego samego przedsiębiorstwa lub w innych przedsiębiorstwach, z którymi system posiada połączenia informatyczne.

2. Procedura homologacji

Działalność homologacyjna wymusza wprowadzenie niezbędnej biurokracji związanej z procedurą sprawdzania zgodności z wymaganiami. Każda biurokracja zmusza do sformalizowania procedur i tworzenia niezbędnej dokumentacji. Działania te będą niewątpliwie uciążliwe dla informatyków, którzy nigdy nie lubili dobrze dokumentować swoich prac ale wymuszą opracowanie dokumentacji systemu i to według jednolitej metodyki. Bedzie to dodatkowa korzyść z wprowadzenia procedur homologacji. Standard dokumentacji na potrzeby homologacji może stanowić zaczyn porządkowania problemów metodycznych, które od dawna wymagają

unormowania. Prezentacja systemów na różnorodnych targach informatycznych czy nieliczne jeszcze przetargi na systemy informatyczne, pokazują, że brakuje standardu niezbędnego minimum dokumentacyjnego. Brak dobrej dokumentacji nie jest tylko usterką techniczną ale może być poważnym utrudnieniem w prawidłowej ocenie jakości systemu. Nieporównywalność dokumentacji jest równoznaczna z brakiem możliwości porównania systemów, co utrudnia podejmowanie decyzji wyboru produktu. Brak możliwości jednoznacznej oceny zwykle premiuje produkty słabszej jakości.

Samodzielnym problemem staje się definicja wymagań jakie powinien spełniać system informatyczny. Wymagania te powinny dotyczyć niezbędnego minimum bez wnikania w szczegółowe metody realizacji, które mogą stanowić tajemnicę handlową producenta. Specyfikacja wymagań może więc dotyczyć jedynie oznak zewnętrznych funkcjonowania systemu a nie szczegółowych rozwiązań informatycznych.

Wymagania te powinny być zgrupowane w trzy działy tematyczne:

- organizacyjno-prawne,
- merytoryczne,
- technologii informatycznej.

Wymagania organizacyno-prawne powinny określać jakie uwarunkowania organizacji prac systemu informatycznego muszą być zgodne z przepisami ogólnymi. Rozwiązania systemu informatycznego muszą być zgodne z obowiązującym prawem regulującym daną dziedzinę działalności. Wymagania te będą różne dla różnych klas systemów informatycznych. Ta grupa wymagań powinna być opracowywana odrębnie dla każdej klasy systemu i w sposób ciągły modyfikowana w miarę wprowadzania nowych uregulowań prawnych.

Wymogiem prawnym powinna być konieczność używania przez producenta systemu informatyznego tylko licencjonowanego oprogramowania operacyjnego i narzędziowego wykorzystywanego przy produkcji oprogramowania. System otrzymujący certyfikat homologacji staje się równocześnie prawnie chronionym. Procedura homologacji będzie również sprzyjała standaryzacji rozwiązań organizacyjnych i prawnych nie tylko w informatyce ale również w dziedzinach stosowania homologowanych systemów.

Wymagania merytoryczne określają podstawowe zasady jakie musi spełniać algorytm realizowany przez system informatyczny. Określenie zasadniczych ram funkcjonalnych nie ogranicza inwencji autorów rozwiązań a jedynie reguluje oznaki zewnętrzne funkcjonowania systemu. W przypadku systemu finansowo-księgowego są to zasady księgowania i prowadzenie ewidencji księgowej przez system informatyczny tak aby zapisy tworzone i przechowywane w systemie gwarantowały możliwość kontroli i uzyskania niezbędnej wymaganej przepisami statystyki. Wymogi te powinny być uzgodnione z odpowiedzialnymi jednostkami administracji państwowej i stowarzyszeniami branżowymi.

Wymagania z zakresu technologii informatycznej powinny się koncentrować na problemach bezpieczeństwa informacji. Bezpieczne powinny być dane konkretnego systemu ale równocześnie istotne jest aby w przypadku błędnego funkcjonowania nie powodowana była awaria w otoczeniu informatycznym. Problem jest szczególnie groźny w przypadku systemów użytkowanych w sieciach komputerowych. Im szerszy zasięg sieci tym ostrzejsze powinny być wymagania ochronne.

Procedura testowania systemów najczęściej opracowywana jest przez producenta, co nie gwarantuje wykrycia wszystkich usterek. Opracowanie standardowych testów przez

niezależny zespół ekspertów może istotnie wpłynąć na jakość oprogramowania systemów. Testowanie w sytuacjach krańcowych zapobiegnie awariom występującym długo po rozpoczęciu eksploatacji i okresie gwarancji producenta. Działania te będą chronić użytkownika systemu przed niesolidnością producenta oraz grożnymi dla funkcjonowania przedsiębiorstwa usterkami systemów informatycznych.

3. Administrowanie procedurą homologacji

Na potrzeby homologacji powinny powstać niezbędne dokumenty homologacyjne, z których pierwszym i najistotniejszym powinien być zbiór wymagań opracowany odrębnie na potrzeby każdej homologowanej klasy systemów informatycznych. Dokument udostępniany producentom systemów informatycznych powinien stanowić pierwszy krok procedury homologacji. Dokument ten w swej podstawowej wersji powinien być stały ale w miarę wprowadzania nowych uregulowań prawnych czy wdrażaniu nowych technologii powinien być na bieżąco modyfikowany. Z uwagi na powszechność stosowania systemów informatycznych w działalności finansowo-księgowej właśnie ta klasa systemów powinna być pierwszą, która może wprowadzić procedurę homologacji. Algorytmy finansowo-księgowe są najlepiej zdefiniowane a skutki złej jakości systemów informatycznych mogą mieć najgroźniejsze konsekwencje.

Prowadzenie procedury homologacji wymaga odpowiedniej organizacji administracji obsługującej cały proces. Administracja powinna obejmować swym zasięgiem cały kraj i mieć możliwości techniczne prowadzenia aktualnej kartoteki systemów, które zostały dopuszczone do użytkowania i tych, które rozpoczęły procedurę homologacji i są w jej trakcie. Wzorce oprogramowania systemów, które otrzymały homologacje muszą być stale przechowywane i stanowią jedyny dokument, który będzie podstawą rozstrzygania wszelkich reklamacji i uwag użytkowników.

Wzorcowe oprogramowanie, zdeponowane w trakcie procedury homologacyjnej, chroni producenta przed nieautoryzowanym wprowadzaniem zmian i modyfikacji do systemu. Rozstrzyganie sporów pomiędzy producentem oprogramowania a użytkownikiem po pewnym okresie eksploatacji, bez posiadania zdeponowanego wzorca oprogramowania jest bardzo uciążliwe. Wzorcowe oprogramowanie staje się równocześnie gwarancją dla użytkownika, że wszelkie wady ukryte, które dały o sobie znać po okresie gwarancji, muszą być usunięte przez producenta.

Baza danych systemów homologowanych wymaga utrzymania i powinna umożliwiać łatwe homologowanie nowych wersji już homologowanego systemu. Biura administrujące procedurą homologacji muszą być wspomagane zespołami specjalistów wysokiej klasy, którzy będą gwarantować wysoką jakość ocen merytorycznych w kategoriach wymienionych wcześniej.

System stopni specjalizacyjnych w informatyce jest działaniem komplementarnym homologacji systemów informatycznych. Polskie Towarzystwo Informatyczne, inicjator stopni specjalizacyjnych powinno rozszerzyć zasięg normalizacji na systemy informatyczne. Należy pamiętać o współpracy Towarzystwa z prawnikami w trakcie początkowych prac nad ochroną praw autorskich oprogramowania. Aktywna współpraca ze Stowarzyszeniami branżowymi może doprowadzić do wykształtowania się zespołów interdyscyplinarnych posiadających niezbędną wiedzę i kwalifikacje dla obsługi procedury homologacji. Pozostaje jedynie zbudowanie odpowiedniej infrastruktury adminstracyjnej i technicznej dla obsługi procesu.

Działalność homologacyjna powinna mieć odpowiednią rangę akceptowalną przez administrację państwową. Rola administracji państwowej powinna być wyraźnie określona ale należy dbać o wagę merytoryczną procedury. Urzędnicy zawsze mają skłonność do biurokratyzacji każdej procedury, często prowadzi to do zaniku podstawowej ideii. Rola administracji w procesie homologacji powinna ograniczać się do działań czysto administracyjnych. Osiągnięcie takiej pozycji będzie możliwe tylko wtedy gdy dominująca stanie się procedura merytoryczna.

Certyfikat homologacji powinien stanowić dokument państwowy posiadający odpowiednią wagę. Certyfikat taki podpisany przez zespół merytoryczny będzie gwarantem odpowiedniej jakości produktu a urzędowa pieczęć administracji państwowej nadaje mu rangę zgodnie z procedurą administracyjną. Wdrożenie certyfikatu homologacyjnego uprości obrót oprogramowaniem i wprowadzi mechanizmy porządkowania zastosowań systemów informatycznych w różnych działalnościach gospodarczych.

Procedura homologacji będzie promować systemy wysokiej jakości a poprzez certyfikat będzie wdrażać znak jakości rozwiązań informatycznych. Pozwoli to na promowanie rozwiązań zgodnych z ogólnymi założeniami polityki państwa. Łatwiejsze stanie się wdrożenie nowych rozwiązań kierunkowych związanych z realizacją umów międzynarodowych czy restrukturyzacją całych gałęzi gospodarki.

Tadeusz Elsner
IDS Prof. Scheer GmbH

Komputerowo wspomagany consulting

Streszczenie

Projekty, których celem jest wprowadzenie elektronicznego przetwarzania danych do rzeczywistości zakładowej służą optymalizacji faktów ekonomicznych oraz wykorzystaniu systemów informacyjnych do ich komputerowego wspomagania. Firmy consultingowe pracują w ramach tych projektów jednak niestety tylko z ołówkiem i papierem. Jest to jedną z przyczyn, dlaczego projekty te wymagają ogromnych nakładów czasu i środków. Dodatkowym problemem jest wątpliwa jakość dokumentacji powstającej w ramach tych projektów. W referacie przedstawione zostanie oprogramowanie ARIS-Toolset firmy IDS Prof. Scheer GmbH służące do wspomagania projektów consultingowych mających na celu wprowadzenie zintegrowanych systemów informacyjnych. Oprogramowanie to bazuje na Architekturze Zintegrowanych Systemów Informacyjnych (ARIS) opracowanej przez Prof. dr A.-W. Scheera.

1. Wprowadzenie.

Rosnąca konkurencja na rynkach światowych zmusza producentów do ciągłej redukcji kosztów wytwarzania. Jednocześnie rosną wymagania klientów odnośnie jakości wyrobów, czasu relizacji zamówień itp. Dla rozwiązania tych w sumie jak się wydaje sprzecznych wymagań, powstało szereg koncepcji, takich np. jak: CIM, czy Lean Production. Z punktu widzenia informacji/organizacji (jako jednego z czterech podstawowych czynników produkcyjnych) wyróżnić można tutaj dwa podstawowe problemy:

- zastosowanie zintegrowanych systemów informacyjnych,
- optymalizację faktów ekonomicznych (realizowanych procesów).

Projekty realizowane w tym zakresie przez firmy consultingowe wymagają ogromnych nakładów czasu i środków. Mimo tych ogromnych nakładów wyniki nie zawsze są zadowalające. Również dokumentacja tych projektów pozostawia wiele do życzenia. Jest ona nieprzejrzysta, obejmuje setki lub tysiące stron papieru i po pewnym czasie zajmuje tylko miejsce w regałach, ponieważ znalezienie konkretnej informacji jest praktycznie niemożliwe. Bardzo trudne jest pielęgnowanie tej dokumentacji. By zachować jej spójność, należy każdą drobną zmianę przeprowadzić w wielu miejscach jednocześnie.

Przedsiębiorstwa mające problemy finansowe coraz rzadziej decydują się na przeprowadzanie wielkich projektów consultingowych. Firmy consultingowe zmuszane są do oferowania swoich usług o lepszej jakości przy znacznym skróceniu czasu realizacji projektów. By rozwiązać ten problem, bazując na doświadczeniach z wielu projektów consultingowych, w firmie IDS Prof. Scheer GmbH powstała idea stworzenia pakietu programowego ARIS-Toolset umożliwiającego komputerowe wspomaganie pracy doradców technicznych.

W artykule przedstawiona zostanie najpierw Architektura Zintegrowanych Systemów Informacyjnych (ARIS), będąca podstawą teoretyczną dla ARIS-Toolset. Po krótkiej prezentacji tegoż oprogramowania, omówiony zostanie sposób realizacji projektów z jego wykorzystaniem. Referat zakończą przykłady zastosowania ARIS-Toolset.

2. Architektura Zintegrowanych Systemów Informacyjnych (ARIS) /1/

W tworzeniu zintegrowanych systemów informacyjnych bierze udział kilku partnerów. Aby zrealizowany przez nich system spełniał później wymagania spójności technicznej oraz organizacyjnej konieczne jest aby partnerzy ci pracowali według jednakowych reguł. Definicja wspomnianych reguł jest konieczna szczególnie wtedy, gdy celem jest kompleksowe wspomaganie komputerowe procesów przetwarzania informacji, np. w przedsiębiorstwie przemysłowym. Całokształt tych reguł do kompleksowego opisu systemów informacyjnych można nazwać architekturą. Analogia do tego elementu sztuki budowlanej wynika z faktu, że od tysięcy lat istnieją w budownictwie reguły planowania i realizacji kompleksowych projektów budowlanych, według których muszą pracować biorący udział w projekcie rzemieślnicy oraz firmy budowlane.

2.1 Opanowanie kompleksowości

Koncepcja ARIS zawiera w sobie dwa modele (rys. 1):

- model podziału,
- model definicji różnych poziomów opisu.

Model podziału bazuje na założeniu, że proces gospodarczy jest tak kompleksowy, że przy opisie i realizacji wspomagającego go systemu informacyjnego nie może być traktowany jako jedna całość. Konieczny jest podział tego procesu poprzez jego analizę z różnych punktów widzenia. Bardzo ważne jest jednoczesne uwzględnienie różnych punktów widzenia na system informacyjny i określenie powiązań między nimi .

W koncepcji ARIS zdefiniowane jest spojrzenie na systemy informacyjne z następujących punktów widzenia: funkcji, organizacji, danych oraz sterowania łączącego pozostałe elementy. Przeprowadzona w pierwszym kroku analiza z różnych punktów widzenia prowadzi do redukcji kompleksowości opisu systemu informacyjnego. Realizowana w drugim kroku analiza z punktu widzenia sterowania obejmuje opis współdziałania pozostałych elementow.

ARIS różni sie od innych znanych architektur systemów informacyjnych głównie tym, że uwzględniona jest tutaj analiza z punktu widzenia organizacji oraz sterowania. Szczególnie problemy organizacyjne odgrywają ważną rolę przy kształtowaniu systemów informacyjnych. W kontekście takich pojeć jak CIM lub Lean Production konieczne jest zajęcie się prolemami struktury organizacyjnej i organizacji pracy. Elementy te wywierają wpływ na kształtowanie systemów informacyjnych. Przy zdecentralizowanych strukturach organizacyjnych preferowane jest przykładowo projektowanie zdecentralizowanych systemów informacyjnych.

2.2 Spojrzenie całościowe

Wprowadzenie analizy z punktu widzenia sterowania umożliwia systematyczne badanie istniejących powiązań między innymi elementami (dane, funkcje, organizacja). Konieczność definicji różnych poziomów opisu wynika z koncepcji cyklu życia projektu (Conception Life Cycle), co oznacza, że architektura powinna towarzyszyć systemowi informacyjnemu od opisu wyjściowej sytuacji ekonomicznej aż do wdrożenia. Pod pojęciem strategii wyjściowej rozumiana jest ogólna koncepcja skierowana na zastosowanie komputerów, która powstaje w wyniku analizy istniejących systemów informacyjnych. Koncepcja ta zawiera w sobie optymalny przebieg organizacyjny możliwy do osiągnięcia poprzez zastosowanie informatyki, tak jak jest to realizowane w koncepcjach CIM lub Lean Production.

W następnym kroku dokonuje się podziału sytuacji wyjściowej z różnych punktów widzenia na poziomie modeli semantycznych. Do opisu pojedynczych punktów widzenia wykorzystuje sie metody bardzo bliskie jezykowi fachowemu, które jednocześnie umożliwiają precyzyjny i spójny opis.
W kolejnym kroku przeprowadza się - na poziomie koncepcji przetwarzania danych - dopasowanie modeli semantycznych do wymagań integrowalności różnych systemów informacyjnych. W tej fazie nie uwzględniane są jednak konkretne systemy informacyjne oferowane na rynku.
Na poziomie implementacji realizowany jest opis konkretnych systemów bazujacy na przyjętej strukturze oprogramowania i sprzętu. Przy pomocy analizy z różnych punktów widzenia oraz modelu definicji różnych poziomów opisu koncepcja ARIS obejmuje wszystkie elementy systemu informacyjnego od powstania idei systemu do jego realizacji. ARIS reprezentuje sobą koncepcję umożliwiającą weryfikację przydatnosci różnych metod do opisu systemów informacyjnych. Poprzez wybór metod projektowania i wdrażania systemów informacyjnych zagwarantowana jest rownież kompletność opisu.

3. ARIS-Toolset

ARIS-Toolset składa się z trzech komponentów służących do analizy, modelowania oraz nawigacji.

3.1 Modelowanie

Celem modelowania jest tworzenie modeli danych, funkcji, organizacji oraz sterowania mających na celu opis rzeczywistości zakładowej bądź sytuacji docelowej, do której doprowadzić ma realizowany projekt. Dla każdego punktu widzenia możliwe jest wykorzystanie różnych typów modeli. Poniżej przedstawione zostaną przykłady modelowania funkcj, danych, organizacji oraz procesów.

W modelu funkcji przedstawione są w przejrzysty sposób realizowane w zakładzie przemysłowym funkcje. Model ten ma charakter statyczny. Na rysuku 2 pokazano przykład drzewa funkcji dla funkcji przyjęcie zlecenia. Drzewa funkcji nie wymagają dodatkowych wyjaśnień. Przy ich tworzeniu rozwiązać trzeba dwa problemy: określić ilość poziomów hierarchii oraz zasady grupowania funkcji.
O ilości poziomów hierachii decyduje znaczenie jakie przypisujemy realizowanej funkcji oraz jej kompleksowść. Wyróżnić można tu (por. rys. 2.) łańcuch funkcji (np. zbyt), funkcje główne (np. przyjęcie zlecenia), funkcje częściowe (np. kontrola kompletności) oraz funkcje elementarne (np. ujęcie zlecenia). Łańcuch funkcji to pewien skomplikowany proces realizowany na pewnym objekcie. Funkcje główne to wchodzące w skład łańcucha funkcji skomplikowane czynności wymagające dokładniejszego opisu. Funkcje główne składają się z funkcji częściowych, które szczegółowo opisywane są przez funkcje elemetarne.
Grupować funkcje można zależnie od realizowanych procesów, według wykonywanych czynności, czy też zależnie od obiektu na którym wykonywane są pewne czynności.

Do opisu danych wykorzystać można ERM (Entity-Relationship Model) zaproponowany przez Chena /2/, który rozszerzony został później przez innych autorów. Podstawowymi elementami opisu danych w tym modelu są TYP-Entity oraz TYP-Relacja (por. rys. 3.). Entity-KLIENT odpowiada w praktyce kartotece z informacjami o wszystkich klientach lub macierzy w programie komputerowym, Entity-PRODUKT jest zbiorem informacji o poszczególnych produktach oferowanych przez pewne przedsiębiorstwo. Relacja-WARUNKI-SPRZEDAŻY informuje o tym,

jakie warunki sprzedaży (np. cena, gwarancja, opakowanie, warunki dostawy) oferowane są różnym KLIENTOM zależnie od nabywanych przez nich PRODUKTÓW.

W modelu organizacji opisana jest struktura organizacyjna analizowanego przedsiębiorstwa (rys. 4.). Podstawowymi symbolami w tym modelu są: jednostki organizacyjne oraz ich siedziby, typy jednostek organizacyjnych, stanowiska pracy, pracownicy i.t.p.

W modelu procesów znaleźć można elementy pozostałych trzech punktów widzenia. Zawiera on w sobie informacje o współdziałaniu ze sobą elementów z modeli danych, funkcji oraz organizacji. Do opisu procesów wykorzystuje się w ARIS-Toolset głównie łańcuch procesów sterowany zdarzeniami. Głównymi jego elementami są funkcje oraz zdarzenia (rys. 5.). Funkcje znamy już z modelu funkcji, zdarzenia opisują warunki rozpoczęcia oraz zakończenia realizacji określonych funkcji, dzięki czemu możliwe jest modelowanie procesów. Dodatkowo pojawiają się tutaj elementy z modelu danych (jako informacje wejściowe i wyjściowe dla funkcji) oraz elementy modelu organizacji (jako jednostki organizacyjne odpowiedzialne za realizację określonych funkcji).

Należy tutaj zwrócić uwagę na dwie cechy charakterystyczne dla ARIS/Toolset:

- ARIS-Toolset umożliwia hierarchizację modeli,
- dwa identyczne objekty w różnych modelach (np. funkcja w drzewie funkcji i w łańcuchu procesów) mają tylko jednego odpowiednika w bazie danych. Jeżeli konieczne jest przeprowadzenie zmian w atrybutach opisujących tą funkcję, wystarczy to zrobić oczywiście tylko w jednym miejscu.

3.2 Nawigacja

Dla lepszego zobrazowania wyników projektowania i wdrażania systemów informacyjnych umożliwia ARIS-Toolset profesjonalną prezentację wyników modelowania. Narzędzie to pomyślane zostało dla specjalistów z dziedziny Information Management oraz użytkowników systemów informacyjnych. Umożliwia ono "nawigację" między różnymi wynikami projektowania i wdrażania konkretnego systemu informacyjnego, które uzyskane zostały na etapie modelowania. Oznacza to konkretnie, że jeżeli punktem wyjścia rozważań będzie przykładowo pewien obiekt z modelu danych, to można się do niego zwrócić z pytaniem, które funkcje korzystają z tego obiektu danych. Znaleźć można także funkcje sąsiednie do rozpatrywanej, w celu ustalenia, która jednostka jest odpowiedzialna za realizację tej funkcji. Na zakończenie można odpowiedzieć na pytanie: za jakie obiekty danych odpowiedzialna jest dana jednostka organizacyjna. Celem jest więc tutaj przejrzysta prezentacja wyników projektowania oraz umożliwienie użytkownikowi swobodnej nawigacji między opracowanymi modelami.

3.3 Analiza

Dla zrozumienia składnika ARIS-Toolset umożliwiający analizę wprowadzić najpierw należy pojęcie modelu referencyjnego. Modele referencyjne to wyidealizowane modele danych, funkcji, organizacji oraz sterowania opracowane dla konkretnej branży przemysłu. Na podstawie tych modeli oraz korzystając z pewnych reguł (mogą być one indywidualnie definiowane) użytkownik wygenerować może model docelowy dopasowany do jego wymagań. Inną możliwością pracy z modelami referencyjnymi jest kopiowanie ich i przeprowadzanie w nich dowolnych zmian. Na zakończenie przeprowadzona może zostać

analiza porównawcza uzyskanego modelu z modelem referencyjnym, w celu oszacowania ilości zmian dokonanych na etapie modelowania.

4. Projekty realizowane z wykorzystaniem ARIS-Toolset

Opisowi przedsiębiorstw przy pomocy technik modelowania strukturalnego przypisuje się coraz większe znaczenie. Główny nacisk kładzie się tutaj na opis faktów ekonomicznych na poziomie modeli semantycznych. Obok opisu (modelowania), aktualnym tematem jest również analiza modeli (porównanie stanu aktualnego ze stanem docelowym, analiza kosztów, analiza czasu trwania procesów). Opis przedsiębiorstw na poziomie modeli semantycznych poprzez tworzenie modeli danych, funkcji, organizacji oraz modeli procesów, jak również analiza tychże modeli staje się coraz częściej podstawą rozwiązywania wielu dyskutowanych w przedsiębiorstwach problemów. Należą do nich przykładowo:

- reorganizacja przebiegających procesów,
- efektywny wybór oraz wprowadzanie oprogramowania standardowego,
- tworzenie oprogramowania indywidualnego,
- dokumentacja istniejącej struktury komputerowej.

Projekty realizowane przez firmy consultingowe w zakresie informacji/ organizacji mają zwykle na celu wprowadzenie zintegrowanych systemów informacyjnych oraz/lub reorganizację faktów ekonomicznych (business process). Projekty takie realizowane przy pomocy ARIS-Toolset mają podobny przebieg, który w skrócie przedstawiony zostanie poniżej.

W pierwszym kroku konieczne jest opisanie aktualnej sytuacji, celem drugiego kroku jest uzyskanie wyidealizowanych modeli opisujących stan docelowy. Oznacza to, że w pierwszej fazie odpowiedzieć trzeba na pytanie jakie funkcje realizowane są aktualnie, jak wygląda komputerowe wspomaganie tych funkcji, jak wygląda przepływ informacji między nimi, a w drugiej bazując na tych informacjach, modelach referencyjnych i na doświadczeniach z innych projektów consultingowych powstają modele wyidealizowane. Do modeli opisujacych rzeczywistość zakładową dochodzi się drogą analizy korzystając z modeli referencyjnych lub poprzez modelowanie danych, funkcji, organizacji oraz procesów. Przy korzystaniu z modeli referencyjych istnieje możliwość przyśpieszenia realizacji projektu, poprzez automatyczne generowanie modeli przy opisie sytuacji wyjściowej.

Wynikami omawianych projektów są zwykle model funkcji, danych, organizacji oraz procesów na poziomie modeli semantycznych. Mogą one być podstawą dalszych prac mających na celu dokumentację struktury przetwarzania danych (tworzenie modeli na poziomie koncepcji przetwarzania danych oraz implementacji), czy też wybór oprogramowania standardowego lub tworzenie oprogramowania indywidualnego. Coraz więcej firm oferujących oprogramowanie standardowe decyduje się na dokumentację tegoż oprogramowania przy pomocy ARIS-Toolset. Najlepszym przykładem może być tutaj firma SAP (największa firma oferująca oprogramowanie standardowe). Jeżeli mamy do dyspozycji modele opisujące rzeczywistość zakładową oraz oprogramowanie standardowe to poprzez porównanie tych modeli łatwiej ocenić przydatność tegoż oprogramowania.

5. Podsumowanie

Firma IDS sprzedała na przestrzeni ostatnich sześciu miesięcy ponad 700 instalacji oprogramowania ARIS-Toolset. Jest to

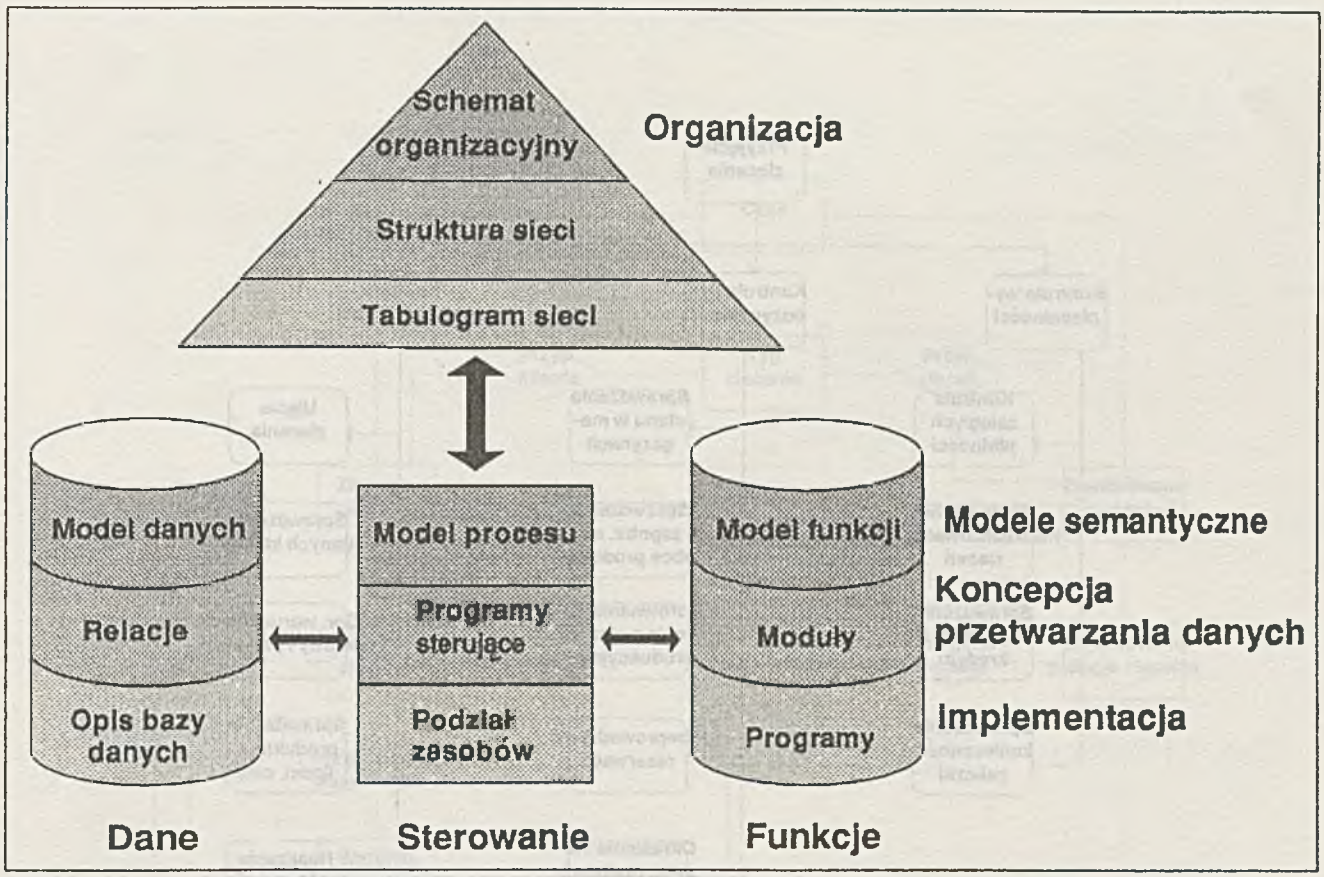
najlepszym dowodem na to, że koncepcja ARIS znalazła już wielu zwolenników. Szereg dużych firm zdecydowało się na wykorzystanie koncepcji ARIS jako podstawy dla opracowania przyszłej strategii przetwarzania danych. Koncepcja ta jest również bazą dla OPEN-CAM-Principle preferowanym przez firmę Hewlett Packard, w którym firmy oferujące produkty CAM połączone są koncepcją integrującą. Na bazie wspólnego modelu danych i modeli procesów realizowane jest, zgodnie z architekturą ARIS, przyporządkowanie określonych rozwiązań poszczególnych partnerów oraz identyfikacja ich wymagań integracyjnych.

Ze względu na możliwości dokumentacji projektów i korzystanie z modeli referencyjnych, również firmy consultingowe coraz szerzej stosują ARIS-Toolset. Dokumentacja wyników projektów oraz możliwości analizy i wykorzystania modeli referencyjnych prowadzi do znacznego skrócenia czasu realizacji projektów, która na bazie dotychczasowych doświadczeń szacowana jest na 60% - 80% (w porównaniu z tradycyjnymi metodami realizacji projektów).

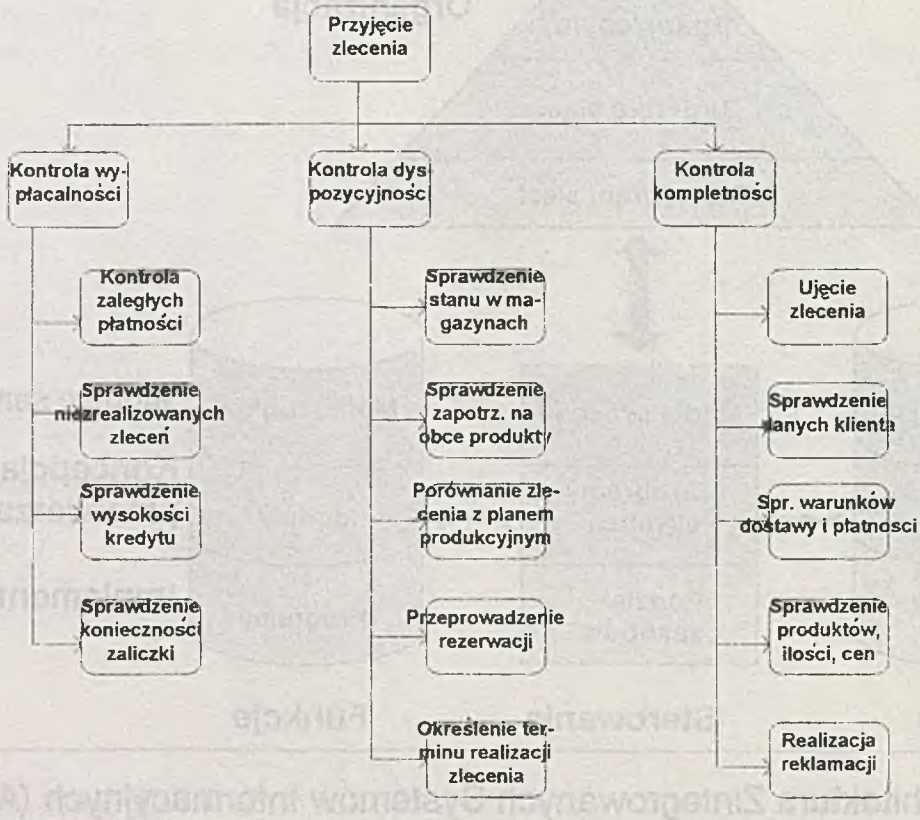
Wyniki badań empirycznych prowadzą do wniosku, że problemy architektury systemów informacyjnych należeć będą do centralnych problemów Information Management w najbliższych latach. Koncepcja ARIS ze swoja spójnością i prostotą może być tutaj wykorzystywana do tworzenia kompleksowych systemów informacyjnych, jako koncepcja z solidną podbudową teoretyczną, będąca przydatną w implementacjach praktycznych.

Literatura

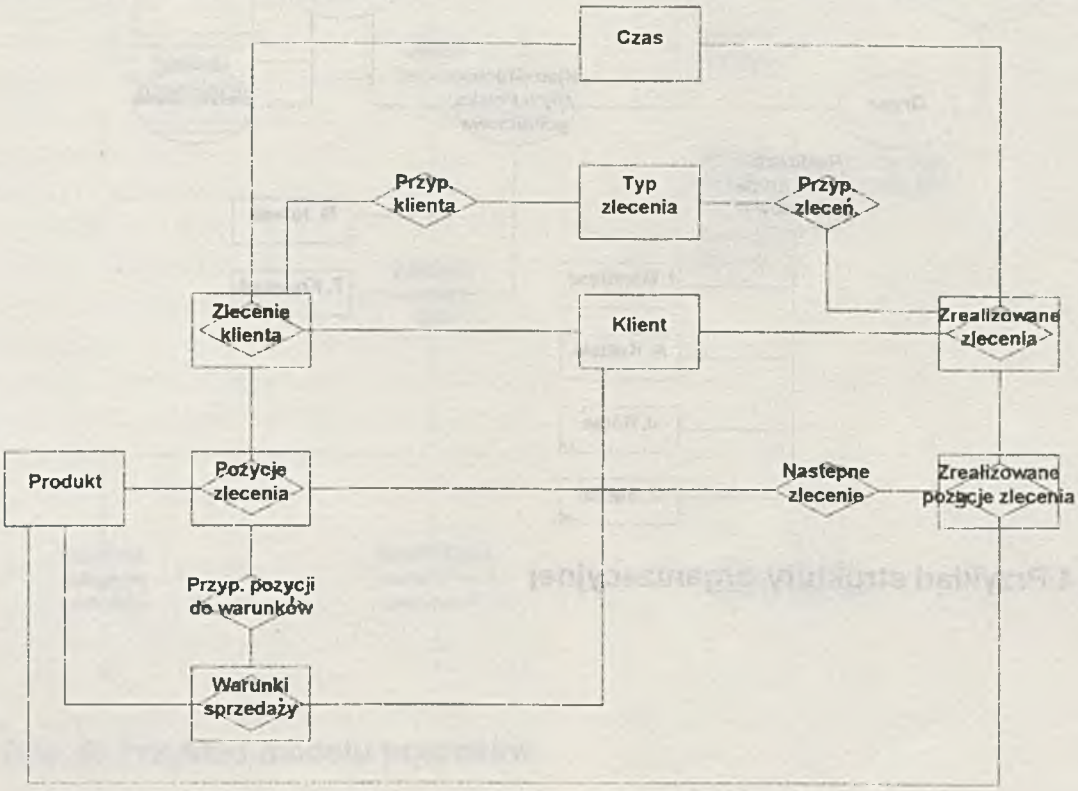
1. Scheer, A.-W.: "Architektur integrierter Informationssysteme", Springer Verlag Berlin Heidelberg 1991, ISBN 3-540-53984-0
2. Chen, P. P.: "The Entity-Relationship Model (Towards an Unified View of Data)", in: ACM (Hrsg); Transactions on Database-Systems, No. 1 (1976)
3. Scheer, A.-W.: "Architektura zintegrowanych systemów informatycznych", Przegląd Organizacji, 4/93, s. 32-34



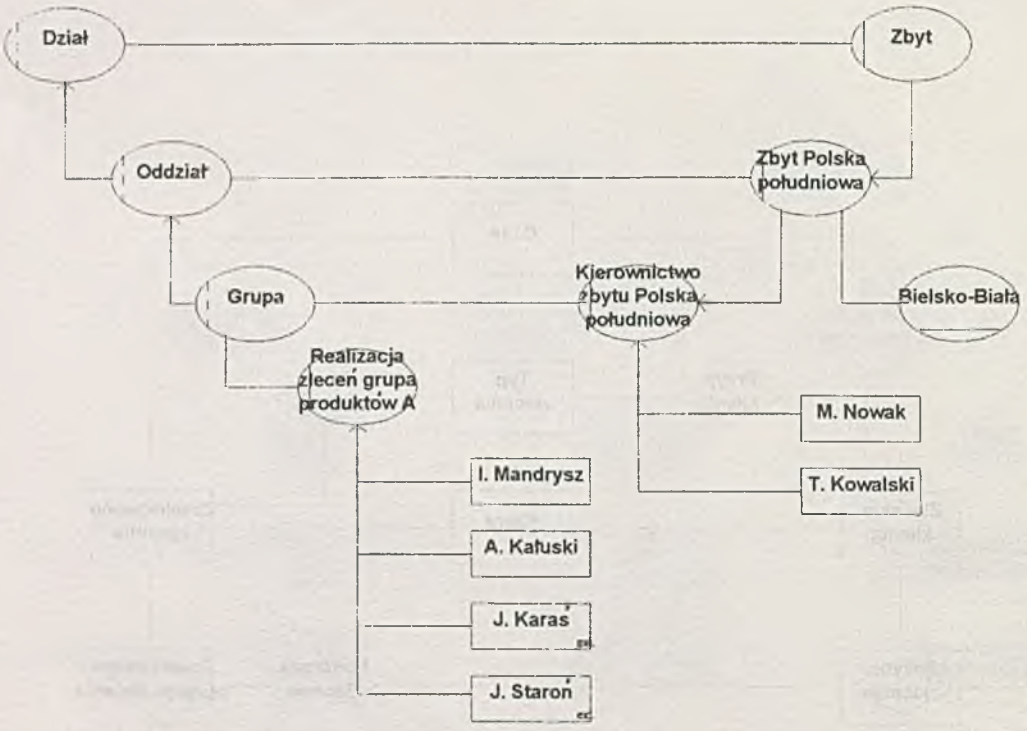
Rys. 1. Architektura Zintegrowanych Systemów Informacyjnych (ARIS) /1/



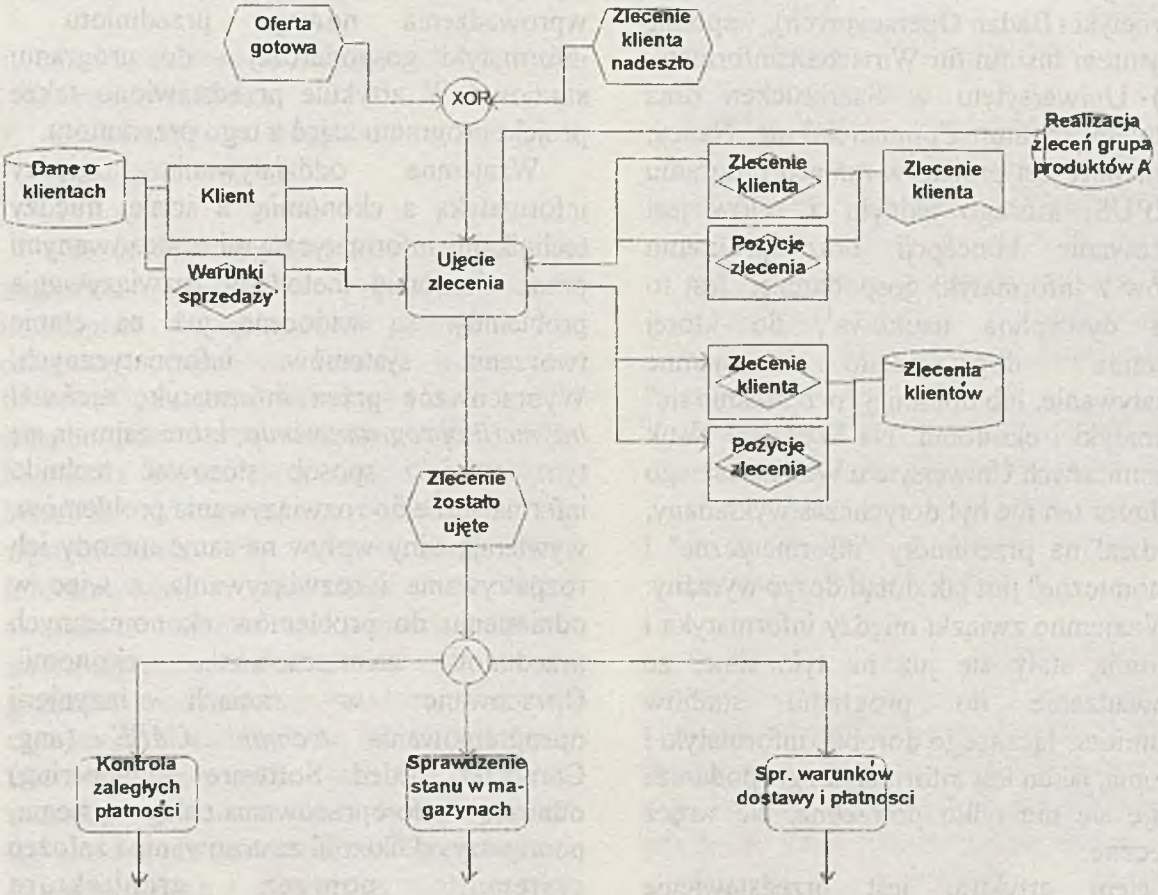
Rys. 2. Przykład drzewa funkcji



Rys. 3. Przykład modelu danych



Rys. 4 Przykład struktury organizacyjnej



Rys. 5. Przykład modelu procesow

Mirosława Lasek
Uiwersytet Warszawski

"Informatyka gospodarcza" - nowy kierunek kształcenia studentów ekonomii

W Wydziale Nauk Ekonomicznych Uniwersytetu Warszawskiego (Katedra Cybernetyki i Badań Operacyjnych), wspólnie z instytutem Institut für Wirtschaftsinformatik (IWi) Uniwersytetu w Saarbrücken oraz instytutem Institut Commercial de Nancy, realizowany jest projekt w ramach programu TEMPUS, którego jednym z celów jest opracowanie koncepcji oraz programu studiów z informatyki gospodarczej. Jest to nowa dyscyplina naukowa[1], do której powstania doprowadziło wzajemne oddziaływanie, lub dobitniej "przenikanie się" informatyki i ekonomii. Na Wydziale Nauk Ekonomicznych Uniwersytetu Warszawskiego przedmiot ten nie był dotychczas wykładany, a podział na przedmioty "informatyczne" i "ekonomiczne" jest jak dotąd dosyć wyraźny.

Wzajemne związki między informatyką i ekonomią stały się już na tyle silne, że wprowadzenie do programu studiów przedmiotu, łączącego dorobek informatyki i ekonomii, jakim jest informatyka gospodarcza wydaje się nie tylko potrzebne, ale wręcz konieczne.

Celem artykułu jest przedstawienie związków między informatyką a ekonomią (w tym w szczególności ekonomiką przedsiębiorstwa), które wskazują, że oddzielne nauczanie przedmiotów "informatycznych" i "ekonomicznych" już nie wystarcza oraz uzasadniają podjęcie wysiłków wprowadzenia nowego przedmiotu - informatyki gospodarczej - do programu studiów.[2] W artykule przedstawiono także projekt programu zajęć z tego przedmiotu.

Wzajemne oddziaływania między informatyką a ekonomią, a ściślej między technikami informatycznymi a stosowanymi przez ekonomię metodami rozwiązywania problemów są widoczne już na etapie tworzenia systemów informatycznych. Wypracowane przez informatykę *techniki inżynierii oprogramowania*, które zajmują się tym, w jaki sposób stosować techniki informatyczne do rozwiązywania problemów, wywierają silny wpływ na same metody ich rozpatrywania i rozwiązywania, a więc w odniesieniu do problemów ekonomicznych przedmiot zainteresowania ekonomii. Opracowane w ramach inżynierii oprogramowania *techniki CASE* (ang. Computer Aided Software Engineering) odnoszą się do opracowania całego systemu, począwszy od filozofii zastosowania i założeń systemu, poprzez architekturę oprogramowania, aż do opracowania baz danych i interfejsów użytkowników. Techniki inżynierii oprogramowania szczególnie silnie oddziaływują na metodę rozwiązywania problemu na etapie definiowania problemu i

[1] W Niemczech pierwsze katedry, zajmujące się informatyką gospodarczą (niem. Wirtschaftsinformatik) powstały w latach 1968-1970 na uniwersytetach Erlangen-Nürnberg, Karlsruhe, Linz.

[2] Pełne przedstawienie związków i wzajemnego oddziaływania informatyka - ekonomia spowodowałoby znaczne przekroczenie dopuszczalnej objętości artykułu. Z tego względu ograniczono się tylko do wymienienia podstawowych problemów.

jego konwersji w projekt systemu. A.-W. Scheer w pracy: "EDV-orientierte Betriebswirtschaftslehre"[3], poświęconej m.in. wzajemnym oddziaływaniom między ekonomiką przedsiębiorstwa a technikami informacyjnymi i komunikacyjnymi, zwraca uwagę, że w tworzeniu nowoczesnych systemów informacji na etapach tworzenia koncepcji systemu pierwszoplanowe role odgrywają:

1. strukturalizowanie danych źródłowych przedsiębiorstwa,
2. analizowanie łańcuchów procesów (zdarzeń).

Dla modelowania wyżej wymienionych zagadnień inżynieria oprogramowania udostępnia szereg technik, przykładowo diagramy przepływu danych (ang. data flowchart), diagramy bąbelkowe (ang. bubble diagram technique), diagramy HIPO (ang. Hierarchy Input-Process-Output), schematy Nassi/Shneidermana. Zostały one wprowadzone, aby ułatwić współpracę informatyków z ekonomistami na etapie tworzenia założeń i projektu systemu informatycznego. Techniki te umożliwiają graficzne modelowanie problemu w sposób czytelny dla nieinformatyka, a jednocześnie na tyle sformalizowany, aby informatykowi umożliwić dalsze tworzenie systemu. Nadają się one dobrze dla modelowania problemów ekonomicznych i dlatego są coraz częściej traktowane jako rozszerzenie werbalnych i matematycznych opisów rozwiązywania problemów, które są tradycyjnie stosowane przez ekonomistów. W takim kontekście są uważane przede wszystkim za narzędzia ekonomisty, a nie informatyka, z zaznaczeniem, że umożliwiają modelowanie problemu w sposób ułatwiający utworzenie odpowiedniego oprogramowania komputerowego.

Przykładem skomputeryzowanego narzędzia inżynierii oprogramowania, umożliwiającego modelowanie danych i procesów w przedsiębiorstwie oraz zaprojektowanie systemu informatycznego jest *ARIS* (niem. Architektur integrierter Informationssysteme).[4] System obejmuje:

* ARIS-Projectmanager - narzędzie wspomagania planowania oraz organizacji projektu
* ARIS-Analyzer - narzędzie analizy stanu obecnego i generowania koncepcji docelowej. Dostępne są przykładowe graficzne modele dla różnych dziedzin gospodarowania przedsiębiorstwem, które mogą być pomocne w zdefiniowaniu wymagań stawianych projektowanemu systemowi
* ARIS-Modeller - narzędzie definiowania oraz prezentacji struktury organizacyjnej, struktury danych, funkcji oraz procesów
* ARIS-Navigator - narzędzie "nawigacji" w systemie informatycznym.

Ponieważ w przedsiębiorstwach przemysłowych często mamy do czynienia z wieloma tysiącami danych, procesów, relacji, modelowanie systemów informatycznych bez wykorzystania skomputeryzowanych narzędzi, takich jak ARIS, nie wydaje się realne.

Dzięki technikom inżynierii oprogramowania problemy ekonomiczne mogą być rozważane w szerszym kontekście niż przy tradycyjnym ujęciu (wyznaczenie celów - budowa modelu - decyzja), obejmującym takie elementy jak pozyskiwanie

[3] A.-W. Scheer, EDV-orientierte Betriebswirtschaftslehre, Springer-Verlag 1990.

[4] Por. A.-W. Scheer, Architektur integrierter Informationssysteme, Springer-Verlag 1991 oraz A.-W. Scheer, Architektura zintegrowanych systemów informatycznych (ARIS), Przegląd Organizacji 4/1993 (s.32-34).

danych, czy integracja z "sąsiednimi" zastosowaniami.

Ze swej strony ekonomia oddziaływuje na techniki inżynierii oprogramowania poprzez:

1. Udostępnianie narzędzi planowania i sterowania projektami, np. technik harmonogramowania, wykorzystujących metody sieciowe, procedur planowania kosztów i zdolności produkcyjnych, metod feasibility study dla wariantów projektów oprogramowania.
2. Formułowanie wymagań odnośnie technik inżynierii oprogramowania, co ukierunkowuje jej rozwój.
3. Wspomaganie poszczególnych faz tworzenia systemu informatycznego za pomocą metod wypracowanych przez ekonomię, np. zastosowanie modeli optymalizacyjnych dla strukturalizacji sieci komputerowych, czy przyjętych w ekonomii metod standaryzacji danych w celu wspomagania wymiany danych.

Wytworzone za pomocą technik inżynierii oprogramowania, *oprogramowanie użytkowe* zawiera niejako "zaprogramowane" w sobie ekonomiczne modele i metody rozwiązań (wiedzę ekonomiczną). Stanowi w ten sposób "nośnik" modeli i metod (wiedzy ekonomicznej). Nośnikiem takim jest zwłaszcza *standardowe oprogramowanie użytkowe*, przeznaczone do wdrażania u wielu użytkowników. Standardowe oprogramowanie użytkowe poprzez wprowadzanie u wielu użytkowników przyspiesza i ułatwia rozpowszechnianie zawartych w nim koncepcji rozwiązywania problemów ekonomicznych. W rezultacie poprzez powszechne stosowanie standardowego oprogramowania użytkowego dominują w praktycznych zastosowaniach te rozwiązania ekonomiczne, które zostały oprogramowane.

Standardowe oprogramowanie daje się wprowadzać tam, gdzie możliwa jest standaryzacja procedur rozwiązywania problemów. W przedsiębiorstwie przemysłowym jako przykłady dziedzin dobrze poddających się standaryzacji można wymienić gospodarkę materiałową, rachunkowość.

Dzięki stosowanej przez informatyków technice modularyzacji, w wielu przypadkach użytkownik może zastosować własną, oryginalną koncepcję rozwiązania problemu i "złożyć" system z gotowych, standardowych modułów oprogramowania, realizujących określone funkcje, np. dla poszczególnych obszarów funkcjonalnych gospodarowania przedsiębiorstwem: produkcji, zaopatrzenia, zbytu, kadr, rachunkowości. Przy podziałach funkcjonalnych problemów wykorzystywane są typologie sporządzane przez ekonomię.

Oprogramowanie użytkowe ułatwia przeniesienie dorobku ekonomii do praktyki. Przykładowo dzięki powstaniu odpowiedniego oprogramowania powszechne w przedsiębiorstwach stało się stosowanie ekonomicznych formuł określania wielkości partii produkcyjnych, czy też technik prognozowania zbytu.

Istnienie lub brak standardowego oprogramowania odzwierciedla często stan badań w określonej dziedzinie ekonomii. Przykładowo, software dla pewnych obszarów przedsiębiorstwa jest bardzo dobrze rozwinięty, np. dla rachunkowości, tak jak i jej koncepcje w ekonomii, dla innych słabo, np. dla kalkulacji kosztów projektowanych wyrobów, dla której w zasadzie brak rozwiniętych koncepcji w ekonomii.

Oprogramowanie użytkowe daje ekonomiście okazję do wniesienia swojej wiedzy do praktyki i jej rozpowszechniania, skłaniając do poszukiwania nowych rozwiązań, jeżeli istniejące nie zadawalają użytkowników oprogramowania.

Ze swej strony ekonomista oddziaływuje na rozwój standardowego oprogramowania:

- oceniając poziom rozwiązań

ekonomicznych i informatycznych oferowanego software i przedstawiając propozycje zmian,
- śledząc tendencje w rozwoju standardowego software i wskazując koncepcje ekonomiczne, które mogą być za jego pośrednictwem zrealizowane.

Szczególną formą rozpowszechniania wiedzy ekonomicznej przez informatykę są *banki modeli i metod*. Za pośrednictwem banków modeli i metod powszechne stało się stosowanie metod programowania liniowego, czy metod statystyki wielowymiarowej. Związek między praktycznym wykorzystywaniem modeli, czy metod i informatyką jest bardzo ścisły, ponieważ możliwość użycia pewnych metod dla rozwiązania rzeczywistych problemów jest uzależniona od obliczeniowych i pamięciowych możliwości użytego sprzętu. Znany jest fakt, że istnieją metody rozwiązywania problemów, które nie mogłyby być stosowane bez komputera. Dotyczy to przykładowo modeli optymalizacyjnych, opracowanych w ramach badań operacyjnych, które operują na dużej liczbie danych i wymagają dużego nakładu obliczeniowego. Innym przykładem są techniki symulacyjne, w których wykorzystuje się technikę Monte Carlo. W wielu przypadkach użycie komputera umożliwiło praktyczne zastosowanie teoretycznie opracowanych procedur, tworząc jednocześnie bodziec dla ich dalszego rozwoju. Jest to dobrze widoczne w przypadku modeli programowania liniowego dla planowania produkcji i zbytu.

Banki modeli i metod poprzez odpowiedni interfejs użytkownika ułatwiają wykorzystywanie modeli i metod, wspomagają przy ich wyborze, a także interpretowaniu uzyskanych wyników.

Barierą dla stosowania wielu modeli ekonomicznych w praktyce, np. modeli optymalizacyjnych w przedsiębiorstwach, jest trudność z zaopatrywaniem ich w dane. Odpowiedź na pytanie co do tego jakie są potrzebne dane i o jakiej strukturze jest domeną ekonomisty. Natomiast informatyka poprzez banki danych i sieci komputerowe stwarza techniczne możliwości pokonania tej bariery.

Kształtowanie banków modeli i metod przez ekonomię polega na dostarczaniu teoretycznie opracowanych modeli i metod rozwiązywania problemów, konstruowaniu procedur doboru metod i określaniu sposobów interpretacji wyników.

Poprzez opracowanie koncepcji *systemów eksperckich* informatyka spowodowała wzrost zainteresowania badaniami nad sposobami rozwiązywania problemów nieustrukturalizowanych lub źle ustrukturalizowanych. Wprowadzenie systemów eksperckich oznacza bowiem nie tylko zmianę architektury oprogramowania, lecz także upowszechnienie w praktyce i nauce nowego podejścia do problemów ekonomicznych, przyznającego, że wiele z nich nie poddaje się algorytmizacji, np. problemy z dziedziny doradztwa inwestycyjnego lub badania zdolności kredytowej klientów banków. Dla rozwiązywania takich problemów niezbędne jest posługiwanie się wiedzą eksperta, jego doświadczeniem i intuicją.

Przedmiotem zainteresowania ekonomii były jak dotychczas przede wszystkim modele dla dobrze ustrukturalizowanych problemów. Systemy eksperckie powodują zmianę tej postawy. Modele dobrze ustrukturalizowane nie pasują bowiem w wielu przypadkach do rzeczywistej sytuacji, którą mają reprezentować, lecz ma miejsce przypadek, że źle ustrukturalizowana sytuacja decyzyjna jest przedstawiana przez dobrze ustrukturalizowany model, który ujmuje zaledwie małą część rzeczywistego problemu.

Rozwój systemów eksperckich kieruje uwagę ekonomistów na tzw. miękkie sposoby rozwiązywania problemów, proponując używanie reguł opartych na doświadczeniu,

stosowanie wiedzy wynikającej z doświadczenia i operowanie na niepewnych informacjach. Uwypuklenie w systemach eksperckich roli modułu interakcji z użytkownikiem (dialogu) podkreśla, że rozwiązywanie problemów jest silnie uzależnione od specyfiki każdego z nich, co jest sprzeczne z klasycznymi modelami rozwiązywania problemów ekonomicznych.

Poprzez realizację koncepcji systemów eksperckich staje się możliwe nie tylko posługiwanie się metodami heurystycznego podejmowania decyzji, lecz także ich dokumentowanie. W rzeczywistości oznacza to, że wiedza wynikająca z doświadczenia nie jest już tracona, lecz poprzez systemy eksperckie może być gromadzona i udostępniana do dalszego wykorzystywania.

A.-W. Scheer zwraca uwagę na fakt, że systemy eksperckie otwierają dla ekonomiki przedsiębiorstwa nowe możliwości zarówno w odniesieniu do badań, jak i praktyki.[5] Problemy z zakresu ekonomiki przedsiębiorstwa, mające duże znaczenie praktyczne, które ze względu na brak odpowiednich narzędzi były "zaniedbywane", wraz z pojawieniem się systemów eksperckich stają się przedmiotem rozważań. Fakt ten potwierdza analiza literatury, dotyczącej aktualnego etapu rozwoju systemów eksperckich. Przykładowo, utworzenie systemu eksperckiego dla kalkulacji kosztów na etapie projektowania wyrobów skierowało uwagę ekonomistów na taki obszar rachunku kosztów, który, mimo jego ogromnego praktycznego znaczenia, nie był przedmiotem badań ekonomiki przedsiębiorstwa.

Oddziaływanie ekonomii na rozwój systemów eksperckich polega na:

1. odkrywaniu odpowiednich obszarów problemowych dla wprowadzania systemów eksperckich,
2. transformacji wiedzy z postaci dostępnej w ramach ekonomii do postaci jej przedstawiania właściwej dla systemów eksperckich,
3. przygotowywaniu ekspertów-ludzi dla akwizycji wiedzy, wynikającej z doświadczenia,
4. opracowywaniu odpowiednich procedur pozyskiwania wiedzy, np. na podstawie zebranych obserwacji; przykładem jest algorytm Quinlana generowania reguł dla bazy wiedzy systemu eksperckiego,
5. udostępnianiu procedur planowania i sterowania projektowaniem i programowaniem systemów eksperckich.

W odniesieniu do przedsiębiorstwa przemysłowego związek między informatyką a ekonomią jest tak ścisły, że zamiast terminu *ekonomika przedsiębiorstwa* wprowadza się pojęcie *komputerowo-zorientowana ekonomika przedsiębiorstwa* (niem. EDV-orientierte Betriebswirtschaftslehre).

W swojej pracy "EDV-orientierte Betriebswirtschaftslehre" A.-W. Scheer wskazuje na strategiczne znaczenie jakie ma użycie technik informatycznych dla przedsiębiorstwa. Po pierwsze, decyzje, dotyczące użycia technik informatycznych, takich jak banki danych, czy sieci komputerowe wiążą na długi okres i determinują możliwości przetwarzania informacji w przedsiębiorstwie w długim okresie czasu. Po drugie, użycie technik informatycznych oddziaływuje na "strategiczną orientację przedsiębiorstwa".

Znaczenie technik informatycznych w odniesieniu do "strategicznej orientacji przedsiębiorstwa" jest widoczne przede wszystkim w sferze walki konkurencyjnej poprzez wpływ na pozycję rynkową przedsiębiorstwa. Sytuacja konkurencyjna w określonej branży jest zdeterminowana przez pięć czynników:

[5] A.-W. Scheer, EDV-orientierte Betriebswirtschaftslehre, jw.

- siłę rynkową nabywców,
- wysokość bariery, utrudniającej wejście na rynek (dla nowych konkurentów),
- stopień substytucyjności produktów przez nowe produkty,
- siłę rynkową dostawców,
- intensywność walki konkurencyjnej między istniejącymi konkurentami.

Koncepcje rozwiązań, opartych na technikach informatycznych mogą oddziaływać na każdy z tych czynników. Sposób tego oddziaływania został dokładnie przeanalizowany przez A.-W. Scheer'a w pracy "EDV-orientierte Betriebswirtschaftslehre".

Istnieje ścisły związek między planowaniem przetwarzania informacji w przedsiębiorstwie a strategicznym planowaniem przedsiębiorstwa. Dla zaplanowania strategii informatycznej proponowane są różne podejścia. Jednym jest *koncepcja Y-CIM*. Centralnym punktem tego podejścia jest ustalenie specyficznych dla przedsiębiorstwa łańcuchów procesów, ich wspomagania przez struktury danych i wbudowanie strategii informacyjnej w ogólną strategię przedsiębiorstwa.

Zgodnie z koncepcją Y-CIM przedsiębiorstwa przemysłowe charakteryzują się dwoma dużymi systemami informacji: systemem planowania i sterowania produkcją, który towarzyszy strumieniowi zleceń oraz systemem projektowania wyrobów i sterowania urządzeniami produkcyjnymi.

Systemy planowania produkcji i sterowania, zdeterminowane strumieniami zleceń, rozpatrują kwestie, mające centralne znaczenie dla ekonomiki przedsiębiorstwa przemysłowego. Wprowadzenie tu na szeroką skalę technik informatycznych spowodowało, że "komputerowo ukierunkowane" systemy planowania i sterowania produkcją przedstawiają inną filozofię planowania i kładą nacisk na inne problemy niż podejścia klasycznej ekonomiki przedsiębiorstwa, np. zupełnie nieistotna jest optymalizacja wielkości partii produkcyjnej, a cała uwaga skierowana jest na elastyczne zarządzanie realizacją zleceń.

Wprowadzenie technik informatycznych w przedsiębiorstwie uwidoczniło fakt, że systemy projektowania wyrobów i sterowania urządzeniami produkcyjnymi, tradycyjnie określane jako techniczne, mają duże znaczenie ekonomiczne. Przykładowo, koszty produkcji wyrobów są w znacznym stopniu zdeterminowane już przez decyzje podejmowane na etapie ich projektowania (a nawet projektowania zakładu).

Poprzez koncepcję integracji Computer Integrated Manufacturing (CIM) dąży się do kompleksowego ujęcia wszystkich funkcji przedsiębiorstwa przemysłowego. Możliwe to jest poprzez wprowadzenie technik informatycznych, dzięki którym tworzony jest spójny i zintegrowany system informacji dla rozwiązywania zadań ekonomicznych i technicznych. Tak więc w koncepcji CIM rozpatrywane są obok siebie zarówno kwestie techniczne, jak i kwestie ekonomiczne, a CIM zmienia w znacznym stopniu przebieg procesów w przedsiębiorstwach przemysłowych. Nowo powstała dziedzina wiedzy komputerowo-zorientowana ekonomika przedsiębiorstwa ma swój udział w kształtowaniu tej koncepcji.

Aspekty CIM, mające zdaniem A.-W. Scheera znaczenie dla ekonomiki przedsiębiorstwa zostały przez niego podsumowane w następujących dziewięciu punktach:[6]

1. CIM określa strategiczną koncepcję dla przedsiębiorstwa.
2. Tworzenie systemów CIM wymaga znacznej wiedzy fachowej (w tym ekonomicznej) i odpowiednich metod rachunku inwestycyjnego.

[6] ScheA.-W. er, EDV-orientierte Betriebswirtschaftslehre, jw.

3. CIM wymaga zdecentralizowanych ekonomicznych koncepcji sterowania.
4. CIM skłania do wymiany danych między przedsiębiorstwami i stosowania standardów dla danych ekonomicznych.
5. CIM wymaga ścisłego powiązania między poziomem procesów i poziomem finansów.
6. CIM prowadzi do niezależnego od zastosowań zarządzania zbiorami danych stałych.
7. CIM zmienia horyzont czasowy procesów planowania.
8. CIM wymaga struktury organizacyjnej zorientowanej na proces lub obiekt.
9. CIM wymusza zastosowanie skomputeryzowanego przetwarzania danych do realizacji funkcji ekonomicznych.

Techniki informacyjne i komunikacyjne zmieniają koncepcje ekonomiki przedsiębiorstwa. Można wymienić tu kilka przykładów. Wprowadzenie skomputeryzowanych systemów planowania i sterowania produkcją zmieniło podejście do procesów planowania w przedsiębiorstwie[7]. Użycie techniki baz danych, przetwarzania w trybie interaktywnym i sieci komputerowych w tych systemach pozwoliło na bardziej efektywne rozwiązania problemów zarządzania danymi i złożoności procesów planowania niż umożliwiały to modele optymalizacyjne planowania produkcji i procesów, oferowane przez ekonomię. Integracja danych i funkcji, wynikająca z użycia systemów baz danych pozwoliła na bardziej niezależne przetwarzanie procedur ekonomicznych, np. zamówień lub sprzedaży. Zastosowanie przetwarzania w trybie interaktywnym umożliwiło poświęcenie większej uwagi nieustrukturalizowanym problemom decyzyjnym. Wprowadzenie sieci komputerowych spowodowało nowy podział uprawnień do podejmowania decyzji, unifikację procedur ekonomicznych między przedsiębiorstwami, nowy (ponad-zakładowy) podział zadań między przedsiębiorstwami poprzez ponad-zakładowe łańcuchy procesów. Wprowadzanie koncepcji CIM powoduje integrację funkcji technicznych i ekonomicznych, np. rozpatrując koszty produkcji bierze się pod uwagę fakt, że o ich wielkości decydują w znacznym stopniu ustalenia przyjęte na etapie projektowania inżynierskiego.

Ekonomika przedsiębiorstwa wpływa na techniki informacyjne i komunikacyjne poprzez formułowanie wymagań wobec tych technik oraz włączanie własnych rozwiązań do zastosowań informatycznych. Przykładowo, modele optymalizacyjne wypracowane przez ekonomię są stosowane do przydzielania zasobów w ramach sieci komputerowych, a koncepcje ekonomiczne struktur danych stanowią podstawę do projektowania modeli danych.

Wzajemne oddziaływania: koncepcje ekonomiczne - rozwiązania informatyczne sprawiają, że rośnie zapotrzebowanie na specjalistów, którzy posiadaliby wiedzę ekonomiczną i informatyczną, pozwalającą wskazywać możliwości wykorzystywania nowopowstających technik informatycznych w zastosowaniach ekonomicznych, stawiać, wynikające z potrzeb ekonomii, konkretne zadania przed informatyką, a także stosować koncepcje i metody ekonomiczne przy kształtowaniu rozwiązań informatycznych.

Zdobycie takiej wiedzy umożliwia "Informatyka gospodarcza".

Informatyka gospodarcza

(projekt programu zajęć dla studentów ekonomii)

Informatyka gospodarcza (wykład, 30 godzin)

[7] Por. A.-W. Scheer, EDV-orientierte Betriebswirtschaftslehre, jw.

Celem jest zapoznanie studentów z dorobkiem informatyki gospodarczej i możliwościami jego wykorzystania w praktyce.

1. Informatyka gospodarcza jako samodzielna dyscyplina naukowa. Powstanie i rozwój. Przedmiot i zakres badań.
2. Technologie informatyczne i ich wpływ na procesy planowania i zarządzania (na przykładzie przedsiębiorstwa przemysłowego i usługowego). Koncepcja komputerowo-zorientowanej ekonomiki przedsiębiorstwa.
3. Analiza poszczególnych metod i technik informatycznych i problemów ich wykorzystania w zastosowaniach ekonomicznych. Systemy baz danych. Przetwarzanie w trybie dialogowym. Sieci komputerowe. Oprogramowanie użytkowe (inżynieria oprogramowania i techniki CASE, standardowe oprogramowanie użytkowe, bazy modeli i metod, systemy eksperckie).
4. Wpływ koncepcji ekonomicznych na rozwój technik informatycznych.
5. Strategiczne planowanie w przedsiębiorstwie a planowanie w zakresie technik informatycznych.
6. Koncepcje zintegrowanego przetwarzania danych. Architektura zintegrowanych systemów informatycznych ARIS.
7. Branżowe systemy informatyczne. Przemysł (planowanie i sterowanie produkcją, Computer Integrated Manufacturing (CIM)). Handel. Banki. Ubezpieczenia. Turystyka.
8. Dziedzinowe systemy informatyczne. Rachunkowość. Marketing. Polityka kadrowa. Planowanie. Biurotyka.
9. Model informacyjny w skomputeryzowanym przedsiębiorstwie.
10. Elektronizacja rynku. Wybrane zagadnienia ekonomiczne, technologiczne i prawne.

Literatura dla studentów:

1. Durlik I., Inżynieria zarządzania, strategia i projektowanie systemów produkcyjnych w gospodarce rynkowej, Wydawnictwo Naukowe amp, Katowice 1993.
2. Kasprzak T., Koncepcje przedsiębiorstwa XXI wieku, mat. konferencji Wydziału Nauk Ekonomicznych Uniwersytetu Warszawskiego, Sopot, październik 1993.
3. Kisielnicki J., Informatyczna infrastruktura zarządzania, PWN, Warszawa 1993.
4. Scheer A.-W., EDV-orientierte Betriebswirtschaftslehre, Springer-Verlag 1990 (tłumaczenie polskie tej książki ukaże się w Wydawnictwach Uniwersytetu Warszawskiego w 1994r.).

Informatyka gospodarcza (laboratorium, 30 godzin)

Celem jest nauczenie studentów posługiwania się wybranym, skomputeryzowanym narzędziem dla modelowania i budowy całościowej koncepcji systemu informatycznego. Studenci opracowują w każdej grupie jeden projekt, sami organizując przebieg prac.

1. Zapoznanie się ze skomputeryzowanym narzędziem modelowania i budowy zintegrowanego systemu informatycznego (np. ARIS).
2. Wybór zagadnienia dla modelowania i budowy systemu.
3. Planowanie oraz organizacja projektu.
4. Analiza problemu i wypracowanie koncepcji docelowej. Zdefiniowanie podstawowych wymagań stawianych przyszłemu systemowi.
5. Modelowanie struktury organizacyjnej, struktury danych, funkcji oraz procesów.
6. Prezentacja opracowanych projektów przez studentów.

Literatura dla studentów:

1. Dokumentacja użytkownika wybranego

oprogramowania dla modelowania i budowy systemu (np. ARIS).

Uwaga: W języku polskim brak podręcznika z informatyki gospodarczej. Przy opracowywaniu projektu programu zajęć przyjęto, że jako podstawowy podręcznik do nauczania przedmiotu będzie wykorzystywana książka: A.-W. Scheer, EDV-orientierte Betriebswirtschaftslehre, Springer-Verlag 1990, przetłumaczona na język polski w ramach realizacji programu TEMPUS.

Literatura

1. Durlik I., Inżynieria zarządzania, strategia i projektowanie systemów produkcyjnych w gospodarce rynkowej, Wydawnictwo Naukowe amp, Katowice 1993.
2. Goliński J., Informatyka gospodarcza, Informatyka nr 9, 1993, s.10-11 i s.15.
3. Kisielnicki J., Informatyczna infrastruktura zarządzania, PWN, Warszawa 1993.
4. Łukasik-Makowska B. (red.), Wskazówki do projektowania mikrokomputerowych systemów użytkowych, Akademia Ekonomiczna im. Oskara Langego we Wrocławiu, Wrocław 1992.
5. Scheer A.-W., EDV-orientierte Betriebswirtschaftslehre, Springer-Verlag 1990.
6. Scheer A.-W., Architektur integrierter Informationssysteme, Springer-Verlag 1991.
7. Scheer A.-W., Architektura zintegrowanych systemów informatycznych (ARIS), Przegląd Organizacji 4/93, s.32-34 (tłumaczenie z "Scheer Magazin" 1/92).
8. Scheer A.-W., Elsner T., Informatyka gospodarcza, artykuł przygotowany do druku w "Przeglądzie Organizacji".

Jan Baranowski
THOMSON POLKOLOR

Techniczne aspekty wdrożenia systemu MRP II w THOMSON POLKOLOR

1. Wprowadzenie.

Od początku swojego istnienia THOMSON POLKOLOR bazował na własnych systemach informatycznych, dostosowanych do jego specyficznych potrzeb. Były to przede wszystkim systemy finansowe, jako że w tej dziedzinie wymagania sprawnego przetwarzania danych były największe. Stworzone systemy pozwoliły w szczególności na prowadzenie równoległych ksiąg finansowych w dwu walutach: PLZ i FRF z uwzględnieniem specyficznych wymagań dla obu walut. Systemy te, pracujące w trybie "on-line" na sieci NOVELL, obejmującej pięć fileserver'ów i około 120 stacji roboczych, spełniły stawiane im wymagania.

Wstępny okres około 2 lat działalności THOMSON POLKOLOR został oceniony bardzo pozytywnie przez centralę THOMSON'a. Tak pozytywnie, że postanowiono przeprowadzić kolejne znaczne inwestycje. Nie zapomniano też o informatyce. Rozwój informatyczny THOMSON POLKOLOR postanowiono oprzeć na kompleksowym, zintegrowanym systemie zarządzania przedsiębiorstwem znanym z teorii pod nazwą MRP II. Wiosną 1993 roku rozpoczęto poszukiwania takiego systemu przy pomocy konsultantów zachodnich. Wytypowano wstępnie 12 systemów tego typu i rozpoczęto akcję ofertową. Na podstawie analizy nadesłanych ofert wytypowano cztery systemy do dalszej analizy. Kryteria jakimi się przy tym kierowano obejmowały m.in.:

- funkcjonalność systemu,
- referencje systemu,
- dostępność systemu na rynku polskim.

Warto zauważyć, że platforma sprzętowa nie stanowiła kryterium wyboru, chociaż ograniczono ją do najbardziej rozpowszechnionych systemów operacyjnych, a więc: DOS, UNIX i AS/400.

Po dwudniowych prezentacjach każdego z wytypowanych produktów, zakwalifikowano dwa z nich do ostatecznej rozgrywki. Obejmowała ona tygodniowe warsztaty, na których analizowano dość dokładnie możliwości funkcjonalne obu systemów. Ostateczna decyzja zapadła w grudniu 1993 roku. Zwycięzcą okazał się system BPCS firmy SSA.

System BPCS dedykowany jest na komputer AS/400, chociaż w niedalekiej przeszłości dostępny będzie również w wersji UNIX'owej. Referencje tego systemu obejmują kilka tysięcy instalacji praktycznie we wszystkich częściach świata.

Wdrożenie systemu BPCS w THOMSON POLKOLOR rozpoczęło się w kwietniu 1994 i przewidywane jest na okres 2 - 3 lat.

Inwestycje towarzyszące:

- zakup i instalacja AS/400 model F35 z dalszą rozbudową do F50,
- instalacja lokalnej strukturalnej sieci komputerowej obejmującej docelowo ok. 500 stacji roboczych,
- instalacja sieci energetycznej zasilającej sieć komputerową,
- przebudowa centrum komputerowego.

2. Konfiguracja sprzętowa systemu.

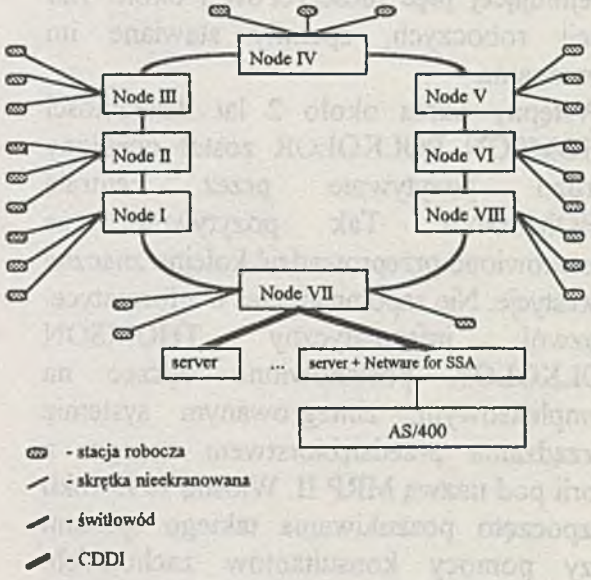
Jak już wspomniano powyżej, system

BPCS dedykowany jest na komputer AS/400 i przystosowany do przetwarzania terminalowego. Pomimo to w skład konfiguracji sprzętowej zainstalowanej w Thomson Polkolor wchodzą, i wchodzić będą docelowo, zaledwie trzy terminale. Użytkownicy pracować będą na mikrokomputerach klasy PC z wykorzystaniem emulacji terminala AS/400. Takie rozwiązanie wynikło zarówno z chęci wykorzystania istniejącego sprzętu, jak i przede wszytkim z potrzeby otwartości i funkcjonalności wykorzystywanego sprzętu. Na tych samych mikrokomputerach można bowiem uruchomić emulację terminala AS/400, jak i normalne programy pod kontrolą sytemów operacyjnych DOS i WINDOWS, w szczególności takie pakiety jak WORD czy EXCEL. Konfiguracja taka umożliwia również wykorzystanie AS/400 w innej roli, jako server'a plików, o czym będzie mowa w dalszej części referatu.

Zdecydowano również o daleko idącej rozbudowie, a właściwie tworzeniu od podstaw, strukturalnej sieci komputerowej ocenianej docelowo na ok. 500 stacji roboczych. Po rozważeniu wielu wariantów zdecydowano zachować system operacyjny Novell jako podstawę tej sieci. Komputer AS/400 został podłączony segmentem sieci Ethernet do servera novelowskiego dedykowanego pod produkt o nazwie NETWARE FOR SAA. Produkt ten spełnia rolę routera pomiędzy transmisją w standarcie IBM i transmisją w standarcie novelowskim (IPX/SPX). Począwszy od tego miejsca cała reszta sieci komputerowej jest typową siecią Novell z transmisją w standarcie IPX. Rozwiązanie takie zapewnia dostęp do AS/400 wszystkim komputerom pracującym w sieci przy pomocy IBM'owskiego produktu o nazwie PC Support. W przeciwieństwie do standardowego sposobu wykorzystania PC Support'u, rozwiązanie takie pozwala na pełną integrację z siecią Novell (jednorodność protokołu trasmisyjnego), możliwość stosowania dowolnych, akceptowanych przez Novell, kart sieciowych i routerów, oraz zmniejsza zajętość pamięci operacyjnej stacji roboczych.

Rozwiązanie sieci komputerowej oparto na ośmiu węzłach połączonych światłowodem (z możliwością łatwego przejścia na szybką transmisję FDDI), i nieekranowanej skrętce kategorii piątej, łączącej stacje robocze z tymi węzłami. W centrum informatyki, stanowiącym jeden z węzłów sieci, zgromadzono 5 (docelowo 8) server'ów Novelowskich (w tym jeden przeznaczony do podłączenia AS/400), połączonych transmisją 100 Mb CDDI. Stanowi to zalążek dalszej rozbudowy sieci do ringu światłowodowego FDDI. Przy budowie sieci komputerowej oparto się na elementach pasywnych AT&T i elementach aktywnych firmy 3COM.

Strukturę sieci ilustruje poniższy rysunek.



3. Harmonogram wdrożenia systemu.

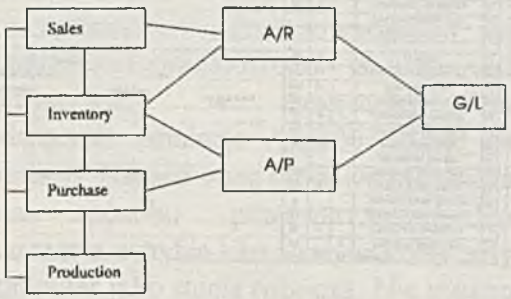
Jak już powiedziano wcześniej, proces wdrożenia systemu przewidziany jest na dwa do trzech lat. Harmonogram wdrożenia podzielony jest na wiele etapów, zgodnych z procedurą dostarczoną przez autorów

systemu, a bazującą na doświadczeniach z dużej ilości wdrożeń tego systemu w różnych warunkach. Do najważniejszych etapów zaliczyć można:

- dokładną analizę możliwości modułów systemu,
- decyzję o zakresie wdrożenia,
- szkolenie grupy wdrożeniowej,
- prototypowanie modułów,
- parametryzowanie systemu,
- ustalenie procedur wykorzystywania systemu,
- ustalenie niezbędnych modyfikacji (w sytemie i w organizacji przedsiębiorstwa),
- ustalenie powiązań z istniejącymi systemami i sposobu przejścia na nowy system,
- szkolenie użytkowników i opracowanie instrukcji wykorzystywania systemu,
- rozpoczęcie użytkowania systemu.

Ze względu na złożoność systemu przewiduje się podział wdrażania systemu na trzy do czterech faz, przy czym zakresy faz nie zostały jeszcze ostatecznie uzgodnione. Wynika to głównie z problemów związanych z wdrażaniem systemów finansowych. Przy tak złożonym i zintegrowanym systemie, wskazane byłoby wdrażanie go w pełnym wymiarze, t.j. również z modułami finansowymi. Jednocześnie jednak moduły finansowe BPCS'a, w aktualnym stanie, nie spełniają wielu wymagań specyficznych dla THOMSON POLKOLOR, a w szczególności warunku podstawowego jakim jest dwuwalutowość rozliczeń finansowych (równoległe prowadzenie ksiąg w dwóch walutach bazowych i walucie orginalnej transakcji). Wdrażanie tych modułów stanowiłoby więc krok do tyłu w stosunku do stanu aktualnego. Rozwiązaniem jest oczywiście modyfikacja systemu. Jest to jednak rozwiązanie ostateczne, ze względu na zakres niezbędnych modyfikacji i praktyczne uniemożliwienie późniejszego przechodzenia na nowsze wersje systemu. Ewentualnym rozwiązaniem może być zapowiadana przez SSA nowa wersja systemu, która, według zapewnień dostawcy, powinna uwzględniać wiele z postulowanych rozszerzeń. Ostateczna decyzja zapaść ma po zapoznaniu się z nową wersją systemu w połowie 1994 roku.

Innym rozwiązaniem (być może przejściowym) może być częściowe wdrożenie systemu. Zgodnie z doświadczeniami firmy SSA podział wrażania systemu może wyglądać tak, jak na poniższym rysunku:

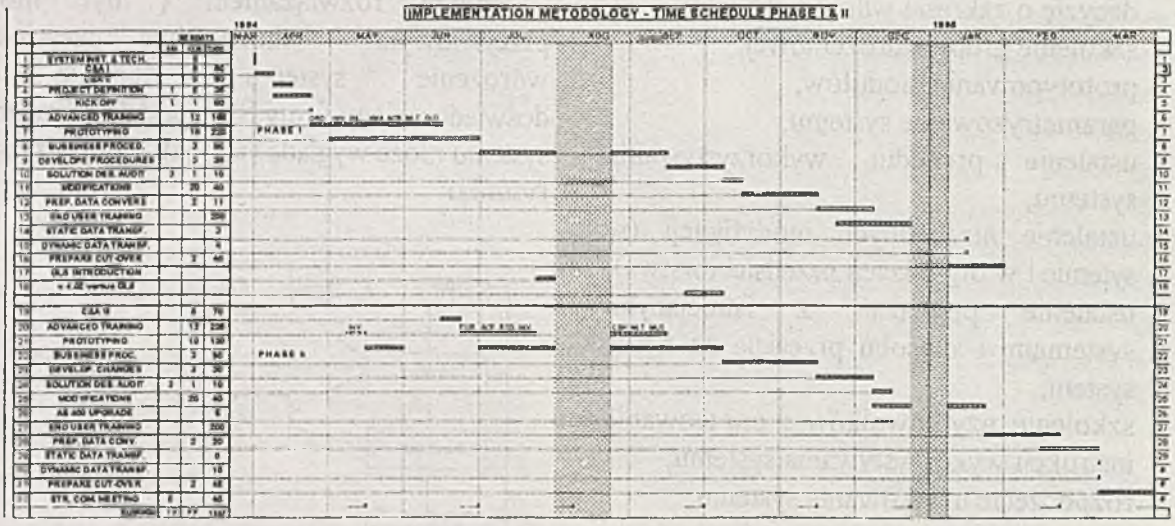


Podział poziomy jest podziałem funkcjonalnym na część dotyczącą sprzedaży (łącznie z magazynem wyrobów gotowych), część dotyczącą zakupów, oraz część produkcyjną. Podział pionowy przedstawia dwie możliwości zintegrowania systemu BPCS z innymi systemami finansowymi. Możliwe jest zachowanie całości "starych" systemów finansowych i integrowanie na styku modułów należności i rozliczeń z modułami sprzedaży, zakupów i zapasów, lub integrowanie na styku modułów rozliczeń należności i zobowiązań z księgą główną. Drugi sposób jest znacznie prostszy, wymaga jednak wdrożenia modułów rozliczeń należności i zobowiązań które, w obecnej postaci, nie uwzględniają wielu wymagań Thomson-Polkolor.

Uwzględniając powyższe uwarunkowania zdecydowano rozpocząć wdrażanie systemu BPCS od modułów sprzedaży (faza I) i

zakupów (faza II) z ewentualnym ich wzbogaceniem (gdyby okazało się to możliwe) o moduły odpowiednio należności i zobowiązań.

Wstępny harmonogram działań na najbliższy okres przedstawia poniższy wykres:



4. Problem modyfikacji systemu i integracji z istniejącymi systemami.

Mimo daleko posuniętej uniwersalności i parametryzowlności systemu BPCS, wydaje się niezbędne wykonywanie pewnych modyfikacji i ulepszeń systemu. Można je podzielić na kilka grup:

- modyfikacje funkcjonalne systemu,
- modyfikacje i rozszerzenia związane z interface'm użytkownika,
- dobudowywanie dodatkowych funkcji bądź możliwości,
- dobudowywanie dodatkowych modułów,
- tworzenie powiązań między różnymi systemami,
- tworzenie raportów dostosowanych do potrzeb użytkowników, w szczególności z danych pochodzących z różnych systemów.

Polityka Thomson Polkolor w zakresie wykonywania powyższych prac jest następująca:

1. wszelkie modyfikacje wymagające zmian w istniejących programach bądź reorganizacji bazy danych, wykonywane będą wyłącznie przez dostawcę systemu lub firmę go reprezentującą,
2. inne modyfikacje i udoskonalenia, w szczególności integracja z innymi systemamii dodatkowe raporty, wykonywane będą własnymi siłami, przy użyciu jednorodnych narzędzi.

Narzędziem wytypowanym do wykonywania prac wymienionych w p.2 jest SQLWindows firmy GUPTA. Narzędzie to zapewnia bezpośredni dostęp do baz danych zarówno AS/400 jak i Netware SQL na sieci Novell, oferując przy tym nowoczesny i efektywny warsztat do tworzenia oprogramowania obiektowo i zdarzeniowo zorientowanego. Dostęp do bazy danych AS/400 umożliwia router do AS/400, dostarczany przez Guptę. Pomimo przejściowych problemów związanych z dostosowaniem tego routera do nowej wersji systemu operacyjnego AS/400, wydaje się on być wręcz idealnym narzędziem do tworzenia aplikacji typu klient-server z wykorzystaniem AS/400 jako servera. Nieco bardziej skomplikowana sytuacja rysuje się po stronie dostępu do baz danych opartych na Netware SQL. Router, dostarczany przez firmę Novell, praktycznie nie nadaje się do użytku. Istnieje jednak możliwość wykorzystania funkcji zewnętrzych zawartych w bibliotekach DLL dostarczanych przez Novell'a, bądź wykorzystania możliwości zaimportowania tzw. customs control z Visual Basic, realizujących dostęp do tych baz danych. Prace nad znalezieniem najwygodniejszego rozwiązania tego problemu trwają.

5. AS/400 a archtektura klient-server.

Opinie na temat komputera AS/400 są mocno podzielone. Od pełnego zachwytu i uznania za najwyższe osiągnięcie techniki komputerowej, do prawie całkowitej negacji i przeznaczenia do muzeum. Prawda jak zwykle leży pośrodku. AS/400 nie jest najnowocześniejszym komputerem i być takim nie może przy polityce IBM zapewnienia zgodności ze starymi systemami. Jest jednak komputerem bezpiecznym i funkcjonalnym. Nie posiada grafiki i jest przstosowany przede wszystkim do przetwarzania terminalowego w starym stylu, ale wcale nie musi być tak wykorzystywany. Przy obecnym błyskawicznym rozwoju sprzętu mikrokomputerowego, niektóre rozwiązania (szczególnie interface użytkownika) stosowane w systemach takich jak BPCS wydają się anachroniczne, należy jednak pamiętać, że są to wyjątkowo złożone systemy a prace nad ich przygotowaniem trwają wiele lat. BPCS nie jest nowoczesnym systemem. Jest systemem dostosowanym do przetwarzania terminalowego i jako taki jest wykorzystywany w tysiącach firm posiadających instalacje AS/400 oparte na terminalach. Nie jest więc możliwa radykalna zmiana sposobu przetwarzania. Do przetwarzania w trybie klient-server konieczny jest komputer jako stacja robocza. Nie można tego zrobić na terminalu. Nie może więc być mowy o szybkim przestawieniu systemów typu BPCS na nowy sposób przetwarzania. Pomimo tych ułomności BPCS jest niewątpliwie jednym z najlepszych systemów w swej klasie. Jest to wynikiem ogromnych funkcjonalnych możliwości systemu. Istnieje jednak możliwość tworzenia w pełni nowoczesnych aplikacji z wykorzystaniem AS/400 jako servera baz danych. Taką możliwość daje m.in. SQLWindows firmy GUPTA. Możliwy jest tu dostęp do bazy danych stworzonej przez BPCS i wykorzystanie zawartych w niej danych, bądź powiązanie tych danych z bazami danych funkcjonującymi na innych platformach sprzętowych. Ten kierunek rozwoju został przyjęty w THOMSON POLKOLOR.

Włodzimierz Adamski
WSK PZL Mielec - Zakład Lotniczy

Nowoczesne formy wytwarzania przy zastosowaniu dużych systemów CAD/CAM

- *Przedstawienie strategi rozwoju przedsiębiorstw w XXI wieku w wyniku zaistnienia nowych technologii informatycznych*
- *Ogólne cechy "aktywnego" wytwarzania (agile manufacturing).*
- *Systemy CAD/CAM podstawą pracy przyszłego przedsiębiorstwa stosującego technikę aktywnego" wytwarzania.*
- *Przykład sieci komputerowej wykorzystującej systemy CAD/CAM.*
- *Rola geometrii numerycznej w projektowaniu i wytwarzaniu wspomaganym komputerowo.*
- *Doświadczenia z wdrażania systemu CAD/CAM CADDS5 firmy Computervision.*
- *Przedstawienie przykładów pilotażowych wdrożeń wraz z rysunkami i zdjęciami.*
- *Jak ekonomicznie wdrażać duże systemy CAD/CAM w przedsiębiorstwie.*

Charakterystyka "aktywnego" wytwarzania

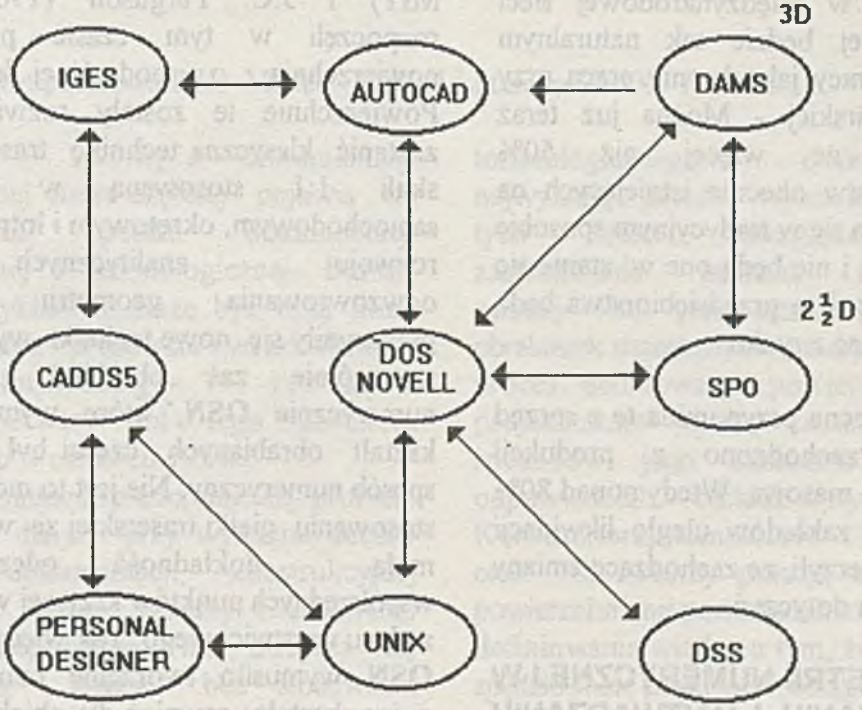
Ostatnie sto lat działalności przemysłowej człowieka było zdominowane koncepcją produkcji masowej, której generalną cechą było wytwarzanie dużej ilości egzemplarzy tego samego produktu wykonanych przy niskich kosztach. Obecnie produkcja masowa wyrobów o wysokiej jakości będzie wypierana przez nowy prężny system produkcyjny wspierany komputerem i siecią komputerową. System ten jest nazywany *aktywnym wytwarzaniem* AM (*Agile Manufacturing*). Jego charakterystycznymi cechami są:

- produkty i usługi są określane i definiowane przez klienta,
- producent szybko reaguje na produkty o krótkim okresie istnienia na rynku (jak np produkty związane z modą),
- produkty nie są przedmiotem tylko jednorazowej sprzedaży, ale także są platformą wysokiej jakości obsługi posprzedażnej,
- wyroby i procesy produkcyjne spełniają wymogi ekologiczne,
- zadania te mogą być osiągalne przez aktywne i kompetentne zespoły pracowników kierowane w tak i sposób aby samodzielnie mogły rozwiązywać problemy, a nie tylko wykonywać polecenia swoich przełożonych.

W świecie produkcji masowej producenci badają upodobania klientów, ale wyrób projektowany i wytwarzany jest przez producenta. W świecie "aktywnego wytwarzania" klient lub jego przedstawiciel np. inżynier sprzedawca w elektronicznym centrum sprzedaży "pracuje" jako projektant. Podobne zalążki takich systemów już istnieją jak np w Toyocie gdzie klient ustala i określa jak ma wyglądać jego samochód, który jest następnie robiony i dostarczony w przeciągu 5 dni, a w przemyśle odzieżowym ten czas reakcji wynosi odpowiednio kilka godzin. To nie znaczy, że można powiedzieć, że nie będzie istnieć produkcja masowa. Jednak wyroby i usługi wysokiej jakości będą bazowały na zasadach organizacyjnych *aktywnego wytwarzania* AM. Dotychczasowe badania potwierdzają, że wśród wielu czynników aktywnego wytwarzania centralną i najważniejszą rolę odgrywają systemy **CAD/CAM**.

Systemy te wyróżniają się następującymi cechami:

- interaktywne systemy komputerowe umożliwiają klientom tworzyć i konfigurować wyrób wg swoich życzeń i upodobań
- niezależnie od położenia miejsca pracy ludzie mogą wspólnie pracować nad jednym projektem
- szerokie wprowadzenie systemów multimedialnych
- elektroniczny handel w międzynarodowej sieci multimedialnej
- szerokie stosowanie i korzystanie z sieciowanej dystrybucji bazy danych
- więcej dostępnych otwartych systemów informacyjnych w zakładach z większymi ilościami danych niż to jest możliwe obecnie
- sprzęt komputerowy i jego oprogramowanie daje obecnie większe możliwości odnośnie symulacji modelowania, przenoszenia danych z konstrukcji do obliczeń i do planowania produkcji
- lepsze zrozumienie stosowanych metod matematycznych w konstruowaniu i projektowaniu
- uzgodnione standardy w oprogramowaniu i przekazywaniu informacji. Na rys.1 przedstawiono powiązania różnych systemów **CAD/CAM** stosowanych w WSK PZL Mielec.



Rys.1 Systemy CAD/CAM stosowane w WSK PZL Mielec

Powstający system aktywnego wytwarzania wprowadza pewne zmiany organizacyjne w przedsiębiorstwie które:

- Dadzą większe możliwości małym przedsiębiorstwom i grupom, które będą współpracowały i współdziałały w celu osiągnięcia wzajemnych korzyści we wspólnych projektach i zadaniach
- Przedsiębiorstwa stworzą takie organizmy, które będą szukały swoich możliwości na rynku i będą natychmiast reagować na pojawiające się potrzeby rynkowe. Obecna powolna hierarchiczna struktura zarządzania będzie bezpowrotnie zanikać.
- Proces produkcyjny samorzutnie będzie obejmował wszystkie struktury organizacyjne przedsiębiorstw. Obecne struktury działów i wydziałów przestaną być ważne, zanikną "granice" podziału tych służb i komórek.
- Pracowanie w międzynarodowej sieci komputerowej będzie tak naturalnym sposobem pracy jak obecnie praca przy desce kreślarskiej - Można już teraz stwierdzić, że więcej niż 50% przedsiębiorstw obecnie istniejących na rynku zamyka się w tradycyjnym sposobie wytwarzania i nie będą one w stanie się przekształcić. I te przedsiębiorstwa będą musiały znikać z rynku.

Sytuacja obecna przypomina tę z sprzed lat, kiedy przechodzono z produkcji rzemieślniczej na masową. Wtedy ponad 80% rzemieślniczych zakładów uległo likwidacji ponieważ nie wierzyli, że zachodzące zmiany będą właśnie ich dotyczyć.

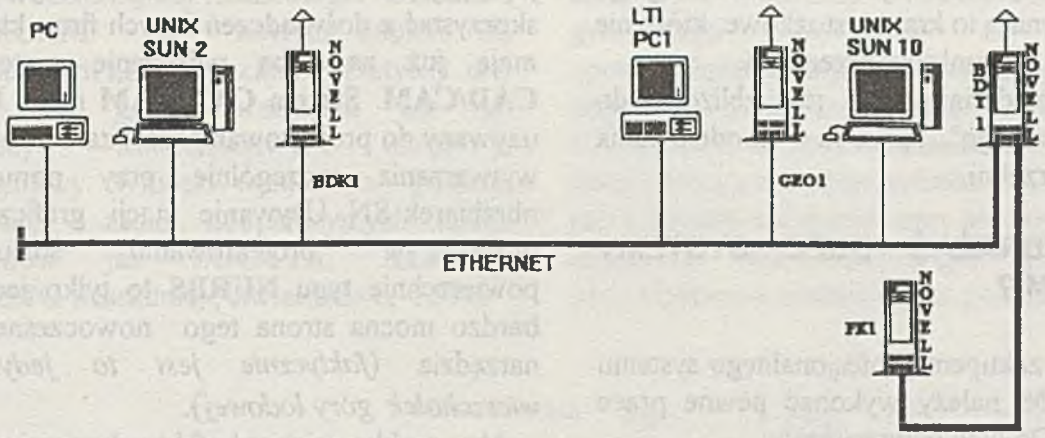
ROLA GEOMETRII NUMERYCZNEJ W PROJEKTOWANIU I WYTWARZANIU WSPOMAGANYM KOMPUTEROWO

Geometria odgrywa główną rolę w projektowaniu i wytwarzaniu wspomaganym komputerem jak i również w analizie wytrzymałościowej. Badania z zakresu optymalnej reprezentacji geometrii, rozwój pewnych i efektywnych metod komputerowego przetwarzania tej geometrii były i ciągle są istotnym obszarem badań i rozwoju od ponad 30 lat. Od wczesnych lat do połowy 1980 technika zwana **NURBS** (**N**on-**U**niform **R**afional **B**-**S**plines) pojawiła się popularna metoda przedstawiania krzywych i powierzchni, szczególnie dla kształtów stosowanych w przemyśle. **NURBS** zostały włączone w wiele przemysłowych standardowych pakietów wymiany danych jak: **IGES, PHIGS i STEP.**

Pojęcie geometrycznego modelowania pojawiło się w użyciu we wczesnych latach 1970 wraz z rozwojem grafiki komputerowej jak i technologii **CAD/CAM.** W połowie 1960 D.T. Ross (1967-MIT) rozwinął i unowocześnił komputer dla graficznego programowania. S.A. Coons (1963,1965-MIT) i J.C. Ferguson (1964-Boeing) rozpoczęli w tym czasie prace nad powierzchnią o podwójnej krzywiźnie. Powierzchnie te zostały rozwinięte aby zastąpić klasyczną technikę trasowania w skali 1:1 stosowaną w przemyśle samochodowym, okrętowym i lotniczym. Do rozwoju analitycznych metod odwzowrowania geometrii samolotu przyczyniły się nowe techniki wytwarzania, szczególnie zaś obrabiarki sterowane numerycznie **OSN**, które wymagają aby kształt obrabianych części był podany w sposób numeryczny. Nie jest to możliwe przy stosowaniu giętki traserskiej ze względu na małą dokładność odczytywanych współrzędnych punktów krzywej wykreślonej z planu warstwicowego. Tak więc stosowanie **OSN** wymusiło tworzenie numerycznego opisu kształtu rowniez dla obiektów, które skonstruowano metodami tradycyjnymi. Obecnie, jednym z najważniejszych problemów, z jakim spotykają się konstruktorzy na całym świecie przy elektronicznym zapisie konstrukcji, jest

graficzna wymiana informacji między kooperującymi zakładami, jak i też integracja miedzy rożnymi systemami komputerowymi. Zagadnienia te zostały opisane dokładnie w [16].

Zastosowanie sieci komputerowej w pracy konstruktora i technologa wraz z odpowiednim oprogramowaniem zmieniło jakość i metody ich pracy. Zamiast



Rys.2 Sieć komputerowa PZL Mielec - Zakład Lotniczy UNIX - NOVELL

"papierowej" formy dokumentacji konstrukcyjnej coraz częściej pojawia się "elektroniczna" postać dokumentacji konstrukcyjnej i technologicznej. Bardzo dobrym przykładem może być tutaj firma BOEING, która opracowała i wykonała swój nowy samolot Boeing - 777 bez użycia papieru. Pierwszy lot tego samolotu zaplanowano w czerwcu 1994r.

Coraz ważniejszą sprawą staje się problem standaryzacji danych przy wymianie między partnerami dokumentacji konstrukcyjnej stosującymi rożne systemy **CAD/CAM.** żadne przedsiębiorstwo nie utrzyma się dzisiaj na rynku bez stosowania odpowiedniej klasy systemow **CAD/CAM.** Automatyzacja prac konstrukcyjnych i procesu wytwarzania samolotu, zmierza głównie do znacznego skrócenia cyklu produkcyjnego, obniżenia pracochłonności opracowania konstrukcyjnego i technologicznego oraz zapewnienia najwyższej jakości wykonania. Realizacja tych celów bewzględnie wymaga zastosowania techniki komputerowej, opartej na rozwiązaniach sieciowych i obrabiarek sterowanych numerycznie. Obecnie proces definiowania powierzchni zajmuje projektantom dużo czasu. Jednym ze sposobów jego skrócenia jest użycie odpowiednich technik optymalizacyjnych. Konstruktorzy samolotów i samochodów oraz Ci którzy pracują nad złożonymi powierzchniami podczas ich komputerowego definiowania wiedzą o tym, że muszą często spędzić wiele czasu nim osiągną zamierzony efekt. Rożne firmy stosują różne metody. Zakład Lotniczy **SAAB-SCANIA** stosuje metodę zwaną **FANGA** (Formela **ANG**le **A**nalyzis). Jest to metoda zbliżona do metody wypracowanej przez autora w Polskich Zakładach Lotniczych. Obydwie

metody bazują na wstępnym modelowaniu powierzchni przy pomocy krzywych stożkowych. Wstępnie powierzchnie definiuje się jako zbiór płaskich krzywych II stopnia. Zaletą tej metody jest:

- mała ilość danych wejściowych
- w jednym kierunku kształt powierzchnii automatycznie jest zadowalajacy. Zapewniają to krzywe stożkowe, które nie posiadają punktów przegięcia.
- sposób definiowania jest zbliżony do "naturalnego" sposobu modelowania powierzchni.

JAK WDRAŻAĆ DUŻE SYSTEMY CAD/CAM ?

Przed zakupem profesjonalnego systemu **CAD/CAM** należy wykonać pewne prace wstępne. Do nich będą należały:

- określenie wyraźnie swoich potrzeb w zakresie projektowania i wytwarzania wspomaganego komputerem
- wybranie oprogramowania i platformy sprzętowej. W tym wypadku najkorzystniej jest "wypożyczyć " system **CAD/CAM** na np. 3 miesiące i testować go w warunkach danego przedsiębiorstwa. Nie jest to obecnie problemem, ponieważ dostawcy oprogramowania zgadzają się na te warunki. W Polsce takie formy działania przyjęły firmy takie jak **ZETO RODAN i COMPUTERVISION**. Bardzo ważne jest przed testowaniem przeprowadzenie wstępnego szkolenia w zakresie obsługi systemu **CAD/CAM**. Trzymiesięczny okres eksploatacji próbnej systemu pozwoli ocenić jego wartość i zalety oraz sprawdzić jego przydatność w warunkach działania danego przedsiębiorstwa. Następnie należy wybrać jakiś projekt jako pilotażowy, dotyczący np. części oprzyrządowania, czy części wyrobu.

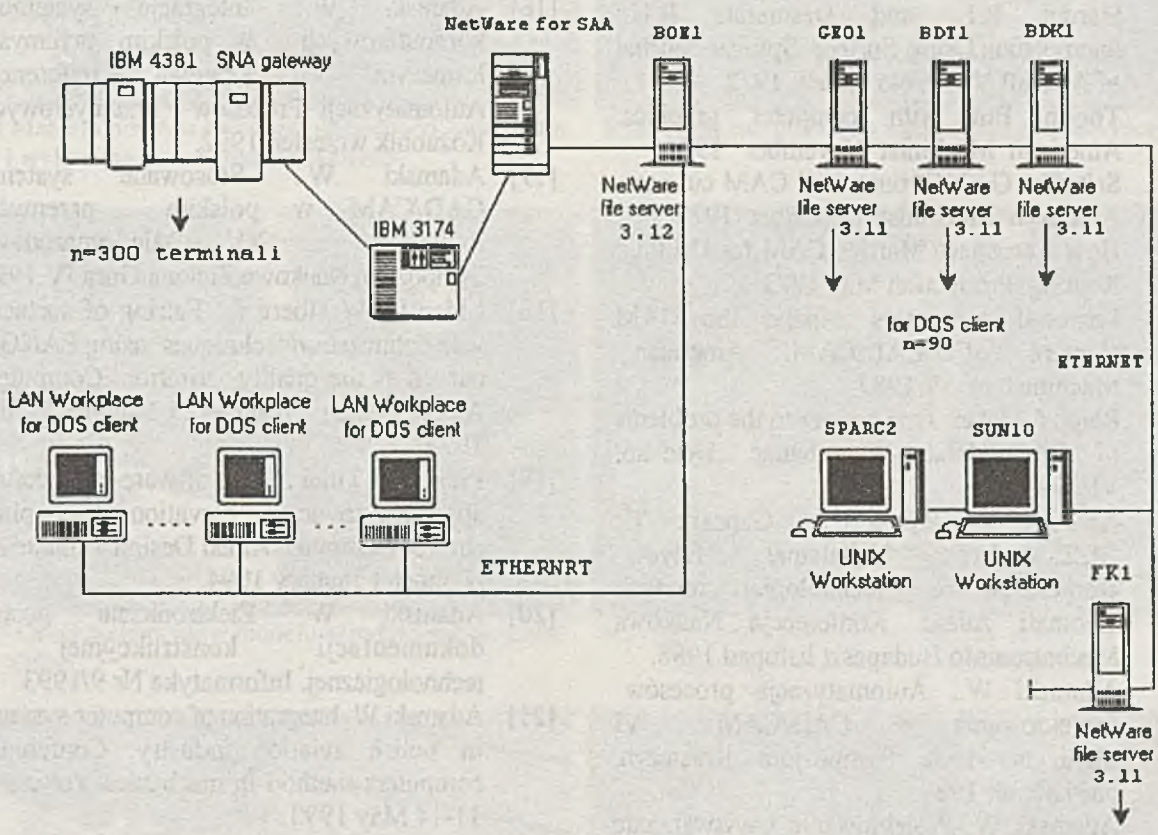
Pełne wdrożenie i opanowanie systemu **CAD/CAM** jest bardzo czasochłonne i kosztowne. Jednocześnie utrzymanie się na rynku przedsiębiorstwa jak i kooperacja międzynarodowa nie pozostawia zakładom lotniczym żadnej innej możliwości wyboru. Można tylko wybrać najtańszą i najszybszą drogę "dojścia" do tego aby istnieć na rynku jako firma. Należy przede wszystkim skorzystać z doświadczeń innych firm, które mają już za sobą wdrożenie systemu **CAD/CAM**. System **CAD/CAM** może być używany do projektowania jak i zarówno do wytwarzania szczególnie przy pomocy obrabiarek SN. Używanie stacji graficznej tylko w projektowaniu stosując powierzchnie typu **NURBS** to tylko jedna bardzo mocna strona tego nowoczesnego narzędzia (*faktycznie jest to jedynie wierzchołek góry lodowej*).

Aby szybko osiągnąć efekty ekonomiczne w WSK PZL Mielec postąpiono następująco:

- wybrano kilka pojedynczych części do zaprojektowania i wykonania (np skomplikowane części z samolotu **IRYDA i ATR-72**),
- zaprojektowano używając systemu **CADDS5** (powierzchnie typu **NURBS** i geometrię bryłową),
- wygenerowano w systemie **CADDS5** drogę narzędzia dla obrabiarki sterowanej numerycznie (OSN),
- wdrożono programy obróbcze na OSN.
- niezależnie opracowano także projekt nowego samolotu
- opracowano i wdrożono do produkcji nową wersję nadwozia samochodu Polonez-truck.

Należy zwrócić uwagę, że dobre systemy **CAD/CAM** w bardzo krótkim czasie i przy małym nakładzie unaoczniają, że nie służą one do tworzenia tylko ładnych obrazków i rysunków ale rzeczywiście są dobrym i sprawnym narzędziem produkcyjnym.

Doświadczenia innych firm jak i nasze własne wyraźnie wskazują, że wdrożenie systemu **CAD/CAM** należy przede wszystkim skoncentrować na stronie ***technnologicznej***. Umożliwia to przedsiębiorstwu natychmiastowy wzrost jego możliwości z zakresu usług kooperacyjnych z najbardziej renomowanymi partnerami w świecie o najnowocześniejszej technologii. Dziedzina **CAD/CAM** jest nowym dopiero się rozwijającym działem techniki. Dotyczy ona wszystkich gałęzi przemysłu jak np. lotniczy, samochodowy, okrętowy, maszynowy. Wnioski wyniesione podczas realizacji zadań kooperacyjnych takich zakładów jak **SOCATA, ALENIA, BOEING** uzasadniają stwierdzenie, że nie tylko prestiż i renoma firmy, która używa znane systemy **AD/CAM** są o wiele wyższe. Firma taka jest automatycznie zaliczana do "rodziny" przedsiębiorstw stosujących najnowocześniejszą technologię i zarazem spełnia wysokie wymagania jakościowe. Łatwiej jest się wtedy porozumieć na poziomie problemów technicznych, ponieważ stosuje się takie same standardy wymiany informacji, gwarantuje najwyższą jakość opracowakonstrukcyjnych i technologicznych (taki bowiem narzuca automatycznie system **CAD/CAM**). Głównym atutem jest wysoka jakość, którą gwarantuje technika **CAD/CAM** jak i bardzo krótkie terminy niemożliwe do uzyskania inną technologią. Programy przyszłościowe przewidują że po roku 1995



Rys.7 Globalna sieć komputerowa w WSK PZL Mielec

2/3 stanowisk pracy będzie ZWIĄZANYCH Z KOMPUTEREM. Specyficznym segmentem rynku obecnie jest pojawiające się zapotrzebowanie na usługi w technice **CAD/CAM**. Rozwój takiego zespołu usługowego w technice **CAD/CAM** może zaprocentować eksportem najwyższej formy pracy jaką jest "***myśl techniczna***" zapisana na magnetycznym nośniku informacji. Po całkowitym wdrożeniu systemu **CADDS5** na jednym stanowisku można dopiero przystąpić do realizacji drugiej fazy tj. rozbudowy sieci o następnych kilka stanowisk. Globalna sieć komputerowa łącząca mainframe'a tj IBM 4381 z siecią NOVELL i ze stacjami graficznymi SUN 10 przedstawiono na rys.7

Literatura:

[1] Harder R.L. and Desmarais R.N.: Interpolation Using Surface Splines. Journal of Aircraft Vol.9, No 2, Feb. 1972

[2] Tooling Built with computer graphics. American Machinist November 1972.

[3] Schaffer G.: NC router and CAM cut cost. American Machinist December 1979.

[4] How Aerospace Marries CAM for Drilling, Routing. Production May 1982

[5] Personal computers expand the CAM element of CAD/CAM. American Machinist no. VI/1987

[6] Ralph J.Mayer: One answer to the problems of CAD database exchange. Byte no. VI/1987

[7] Adamski W., Rybak Cz., Gancarz T.: "PZL-Mielec" Vallalatnal folyo szerkesztesi es technologiai munkak automati zalasa. Konferencja Naukowa Mechatroninfo Budapeszt listopad 1988.

[8] Adamski W.: Automatyzacja procesów projektowania CAD/CAM. VI Międzynarodowe Sympozjum Krasiczyn, październik 1989.

[9] Adamski W. Projektowanie i wytwarzanie samolotów wspomagane komputerem Aerotechnika Lotnicza nr 11/91

[10] Elliot W.S.: Computer-aided machanical engineering: 1958 to 1988 Computer-Aided Design Volume 21 number 5 June 1989.

[11] Adamski W.: Zasady numerycznego modelowania zewnętrznych kształtów obiektów. Przeglad Mechaniczny nr 3/92

[12] Cesky Vybor Strojnicke Spolecnosti CSVTS. Stav a Vyuziti Vypocenti Techniky Pri Konstruovani a Vyrobe Letadel. Duben 1984

[13] Adamski W.: Projektowanie i wytwarzanie samolotów wspomagane komputerem w WSK "PZL-Mielec". Konferencja Naukowa Rydzyna 1988

[14] Adamski W.: Integracja systemów komputerowych w przemyśle lotniczym. III Konferencja Naukowa aktualnych problemów lotnictwa polskiego Warszawa październik 1990

[15] Saab Aicraft Division: CAD/CAM and Geometry Saab Aicraft Division application of advanced technology II 1991.

[16] Adamski W.: Integracja systemów komputerowych w polskim przemyśle lotniczym. VII Krajowa Konferencja Automatyzacji Procesów Przemysłowych Kozubnik wrzesień 1992.

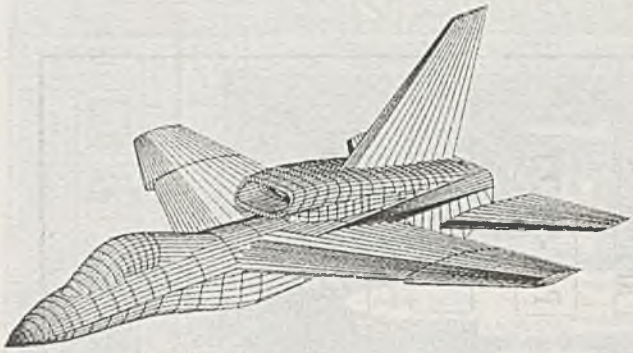
[17] Adamski W.: Stosowane systemy CAD/CAM w polskim przemyśle lotniczym. XV Międzynarodowe Sympozjum Naukowe Zielona Góra IV 1993

[18] Liden G. Westberg S.: Fairing of surfaces with optimization techniques using FANGA curves as the quality criterion. Computer-Aided Design Volume 25 Number 7 July 1993.

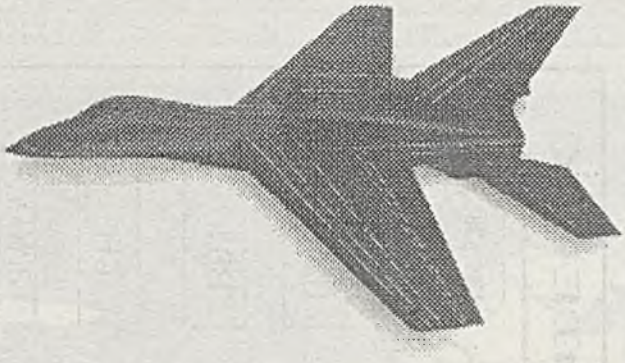
[19] Piegl L., Tiller W.: Software-engineering approach to degree elevation of B-spline curves. Computer-Aided Design Volume 26 Number 1 January 1994.

[20] Adamski W. Elektroniczna postać dokumentacji konstrukcyjnej i technologicznej. Informatyka Nr 9/1993

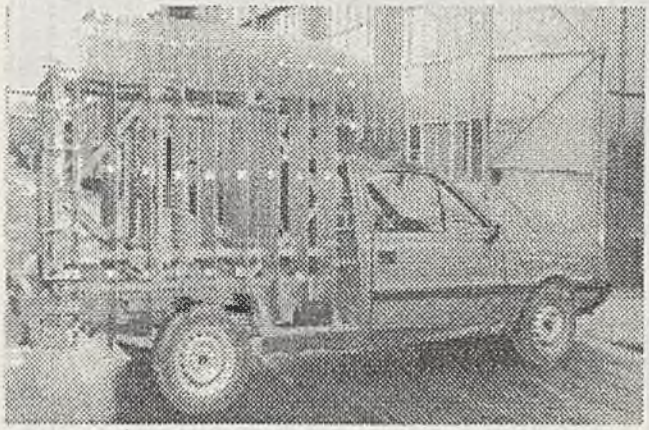
[21] Adamski W. Integration of computer systems in polish aviation industry. Conference computers method in mechanics Volume 1 11-14 May 1993



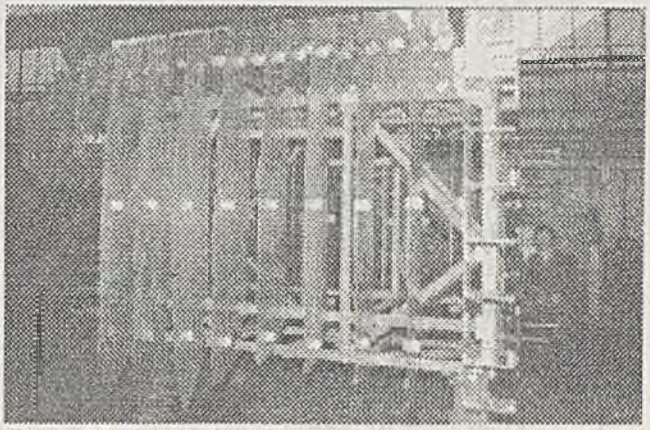
Rys.3 Geometria numeryczna samolotu KOBRA



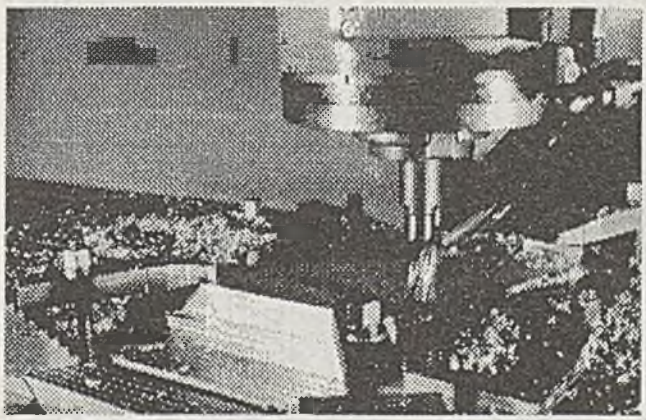
Rys.3a Model aerodynamiczny samolotu KOBRA wykonany na OSN



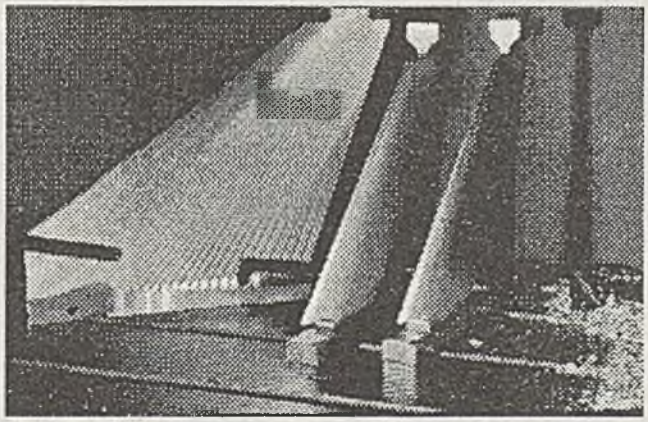
Rys.4 Makieta nowego nadwozia samochodu zaprojektowana i wykonana przy pomocy CADDS5



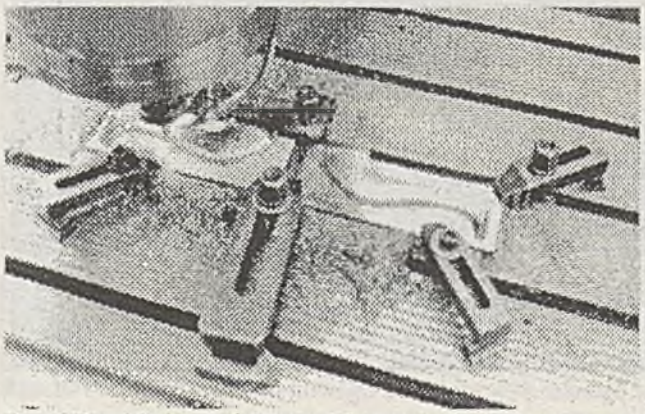
Rys4a Makieta nadwozia samochodu podczas montażu

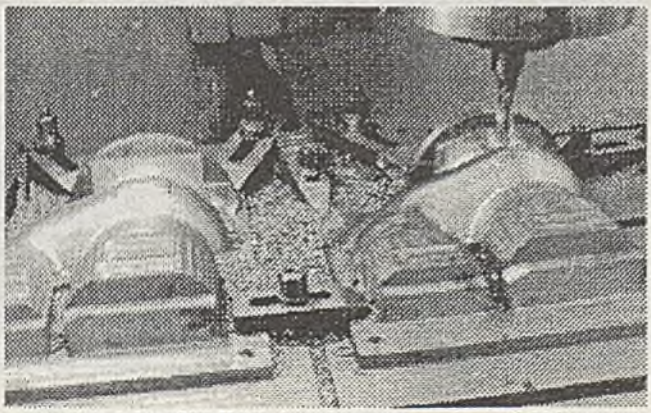


Rys.5 Obróka elementów modelu aerodynamicznego na OSN

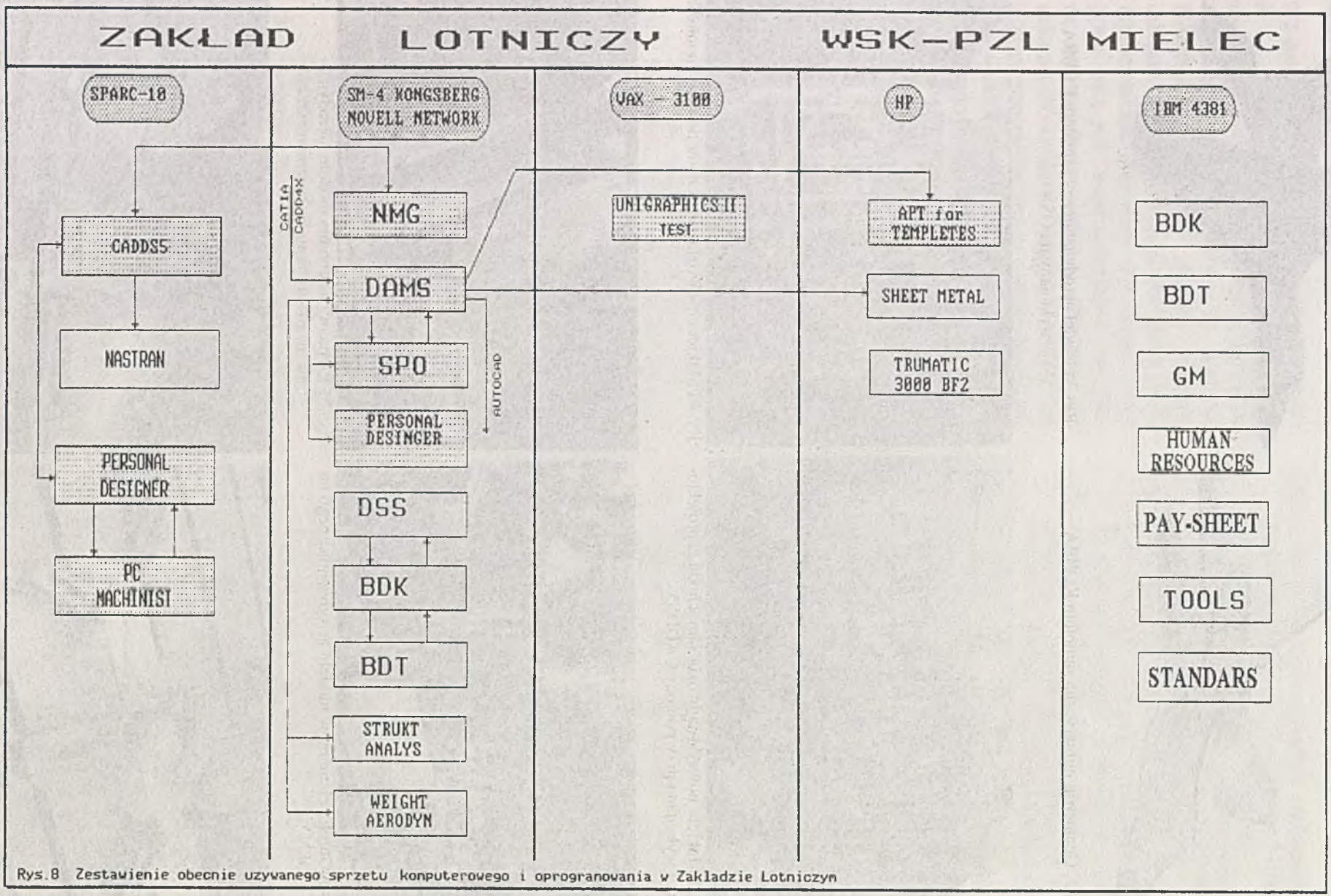


Rys.5a Elementy usterzenia modelu aerodyna - micznego samolotu wykonane na OSN

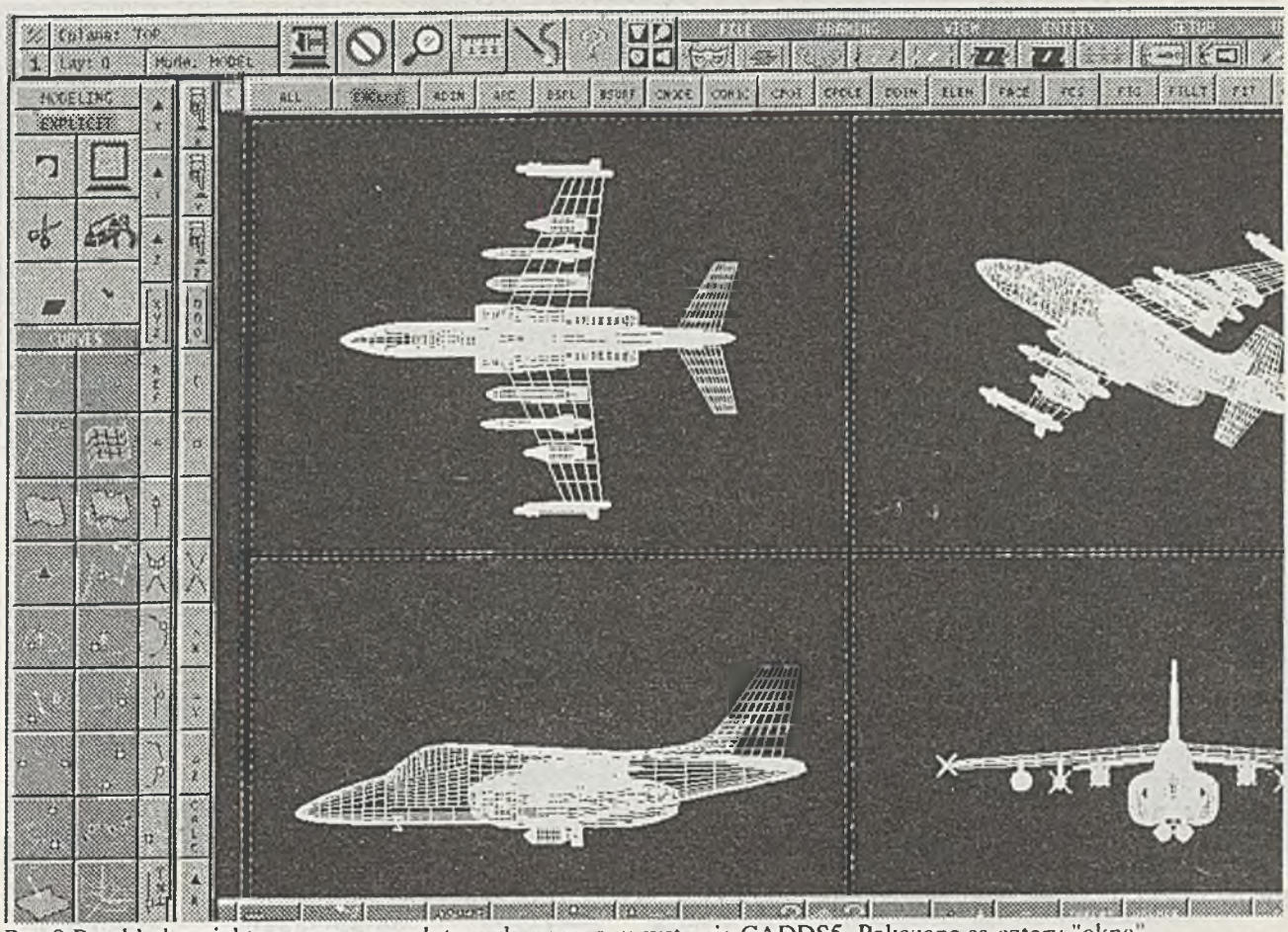




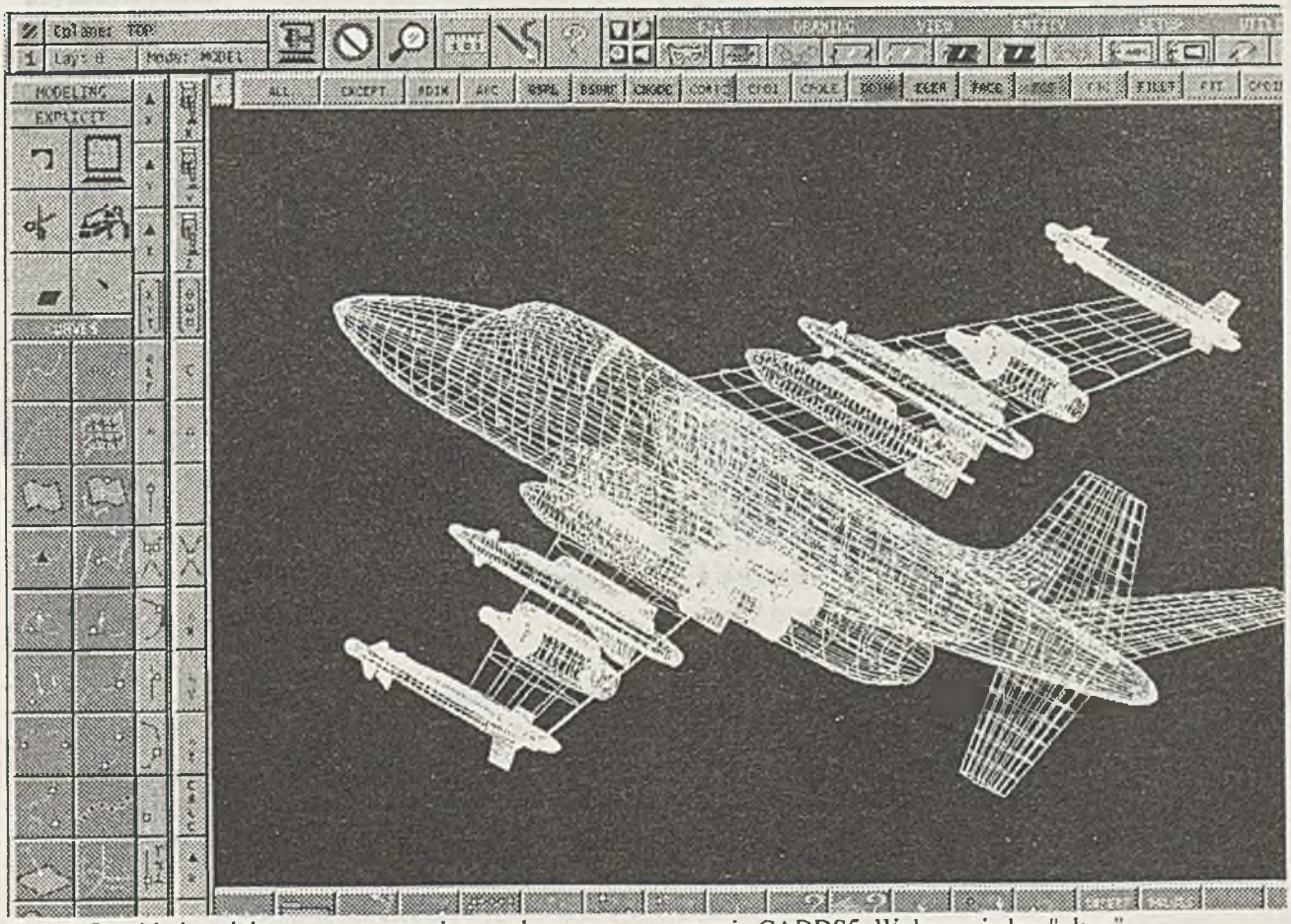
Rys.6 Przykłady części zaprojektowanych przy pomocy systemu PC MACHINIST i wykonywanych na OSN



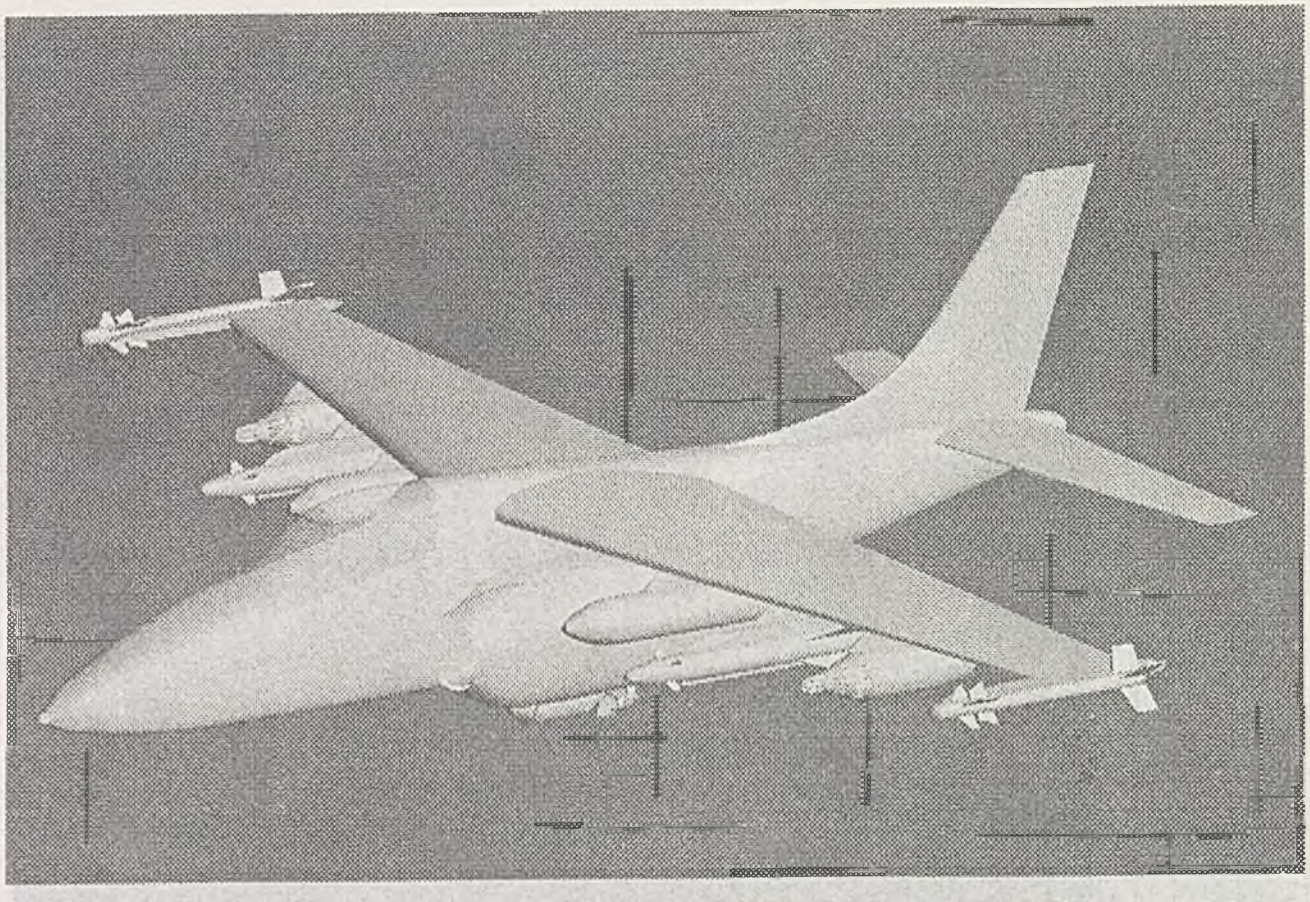
Rys.8 Zestawienie obecnie uzywanego sprzetu konputerowego i oprogranovania w Zakladzie Lotniczyn



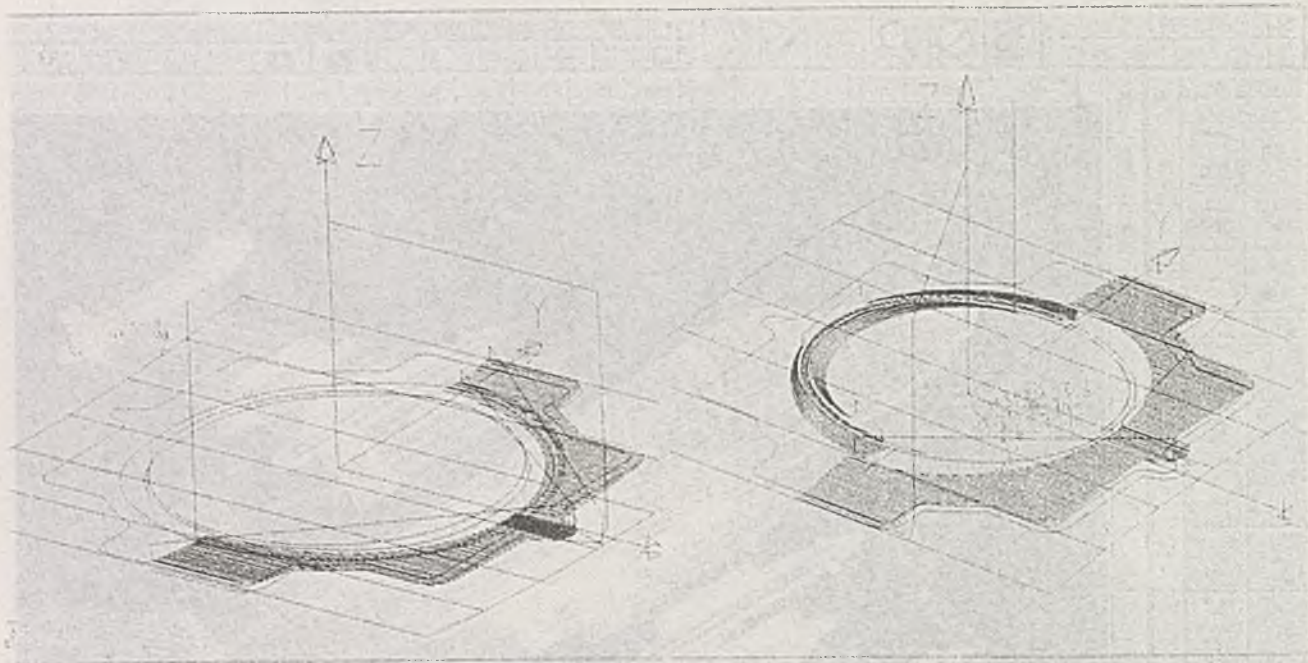
Rys.9 Przykład projektu nowego samolotu wykonanego w systemie CADDS5. Pokazane są cztery "okna"



Rys.10 Przykład projektu nowego samolotu wykonanego w systemie CADDS5. Wybrano jedno "okno"



Rys.11 Przykład "komputerowego" projektu samolotu przedstawionego jako obiekt rzeczywisty



Rys.12 Droga narzędzia obrabiarki SN wziernika pokrywy łuku samolotu włoskiego ATR-72 wykonanego przy pomocy systemu CADDS5

Michał Bayer, Antoni Kozok, Adam Lipiński, Hanna Lubecka, Lech Pławecki

Kryteria wyboru narzędzi typu CASE

1. Wprowadzenie.

Systemy CASE są złożonymi produktami software'owymi oferującymi bardzo szerokie możliwości w zakresie wspomagania tworzenia oprogramowania. Przy wyborze konieczna jest zarówno ogólna znajomość metod i technik w nich stosowanych jak również świadomość istnienia wielu odmian implementacyjnych o różnych zaletach i wadach. W oparciu o taką wiedzę można następnie poddać systemy CASE gruntownej ocenie z punktu widzenia własnych potrzeb i uwarunkowań. Można się spodziewać, że wybór przeprowadzony w taki sposób przyniesie maksymalne korzyści użytkownikom.

2. Podstawowe zagadnienia uwzględniane przy wyborze systemu CASE.

Wysokie ceny pakietów CASE pociagają za sobą konieczność dokonania starannego wyboru spośród całego szeregu różnorodnych produktów spotykanych na rynku w celu zakupienia produktu spełniającego oczekiwania zespołu zajmującego się tworzeniem aplikacji. Jeśli chcemy aby wybór był racjonalny powinniśmy ocenić go z różnych punktów widzenia, stosując szeroką gamę kryteriów. Waga stosowanych kryteriów oceny od szeregu czynników takich jak:

1. Użytkownik systemu (inaczej będzie oceniać CASE firma software'owa świadcząca usługi dla różnych przedsiębiorstw, inaczej duże przedsiębiorstwo o sprecyzowanych potrzebach w zakresie informatyki ale rozpoczynające dopiero informatyzację a jeszcze inaczej duże przedsiębiorstwo eksploatujące szereg systemów informatycznych na sprzęcie którego nie chce na razie wymieniać).
2. Wielkość projektu oraz wielkość i kwalifikacje zespołu projektowego.
3. Geograficzne rozmieszczenie przedsiębiorstwa i zespołu tworzącego aplikację
4. Posiadane już pakiety CASE

Dla ułatwienia analizy można rozpatrywane kryteria oceny połączyć w grupy, którym nadajemy różne wagi zależnie od znaczenia jakie chcemy im przypisać.

Ogólnie jednak w trakcie oceny systemów CASE należy uwzględnić:

Informacje ogólne dotyczące dostawcy i jego pozycji na rynku (np. liczba sprzedanych kopii, liczba wdrożonych systemów, lista użytkowników CASE (z uwzględnieniem branży) i referencje zwłaszcza w odniesieniu do systemów zrealizowanych dla przedsiębiorstw o tym samym lub podobnym do naszego charakterze, plany dostawcy w zakresie rozwoju posiadanych narzędzi CASE ze szczególnym uwzględnieniem kierunku tego rozwoju (metodyka, platforma sprzętowa, architektura systemów, języki programowania, systemy operacyjne, DBMS itp.), oferowana współpraca w zakresie szkoleń i realizacji pierwszych projektów, praktyczne doświadczenie wykładowców w zakresie tworzenia systemów informatycznych, język szkoleń i cena systemu.
Informacje tego typu mogą być użyteczne w

trakcie sporządzania wstępnej listy produktów które będą brane pod uwagę w trakcie dalszej analizy.

Stopień integracji systemów CASE. Rozpatrujemy podział na pakiety narzędziowe (CASE toolkit) i pakiety zintegrowane (CASE workbench).
Powyższy podział ma zawsze duże znaczenie jako kryterium wyboru narzędzia CASE. Duże i poważne przedsięwzięcia wymagają pakietów zintegrowanych, gdyż w innym przypadku mogą wystąpić trudności z uzyskaniem projektów spójnych, kompletnych i nie zawierających błędów.

Etapy cyklu życia systemu wspierane przez CASE (planowanie, analiza, projektowanie, implementacja, utrzymanie, zarządzanie projektem).
W niektórych przypadkach może okazać się, że nie jest potrzebny system CASE obsługujący pełny cykl życia systemu, gdyż posiadamy już wcześniej zakupione narzędzia realizujące część potrzebnych zadań. W takim przypadku ocena narzędzia obejmującego brakującą część cyklu życia systemu zależy od jego zgodności z posiadanymi już narzędziami i ewentualnie od kosztu zakupu modułów łączących .

Metodyki wspierane przez CASE. Istnieją systemy CASE wspierające określoną metodykę. Takie systemy mają dość ściśle określoną kolejność postępowania w trakcie tworzenia aplikacji. Istnieją też systemy CASE oferujące zestaw różnorodnych narzędzi dość luźno ze sobą powiązanych a wybór i kolejność ich stosowania ustala się w zależności od wybranej metodyki.

Techniki wspierane przez CASE. Kryterium to ma zasadnicze znaczenie w połączeniu z metodyką, która ma być stosowana. Szczególnie starannie należy sprawdzić proponowane techniki w narzędziach uniwersalnych, które nie wymuszają stosowania konkretnej metodyki.

Możliwości tworzenia prototypów.

Środowisko projektowe w którym pracuje CASE ma szczególne znaczenie dla organizowania współpracy dużych grup informatyków (zwłaszcza grup rozproszonych geograficznie) w ramach dużych i bardzo dużych projektów. Również graficzny interface użytkownika związany ze środowiskiem może mieć zasadniczy wpływ na ocenę systemu CASE ze względu na stwarzany komfort pracy a co za tym idzie - zwiększenie wydajności i zmniejszenie ilości błędów. Budowa bazy danych projektu (repozitorium) jest również zależna od środowiska i może wpływać na organizację pracy oraz sposób zarządzania projektem.

Środowisko implementacyjne w którym będą pracować systemy opracowane przy użyciu CASE. Kryteria tej grupy mają duże znaczenie, gdy przewidujemy dużą różnorodność platform sprzętowych i softwarowych jak to ma miejsce w wypadku firmy tworzącej oprogramowanie na zamówienie. W takim przypadku narzędzia CASE pozwalające na tworzenie aplikacji pracujących na różnorodnym sprzęcie, z różnymi systemami operacyjnymi i różnymi RDBMS będą uzyskiwać znacznie wyższe oceny.

Generowanie kodu programu (szkielety programów, kompletne programy, oferowane języki programowania, programy on-line, programy wsadowe).
Wykonywalny kod jest obok dokumentacji zasadniczym celem, który chcemy osiągnąć i problemy z tym związane należą zawsze do bardzo ważnych. Jeżeli nie jest generowany kompletny kod i konieczne jest (jak to zazwyczaj bywa) uzupełnienie go metodą tradycyjną to mogą wystąpić problemy

związane z utrzymaniem programów jeżeli narzędzie CASE nie przechowuje tradycyjnie dopisanych fragmentów po wprowadzeniu do projektu poprawek i ponownym wygenerowaniu modułów. Sposób rozwiązania tych trudności ma bardzo duże znaczenie przy ocenie systemu CASE.

Reengineering i inżynieria odwrotna ma duże znaczenie nie tylko dla użytkownika który stosował poprzednio tradycyjne metody tworzenia systemów i ma w tym zakresie duży dorobek, który chce utrzymywać i modyfikować przy użyciu narzędzi CASE. Utrzymanie aplikacji które utworzono używając systemu CASE może również często wymagać technik tego typu.

Możliwości oferowane przez grafikę (kolory, mysz, windows).

Możliwości kontroli błędów (składnia, kompletność, integralność, możliwości w zakresie ustalania przyczyn błędów i ich związków z wcześniej zaprojektowanymi obiektami, kontrola i zapewnienie jakości) są ważnym czynnikiem zapewniającym wykrycie popełnionych błędów na wczesnych etapach tworzenia systemu co wpływa zasadniczo na koszty tworzenia systemu.

Encyklopedia systemu (Repozytorium). Należy zawsze zwrócić uwagę na DBMS używany przez encyklopedię systemu i jej architekturę, raportowanie, kontrola zmian, próbne rewizje (np. bilansowanie przepływów danych) i gospodarowanie wersjami. Bardzo użyteczna może się okazać możliwość definiowania przez użytkownika własnych raportów oraz bezpośredni dostęp do repozytorium wykorzystujący DBMS (np. język SQL, QUERY BY EXAMPLE).

Typ systemu docelowego (on-line, wsadowy, przetwarzania transakcji, czasu rzeczywistego, wbudowany). Możliwości systemu CASE w tym zakresie są trudne do oceny ze względu na zmieniające się wymagania stawiane systemom informatycznym. Zmiany wymagań są wynikiem postępu technicznego w dziedzinie sprzętu i oprogramowania, który to postęp jest w zasadzie dostrzegany przez firmy produkujące systemy CASE wytwarzające dość szybko nowe generacje swoich produktów. Z tego względu oprócz potrzeb bieżących i dotyczących możliwej do przewidzenia przyszłości należy tutaj brać pod uwagę ogólną ocenę producenta systemu CASE i tempo jego nadążania za postępem.

Zarządzanie projektem. Wiele systemów CASE jest wyposażonych w narzędzia do planowania prac i zapotrzebowania na zasoby, śledzenia postępu prac, analizy ryzyka itp. Zintegrowanie tych narzędzi z systemem CASE może znacznie zmniejszyć nakład pracy na planowanie i zarządzanie.

3. Ocena zastosowanych metod i technik.

W systemach CASE znalazło zastosowanie wiele różnorodnych technik przeznaczonych do wykorzystania zarówno na etapie analizy jak i projektowania. Wiele z nich to techniki bardzo popularne, występujące bardzo często w różnych produktach, inne znów to unikalne techniki stosowane przez konkretnego twórcę systemu CASE. Poniżej omówione zostaną techniki najczęściej spotykane. Może się również zdarzyć tak, że ta sama technika zastosowana w różnych produktach będzie miała za każdym razem trochę inną formę.

Przyglądając się technikom i metodom zastosowanym w systemach CASE należy skonfrontować je z wymaganiami stawianymi przyszłemu systemowi. W ramach metodyk istnieje wiele **metod analizy i projektowania** systemów informatycznych. Mogą to być np:

- **metoda analizy i projektowania strukturalnego,**
- **metoda analizy i projektowania**

zorientowanego obiektowo,
- **metoda Jacksona,**
- **metoda Warniera-Orra,** itp.

Wybór odpowiedniej metody zależeć będzie od rodzaju projektów które będą realizowane za jej pomocą; od ich wielkości, stopnia złożoności danych, ilości realizowanych funkcji itd. Metody te oparte są o dwa podejścia do analizy systemu: podejście od strony procesów i podejście od strony danych. Na etapie analizy często stosowanymi **technikami** są:

- **diagram przepływu danych** (Data Flow Diagram - podstawowy diagram stosowany w metodach analizy strukturalnej),
- **diagram hierarchii funkcji** (Function Hierarchy Diagram - stosowany w metodach wykorzystujących podejście od strony danych)
- **diagram encje-związki** (Entity-Relationship Diagram).

Dodatkowymi technikami mogą być **diagramy przedstawiające historię życia encji** (Entity Life History) , **diagramy zmian stanów** (State Transition Diagram) i **macierze IURD** (Insert, Update, Read, Delete) opisujące sposób wykorzystania danych przez procesy, funkcje lub kategorie użytkowników analizowanego systemu. Analizując te techniki należy zwrócić uwagę na to, czy system CASE zapewnia kontrolę spójności danych pomiędzy nimi. Szczególną wagę ma tu kontrola spójności pomiędzy diagramem przepływu danych lub diagramem hierarchii funkcji a diagramem encje-związki. Dla diagramu przepływu danych jest ona przeważnie realizowana w oparciu o parę magazyn danych - encja. Jednak zbyt silna kontrola integralności danych pomiędzy tymi elementami może niekiedy spowodować trudności w zarządzaniu zgromadzoną informacją. Innym ważnym elementem jest sprawdzenie dokładności bilansowania diagramu przepływu danych, zarówno pionowego (poprzez wszystkie poziomy diagramu), jak i poziomego (w ramach jednego poziomu). Dotyczy ono zazwyczaj magazynów danych i przepływów. Chcąc zastosować analizę strukturalną opartą o zdarzenia (metoda Yourdona) warto zwrócić uwagę na to, czy CASE ma możliwość łączenia diagramów przepływów danych i encje-związki. Diagram przepływu danych może umożliwiać również przedstawienie procesów i przepływów sterujących. Jest to ważne dla projektowania systemów czasu rzeczywistego. Istotną sprawą jest wtedy kontrola spójności pomiędzy nim a diagramem zmian stanów. Matryce IURD powinny określać sposób wykorzystania danych raczej na poziomie atrybutu encji niż tylko na poziomie encji. Diagramy encje-związki mogą być realizowane w oparciu o różne konwencje notacyjne (np. Chena, Bachmana). Mogą również umożliwiać tworzenie pod-encji i super-encji. Analiza realizowana w oparciu o metodę obiektowo zorientowaną powinna być wspierana poprzez diagramy obiekty-związki, diagramy akcji, diagramy zmiany stanów, itp. Analizując możliwości tych diagramów należy zwrócić uwagę na to, aby umożliwiały one przedstawienie hierarchii obiektów, ich komponentów i związków pomiędzy obiektami, akcji dotyczących poszczególnych obiektów i cyklu życia obiektów.

Istotnym elementem analizy jest również **prototypowanie**. Umożliwia ono pokazanie przyszłemu użytkownikowi jak będzie wyglądał docelowy system. Wagę jaką należy przywiązywać do możliwości prototypowania trudno jest krótko określić. Jest elementem cennym zarówno dla użytkownika nie zaznajomionego z techniką komputerową gdyż może on wtedy oswoić się z nową technologią i skonfrontować swoje dotychczasowe przyzwyczajenia z tym, co w przyszłości zostanie mu zaoferowane, jak i dla użytkownika doświadczonego w posługiwaniu

się narzędziami informatycznymi. Może on wtedy dostatecznie wcześnie wnieść cenne uwagi na temat funkcjonowania przyszłego systemu. Jednak zbyt duże możliwości w tym zakresie mogą nieraz (zwłaszcza gdy pracujemy z użytkownikiem mało obytym z informatyką) nastręczyć wiele kłopotów. Może on po oglądnięciu prototypu dojść do wniosku, że do uruchomienia systemu pozostało już niewiele do zrobienia i wywierać nacisk na szybkie ukończenie aplikacji. Możemy jednak zdecydować się na tworzenie systemu metodą inkrementacyjną, gdzie w kolejnych iteracjach tworzymy wyodrębnione jednostki implementacyjne systemu w oparciu o uzgodniony z użytkownikiem prototyp. Wymaga to posiadania zupełnie innego rodzaju narzędzia CASE, w którym prototyp przekształcany automatycznie lub półautomatycznie w aplikację odgrywa zasadniczą rolę. W systemach CASE stosowane są dwa rodzaje prototypowania: dynamiczne i statyczne. Prototyp statyczny pozwala jedynie na zaprojektowanie wyglądu przyszłego systemu i określenie sposobu nawigacji pomiędzy poszczególnymi ekranami. Prototyp dynamiczny pozwala ponadto na konwersację i symulację niektórych funkcji przyszłego systemu. Ważną sprawą jest również możliwość tworzenia prototypów raportów.

W fazie projektowania system CASE powinien zapewniać możliwie najdokładniejszą kontrolę integralności danych z fazą analizy. Jest to bardzo ważne przy projektowaniu modułów systemu. Jedną z najczęściej stosowanych do tego technik w systemach CASE jest **diagram struktury** (Structure Chart). Występuje on jako element metody analizy i projektowania strukturalnego. Analizując możliwości tego diagramu należy zwrócić uwagę na to, czy umożliwia on zdefiniowanie styków pomiędzy modułami diagramu, w postaci danych wejściowych i wyjściowych oraz sterowania. Rolę diagramu struktury może pełnić w niektórych systemach CASE uszczegółowiony diagram hierarchii funkcji. Innymi ważnymi elementami są wszelkie techniki wspomagające przekształcenie logicznego modelu danych w fizyczny model danych. Bardzo pomocne są tu wszelkiego rodzaju narzędzia wspomagające rozkład związków N do N, migrację kluczy lub wspomagające proces normalizacji modelu danych. Projektowanie związane jest także z definiowaniem algorytmów modułów projektowanego systemu. Stosowane są tu zarówno techniki graficzne jak i języki formalne (pseudokody). Jako techniki graficzne można spotkać schematy blokowe (Flow Chart), diagramy Nassi-Schneidermana, diagramy Ferstla itp. Z technik tych najlepsze rezultaty dają diagramy Nassi-Schneidermana wymuszające programowanie strukturalne i podział na małe, łatwe w testowaniu fragmenty.

4. Ocena środowiska projektowego

Repozytorium jest centralną częścią systemów CASE, obejmującą wszystkie aspekty procesu tworzenia systemów i zarządzającą definicjami będącymi ich odzwierciedleniem. W związku z tym konieczna jest wszechstronna analiza jego możliwości. Diagramy występujące w systemach CASE powinny być w pełni zintegrowane z repozytorium; zmiany na diagramach powinny powodować natychmiastową aktualizację repozytorium. Głównymi zadaniami tego narzędzia powinny być:

- zapewnienie spójności nowo wprowadzanych danych z tymi, które uprzednio wprowadzono, jak również umożliwienie sprawdzenia ich kompletności i poprawności,
- kontrola wprowadzanych zmian,
- tworzenie raportów i aktualnej dokumentacji systemowej.

Jest szereg aspektów, które należy wziąć pod uwagę analizując możliwości repozytorium:

Wielodostęp (multiuser environment). Cecha ta jest bardzo istotna przy pracy zespołowej - pozwala ona członkom tego samego zespołu projektowego na równoczesną pracę nad tym samym projektem. Aktualizacja np. diagramu jest możliwa dla jednego użytkownika w danym czasie, pozostali mogą go przeglądać i wykorzystywać dla uzyskania dostępu do innych obiektów oraz modyfikować inne elementy zawarte w repozytorium. Nowo tworzone obiekty są bezpośrednio dostępne dla innych użytkowników. Cecha ta pozwala także na współdzielenie danych z innymi zespołami projektowymi, przy czym ich aktualizację może wykonać jedynie właściciel. Ważne jest to szczególnie w nowoczesnych organizacjach gdzie nowe systemy tworzone są często równocześnie przez różne zespoły projektowe i cecha ta zapewnia spójność tworzonego oprogramowania.
Innym rozwiązaniem jest centralne repozytorium na serwerze służące jedynie do przechowywania wszystkich definicji tworzonego systemu (mechanizm "check in/check out"). Praca nastręcza wtedy pewne trudności: w sytuacji, gdy zespół projektowy ma do opracowania pewną część tworzonego systemu, wymagane definicje są kopiowane z centralnego repozytorium i powielane na stacji roboczej. Zespół pracuje w kompletnej izolacji od innych grup bez współdzielenia danych. Nie budzi zdziwienia fakt, że tworzone specyfikacje systemowe są niezgodne ze specyfikacjami innych zespołów projektowych. Ponieważ trudna jest identyfikacja tych niespójności na bieżąco, taka czynność jest wykonywana raz, pod koniec tworzenia projektu. Wtedy rezultaty końcowe przekazywane są do centralnego repozytorium i w czasie tego procesu następuje automatyczna identyfikacja niespójności.

Zarządzanie wersjami.

System CASE powinien mieć tą własność. Jest ona korzystna z różnych punktów widzenia tworzenia cyklu życia systemu, różnorodności grup użytkowników .
Możliwość zapamiętywania tylko aktualnej wersji projektu, uniemożliwia przechowywanie zatwierdzonych przez użytkowników wyników każdej z faz jego tworzenia (tj. analizy, projektowania, implementacji). Nie ma możliwości "zamrażania" wersji przed przystąpieniem do wprowadzania zmian. Nie ma też możliwości równoczesnego tworzenia kilku odmian tego samego projektu, ze względu na specyficzne wymagania różnych grup użytkowników końcowych.

Kontrola dostępu.

Jest to ważna cecha systemów CASE, zabezpieczająca projekt przed przypadkową aktualizacją. W systemach CASE wspierających ją definiowane są prawa dostępu, w wyniku których różni użytkownicy systemu CASE mają ściśle określony typ dostępu do definicji i diagramów projektu.

Łączenie model danych.

Jest to szczególnie ważne przy równoczesnym tworzeniu przez kilka zespołów projektowych dużych projektów.

Przy wyborze systemu CASE potencjalny nabywca może spotkać się z określeniami ich typów, a mianowicie: toolkit i workbench .
Systemy **Case toolkit** obejmują jedną fazę tworzenia oprogramowania lub jeden typ zadania projektowego, np.: analizę, projektowanie bazy danych,implementację. Często zdarza się, że użytkownicy korzystają z kilku różnych tego typu systemów a także używają innych narzędzi jak 4GL, słowniki danych, DBMS. Przy takim wyborze istotne

jest jak ścisła może być ich współpraca.
Systemy **CASE workbench** są zbiorami zintegrowanych narzędzi tworzenia oprogramowania, które dostarczają zautomatyzowanej pomocy w procesie analizy systemu, projektowania i implementacji. Dysponują wspólnym repozytorium zawierającym wszystkie informacje potrzebne do budowania i wspierania systemu. Oferują automatyczne wsparcie poprzez cały proces tworzenia oprogramowania w celu dostarczenia udokumentowanego, wykonywalnego systemu.

Ważna przy ocenie środowiska projektowego jest analiza **interface'u graficznego użytkownika** (GUI). Używany interface wpływa bardzo mocno na komfort pracy, jej wydajność i jakość. Korzystne jest np. wykorzystanie standardu Windows (najpopularniejszego dla PC graficznego interface'u użytkownika), przynoszącego wiele udogodnień (np. w sytuacji gdy potrzebny jest dostęp do kilku funkcji systemu równocześnie). Użycie myszy przyśpiesza poruszanie się po wielopoziomowych menu, koniecznych w tak złożonym narzędziu. Bardzo dużym udogodnieniem jest protokół Dynamic Data Exchange (DDE), umożliwiający wymianę tekstu lub grafiki między systemem Case a innymi produktami, takimi jak arkusze kalkulacyjne czy edytory tekstowe i graficzne. Mechanizm ten może okazać się bardzo przydatny w tworzeniu dokumentacji projektowej.
Przy ocenie środowiska operacyjnego stacji roboczej należy wziąć pod uwagę m.in. to czy CASE bazuje na PC, czy wymaga stacji o dużej mocy obliczeniowej, monitorów o dużej rozdzielczości ekranu itp., a co za tym idzie, poważnych wydatków. Jeżeli w systemie CASE została zastosowana architektura klient-serwer (jedna z najbardziej nowoczesnych aktualnie technologii) jest to jego zaletą, ale nie powinno to w sposób decydujący wpłynąć na jego wybór. Należy także przejrzeć listę dostępnych sterowników drukarek, ploterów itp. aby upewnić się, że istniejący w przedsiębiorstwie sprzęt będzie wykorzystany. W zakresie oprogramowania ważne są systemy operacyjne i bazy danych.

5. Ocena pod kątem docelowego środowiska implementacyjnego.

Stojąc przed wyborem narzędzia CASE musimy brać pod uwagę różne aspekty docelowego środowiska implementacyjnego:

- **sprzęt i system operacyjny**
  system oparty o duży komputer (mainframe)
  system oparty o minikomputery z systemem operacyjnym typu Unix
  system oparty o środowisko PC/LAN
  system wbudowany (embedded)

Wprawdzie istnieją narzędzia CASE ograniczone do jednej docelowej platformy sprzętowej, ale obecnie większość z nich jest bardziej uniwersalna i zasadniczym aspektem docelowego środowiska jest system zarządzania bazą danych.

- **docelowy system zarządzania bazą danych**
  RDBMS (Sybase ,Oracle, Ingres, Informix, Gupta, SQLBase, DB2,...)
  inne typy baz danych

Jednym z najważniejszych kryteriów wyboru systemu CASE będzie konkretny system zarządzania bazą danych, na jakim ma być oparta tworzona aplikacja. Istnieją jednak dodatkowe kryteria. Absolutna większość narzędzi CASE generuje aplikacje oparte o relacyjne bazach danych, ale występują między nimi duże różnice co do zakresu generowanego kodu. Zawsze będziemy mieli do czynienia z generowaniem definicji tablic bazy danych, ale ważne jest aby były generowane także reguły integralności na

poziomie definicji tablic. Większość nowoczesnych relacyjnych baz danych posiada obecnie takie mechanizmy jak wyzwalacze (triggers) i składowane procedury (stored procedures); warto więc zwrócić uwagę, czy są one generowane przez CASE (niektóre narzędzia mogą generować tylko części deklaratywne wyzwalaczy i składowanych procedur). Istotną sprawą jest, czy CASE potrafi generować wyłącznie proste funkcje aplikacji, oparte na samym modelu danych (zapytania do bazy danych, dodawanie, usuwanie i modyfikacja danych), czy też potrafi generować złożone funkcje w oparciu o ich specyfikacje w różnych technikach (diagramy struktury, pseudokod i inne).

- **typ docelowego systemu (architektura logiczna)**
  - on-line
  - wsadowy
  - przetwarzanie transakcyjne
  - klient/serwer
  - real-time
  - wbudowany (embedded)

Od typu architektury logicznej tworzonej aplikacji zależą w znacznym stopniu wymagania stawiane przed narzędziem CASE w zakresie wspomaganych technik i metodyki.

- **typ współpracy z użytkownikiem projektowanych systemów (End User Interface)**
  - nieinteligentne terminale (tryb znakowy)
  - graficzne interface'y z użytkownikiem (GUI)
  - multi-media (obraz, dźwięk)

Istotną sprawą są możliwości narzędzia CASE w zakresie współpracy z użytkownikiem (formatki ekranowe, raporty, struktura menu, help). Mogą też opierać się na różnych standardach GUI (Microsoft Windows, OSF/Motif, CUA i inne).

- **docelowe narzędzia programowania**
  - generacja kodu w językach 3GL
  - współpraca z narzędziami 4GL

Często możemy stać przed wyborem różnych narzędzi programowania, np. dostarczanych przez twórców systemu zarządzania bazą danych oraz przez firmy niezależne.

- **uniwersalność kontra optymalność narzędzia**
  - CASE ukierunkowany na jedno konkretne środowisko implementacyjne
  - CASE bardziej uniwersalny

Ponieważ systemy zarządzania bazą danych i związane z nimi narzędzia programowania bardzo szybko rozwijają się, narzędzia CASE są zawsze o krok do tyłu i przy generowaniu kodu nie są w stanie uwzględnić wszystkich możliwości oferowanych przez bazy danych. Wydaje się, że narzędzia ukierunkowane na konkretny system zarządzania bazą danych są pod tym względem bliższe ideału. Z drugiej strony jednak, uniwersalność narzędzia też ma swoje zalety (chociażby przenoszalność aplikacji między różnymi bazami danych).

6. Wnioski i uwagi końcowe

Wybór dobrego i właściwego systemu CASE wymaga sprecyzowania zadań, które będą przy jego użyciu rozwiązywane, co pociąga za sobą wybór odpowiedniej metodyki oraz środowiska implementacyjnego i jego architektury. To z kolei determinuje techniki, które powinien realizować wybrany system CASE w celu właściwego wsparcia wybranej metodyki. Istotne jest też, aby wybrany system CASE umożliwiał tworzenie aplikacji o architekturze dobrze wykorzystującej środowisko implementacyjne. Są to kryteria których spełnienie należy uznać za konieczne. Wskazane też jest, aby wybrany system CASE umożliwiał jak najlepszą

organizację pracy zespołu tworzącego aplikację i pozwalał na osiągnięcie wysokiej wydajności pracy przy małej ilości błędów. Można to osiągnąć przez wybór właściwego środowiska projektowego, odpowiednie repozytorium, dobry interface graficzny, system wczesnego wykrywania błędów umożliwiający prześledzenie przyczyn ich powstania.

W świecie szybko rozwijającej się techniki ważne jest też, aby wybrać producenta, który nie pozwala się wyprzedzić szybko nadchodzącym zmianom. Duże znaczenie ma również dobra współpraca z producentem zakupionego systemu CASE w zakresie szkoleń i konsultacji.

Literatura:

1. "The CASE Philosophy", Michael Lucas Gibson, BYTE 4/89
2. " Methodology:The Experts Speak", Ken Orr, Chris Gane, Edward Yourdon, Peter P.Chen, Larry L. Constantine, BYTE 4/89
3. "The CASE Experience ", Carma McClure, BYTE 4/89
4. "Principles of Object - Oriented Analysis and Design", James Martin, Prentice- Hall, 1993
5. "Modern Structured Analysis", Edward Yourdon, Prentice-Hall, 1989

Jacek Irlik
KATOWICE

Wdrożenia systemów informatycznych Uwagi konsultanta

Przedmiotem wykładu są uwagi, będące wynikiem kilkuletniej praktyki opracowywania ekspertyz dotyczących przebiegu i sposobu wykonania zawartych umów o usługi tzw. "kompleksowej komputeryzacji", a także doświadczeń opiniowania wyników prac wykonywanych w ramach takich umów.

Omówione zostaną najczęściej spotykane w toku wykonywania umowy problemy i związane z nimi spory. Spory te pojawiały się w toku odbioru poszczególnych etapów pracy lub w toku odbioru końcowego. We wszystkich prawie przypadkach można było zakwalifikowć je jako skutek braku staranności w zawieraniu umowy, zwłaszcza przy specyfikowaniu świadczeń wykonawcy. Niestarannie zawarta umowa nie mogła być bowiem instrumentem, pozwalającym jednoznacznie rozstrzygnąć - w interesie obu stron - czy wykonawca wykonał prace zgodnie ze swoim zobowiązaniem.

Można w tym zakresie wyróżnić następujące sytuacje typowe.

A. Sytuacja krańcowa:

Zawierana umowa ogranicza się do następujących postanowień (w pewnym uproszczeniu redakcyjnym):

- wykonawca zapewni dostawę sprzętu komputerowego (specyfikacja w załączniku), sprzeda moduły oprogramowania (specyfikacja w załączniku),
- wykonawca wykona oraz wdroży oprogramowanie dla potrzeb zamawiającego w zakresie A, B, C, ... ,
- prace fakturowane będą w terminach (harmonogram w załączniku),
- całość prac zostanie wykonana w terminie 99.99.99.

Zawierana w ten sposób umowa czyni wykonawcę całkowicie odpowiedzialnym za uzyskanie "w terminie 99.99.99" wyniku satysfakcjonującego zamawiającego. Wynik taki jednak często nie pojawia się i stąd problemy.

B. Sytuacja typowa:

Postanowienia umowy nie są bardziej precyzyjne niż w sytuacji opisanej powyżej, nazwanej krańcową. Dla uzyskania większego komfortu psychicznego jako pierwszy w umowie wyróżnia się etap, który obejmuje wykonanie projektu (realizacyjnego, technicznego, ... ?). Pozytywny odbiór tego projektu zostaje wówczas zastrzeżony jako warunek wykonania oraz podstawa dalszych prac. Jeżeli jednak w umowie nie zastrzeżono szczegółowych wymagań funkcjonalnych w stosunku do projektowanego systemu to zamawiającemu trudno jest uzasadnić odmowę odbioru projektu, nawet jeżeli zawiera on rozwiązania nie odpowiadające zamawiającemu, ale jest technicznie poprawny i wykonany zgodnie z regułami inżynierii programowania.

C. Bardzo często:

Jak wynika z doświadczeń autora niniejszych uwag, spory pomiędzy stronami powstają często w toku odbioru wykonanego

oprogramowania. Spory te dotyczą najczęściej dwóch kwestii. Jedna z nich to ustalenie co stanowi przedmiot odbioru i ma być wydane zamawiającemu. W związku z brakiem powszechnie obowiązujących norm w tym zakresie proponuje się regulować te sprawy w umowie. Źródłem sporów jest również brak umownego określenia trybu i kryteriów dokonywania odbioru. Wykonawca zwykle zastrzega, że niedokonanie odbioru przez zamawiającego w określonym terminie upoważnia wykonawcę do dokonania przez wykonawcę odbioru jednostronnego, będącego podstawą fakturowania. W umowie warto więc szczegółowo określić wymagania, które wykonane oprogramowanie ma spełniać, oraz procedurę rozstrzygnięcia, czy są spełnione.

D. Prawie zawsze:

Umowa z reguły przewiduje (zwykle jako ostatni) etap "wdrożenia" systemu, którego treść jest najczęściej całkowicie niezdefiniowana, nawet założywszy potoczne rozumienie terminu wdrożenie. Pomija się całkowicie fakt, że z punktu widzenia użytkownika to właśnie wdrożenie jest tym etapem, którego wyniki są podstawą efektywnej eksploatacji systemu. Współpraca stron w trakcie tego etapu powinna więc być zdefiniowana szczególnie precyzyjnie poprzez szczegółowe określenie wzajemnych zobowiązań.

***

Umowy zawierane w dziedzinie informatyki były przedmiotem opracowań książkowych (np. [1], [2]). Niniejsze uwagi nie pretendują do roli kolejnego opracowania prawniczego na ten tamat. Wynikają one z obserwacji konkretnych sytuacji spornych i zawierają propozycje takiego sposobu specyfikowania świadczeń w umowie oraz stopnia dokładności, który mógłby pozwolić na uniknięcie późniejszych kłopotów.

1. Znaczenie specyfikacji wymagań

Brak właściwego kryterium oceny zgodności wszelkich usług, świadczonych przez wykonawcę tzw. "kompleksowej komputeryzacji", z potrzebami zamawiającego jest podstawową przyczyną późniejszych sporów i nieporozumień pomiędzy zamawiającym i wykonawcą.

Umowa zawarta zostaje najczęściej wyłącznie na podstawie oferty, w której główną część stanowi specyfikacja oferowanego sprzętu komputerowego i standardowych modułów oprogramowania. W zawartych w taki sposób umowach szczegółowe postanowienia dotyczą zwykle sprzedaży sprzętu komputerowego oraz standardowych modułów oprogramowania, zestawionych najczęściej zgodnie z ofertą, oraz adaptacji modułów standardowych lub opracowania programów na zamówienie. Umowy takie zawarte zostają zwykle w oderwaniu od profesjonalnie przeprowadzonej analizy potrzeb.

W związku z tym jako pierwszy etap prac przewiduje się wykonanie projektu technicznego, którego odbiór jest wówczas warunkiem dalszych prac. Nie ma jednak podstaw aby odmówić odbioru projektu, który zakłada rozwiązania nie odpowiadające zamawiającemu, jeżeli jest on technicznie poprawny, wykonany zgodnie z regułami inżynierii programowania. Należy więc w konsekwencji uznać, że przedstawiony do odbioru projekt techniczny wykonany został zgodnie z umową pomimo, że nie odpowiada potrzebom zamawiającego. Zamawiający jest wówczas zmuszony zażądać dokonania zmian projektu, a wykonawca nie jest skłonny wykonać tego za darmo. Stosowane w praktyce zawierania umów rozwiązanie, polegające na załączeniu do umowy "specyfikacji funkcjonalnej" lub "opisu technicznego" nie jest najczęściej skuteczne,

specyfikacja taka czy opis są bowiem na ogół zbyt ogólne.

Aby uniknąć takich problemów należy - przed zawarciem umowy z wykonawcą - przeprowadzić analizę potrzeb i dokładnie określić wynikające z nich wymagania w stosunku do systemu, który ma być wynikiem "kompleksowej komputeryzacji" (zarówno co do konfiguracji sprzętu jak i co do oprogramowania). Wymagania takie należy zestawić w odrębnym dokumencie "specyfikacji wymagań", który powinien stanowić załącznik do umowy o wykonanie kompleksu usług "komputeryzacji".

Zgodnie ze standardem IEEE dotyczącym cyklu życia oprogramowania [3] dokument specyfikacji wymagań jest jednym z podstawowych, obligatoryjnie tworzonych w tym cyklu dokumentów.
Zawartość i strukturę takiego dokumentu określa na przykład standard IEEE "Specyfikacja wymagań dla oprogramowania" [4] lub zalecenia EWICS dotyczące "Specyfikacji wymagań dla systemu" [5].

Brak oczywiście podstaw prawnych aby tworzenie dokumentu specyfikacji wymagań traktować jako nakaz. Brak też podstaw aby, tworząc taki dokument z własnej woli, przestrzegać zaleceń wspomnianych standardów. Należy jednak zapewnić, aby w dokumencie specyfikacji wymagań znalazły się wszystkie te wymagania, które zamawiający chce zastrzec jako kryterium odbioru późniejszych prac. Standardy te można natomiast traktować jako pożyteczną wskazówkę co do zawartości i struktury dokumentu specyfikacji wymagań.

Nie analizując szczegółowo treści wspomnianych standardów warto zaznaczyć, że zgodnie z ich ogólnymi zaleceniami specyfikacja wymagań powinna odnosić się do następujących kwestii.

1.1. Funkcje systemu

Specyfikacja wymagań powinna określać w jednoznaczny sposób funkcje, których wykonywanie oprogramowanie ma zapewnić. Poszczególne funkcje powinny zostać określone w taki sposób, jaki zamawiający uważa za dostateczny w celu późniejszego rozstrzygnięcia czy jego wymaganie w tym zakresie zostało spełnione.

1.2. Współdziałanie z otoczeniem

Wymagania w tym zakresie powinny sił odnosić do:

- kategorii stanowisk pracy, które wspomagane być mają przez przyszły system, oraz zakresu i sposobu obsługi każdego ze stanowisk,
- współdziałania przyszłego systemu z innymi programami, eksploatowanymi zarówno przez zamawiającego jak i innych użytkowników,
- wykorzystania, w ramach przyszłego systemu, modułów już eksploatowanego sprzętu komputerowego i oprogramowania jako środowiska tego systemu,
- pożądanej lokalizacji modułów.

1.3. Sposób działania

Wymagania tego rodzaju powinny odnosić się do takich parametrów przyszłego systemu jak na przykład: czas wykonania poszczególnych funkcji, liczby użytkowników mogących równocześnie korzystać z danej funkcji, czas reakcji systemu na zgłoszenie użytkownika itp..

1.4. Ograniczenia rzutujące na projekt i implementację

Wymagania tego rodzaju mogą dotyczyć na przykład pożądanego zakresu standaryzacji makiet ekranu i wydawnictw, rodzaju i objętości baz danych, rozproszenia systemu itp..

1.5. Inne cechy

Inne wymagane cechy, do których należy się odnieść mogą dotyczyć na przykład zakresu i stopnia ochrony i bezpieczeństwa danych, niezawodności ciągłości działania, możliwości szybkiego przeniesienia, łatwości pieęlgnacji i rozwoju itp..

2. Wykonanie projektu technicznego

Umowy o wykonanie usług "kompleksowej komputeryzacji" przewidują najczęściej wykonanie projektu technicznego jako wydzielony etap prac i warunkują dalszą realizację prac wynikami odbioru tego etapu. W przypadkach, w których nie dokonano szczegółowej analizy potrzeb i w których nie specyfikowano wymagań dostatecznie dokładnie, projekt techniczny ma pełnić - poza swoją zwykłą rolą - rolę protezy specyfikacji wymagań.

Opracowany projekt techniczny ma więc w takich przypadkach podwójne znaczenie:

- dla wykonawcy: znaczenie dokumentu technicznego, w oparciu o który tworzone będą programy,
- dla zamawiającego: znaczenie opisu, na podstawie którego akceptuje on ostatecznie kształt zamawianego oprogramowania.

Konsekwencją wspomnianego wyżej znaczenia projektu technicznego dla zamawiającego jest, aby projekt ten prezentował w czytelny dla zamawiającego i jednoznaczny sposób:

a) kategorie użytkowników (stanowisk pracy w systemie) przewidywanych przez projektowane oprogramowanie,
b) dostęp do danych oraz funkcji określony dla użytkownika danej kategorii,
c) makiety ekranów, wyświetlanych użytkownikowi w celu wprowadzenia danych lub przedstawienia wyników - dla każdej z kategorii stanowisk,
d) wzory wydruków lub innych wydawnictw graficznych emitowanych przez system.

Projekt taki powinien wówczas ponadto szczegółowo prezentować zamawiającemu rozwiązania, projektowane w celu spełnienia innych niż funkcjonalne wymagań w stosunku do oprogramowania (ochrona danych, liczba użytkowników, objętość zbiorów danych, czasy reakcji, współpraca z innymi programami, łatwość modyfikowania i rozwoju itp.).

Uwzględniając powyższe, zamawiający powinien w umowie zastrzec te wymagania w stosunku do dokumentacji projektu technicznego, które uznaje on za istotne dla dokonania oceny przyszłego oprogramowania.

Zgodnie z tym co powiedziano wyżej, podstawowym jednak kryterium odbioru wykonanego projektu powinna być zgodność z ustalonymi wcześniej wymaganiami. Brak takich wymagań może - jak wspomniano wcześniej - rodzić spory przy dokonywaniu odbioru projektu.

3. Wykonanie oprogramowania

Zgodnie z tym co powiedziano dotychczas oprogramowanie wykonane zgodnie z umową powinno:

- zostać wykonane zgodnie z projektem technicznym (po to został on opracowany), a ponadto uwzględniać ewentualne zalecenia realizacyjne zgłoszone podczas jego odbioru,
- działać zgodnie z wymaganiami określonymi w specyfikacji wymagań (wynika z przeprowadzonej analizy potrzeb i była podstawą zawarcia umowy).

Jak wynika z doświadczeń autora niniejszych

uwag, w toku odbioru wykonanego oprogramowania często rodzą sił spory pomiędzy stronami. Wiele sporów jest konsekwencją braku specyfikacji wymagań lub niestarannego odbioru projektu technicznego. Inne jednak spory dotyczą bezpośrednio etapu wykonania oprogramowania. Są to spory dotyczące kwestii materiałów stanowiących przedmiot odbioru przez zamawiającego oraz kryteriów dokonania odbioru wykonanego oprogramowania.

Strony powinny więc zastrzec w umowie co będzie przedmiotem odbioru; na przykład, że przedmiotem odbioru oprogramowania będzie:

a) oprogramowanie zainstalowane na sprzęcie Zamawiającego,
b) oprogramowanie zakodowane na nośniku zewnętrznym, w wersji umożliwiającej instalację,
c) instrukcja instalacji systemu,
d) dokumentacja użytkowa.

W kwestii kryterium odbioru strony mogą zastrzec ponadto na przykład, że wykonawca przygotuje i zaproponuje zamawiającemu przed przystąpieniem do odbioru sposób zademonstrowania, że wymagania w stosunku do oprogramowania zostały w implementacji spełnione (program testowania).

4. Wdrożenie oprogramowania aplikacyjnego

Pojęcie "wdrożenia" systemu jest często rozumiane w rozmaity sposób przez oferentów usług informatycznych.

Oferenci rozumieją najczęściej wdrożenie jako:

a) zainstalowanie poszczególnych modułów systemu w sposób umożliwiający rozpoczęcia wdrażania systemu,
b) przeszkolenie pracowników zamawiającego w zakresie korzystania z systemu,
c) pomoc przy wprowadzeniu do systemu danych rzeczywistych, umożliwiających rozpoczęcie bieżącej eksploatacji systemu.

Doświadczenia autora wskazują jednak, że w ramach umowy należy szczegółowo określić zadania składające się na przedsiłwzięcie wdrożenia. Powinno to zostać dokonane w formie harmonogramu wdrożenia, w którym wymienione zostają konkretne zadania oraz terminy ich wykonania. Harmonogram taki powinien zostać opracowany przez wykonawcę i przedłożony zamawiającemu nie później niż w czasie odbioru wykonanego oprogramowania. Przygotowanie harmonogramu ma znaczenie dla obu stron umowy. Pozwala na ocenę wykonalności przedsięwzięcia wdrożeniowego, ustala świadczenia wykonawcy i zakres współdziałania zamawiającego, a także pomaga wykonawcy przygotować się organizacyjnie do wdrożenia i rozpoczęcia normalnej eksploatacji.

Wdrożenie prostego systemu (parę programów, nieduża baza danych, jeden użytkownik) w praktyce nie powinno stanowić źródła sporów. Sprowadza się ono najczęściej do zainstalowania kilku programów i przeszkolenia użytkownika w zakresie ich obsługi. Przygotowywanie szczegółowego harmonogramu zadań (świadczeń wykonawcy) nie ma w takich przypadkach większego znaczenia. Szczególna staranność w zaplanowaniu takich zadań wymagana będzie jednak w sytuacji, w której wdrożenie dotyczyć ma oprogramowania o złożonej strukturze, zarządzającego rozproszoną bazą danych, przewidzianego do eksploatacji przez wielu użytkowników, przeznaczonego do wspomagania pracy instytucji o złożonej strukturze organizacyjnej.

Harmonogram wdrożenia powinien wyróżniać zadania związane z instalacją modułów (dotyczyć to powinno zarówno modułów sprzętowych jak i programowych), szkoleniem pracowników zamawiającego oraz

przygotowaniem bazy danych do rozpoczęcia eksploatacji.

4.1. Opis zadań związanych z instalacją

Dla każdego z zadań proponuje się określić:

a) przedmiot instalacji,
b) strona odpowiedzialna za przeprowadzenie instalacji,
c) warunki rozpoczęcia instalacji,
d) termin zapewnienia warunków instalacji,
e) strona odpowiedzialna za zapewnienie warunków instalacji,
f) termin,
g) warunki i procedura odbioru instalacji.

Zadania związane z instalacją dotyczą w tym przypadku modułów wykonywanego oprogramowania aplikacyjnego, a poza nimi tych modułów sprzętu lub oprogramowania systemowego, które wymagane są dla zainstalowania wykonanych programów aplikacyjnych i nie były instalowane wcześniej.

4.2. Opis zadań związanych ze szkoleniem

Dla każdego z zadań proponuje się określić:

a) przedmiot szkolenia,
b) miejsce szkolenia,
c) liczba godzin szkolenia,
d) terminy rozpoczęcia/zakończenia,
e) prowadzący szkolenie,
f) wykaz materiałów szkoleniowych,
g) wykaz słuchaczy,
h) sposób stwierdzenia efektów szkolenia.

4.3. Opis zadań związanych z przygotowaniem bazy danych

Zadania powinny zostać określone dla każdego ze stanowisk pracy (terminala systemu).
Dla każdego z zadań proponuje się określić:

a) stanowisko,
b) wprowadzane dokumenty źródłowe,
c) zapisywane pliki danych,
d) organizacja wprowadzania danych,
e) zasady aktualizacji danych w okresie wprowadzania,
f) terminy rozpoczęcia/zakończenia wprowadzania,
g) strona odpowiedzialna za wprowadzenie danych.

4.4. Wdrożenie i sposób jego odbioru - ujęcie w umowie

Autor sugeruje, aby w umowie zastrzegać, że przedstawiając do odbioru wykonane oprogramowanie wykonawca przygotuje i zaproponuje zamawiającemu harmonogram wdrożenia. Umowa powinna określać stopień szczegółowości i postać takiego harmonogramu, na przykład zgodnie z propozycjami zawartymi powyżej w pktach 4.1 do 4.3 oraz zastrzegać, że będzie on przedmiotem odbioru wraz z oprogramowaniem. Jak już powiedziano, zadania związane z instalacją (pkt 4.1) dotyczą w tym przypadku modułów oprogramowania aplikacyjnego wykonywanego na zamówienie, a także tych modułów sprzętu lub oprogramowania systemowego, które wymagane są dla zainstalowania wykonanych programów aplikacyjnych i nie były instalowane wcześniej.

Harmonogram instalacji wszelkich modułów sprzętu i oprogramowania, instalowanych niezależnie od oprogramowania aplikacyjnego wykonywanego na zamówienie (dotyczyć to może również modułów oprogramowania aplikacyjnego gotowego), powinien stanowić załącznik do umowy już w chwili jej zawierania.

Co do innych kwestii związanych z wdrożeniem wątpliwości i spory dotyczą ponadto często sposobu i kryteriów odbioru etapów wdrożenia. Proponuje się, aby w umowie zastrzegano na przykład obowiązek

prowadzenia dziennika wdrożenia w którym wykonawca odnotowuje, a zamawiający potwierdza wykonanie kolejnych zadań wdrożenia.

Literatura

1. Umowy w zakresie informatyki i ochrona programów komputerowych, B.Czachórska, Warszawa (1980),
2. Jak zawierać umowy w zakresie informatyki - poradnik, B.Czachórska, J.Irlik, R.Pesel, Wyd.IOPM, Warszawa (1985)
3. Standards for Software Life Cycle Processes, IEEE Computer Society, (1988),
4. A Guide to Software Requirements Secification (SRS), IEEE/ANSI Std 830-1984,
5. Dependability of Critical Computer Systems, Ed.F.J.Redmill, Elsevier, London (1988).

Andrzej Zaliwski Marian Kuraś
Akademia Ekonomiczna w Krakowie

Pakiet VIB-CASAD

I was in time to catch anevanescent glimpse of my white hat left behind to mark the spot where the secret sharer of my cabin and of my thoughts, as though he were my second self, had lowered himself into the water to take his punishment; a free man, a proud swimmer striking out for a new destiny.

J. Conrad "The Secret Sharer"[1]

Wprowadzenie

Niniejsza publikacja jest prezentacją założeń prototypu oprogramowania narzędziowego przeznaczonego do wspomagania procesu analizy systemów informacyjnych. Efektem wykonanych prac jest pełna koncepcja i moduły główne środowiska projektowego typu ES-GDSS o roboczej nazwie VIB-CASAD. Omawiany pakiet w znacznym stopniu powinien umożliwić pokonanie trudności i zakłóceń w przebiegu prac projektowych, na jakie napotkali autorzy korzystając z metodyki ISAD [Kuraś i in. 1992]. Postęp prac nad pakietem VIB-CASAD spowodował konieczność zaktualizowania rozwiązań przedstawionych w [Kuraś, Zaliwski, 1993][Zaliwski, Kuraś, 1993].

Projektowanie systemów informacyjnych

Projektowanie SI jest w swej zasadniczej fazie - analizie - transferem wiedzy między uczestnikami procesu: użytkownikami i analitykami. Ma to na celu opanowanie złożoności systemu będącego przedmiotem prac poprzez analizę logiczną i strukturyzację (patrz: [Yourdon, 1986,89]. Analiza ma prowadzić do porządkowania postrzeganej rzeczywistości, jej uproszczenia poprzez zreformułowanie wstępnie postawionego problemu [Nerson, 1992]. Temu służą nowe podejścia do analizy systemów - jak w latach '70 i '80 podejście strukturalne a obecnie obiektowe [Coad, Yourdon, 1991][Martin, 1993].

Modelowanie jako metoda i technika akwizycji oraz transferu wiedzy o SI pojawiło się z końcem lat '70. Od tego czasu powstało wiele metod, które pozwalają na prezentację wszelkich aspektów, jakie należy uwzględnić w toku analizy SI. Można przy tym łatwo zauważyć, że znaczny nacisk kładzie się na aspekt organizacyjny i społeczny. W pierwszym przypadku wymusza to konieczność podporządkowania zastosowań potrzebom organizacji, która wprowadza i rozwija zastosowania nowoczesnej techniki TI po to, by uzyskać wymierne korzyści i by zdobywać przewagę konkurencyjną. Uwzględnienie aspektów społecznych jest koniecznością wynikającą z podporządkowania zastosowań potrzebom użytkowników oraz z potrzeby ich uczestniczenia w procesie tworzenia SI.

Potrzeby zmian w praktyce projektowania

Zmiany w gospodarce powodują przyspieszone uczenie się przez organizacje zasad zarządzania oraz korzystania z techniki informacyjnej. Zainteresowanie użytkowników techniką informacyjną oraz uczenie się jej

[1]) J.Conrad: Selected novels and stories. Hamlyn Publishing 1986.

wykorzystania zwiastuje przyspieszenie rozwoju zastosowań. Będą to - jak należy przypuszczać - inne zastosowania niż obecne, gdyż elementarne potrzeby zostaną zaspokojone. Będzie się wzmagał popyt na aplikacje wspomagające zarządzających przy zwiększaniu wymagań wobec dotychczasowych. Należy sądzić, że będzie się to odbywało w mniej komfortowych warunkach, przy ostrzejszych wymaganiach, szczególnie finansowych.

Oznacza to, że pojawi się zapotrzebowanie na metody projektowania zapewniające systematyczne podejście do **projektowania systemów informacyjnych organizacji**. Zastąpi to dotychczas dominujące podejście do komputeryzacji firmy, komórek organizacyjnych czy stanowisk, polegającego na doraźnym doborze lub tworzeniu oprogramowania.

Pojawi się zapotrzebowanie na wiedzę a przede wszystkim praktyczne umiejętności rozpoznawania potrzeb użytkowników i ich analizę prowadzącą do określenia wymagań wobec systemu informacyjnego. Rozwiązania techniczne będą w coraz większym stopniu podporządkowane potrzebom funkcjonowania organizacji i zarządzania. Zastosowania będą stopniowo oceniane na podstawie kryteriów ustalonych przez użytkowników.

Wnioski z doświadczeń i obserwacji trendów rozwojowych

W przewidywaniu zmiany podejścia do tworzenia SI autorzy od kilku lat pracują nad stworzeniem metodyki i komputerowego jej wspomagania.
Przyjęto założenie, że metodyka powinna:

- obejmować całość prac wykonywanych podczas modernizacji SI;
- być raczej przewodnikiem po metodach, technikach i narzędziach, jakie są dostępne w literaturze, niż nową metodyką zalecającą jedyny sposób postępowania i jedynie dostępne techniki i narzędzia;
- oferować konwencję graficznego opisu struktur SI wymuszającą strukturyzację problemów;
- wymuszać standardowe rozwiązania w ramach dowolności typowej dla miękkiego podejścia systemowego;
- oferować komputerowe wspomagania najbardziej pracochłonnych czynności prac analitycznych.
- być sprawdzona przed przystąpieniem do tworzenia wspomagania komputerowego;
- zawierać dostępną dla użytkownika wiedzę o metodach, technikach i narzędziach (podręcznik, kontekstową 'ściągę' oraz diagnostykę błędów);
- pozostać rozwiązaniem otwartym - w szczególności w zakresie wspomagania;
- przekształcać się w system oparty na wiedzy, wspomagający współpracę użytkowników i informatyków.

Wcześniejsze próby stosowania oryginalnych metodyk zachodnich (ISAC, Gane+Sarson) zakończyły się niepowodzeniem. Jego przyczyny były dwojakiego rodzaju. Pierwszą był negatywny syndrom NIH (Not Invented Here) powodujący, że użytkownicy nie ufają metodzie stworzonej i wykorzystywanej w innych warunkach. Druga grupa przyczyn niepowodzeń to niedostępność narzędzi CASE (lata 80). To stało się przyczyną podjęcia prac nad własną metodyką a następnie nad jej wspomaganiem komputerowym. (Obecnie narzędzia stały się bardziej dostępne ale ich cena jest znaczna, brak doświadczeń w korzystaniu z tych narzędzi, czy wreszcie nieodczuwanie potrzeby wykorzystywania narzędzi tego typu.

Z tworzeniem i rozwojem SI jest związane wiele barier. Ważniejsze z nich to trudności wynikające ze złożoności systemu oraz trudności techniczne, ludzkie i organizacyjne obejmujące m.in. komunikację z użytkownikiem w zespole projektowym,

niezrozumienie potrzeb informacyjnych organizacji oraz jej celów strategicznych, zwłaszcza, że często wśród samych użytkowników nie ma zgodności co do celów i wymagań wobec modernizowanego systemu informacyjnego. Wszystkie problemy jakie dotyczą produkcji oprogramowania w szczególności i w zwielokrotnionej skali dotyczą tworzenia systemów informacyjnych. Jako, że złożoność tych systemów przekracza o rząd wielkości złożoność innych grup oprogramowania, ponadto proces wytwarzania oprogramowania sam w sobie jest złożony.

Nasza koncepcja wspomagania projektowania

Punktem wyjścia dla naszej koncepcji wspomagania projektowania jest potraktowanie rozwoju oprogramowania jako procesu pozyskiwania wiedzy [Jaworski, 87-92]. Zakładamy, że istnieje baza wiedzy zawierająca aktualnie znane informacje na temat realizowanego SI. Postęp następuje poprzez zdobywanie dalszych informacji i włączanie ich w strukturę bazy wiedzy, lub przez przetwarzanie informacji już posiadanej. Formalna notacja umożliwiająca reprezentację wiedzy o projekcie SI na każdym jego etapie jest zapewniona przez notację jMAP (or infoMAPs) [Jaworski,1988-1993].
Drugim elementem naszego rozwiązania jest tzw. VIB - Virtual Intelligent Board - stanowiąca inteligentne medium komunikacji między analitykiem a użytkownikiem.
[Zaliwski, Kuras, 93][Kuras, Zaliwski, 93].
Trzecim elementem jest specjalnie zaprojektowana metodyka obiektowa [Kuras, 91-93][Sarga, 93].

CASAD - Środowisko projektowe typu ES-GDSS

Zarówno użytkownik jak i projektant tworzą we własnej świadomości model będący wyobrażeniem tworzonego lub modernizowanego systemu. By uniknąć błędów wynikających ze wzajemnego niezrozumienia [Kuraś, Zaliwski, 1993] konieczne jest nałożenie się tych modeli na siebie. W tym celu konieczny jest szereg przedsięwzięć umożliwiających nałożenie się na siebie modeli użytkownika i analityka/projektanta.
Wyobraźmy sobie system ze sztuczną inteligencją, który tworzy model systemu na podstawie wprowadzonych do niego informacji. Informacje te byłyby gromadzone w tzw. przestrzeni informacyjnej projektu. Przestrzeń ta istnieje w każdej fazie rozwoju projektu. Jest wielowarstwowym i wielowymiarowym modelem aktualnego stanu projektu.
Model w Przestrzeni Informacyjnej może być potraktowany jako zbiór hipotez i stanowić niezależnie od innych możliwych działań przedmiot badań prowadzonych przez system doradczy (ES), w którego bazach wiedzy powinny znaleźć się m.in.:

- Model środowiska zewnętrznego tj. informacje o uwarunkowaniach zewnętrznych w jakich działa organizacja a jakie mają lub mogą mieć wpływ na jej działanie.
- Model środowiska wewnętrznego systemu informacyjnego tj.
  - fakty o analizowanej organizacji,
  - specyfikacja potrzeb informacyjnych,
  - wymagania użytkownika/klienta.

Ponadto w bazach wiedzy powinny znaleźć się:

- ograniczenia i warunki brzegowe projektu,
- informacje o podobnych projektach,
- podręcznik(i) metodyki.
- zasady tworzenia diagramów,
- Schemat organizacji zbiorów programu
- struktura zbiorów dla potrzeb

ewentualnego rozszerzenia interfejsów programu.

- Słownik definiowany i rozszerzany przez użytkownika.
- POST-IT. Mechanizm umożliwiający gromadzenie i porządkowanie notatek robionych na bieżąco w trakcie sesji z programem.

Wprowadzając do rozważanego systemu użytkownika i analityka dochodzimy do sformułowania koncepcji środowiska projektowego typu ES-GDSS.

VIB - "Virtual Intelligent Board" jako narzędzie komunikacji w zespole projektowym

Środowisko projektowe wykorzystujące VIB ma wykrywać błędy logiczne i merytoryczne, ma być z założenia bezstronnym uczestnikiem dyskusji między użytkownikiem a projektantem. Zadaniem VIB jest przechwytywanie i zbieranie informacji wymienianych między członkami zespołu projektowego a także stanowienie inteligentnego interfejsu między zespołem projektowym a innymi programami wchodzącymi w skład zintegrowanego środowiska projektowego typu ES-GDSS.
By system mógł spełnić zakładane cele niezbędne jest:

- przechowywanie danych o podobnych do analizowanego przypadkach i ich udostępnianie.
- przechowywanie i udostępnianie 'wiedzy podręcznikowej'
- wykrywanie sprzeczności w nowo projektowanym systemie (np. przez użycie jMapy).
- gromadzenie faktów uzyskanych w trakcie sesji oraz zachowanie ich integralności (integrity). Sugerowanie jakie dane mogą być jeszcze potrzebne.
- tworzenie specyfikacji SI na podstawie danych zebranych w bazach podczas wielu sesji (Oczywiście zakładana jest pewna ingerencja człowieka).

W systemie VIB-CASAD wydzielono trzy niezależne warstwy (Rys.1):

- Warstwę projektową (Design Layer) (look & feel)
- Warstwę transformacji (Transformation Layer)
- Warstwę aplikacji (Application Layer)

Zadaniem pierwszej warstwy jest stanowienie narzędzia komunikacji między użytkownikiem a projektantem. Głównym elementem tej warstwy jest Edytor Diagramów Przepływu Danych (Rys 2). Jest to wyspecjalizowany edytor graficzny wspomagający pierwszy etap analizy tj. gromadzenie danych o rozpatrywanym SI. Edytor ten działa w sposób interakcyjny i komunikuje się bezpośrednio z członkami zespołu projektowego. Komunikacja odbywa się poprzez użycie i wspólne tworzenie diagramów opartych o specjalnie do tego celu opracowaną metodykę [Kuraś, 1992][Sarga, 1993]. Narysowanie na diagramie tzw. OMSI (Obiekt Metodyki Systemu Informacyjnego) powoduje automatyczną konstrukcję stowarzyszonego obiektu w Obiektowej Bazie Danych (ODBMS). Obiektowa Baza danych wraz z Przestrzenią Informacyjną Projektu (Project Information Space (PIS)) stanowi tzw. Bazę Danych Projektowych (BDP) (innymi słowy Repository). Jako Przestrzeń Informacyjna Projektu może być użyty w najprostszym przypadku Microsoft Excel z załadowanym programem InfoFARM. Jako moduł współpracujący z Excelem za pośrednictwem DDE (Dynamic Data Exchange) jest realizowany w języku C moduł (HNN) (tzw. Neural Server). HNN wraz z bazą zawierającą stare projekty (Old jMAP) stanowi następną warstwę - Warstwę Transformacji.
Idea "Przestrzeni Informacyjnej" pochodzi od

Webstera [1987]. W odniesieniu do naszego systemu Przestrzeń Informacyjna Projektu jest pewnym znanym i dobrze zdefiniowanym w danej chwili podzbiorem ogólnej przestrzeni informacyjnej opisującej badaną organizację, zaimplementowanym przy użyciu techniki jMAP [Jaworski, 1986-93].
W tej warstwie przewidziano mechanizmy wspierające przebieg procesu projektowania poprzez:

- wyszukiwanie i przypominanie rozwiązań pochodzących z uprzednio realizowanych projektów, podobnych do aktualnie realizowanego.

To zadanie jest realizowane przez wyszukiwanie asocjacyjno-rozproszone implementowane w module HNN (Rys. 2). Obraz realizowanego projektu jest reprezentowany jako jMapa i stanowi wejście dla procesu rozpoznawania obrazu).

- notowanie niedokończonych pomysłów oraz przechowywanie rozwiązań odrzuconych, by zapobiec 'gubieniu informacji' (POST-IT Enginee),
- zapewnienie w każdej chwili możliwości cofnięcia się do rozwiązań poprzednich (Extended Backup & Recovery).

Następna warstwa - Warstwa Aplikacji zawiera generator aplikacji korzystajacy z informacji pochodzących z warstwy poprzedniej.
Generator aplikacji może wykorzystywać definicje klas i obiektów wybrane z ogólnej bazy 'Reusable Component Library' na podstawie podobieństw do starych projektów przefiltrowanych przez jMapę wybraną wcześniej z 'Old jMAP'.

Wnioski

Proponowane w niniejszej publikacji rozwiązanie łączy w sobie wiele nowych elementów takich jak na przykład koncepcja VIB oraz implementacja przestrzeni informacyjnej. System VIB-CASAD jest w tej chwili koncepcją najbardziej rozbudowanego środowiska gromadzą-cego, przechowującego i udostępniającego wiedzę o tworzonym SI. Wykorzystuje notację infoMAP (czy inaczej jMAP) jako nośnik tej wiedzy (infoMAPY są doskonałym narzędziem do pozyskiwania i gromadzenia wiedzy). Zalety notacji infoMAP (m.in. to, że cały proces rozwoju oprogramowania może być dokumentowany w jednolitej postaci) czynią możliwym implementację wielu nowych funkcji. Możliwe jest na przykład 'rozmyto'-asocjacyjne wyszukiwanie roz-wiązań podobnych, czy też łatwe przechodzenie z jednej notacji na inną. Sprawdzenie przydatności fuzzy & associative search wymaga jednak wcześniejszego wypełnienia baz wiedzy systemu VIB-CASAD informacjami o wcześniej wykonanych projektach. Zachodzi zatem potrzeba zgromadzenia opisów z kilku projektów wykonywanych z wykorzystaniem tego narzędzia. VIB-CASAD jest obecnie w fazie opracowywania prototypów podstawowych modułów (w szczególności z warstwy pierwszej i drugiej). Najpoważniejszym zagrożeniem dla wypracowanej koncepcji (częściowo zaimplementowanej) jest brak odpowiednich środków finansowych.
W najbliższym czasie przewiduje się dokończenie modułów systemu.
Szczegółowego opracowania wymaga także koncepcja VIB od strony logicznej jako koncepcji pracy zespołu projektowego opartej o wykorzystanie inteligentnej tablicy zrealizowanej np. przy pomocy komputera z panelem oraz kamery-scannera skierowanej na ekran[2].

[2]) Wspomniane urządzenie wejściowe jest w trakcie realizacji prototypu. Kamera Scanner jest wycelowana na ekran na którym jest obraz rzucany przez panel. Analityk jak i użytkownik posługują się typowym

Bibliografia

Coad, P., Yourdon, E. (1991). Object Oriented Analysis. 2-nd ed. Englewood Cliffs. Prentice-Hall, Inc.

Jaworski W.M., Cummings T., (1991). Program Normalization and Optimization: Using infoMaps AS AN INSPECTION AND PROGRAM PROCESSING TOOL. Canadian Conference on Electrical and Computer Engineering, Quebec City, September 1991.

Jaworski W.M., Deslauriers B., (1990). Software Process, Maintenance, Products: Software Development Environment based on infoMAPs. Control and Electrical Eng. Conference, Ottawa, September 1990.

Jaworski W.M., Grogono P.D., (1991). infoMAPS: a Pragmatic Environment for Seamless and Nondeterministic Software Development, Concordia University, Computer Science Dept., January 1991.

Jaworski W.M., Zaliwski A., Kuraś M., (1993). InfoMapy jako nowa metoda pozyskiwania wiedzy. Informatyka Nr. 11.

Kuraś M., (1989). Zakres modernizacji systemu informacyjnego. Zeszyty Naukowe AE Kraków. Nr. 286.

Kuraś M., (1991), Problemy komunikacji z użytkownikami wewnątrz zespołu., Informatyka 10,11, 1991

Kuraś M., (1992). Koncepcja komputerowego wspomagania analizy i projektowania systemów informacyjnych. Zeszyty Naukowe AE Kraków. Nr. 366.

Kuraś, M., Sarga, D., Zaliwski, A. (1992). Metoda planowania i tworzenia systemów informacyjnych zarządzania. Raport z pracy badawczej 29/ISEI/7/91/S. Katedra Informatyki. Akademia Ekonomiczna. Kraków.

Kuraś M., (1990). Wybór strategii rozwoju zastosowań informatyki do obsługi systemu informacyjnego. Zeszyty Naukowe AE w Krakowie. Nr. 330.

Kuraś M., Maciołek W., Sarga D., Zaliwski A., (1991). Komputerowe wspomaganie specyfikacji SI. Ogólnopolska Konferencja Instytutów, Katedr i Zakładów Informatyki. Rogi k/Łodzi, 9-11 września 1991.

Kuraś M., Sarga D., Węgrzynowski A., (1990). Specyfikacja potrzeb informacyjnych z udziałem użytkowników. Zeszyty Naukowe AE w Krakowie. nr. 330.

Martin J., (1993). Principles of Object Oriented Analysis and Design. P T R Prentice Hall. Englewood Cliffs, New Jersey.

Nerson, J.-M. (1992). Applying Object-Oriented Analysis and Design. Communications of the ACM. Vol.55. No.9. September 1992.

Sarga D., (1993). Opis funkcji hipotetycznego przedsiębiorstwa.
AE Kraków.

Webster D.E, (1987). Mapping the Design Representation Terrain., Technical Report STP-093-87, MCC, Software Technology Program., July 1987.

Webster, D., (1988). Mapping the design representation terrain: a survey. IEEE Computer, (December 1988). pp. 8-23.

Yau, S., R.Nicholl, J.Tsai, and S.S.Liu, (1988). An integrated life cycle for software maintenance. IEEE Trans. Soft. Eng., 14, 8, 1128-44.

Yourdon, E. (1986). Managing the Structured Techniques. Strategies for Software Development for 1990's. 3-rd ed. New York -London. Yourdon Press.

Yourdon, E. (1989). Modern Structured Analysis. Englewood Cliffs. Prentice-Hall, Inc.

Zaliwski, A., Kuraś, M., (1993). CASAD - Nowe podejście do modernizacji SI. Materiały Wiosennej Szkoły PTI. Szczyrk.

---

wskaźnikiem laserowym używanym na wykładach. Scanner odbiera obraz a na nim lokalizuje położenie wyraźnie jaśniejszej plamki. Współrzędne plamki są podawane przez odpowiedni driver do aplikacji w sposób identyczny jak to się dzieje w przypadku otrzymywania przez aplikację współrzędnych myszy. Urządzenie nie wymaga żadnych przeróbek oprogramowania. Driver jest widziany przez aplikację jako zwykła szara mysz. Wciśnięciu klawisza na myszy odpowiada wysłanie impulsu przez jednokanałowego pilota sprzężonego ze wskaźnikiem laserowym. Samo urządzenie wskaźnikowe nie wymaga kabli.

Mariusz Grabowski
Katedra Informatyki, Akademia Ekonomiczna

Zastosowanie sieci neuronowych w zarządzaniu

Streszczenie

W referacie zostaną omówione podstawowe cechy sieci neuropodobnych[1] oraz klasy problemów rozwiązywanych przy ich pomocy. Zostaną również wskazane i skrótowo omówione przesłanki tworzenia, oraz podstawowe obszary zastosowań sieci neuropodobnych w zarządzaniu.

Wstęp

Sieci neuronowe są dziedziną rozwijającą się na przestrzeni ostatniego półwiecza. Jednak szczególne zainteresowanie problematyką, obserwuje się od zaledwie ostatnich kilku lat. Po przezwyciężeniu problemów natury metodologicznej związanej z opublikowaniem pracy M. Minsky'ego i S. Paperta *Perceptrons* w roku 1969, które spowodowało zastój w badaniach w latach siedemdziesiątych [Tadeusiewicz 1993], nastąpił wzrost zainteresowania sieciami neuronowymi na przełomie lat osiemdziesiątych i dziewięćdziesiątych. Można dopatrywać się go w:

- powstaniu bariery technologicznej w zwiększaniu wydajności systemów opartych o architekturę sekwencyjną [Wilusz 1987],
- istnieniu bariery w oprogramowaniu systemów współbieżnych [Ben-Ari 1989],
- potrzebie automatycznego przetwarzania danych niekompletnych, lub takich których natura jest nieopisywalna, bardzo trudna do opisania, lub rozmyta [Wołoszyn 1990] [Pawlak 1991],
- potrzebie rozwiązywania problemów trudno poddających się algorytmizacji,
- rozwojem metod opisu wiedzy rozmytej [Mulawka i Kopertowski 1994].

Do wzrostu zinteresowania sieciami neuronowymi w ostatnich latach, oprócz wyżej wymienionych, przyczynił się niewątpliwie rozwój technologiczny, umożliwiający praktyczną realizację doświadczalnych i komercyjnych systemów opartych na sieciach neuropodobnych.

Obecnie klasy zagadnień rozwiązywanych przy pomocy sieci neuronowych można sprowadzić do nastepujących [Tadeusiewicz 1993]:

- rozpoznawanie (obrazów), klasyfikowanie, filtracja,
- robotyka, automatyka, teoria sterowania,
- optymalizacja,
- pamięci asocjatywne (BAM), problematyka pamięci rozproszonych,
- sztuczna inteligencja.

Przykłady zastosowań sieci neuropodobnych prezentowane jako owoc prac badawczych na różnorodnych konferencjach naukowych koncentrują się wokół zagadnień technicznych [Materiały 1994]. Jest to rzeczą naturalną, gdyż tematyka sieci neuronowych jest rozwijana głównie przez przedstawicieli tych nauk. Także

[1] Sieci neuropodobne są modelem (bardzo daleko odbiegającym od doskonałości) naturalnych sieci neuronowych w organizmach żywych. W niniejszym opracowaniu, termin sieci neuropodobne odnosi się do modeli sieci neuronowych, a termin sieci neuronowe do dziedziny zajmującej się modelowaniem naturalnych sieci neuronowych.

zagadnienia będące tematem modelowania przy pomocy sieci neuronowych: rozpoznawanie obrazów, optymalizacja czy teoria sterowania, wyrastają *stricte* z gruntu technicznego.

Czy sieci neuropodobne mogą znaleźć zastosowanie w dziedzinach biznesu, takich jak: marketing, analiza finansowa, planowanie? Jeśli tak to dlaczego nie ma na rynku literatury dotyczącej powyższych zagadnień, a przynajmniej nie ma jej w stopniu wystarczającym?

Powodów jest zapewne wiele, lecz nie należy do nich na pewno brak komercyjnych systemów opartych o sieci neuropodobne (SOSN). Trzeba z całym naciskiem podkreślić, że komercyjne zastosowania SOSN w zarządzaniu istnieją i z powodzeniem działają. Podstawowym powodem braku literatury dotyczącej zastosowań sieci neuropodobnych w zarządzaniu jest fakt, że wiele prac, z zakresu SOSN jest chronionych prawami autorskimi i jeśli powstają nawet publikacje na ich temat to są one z przyczyn oczywistych bardzo ogólnikowe.

Przesłanki stosowania sieci neuropodobnych w zarządzaniu

Systemy ekonomiczne w przeciwieństwie do systemów technicznych, są mało zdeterminowane. Wielokrotnie nasza wiedza dotycząca natury zjawisk ekonomicznych jest niewystarczająca, do podjęcia na jej podstawie określonych wniosków. Często nie znamy zależności funkcjonalnych, a nawet liczby zmiennych występujących w danym modelu. W związku z powyższym zachodzi potrzeba poszukiwania nowych metod badawczych, które pozwalałyby na analizę zjawisk ekonomicznych bez konieczności znajomości matematycznego modelu opisującego dane zjawisko. Wydaje się, że zastosowanie sieci neuropodobnych w dużej mierze pozwala na przezwyciężenie tego problemu[2].

Zarządzanie zajmuje się przede wszystkim podejmowaniem decyzji. Wszystkie metody rozwijane przez tę dziedzinę i dziedziny pokrewne, dążą do tego aby dać decydentowi aparat pozwalający na podjęcie decyzji optymalnej[3], mając na uwadze zminimalizowanie ryzyka podjęcia decyzji błędnej. W tym celu jest konstrułowany model sytuacji decyzyjnej i na jego podstawie wnioskuje się na temat podejmowanych działań. Na model ma wpływ wiele czynników takich jak: ogólna sytuacja związana z otoczeniem modelu (warunki zewnętrzne w jakich model funkcjonuje) oraz sama konstrukcja modelu (relacje i zmienne w nim uwzględnione). Na gruncie metod ilościowych, w skład parametrów modelu wchodzą [Bierman, Bonini, Hausman 1986]:

- zmienne decyzyjne *(ang. decision variables)*; czynniki które mają decydujący wpływ na wybór decyzji; reprezentują zbiór możliwości decydenta,
- zmienne egzogeniczne *(ang. exogenous variables)*; zmienne ważne dla sytuacji decyzyjnej, ale pozostające poza wpływem decydenta,
- przepisy i ograniczenia *(ang. policies and constraints)*; wyrażają procedury i regulacje np. w danej firmie oraz ograniczenia fizyczne np. wartość wyprodukowanych sztuk wyrobów nie może być mniejsza od 0,
- jednostki miary *(ang. performance masures)*; w metodach ilościowych

[2] Sieci neuropodobne z powodzeniem stosuje się do gry na giełdzie. Otrzymuje się w ten sposób prognozy z ok. 70% dokładnością [Tadeusiewicz 1993].

[3] Jako kryterium optymalizacji decyzji stosuje się dwie zasady: minimalizacji kosztów przy ustalonym zysku lub maksymalizacji zysku przy ustalonych kosztach.

decyzja wyrażona jest w jednostkach (np w zł);
- zmienne pośrednie *(ang. intermediate variables)*; zmienne wiążące zmienne egzogeniczne ze zmiennymi decyzyjnymi.

Jedną z podstawowych cech modelu jest to, że jest on abstrakcją i najczęściej uproszczeniem rzeczywistości. W przypadku modeli bardziej złożonych, oraz tych które mają naturę probabilistyczną, niesie to z sobą na pewno szereg korzyści, gdyż możliwa jest analiza problemu poprzez odrzucenie nieistotnych związków, które jedynie zaciemniają pole widzenia. Z drugiej jednak strony istnieje niebezpieczeństwo, że w trakcie budowy modelu celowo, upraszczając rzeczywistość, odrzuci się istotne związki i zmienne. Obecnie najczęściej stosowaną alternatywą dla klasycznych metod modelowania są metody wykorzystujące sieci neuropodobne. W sytuacji gdy znamy odpowiednią liczbę przykładów danych wejściowych modelu oraz odpowiadające im wartości wyjściowe, możliwa jest konstrukacja modelu w oparciu o sieci neuropodobne. Mechanizmy dostępne przy modelowaniu sieci neuropodobnych umożliwiają kojarzenie danych, uogólnianie doświadczeń, dostrzeganie pozornie nie istniejących związków pomiędzy danymi obrazującymi konkretne zjawisko [Tadeusiewicz 1993], co było poza zasięgiem metod tradycyjnych. Sieć neuropodobna tworzy w ten sposób model, który jest wyrażony nie poprzez zależności funkcjonalne, ale jest on zawarty w strukturze sieci i wyrażony odpowiednimi wagami synaptycznymi wejść poszczególnych neuronów. W tym podejściu sam proces modelowania sprowadza się do przygotowania odpowiedniej liczby danych i uruchomienia algorytmu uczenia sieci. Oczywiście celowość użycia sieci neuropodobnej, jak i wybór jej typu i topologii zależy od konkretnych zastosowań.

Obszary zastosowań

Wiele osiągnięć nauk technicznych z zakresu sieci neuronowych można bez większych modyfikacji przenieść do zastosowań w zarządzaniu. Dotyczy to przede wszystkim sieci rozpoznających obrazy, prognozujących i optymalizujących (np. sieci *Hopfielda*).

Obszarów zastosowań sieci neuronowych w zarządzaniu można szukać przede wszystkim w dziedzinach, gdzie stosuje się opisane wyżej modelowanie oraz metody statystyczne [Cherkasky & Lari-Najafi 1991]. Zaliczyć do nich należy:

- optymalizację,
- prognozowanie,
- filtrację danych,
- określanie brakujących danych w szeregach czasowych,
- statystyczną kontrolę jakości,
- taksonomię.

Zagadnienie optymalizacji jest często wykorzystywane przez wiele dziedzin związanych z zarządzaniem. Należy tutaj przede wszystkim zaliczyć problemy związane z wyborem procesu technologicznego, harmonogramowaniem i planowaniem (problem komiwojażera). Jako modele sieci neuronopodobnych stosuje się tutaj najczęściej sieci *Hopfielda*. Sieci neuropodobne można z powodzeniem stosować w predykcji. Poprzez zdolność sieci do wykrywania związków pomiędzy danymi, możliwe jest dokonanie prognozy bez znajomości nie tylko parametrów funkcji trendu ale nawet jej klasy.

W procesie podejmowania decyzji, bardzo ważnym jest oczywiście zagadnienie informacji zmniejszającej ryzyko podjęcia błędnej decyzji. Nadmiar danych jest tak samo szkodliwy, a może nawet bardziej, niż jej brak. Dlatego ważnym aspektem jest usuwanie szumów z informacji decyzyjnej. Sieci służące do rozpoznawania obrazów są w stanie ignorować w trakcie rozpoznawania składowe

nieistotne czyli te, które zawierają redundancję.

Innym obszarem metod ilościowych, gdzie klasyczne metody zawodzą jest próba szacowania brakujących danych. Klasyczne metody statystyczne napotykają np. na problemy natury obliczeniowej. Po prostu niektóre algorytmy są bardzo złożone obliczeniowo. Sieci neuropodobne posiadają zdolność do równoległego[4] (tzn. bardzo szybkiego!) przetwarzania danych. Co więcej, zdolność ta leży w samej naturze sieci, dlatego bez konieczności dodatkowych nakładów pracy możliwa jest realizacja rozwiązań będących w swej naturze równoległymi. W systemach statystycznej kontroli jakości opartych o sieci neuropodobne możliwe jest śledzenie kilku zmiennych jednocześnie a nie tylko jednej, jak to ma miejsce w metodach klasycznych. Przy czym nie jest to jedynie równoległy monitoring kilku zmiennych ale śledzenie wzajemnych współzależności pomiędzy tymi zmiennymi, co może mieć istotny wpływ na decyzje podejmowane w procesie statystycznej kontroli jakości [Carig 1991].

Taksonomia jest dziedziną zajmującą się zagadnieniami grupowania. Grupowanie jest po prostu kwalifikowaniem obiektu spełniającego pewne kryteria do określonej grupy. Dlatego modele sieci neuropodobnych wykorzystywanych w rozpoznawaniu będą miały tutaj z pewnością szerokie zastosowanie. Istotną rzeczą jest, że sieci neuronowe są w stanie klasyfikować obiekt do określonej grupy bez wcześniejszej znajomości tego obiektu.

[4]Problem równoległego przetwarzania informacji jest realizowany technicznie przy pomocy systemów współbieżnych. Jednak stopień współbieżności jest w sieciach neuronowych setki razy większy [Tedeusiewicz 1990].

Systemy informacyjne

Omówione wyżej zastosowania sieci neuropodobnych w zarządzaniu skupiały się wokół problematyki modelowania, prognozowania, taksonomii i statystycznej kontroli jakości. Jeszcze jednym istotnym polem dla sieci neuronowych są zastosowania informatyki w zarządzaniu. Już w roku 1977 James Anderson z Uniwersytetu Browna zbudował sieć *Brain State in Box* służącą do wyszukiwania informacji w bazie danych. Sieć ta pod względem funkcjonalnym przypominała pamięci asocjatywne (BAM) [Tadeusiewicz 1993]. *Brain State in a Box* znacznie odbiegał od ówczesnych zastosowań sieci neuropodobnych, które skupiały się przede wszystkim wokół rozpoznawania. Zastosowanie sieci neuropodobnej do wyszukiwania podobieństw w bazie zawierającej informacje o rozwiązaniach projektowych jest omówione w pracy [Zaliwski i Kuraś 1993]. W związku z zastosowaniem sieci neuropodobnych do selekcji informacji należy oczekiwać wzrostu zastosowań sieci neuropodobnych w systemach wspomagania decyzji, w których skład wchodzą również modele pochodzące z omówionej wyżej analizy ilościowej, oraz w systemach doradczych.

Systemy doradcze budowane jako SOSN mają tę przewagę nad klasycznymi systemami opartymi o reguły wnioskowania, że:

- są w stanie w efektywny sposób przetwarzać wiedzę rozmytą,
- pozwalają na budowę systemu bez udziału eksperta,
- charakteryzują się krótkim czasem budowy (kilka tygodni lub miesięcy),

Wadą ich natomiast jest:

- bardzo małe, lub żadne możliwości objaśniania proponowanych rozwiązań,
- do budowy potrzebna jest duża liczba przykładów.

Dlatego w praktyce łączy się obydwa podejścia tworząc systemy hybrydowe [Caudill 1991].

W pracy [Mulawka i Kopertowski 1994] jest przedstawiony opis szkieletowego systemu doradczego zbudowanego w oparciu o sieć *Hamminga*. W rozwiązaniu tym, sieć neuropodobna jest zarówno bazą wiedzy (zbiór wag poszczególnych synaps) jak i motorem wnioskowania (działanie w fazie rozpoznawania).

Aspekty wdrożeniowe

Sieci neuropodobne nie wymagają programowania. Programowanie sieci jest zastąpione procesem uczenia, który dokonuje się automatycznie bez udziału człowieka. Oznacza to, że nakłady ponoszone na prace związane z opracowywaniem algorytmu rozwiązywanego problemu w porównaniu z technologią tradycyjną są o wiele mniejsze. Można w ten sposób uniknąć wielu problemów związanych z testowaniem algorytmu, a co za tym idzie wdrażaniem systemu[5].

Zakończenie

Niewątpiwie zarządzanie jest dziedziną, w której zasosowanie sieci neuronowych jest jak najbardziej celowe i potrzebne. Należy w związku z tym spodziewać się w najbliższych latach wzrostu zainteresowania sieciami neuronowymi przez pracowników i badaczy z dziedziny biznesu, oraz licznymi nowymi wdrożeniami SOSN. Niniejszy referat miał na celu jedynie przybliżenie tematyki i wskazanie na sens stosowania sieci neuropodobnych w zarządzaniu. Zestawienie systemów komercyjnych zbudowanych w oparciu o sieci neuronowe, a stosowanych w takich dziedzinach jak: marketing, badania operacyjne, analiza finansowa i auditing można znaleźć w pracy [Harston 1991].

Literatura

[1] M. Ben-Ari, *Podstawy programowania współbieżnego*, WNT, Warszawa, 1989,

[2] Maureen Caudil, *Expert Networks*, Byte, October 1991,

[3] Harold Bierman, Jr., Charles P. Bonini, Warren H. Hausman, *Quantitative Analysis For Business Decisions*,

[4] Vladimir Cherkassky, Hossein Lari-Najafi, *Constraind Topological Mapping for Nonparametric Regression Analysis*, Neural Networks, Vol. 4. pp.27-40, 1991,

[5] Carig T. Harston, *Business with Neural Networks*, Handbook of Neural Computing Applications, 1991, pp. 391-399,

[6] Zdzisław Pawlak, *Rough Sets*, Warsaw 1991,

[7] Materiały konferencyjne I Krajowej konferencji: *Sieci neuronowe i ich zastosowania*, Kule (woj. częstochowskie) 12-15 IV 1994, Częstochowa 1994,

[8] Jan Mulawka, Zbigniew Kopertowski, *Szkieletowy system doradczy wykorzystujący sieć neuronową*, Materiały konferencyjne I Krajowej konferencji: Sieci neuronowe i ich zastosowania, Kule (woj. częstochowskie) 12-15 IV 1994, Częstochowa 1994, pp. 371-376,

[9] Ryszard Tadeusiewicz, *Sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa 1993,

[10] Tadeusz Wilusz, *Algorytmy komputerowego wspomagania syntezy sieci neuropodobnych*, Zeszyty naukowe Akademii Ekonomicznej w Krakowie, Nr 248, Kraków 1987, ss. 43-65,

[11] Jacek Wołoszyn, *Grafy rozmyte i możliwości ich wykorzystania w ekonomii*, Zeszyty naukowe - Monografie Nr 90, Kraków 1990,

[12] Andrzej Zaliwski, Marian Kuraś, *CASAD - nowe podejście do modernizacji SI*, V Jubileuszowa Wyższa Międzynarodowa Górska Szkoła PTI, Szczyrk 1993.

[5] Oczywiście dotyczy to tylko tych fragmentów systemu, które w naturalny sposób dają się modelować jako sieci neuropodobne (rozpoznawanie, optymalizacja, itp.).

Stanisław Kędzierski
Akademia Ekonomiczna w Katowicach

Specyfikowanie systemu informatycznego w języku Z

1. Wstęp

System informatyczny można traktować jako pewien model fragmentu interesującej jego twórców rzeczywistości (świata realnego). W procesie modelowania stosuje się wiele metod i technik formalnych, które można podzielić na dwie zasadnicze grupy:

- zorientowane na model, gdzie specyfikacja jest abstrakcyjną definicją modelu rzeczywistości (w tej grupie mieszczą się języki VDM i Z),
- zorientowane na własności, gdzie specyfikacje są aksjomatami definiującymi relacje pomiędzy operacjami (język OBJ).

Metoda formalna to zbiór ścisłych reguł inżynierskich stosująca techniki formalne a te z kolei powinny podlegać mechanicznym manipulacjom zgodnie z pewnymi rachunkami (np. matematycznymi) [Bjorner91]. Inna definicja metody formalnej charakteryzuje ją jako posiadającą logiczne podstawy matematyczne [Barroca92]. W metodach formalnych mają zastosowanie języki formalne (o precyzyjnej składni). Istnieją conajmniej dwa powody, dla których należy zacząć stosować metody formalne:

- edukacja; stosowanie metod formalnych wymaga pewnej wiedzy z matematyki dyskretnej i logiki a część twórców SI jej nie posiada w przeciwieństwie do innych dyscyplin inżynierskich gdzie konstruowanie produktu opiera się na solidnych podstawach matematycznych,
- automatyzacja; możliwe jest wsparcie metod formalnych narzędziami informatycznymi (np. edytorami syntaktycznymi), co znacznie przyśpieszy proces tworzenia SI.

Stosowanie formalizmu przynosi wiele korzyści a do najbardziej znanych zaliczyć można: zwięzłość opisu, precyzję, możliwość manipulacji, abstrakcję.

2. Podstawy języka Z

Język Z jest wynikiem prac naukowców skupionych w Programming Research Group z Uniwersytetu Oxfordzkiego pracujących pod kierunkiem J.M. Spivey'a. Język ten zdobywa coraz większą popularność zarówno w środowisku akademickim jak i przemysłowym. Z charakteryzuje się prostotą zapisu i jest intuicyjnie zrozumiały. Język Z bazuje na podstawowych pojęciach z teorii zbiorów, relacji, rachunku predykatów, funkcji i sekwencji. Istnieje dosyć bogata literatura o języku Z [Hayes87, Imperato91, Ince88, McMorran93, Potter91, Spivey89, Spivey88, Woodcock88]. Przedstawianie elementów języka Z rozpoczyna obiekt.

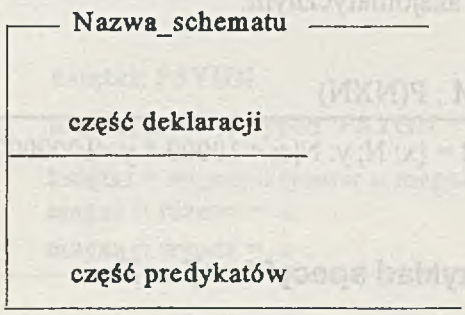
Obiekt w Z jest abstrakcyjną reprezentacją pewnej konkretnej jednostki (ze świata realnego - UoD) i jest skojarzony z pewną identyfikowalną rzeczą (realną lub sztuczną) przydatną dla specyfikacji.

Deklaracja wprowadza nowy *identyfikator* poprzez podanie jego (unikalnej) nazwy i podanie typu:

x : A (x jest typu A)

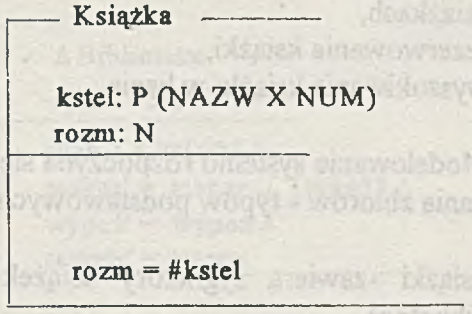
Kluczowym pojęciem w języku Z jest *schemat* , który przedstawia obiekty i relacje zachodzące pomiędzy nimi. Graficznie schemat składa się z dwu części: deklaracyjnej

i warunków (opcjonalnie).

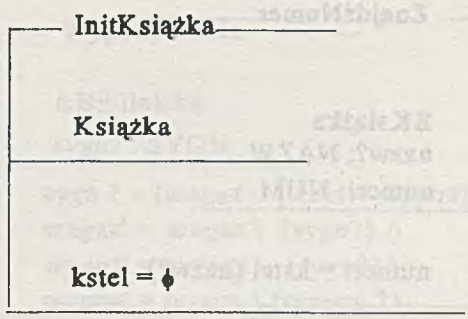


Schematy ze swej natury są modularne, każdy schemat opisuje część specyfikacji. Schematy mogą być traktowane jako obiekty i wykonywać na nich operacje.

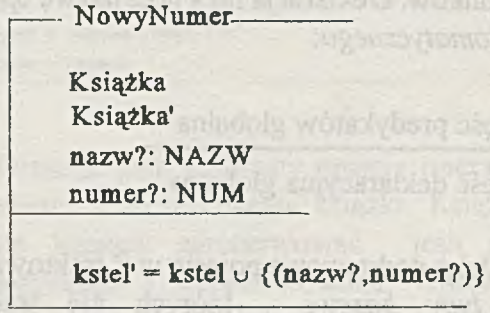
W języku Z modelując wycinek świata realnego posługujemy się obiektami i niezmiennymi (w czasie) relacjami zachodzącymi pomiędzy nimi. Opis ten podaje wszystkie możliwe dopuszczalne stany systemu, który jest *przestrzenią stanów* modelu. Przestrzeń stanów stanowi statyczny element specyfikacji. Przykładowo uproszczona książka telefoniczna to nic innego jak zbiór uporządkowanych par (nazwisko, numer), przy czym nazwisko pochodzi ze zbioru NAZW a numer ze zbioru NUM, co sprawia że książka tel. jest typu P(NAZW X NUM)



Model powinien opisać *stan inicjujący* (zerowy) systemu. W Z podaje się to schematem charakteryzującym system w stanie zerowym.



Zmiany (dynamikę systemu) można zapisać jako dwa stany: przed i po operacji gdzie stan po dekorowany jest znakiem ' (prim). Wprowadzanie i wyprowadzanie informacji oznaczane jest sufixami ? i !. Modelując pojedyńczą operację podaje się warunki wstępne - przed (preconditions), które muszą być spełnione aby można było wykonać operację oraz warunki końcowe - po (postconditions), które opisują zbiór stanów systemu możliwy po prawidłowym wykonaniu operacji. Przykładowo dopisanie nowego numeru do ksiażki telefonicznej może wyglądać następująco:



Schematy (a precyzyjniej operacje) mogą lecz nie muszą zmienić stan systemu. Fakt ten oprócz sygnalizacji znakiem ' może być uwidoczniony w nazwie schematu poprzez dodanie prefixu Δ dla schematu zmieniającego stan Ξ i dla pozostawiającego stan niezmieniony.

ZnajdźNumer

Ξ Książka
nazw?: NAZW
numer!: NUM

numer! = kstel (nazw?)

Zmienne w specyfikacjach Z mają zasięg *lokalny* czyli ograniczony do części predykatów schematu, w którym zostały zadeklarowane. Poprzez łączenie schematów zasięg ten może być zwiększany. Zmienne *globalne* rozciągają swój zasięg na wszystkie specyfikacje od momentu ich deklaracji aż do końca specyfikacji:

| x : A

Podobnie jak zmienne globalne można wprowadzać *ograniczenia globalne* rozumiane jako globalną część warunków co powodować będzie, że wszystkie schematy specyfikacji Z muszą spełniać globalną część warunków. Deklaracja taka nosi nazwę *opisu aksjomatycznego*:

| część predykatów globalna
|---
| część deklaracyjna globalna

Jako podstawowe pojęcie w Z traktowane są *typy bazowe* , których nie trzeba specyfikować:

[A, B, C] (oznacza wprowadzenie trzech typów A, B, C)

Przykładowo w kstel rozumianej jako P(NAZW X NUM) wiadomo, że istnieją zbiory NAZW i NUM o nieznanej zawartości. Deklaracja typów bazowych:

[NAZW, NUM]

wprowadza nowe typy nic nie mówiąc o ich precyzyjniejszej postaci, co można uzyskać w opisie aksjomatycznym:

| NUM : P(NXN)
|---
| NUM = {x: N; y: N| x < 10000 ∧ y <10000000}

3. Przykład specyfikacji w Z

Jako przedstawienie możliwości i siły języka Z wybrany został prosty system biblioteczny o następujących założeniach:

- książka może znajdować się w magazynie, być wypożyczona lub rezerwowana,
- książka rezerwowana nie znajduje się w magazynie,
- książka występuje tylko w jednym egzemplarzu,
- książkę może zarezerwować tylko jedna osoba.

Od systemu oczekuje się, że powinien realizować następujące funkcje:

- dodania nowej książki do bazy,
- przechowywania informacji o wypożyczonej książce,
- przechowywania informacji o zwracanych książkach,
- rezerwowania książki,
- wyszukiwania książki w bazie.

Modelowanie systemu rozpoczyna się od wybrania zbiorów - typów podstawowych:

książki -zawiera sygnatury książek w bibliotece,
wypoż, rezerw, magaz - zawierają sygnatury odpowiednich książek.

Wprowadzany jest zatem typ podstawowy [SYGN].

Biblioteka

książki: PSYGN
magaz, rezerw, wypoż: PSYGN

---

książki = wypoż ∪ rezerw ∪ magaz
magaz ∩ rezerw = ∅
magaz ∩ wypoż = ∅

oraz stan początkowy:

InitBiblioteka

Biblioteka

---

magaz = ∅ ∧ wypoż = ∅ ∧ rezerw = ∅

Wprowadzenie nowej książki do biblioteki odbywa się przez nadanie jej nowej sygnatury i wprowadzenie do bazy oraz położenie w magazynie:

NowaKsiążkaa

ΔBiblioteka
sygn? : SYGN

---

sygn ? ∉ książki ∧
magaz' = magaz ∪ {sygn?} ∧
wypoż' = wypoż ∧
rezerw' = rezerw

Wypożyczenie książki następuje poprzez podanie sygnatury, która usuwana jest ze zbioru magaz lub rezerw i dodanie jej do zbioru wypoż:

Wypożyczenie

ΔBiblioteka
sygn? : SYGN

---

sygn ? ∈ (magaz ∪ (rezerw\wypoż)) ∧
magaz' = magaz \ {sygn?} ∧
wypoż' = wypoż ∪ {sygn?}∧
rezerw' = rezerw \ {rezerw ?}

Zwroty książek następują po zakończeniu wypożyczeń i polegają usunięciu ze zbioru wypoż konkretnych sygnatur i powiększeniu zbioru magaz; aby operacja mogła się odbyć zwracane książki muszą być uprzednio wypożyczone (sygn ? wypoż), zaś do magazynu trafiają tylko książki nierezerwowane:

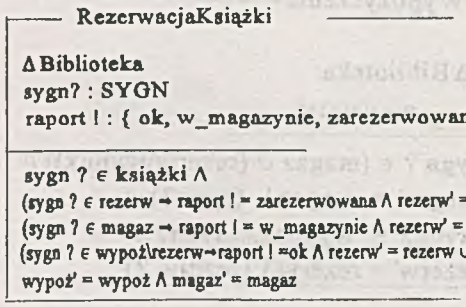
Zwroty

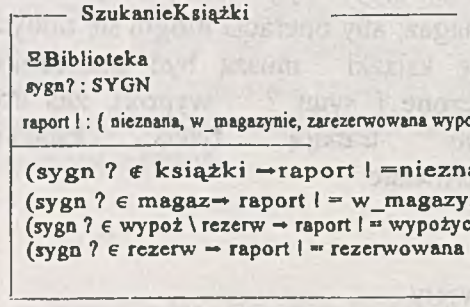
ΔBiblioteka
sygn? : PSYGN

---

sygn ? ⊂ wypoż ∧
magaz' = magaz ∪ {x : SYGN | x ∈ sygn? ∧ x ∉ rezerwy}
wypoż' = wypoż \ sygn ? ∧
rezerw' = rezerw

Ostatnie dwa schematy opisują operacje rezerwacji i wyszukiwania książki. Książkę wolno legalnie zarezerwować jeśli jest wypożyczona i nierezerwowana. Operacja wyszukiwania daje odpowiedź na pytanie : gdzie znajduje się książka?





4. Zakończenie

Przedstawiony skrótowo język Z specyfikacji formalnych systemów informatycznych potwierdza olbrzymie możliwości tkwiące w językach formalnych. Stosowanie metod formalnych pociąga za sobą konieczność pewnej znajomości podstaw matematycznych. Korzyści płynące z wykorzystania tych metod mogą wynagrodzić trud włożony w konstruowanie SI tym sposobem. Dalsze kierunki prac nad wykorzystaniem omawianych metod (w tym także języka Z) idą w stronę stworzenia narzędzi typu CASE wspomagających pracę projektantów.

Literatura:

[Barroca92] Barroca L.M., McDermid J.A., Formal methods: use and relevance for the development of safety-critical systems, The Computer Journal vol. 35, no. 6, 1992 S. 579 - 599

[Bjorner91] Bjorner D., Prehn S., Formal methods in software development requirements for a CASE, w: Endres A., Weber H. (eds), Software develpoment environments and CASE technology, Springer Verlag, Berlin 1991

[Hayes87] Hayes I. (ed), Specification case studies, Prentice Hall International, London, 1987

[Imperato91] Imperato M., An introduction to Z, Chartwell-Bratt, Studentlitteratur, Lund, 1991

[Ince88] Ince D.C., An introduction to discrete mathematics and formal system specification, Clanderon Press, Oxford, 1988

[McMorran93] McMorran M., Powell S., Z guide for begginers, Blackwell Scientific Publications, London, 1993

[Potter91] Potter B., Sinclair J., Till D., An introduction to formal specification and Z, Prentice Hall International, London, 1991

[Spivey89] Spivey J.M., The Z notation; A reference manual, Prentice Hall International, London, 1989

[Spivey88] Spivey J.M., Understanding Z; A specification language and its formal semantics, Cambridge University Press, Cambridge, 1988

[Woodcock88] Woodcock J., Loomes M., Software Engineering mathematics; Formal methods demystified, Pitman, London, 1988

Jerzy Gołuchowski Krzysztof Kania
Akademia Ekonomiczna w Katowicach

Aktywne Bazy Danych w Inteligentnych systemach wspomagania decyzji gospodarczych

Inteligentne systemy wspomagania zarządzania działalnością gospodarczą stawiają nowe wymagania wobec technologii gromadzenia, przetwarzania i udostępniania dużych kolekcji danych gospodarczych. Niewystarczające okazują się powszechnie stosowane obecnie systemy baz danych oraz klasyczne dziś metodyki tworzenia zastosowań tych baz. Coraz częściej pojawiają się postulaty tworzenia nowej technologii baz danych, określanej mianem technologii postrelacyjnych baz danych albo aktywnych baz danych. Nowa technologia tworzona ma być w oparciu o propozycje programowania obiektowego, tradycyjnej technologii baz danych oraz metod sztucznej inteligencji, zwłaszcza technologii reguł produkcji. Pojawiają się koncepcje integracji obiektowych, dedukcyjnych, czasowych i regułowych baz danych w systemy zwane systemami aktywnych baz danych.

Opracowanie stanowi wprowadzenie w problematykę badań prowadzonych nad konstruowaniem i wykorzystaniem aktywnych baz danych w inteligentnych systemach wspomagania decyzji.

1. Wprowadzenie

Inteligentne systemy wspomagania zarządzania działalnością gospodarczą, wykraczające poza standardowe przetwarzanie prostych transakcji, stanowią nowe wyzwanie dla informatyków - konstruktorów tych systemów. Usiłowaniu wspomagania (z pomocą tej klasy systemów) zarządzania na szczeblu strategicznym towarzyszyć musi dążenie do tworzenia formalizmów modelowania konceptualnego i narzędzi zwiększających funkcjonalność systemów baz danych. Bazy danych stanowią dziś zasadniczy element tworzonych aplikacji, a także podstawowe narzędzie budowy tych systemów.

Zwiększenie funkcjonalności systemów baz danych wynika z konieczności stworzenia możliwości bardziej wiernego odwzorowywania rzeczywistości gospodarczej, w sposób możliwie najbardziej dostosowany do widzenia świata ekonomii oczyma ekonomisty. Unikać należy rozwiązań ujmujących w modelu konceptualnym konstrukcje wynikające z potrzeb technologii organizacji i przetwarzania danych. Z tego punktu widzenia klasyczne relacyjne systemy baz danych stanowią poważne ograniczenie dla tego typu zastosowań.

Z dotychczasowych doświadczeń wynika, że modelowanie zarządzania dla potrzeb budowy systemów informatycznych wspierających kierownictwo naczelne organizacji wymaga wzbogacenia modelowania konceptualnego o narzędzia umożliwiające:

- reprezentację w bazie danych bardziej złożonych obiektów niż udostępniają to obecnie relacyjne bazy danych,
- bezpośrednią, jawną reprezentację w bazie danych reguł /wiedzy/ opisujących: prawidłowości /zasady/ rządzące światem biznesu /ang. business rules/, uprawnienia do posługiwania się danymi, poprawność danych, wyprowadzanie nowych danych,

tworzenie wersji procedur itp.
- modelowanie i jawną reprezentację w bazie danych zjawisk przebiegających w czasie,
- wnioskowanie /dedukcję/ na podstawie wiedzy dostępnej systemowi bazy danych, w tym również wnioskowania czasowego.

Pojawiające się w literaturze projekty baz danych spełniają wybrane postulaty. Bazy takie są określane mianem obiektowych baz danych, dedukcyjnych baz danych, postrelacyjnych baz danych, czasowych /temporalnych/ baz danych, aktywnych baz danych, itp. Wnikliwa analiza proponowanych rozwiązań pozwala dostrzec dużą zbieżność rozważań, możliwą do integracji różnorodność ujęć.

Ostatni z przytoczonych terminów - aktywne bazy danych - wiąże się z systemami baz danych, które są wyposażone w mechanizm wyzwalania pewnych działań aplikacji w sposób automatyczny na podstawie reguł, opisujących niezbędne reakcje systemu. W opracowaniu przyjęto rozszerzoną interpretację tego terminu, obejmując nim bazy, które oprócz mechanizmów przetwarzania reguł dysponują mechanizmami pracy z obiektami oraz mechanizmami reprezentacji czasu i wnioskowania czasowego. Takie ujęcie systemu bazy danych wynika z potrzeby wyposażenia konstruktora aplikacji bazodanowych w wymienione mechanizmy w jednym systemie. Potrzeba taka powstaje m.in. w sytuacji gdy modeluje się rzeczywistość gospodarczą dla potrzeb inteligentnych systemów wspomagania decyzji /zarządzania/.

2. Reguły w bazach danych

Podstawową korzyścią, jaką daje jawna reprezentacja reguł w bazie danych jest oddzielenie postępowania decydenta i zachowania się modelowanego systemu od jego implementacji w programach aplikacyjnych. Dzięki temu zmiany zasad działania w modelowanej dziedzinie mogą być łatwiej wprowadzone do systemu informatycznego niż wtedy, gdy reguły te są zaimplementowane w programie.

Dla potrzeb modelowania dziedzin ekonomicznych użyteczne jest wyróżnienie kilku głównych kategorii reguł /por.[1],[2] /:

- generowania wersji programów /procedur/,
- kontroli uprawnień dostępu,
- ograniczeń /więzów/ integralności /ang. constraints rules/,
- wywodu /ang. derivation rules/,
- działania /ang. action rules/.

Reguły generowania wersji programów dostępne są w niektórych systemach baz danych np. jako reguły preprocesora. Pozwala to na przechowywanie jednej spójnej aplikacji z wariantami tych rozwiązań, które są specyficzne dla danej wersji aplikacji /np. wersji demonstracyjnej i komercyjnej, albo wersji dla poszczególnych typów firm/.

Reguły kontroli dostępu do danych odgrywają zasadniczą rolę w systemach wielodostępnych, zwłaszcza systemach kładących duży nacisk na bezpieczeństwo i autoryzację dostępu do danych. Zapis reguł dostępu w programach, zwłaszcza w systemach typu klient - serwer - jest kłopotliwe i pracochłonne w toku modyfikacji aplikacji.

Reguły więzów integralności są stosowane w modelach baz danych i zazwyczaj są one reprezentowane nie w bazach lecz w programach aplikacyjnych. Występować mogą w dwu postaciach: w postaci statycznych oraz w postaci dynamicznych więzów integralności /ang. transition constraint rules/. Przykładem reguły statycznej jest zdanie: "Pełny etat pracownika zatrudnionego w produkcji wynosi zawsze 42 godziny". Reguła dynamiczna może być stwierdzeniem, że np. "wynagrodzenie pracownika nie może być mniejsze niż płaca minimalna" albo

wyrażeniem, stwierdzającym, że "stan kasy na koniec dnia musi być równy stan_poprzedni + przychody - wydatki".

Warto dodać, że w wielu systemach informatycznych istnieje potrzeba definiowania reguł na różnych poziomach, np. reguł i metareguł więzów integralności. Warto podkreślić, że rola reguł integralności wzrasta w systemach baz danych typu klient-serwer.

Reguły wywodu są wyrażeniami, które pozwalają wyprowadzać nowe elementy modelu z dostępnych w bazie danych. Niezbędne jest by dla każdego wyprowadzenia nowego elementu modelu stosować jedną regułę. Podobnie jak wśród reguł więzów integralności można wyróżnić reguły wywodu statyczne i dynamiczne. Przykładem statycznej reguły wywodu jest zdanie "dostawca jest najtańszym dostawcą danego towaru jeśli oferuje swój towar po najniższej cenie". Przykładem reguły dynamicznej jest stwierdzenie: "klient jest najlepszym klientem w tym miesiącu jeśli suma złożonych przez niego zamówień jest maksymalną w zbiorze zamówień".

Reguły działania związane są z wywoływaniem procedur. W szczególności reguła działania może wyrażać warunek, który musi być spełniony by procedura mogła być użyta. Przykładem reguły działania może być zdanie: "Gdy zapas towarów spada poniżej poziomu minimalnego dla tego produktu wykonaj procedurę zamówienia". Reprezentacja tego typu stwierdzeń w bazie danych, a nie w programie nabiera szczególnego znaczenia w systemach, których działanie musi być modyfikowane w trakcie użytkowania lub użytkownik sam chciałby dokonywać takich modyfikacji.

3. Obiekty w systemach baz danych

W każdej dziedzinie przedmiotowej mamy do czynienia z modelowaniem i zarządzaniem mniej lub bardziej złożonymi obiektami. Obiekty proste nie są dekomponowane na inne obiekty, co jest równoznaczne z tym, że ich istnienie nie zależy od innych obiektów. Obiekty złożone składają się z obiektów innego typu, ich istnienie zależy więc od istnienia obiektów składowych. Powiązanie takie jest modelowane za pomocą powiązań IS_PART_OF. Dość często zachodzi potrzeba wyodrębniania klas obiektów oraz tworzenią hierarchii klas. Przykładem takich hierarchii są systemy klasyfikacyjne, np. kosztów czy też np. schematy konstrukcyjne wyrobów.

W relacyjnych bazach danych obiekty złożone /a niekiedy i proste: np. faktura wielopozycyjna/ są dekomponowane i przechowywane w kilku tablicach po to, by sprostać wymaganiom nakładanym przez normalizację. Dekompozycja ta nie jest bynajmniej dokonywana dla realizacji potrzeb użytkownika lecz z punktu widzenia potrzeb systemu bazy danych /łatwości dodawania, usuwania i aktualizacji danych/. Rozszerzenia modeli relacyjnych idą w dwu kierunkach: dodaje się nowe typy atrybutów oraz rezygnuje z ograniczeń pierwszej postaci normalnej. W obu przypadkach jednakże modelowanie złożonych obiektów jest podejmowane bardziej z perspektywy baz danych niż użytkownika. Ponadto podejmuje się prace nad systemami baz danych nierelacyjnymi ukierunkowanymi na reprezentację obiektów /obiektowe bazy danych/.

4. Czas w bazach danych

W wielu zastosowaniach trudno wyobrazić sobie wierny obraz rzeczywistości bez przedstawienia historii działań, które doprowadziły do aktualnego stanu oraz świadomości możliwych, przyszłych kierunków zmian. Migawkowy obraz rzeczywistości, ukazywany przez stosowane obecnie systemy baz danych, może służyć jedynie wspomaganiu decyzji na szczeblu operacyjnym. Głębsze zrozumienie zasad współdziałania elementów systemu i ustalenie

podstaw do podejmowania właściwych decyzji dotyczących różnych horyzontów czasowych jest możliwe tylko w oparciu o dane historyczne, pozwalające na analizę czasową trendów rozwoju, opis historii poszczególnych obiektów i badanie zmian ich powiązań między sobą.
Integralną częścią takiego systemu powinna być baza danych zdolna do przechowywania i manipulacji danymi o charakterze czasowym. Bazy, w których czas jest traktowany jako dodatkowy, nieodłączny atrybut modelowanej rzeczywistości i konsekwentnie ujmowany w modelach danych oraz odpowiednich systemach zarządzania danymi określane są jako czasowe bazy danych. Czasowe modele danych powinny być zdolne do odzwierciedlania dynamicznej natury zdarzeń. Aby było możliwe wykorzystanie takich modeli spełnione muszą być trzy podstawowe wymagania[6][7]:

- Można uchwycić i interpretować czasowy charakter danych.
- Należy dane zapisywać tak, by zachować ich czasowy kontekst.
- Należy dysponować mechanizmem odzyskiwania danych o charakterze czasowym, bądź to przez bezpośrednie pytania, bądź przez mechanizny indukcyjno-dedukcyjne.

Fakt uwzględnienia czasu w bazie danych niesie ze sobą konieczność kontroli oprócz tradycyjnych powiązań między danymi także powiązań określających zależności czasowe. Pojawienie się nowych danych w bazie może być związane czasowo z:

- wartościami danych już występujących,
- naturalnym upływem czasu,
- czasem zapisu do bazy innych danych.

Odpowiednio powinny być zmodyfikowane reguły opisujące więzy integralności takiej bazy. Dotyczą one funkcji przejścia z jednego stanu bazy do drugiego, powiązań pomiędzy zdarzeniami i stanami oraz samych zdarzeń. Aby wyrazić te ograniczenia do standardowych mechanizmów zarządzania bazą danych wprowadzić należy operatory czasowe, które pozwolą odnieść reguły postępowania do przeszłości i przyszłości.

5. Reguły czasowe w ineligentnych systemach informatycznych

Zarządzanie danymi czasowymi w aktywnych bazach danych wymaga określenia takich reguł, które uwzględniałyby ich szczególny charakter. Konstrukcja reguł wymaga zdefiniowania pojęć określających zależności czasowe takie jak: teraz, przed, po, nie później, jednocześnie, razem rozpoczynając, następnie itp., tak aby bieżące zarządzanie bazą danych pozwalało uwzględniać zależności czasowe między zdarzeniami (transakcjami). Konieczne jest również wprowadzenie odpowiedniego systemu kalendarza wraz ze zdefiniowanymi metrykami pozwalającymi na obliczanie "odległości" w czasie i porównywanie czasu wystąpienia zdarzeń.
Ogólnie regułę czasową można przedstawić jako klauzulę:

$$B \leftarrow A_1 \wedge A_2 \wedge \ldots \wedge A_n$$

gdzie co najmniej jeden z warunków A_i jest warunkiem o charakterze czasowym, a B określa działanie lub obowiązujące ograniczenie. W skrajnym przypadku wszystkie warunki mogą określać zależności czasowe[8]. Regułę taką można również przedstawić w formie zdania:

[[WHEN warunek_czasowy] [IF warunek] THEN] wyrazenie logiczne

Przykładowo jeśli zdefiniowana jest relacja

R(nazwisko, płaca, data) i ograniczenie mówiące, że płaca minimalna z dniem 01.02.1994 została ustalona na 2000000 to zapisać je można:

```
R(płaca)>= 2000000 data<01.02.94, teraz)
lub jako:
 IF insert_płaca WHEN data>01.02.94 and
   data<=dziś, THEN płaca >=2 000 000.
```

Reguły te można określić jako reguły bierne-strzegące poprawności wskazanych wartości. Natomiast z punktu współdziałania z użytkownikiem coraz większe zastosowanie powinny mieć reguły wyzwalające pewne działania. Format takiej reguły w syntaktyce zbliżonej do języka PROLOG wygląda następująco:

```
rule(Nazwa_reguły, Priorytet )
        event(Opis_zdarzenia),
        condition(Lista_warunków),
        action(Opis_działania).
```

Opis_zadarzenia określa w jakich przypadkach reguła ma zastosowanie. Lista_warunków jest zbiorem wyrażeń logicznych wbudowanych na stałe w strukturę bazy lub definiowanych przez użytkownika. Jeżeli warunki te są spełnione wykonywane są działania określone w Opisie_działań. Zapis przykładowego ograniczenia byłby następujący:

```
rule(Płaca,1)
      event(insert,R(płaca)),
      condition(   R(płaca)<2000000,
                   R(data)<01.02.94, teraz),
      action(komunikat, abort).
```

Ponieważ reguły również mogą zmieniać się w czasie i mogą być definiowane także przez użytkownika, a więc mogą być w określonym czasie zmieniane lub poprawiane, powinien zostać określony zbiór meta-reguł niezależnych czasowo i niedostępnych dla użytkownika określający zasady zmiany i manipulacji samymi regułami./ang. meta static constrains rules/. Problem czasu pojawia się również wtedy, gdy reguły zmieniały się w czasie i istotna staje się historia zmian samych reguł oraz wpływ tego faktu na odpowiedź o historię obiektów w systemie.

Ze względu na charakter zastosowań reguł czasowych można podzielić je na trzy podstawowe grupy [1]:

- Podstawowe - opisujące warunki dopisywania do bazy nowych explicite podanych faktów (transakcje).
- Dedukcyjne - działające na istniejących historiach obiektów i czasie ustalające położenie historii w czasie, wnioskujące o zależnościach czasowych obiektów nie zapisanych wprost w bazie.
- Indukcyjne - aproksymujące wartości nie zapisane wprost w bazie np. metoda najmniejszych kwadratów lub netoda regresji.

Podział ten nawiązuje do opisanych już reguł kontroli integralności oraz reguł wywodu i działania.

Szczególny charakter czasowych baz danych powoduje, że jedyną dostępną dla użytkownika operacją jest dopisanie nowego wystąpienia do bazy. Możliwość rzeczywistej (fizycznej) zmiany wartości atrybutów pozostawia się do wyłącznej dyspozycji administratora[8].

Użycie reguł czasowych wymaga mechanizmów opartych o nowe techniki - przede wszystkim logiki temporalne i odpowiednie modele danych.

6. Obiekty, czas i reguły w inteligentnych systemach podejmowania decyzji

Przedmiotem zainteresowania decydenta są określone sytuacje, przedmioty lub obiekty, które są oceniane w aspekcie realizacji zaplanowanych celów. Z punktu widzenia projektanta systemu zainteresowanie koncentruje się na różnorodnych obiektach

/sytuacjach, przedmiotach, organizacjach, itp./, które istnieją i funkcjonują w określonym czasie, które zmieniają się i mogą zmieniać się w przyszłości. Każdy z obiektów ma określone zasady istnienia, zmian, łączenia się w klasy /np. sytuacji/ i współdziałania, które wygodnie jest opisywać i przechowywać w bazie danych w postaci reguł: więzów, wywodu, działania. Reguły mogą więc być związane zarówno z czasem jak i z obiektami i klasami obiektów stanowiącymi przedmiot zainteresowania modelujących i użytkowników modeli.
Wiążąc reguły z obiektami można wyróżnić następujące typy reguł:

- reguły działające na obiektach klasy,
- reguły działające na dziedzinie atrybutów wystąpień danej klasy,
- reguły właściwe dla działań na powiązaniach klas.

Wykorzystanie aktywnych baz danych oraz związanej z nią metodyki tworzenia systemów informatycznych, modelujących dynamikę rzeczywistości w postaci bazy danych /bazy wiedzy/ może stać się znaczącym stymulatorem dla tworzenia nowej klasy systemów informatycznych zarządzania, pozwalających skuteczniej niż dotychczas wspierać rozwiązywanie złożonych zadań wiążących się ze strategicznym szczeblem zarządzania organizacją gospodarczą.

ZAKOŃCZENIE

Teoria i technologia aktywnych systemów baz danych jest dopiero w stadium dojrzewania. Prace nad tym zagadnieniem podejmowane są przez wiele instututów i firm informatycznych. Najbardziej rozwiniętym systemem komercyjnym, dostępnym na naszym rynku jest INGRES oraz ORACLE [3], [4]. Udostępniają one pewne mechaniźmy zarządzania regułami. Reguły np. w systemie INGRES są definiowane za pomocą nowego zlecenia języka SQL - CREATE RULE [ ] i mogą być udostępniane przez różne reguły działania. Przedsięwzięcia prowadzone na Zachodzie dotyczą również obiektowych baz danych [np. Ariel, HiPAC,Ode, Sybase]. Prace zmierzają m.in do:

- stworzenia narzędzi wspierających tworzenie aplikacji (CASE),
- zwiększenia mocy ekspresji reguł,
- większej integracji metod sztuczej inteligencji z systemami zarządzania bazami danych,
- poprawy algorytmów stosowanych w systemach zarzadzania bazami danych.
- wykorzystania współbieżności.

Zwiększenie funkcjonalności baz danych dzięki rozbudowie mechanizmów operowania regułami, obiektami i czasem stwarza dobre podstawy do budowy inteligentnych systemów informatycznych zarządzania, wspierających monitorowanie, diagnozowanie i prognozowanie sytuacji firmy. Badania nad tymi zastosowaniami są niewątpliwie warte kontynuacji.

Literatura:

[1] B. Theudoulidis, P. Loucopoulos, V. Kopanas: A Rule-Oriented Formalism for Active Temporal Databases. Next Generation CASE Tools. IOS Press, 1992, pp. 144-167.
[2] E. Hansosn, J.Windom: An Overview of production rules in database systems. The Knowledge Engineering Review. Vol 8, No 2, 1993, pp. 121-143.
[3] J. Stochlak: Relacyjna baza danych INGRES. Architektura i narzędzia tworzenia systemów aplikacyjnych. w [5]
[4] W. Iszkowski: Relacyjna baza danych ORACLE. w [5]
[5] Systemy relacyjnych baz danych: INFORMIX, INGRES, ORACLE, PROGRESS. Materiały trzeciej i czwartej

Górskiej Szkoły PTI. PTI, Katowice 1991-92.

[6] R. Snodgrass, I.Ahn: Temporal Databases. Computer 19, Wrzesień, 1986, s. 35-42

[7] R. Maddison, S. Stanczyk: Time in information systems and its impact on modelling processes and data. Information and software technology. Vol 30, No 1, Jan/Feb 1988.

[8] T. Abbod, K. Brown, H. Noble: Providing Time-Related Constraints for Conventional Database Systems. Proceedings of the 13th VLDB Conference, Brighton 1987.

Wiesław Byrski
Telekomunikacja Polska S.A. Kraków

Jak naliczane są należności za rozmowy telefoniczne w TP S.A.

1. Wstęp

W artykule opisano stosowane w TP S.A. metody rejestrowania rozmów prowadzonych przez abonentów i system wystawiania rachunków telefonicznych. Metody rozliczania rozmów związane są ze środkami technicznymi, a w szczególności rodzajem centrali telefonicznych.

Problem rozliczania zrealizowanych usług należy do sfery obsługi klienta. Poziom obsługi klientów jest w TP S.A na niezadowalającym poziomie. Jego podniesienie jest jednym z celów, jakie TP S.A postawiła sobie na najbliższe lata. Część problemów związanych z tym zagadnieniem można rozwiązać w sferze technicznej, jednak spora ich ilość wykracza poza nią i znajduje się w sferze organizacyjno - prawnej. O ile problemy techniczne są uniwersalne i mogą być rozwiązane według sprawdzonych w świecie wzorów, o tyle rozwiązań problemów organizacyjno - prawnych dopiero się szuka /BAB, KAM, WIER, ZAL/. Wiążą się one z określeniem czym właściwie ma się zajmować TP S.A? Pozycja monopolisty ma swoje dobre strony, ale ma również wady zmuszając monopolistę do zajmowania się problemami dla niego niewygodnymi. W szczególności trudno jest określić jakie nakłady przeznaczyć na poprawę tzw. ogólnego obrazu firmy. Nie mając konkurencji pojawia się skłonność do bagatelizowania wszystkiego, co nie jest związane z techniczną stroną działalności - niezadowolony klient nawet jak się obrazi to i tak nie ma wyboru. Krytykując różne aspekty pracy TP S.A. trzeba pamiętać o tym, że każdą działalność można racjonalnie wyjaśnić z wyjątkiem założeń - założenia są dla przedsiębiorstwa zewnętrzne, a te dopiero się tworzy /BAB, KAM/.

W artykule chciałbym przedstawić sposób w jaki obecnie TP S.A. rozlicza się ze swoimi klientami. Proces ten składa się z trzech faz :

1. naliczanie należności za zrealizowane połączenia;
2. wystawianie rachunku - faktury;
3. rozliczania dokonanych wpłat.

2. Usługi telekomunikacyjne o charakterze powszechnym wykonywane przez TP S.A.

Jednym z zadań TP S.A jest świadczenie usług telekomunikacyjnych o charakterze powszechnym. Stwierdzenie "o charakterze powszechnym" jest założeniem silnym. Wynika z niego, że każdy klient TP S.A. musi mieć możliwość skorzystania z oferowanej usługi, nie może być wyjątków.

Paragraf 4 Regulaminu wykonywania usług /REG/ określa jakie usługi telekomunikacyjne o charakterze powszechnym wykonuje TP S.A.:

1. w ruchu telefonicznym :
   a) połączenia telefoniczne i radiotelefoniczne,
   b) abonament telefoniczny i radiotelefoniczny
2. w ruchu telegraficznym :
   a) telegramy, radiotelegramy,
   b) połączenia teleksowe i radioteleksowe,
   c) abonament teleksowy.

Połączenia telefoniczne i radiotelefoniczne oraz telegramy i połączenia teleksowe

realizowane są w placówkach telekomunikacyjnych lub urzędach pocztowych. Usługi po zrealizowaniu opłacane są na bieżąco wg. określonej taryfy i tym problemem nie będziemy się zajmować.

W dalszym ciągu zajmiemy się problemem rozliczania abonenta tj. posiadacza zainstalowanego przez TP S.A. aparatu telefonicznego. Analogicznie, z pewnymi modyfikacjami, rozlicza się posiadaczy aparatów teleksowych. Wspomniany już Regulamin określa co wchodzi w skład opłat za korzystanie z abonamentu (§ 109):

1) jednorazowa opłata z tytułu
   a) uzyskania dostępu do sieci telekomunikacyjne TP S.A. (...)
   b) instalacji stacji telefonicznej lub radiotelefonicznej,
2) miesięczna opłata abonamentowa,
3) opłaty za połączenia telefoniczne i świadczenia dodatkowe do tych połączeń
4) opłaty za dodatkowe czynności (...)

Na opłaty za korzystanie z abonamentu składają się ponadto:

1) w sieci KOMERTEL - miesięczna opłata dodatkowa za zastosowanie międzycentralowego łącza telekomunikacyjnego,
2) w sieci radiokomunikacji ruchomej lądowej - opłaty za wyszukanie i wywołanie stacji radiotelefonicznej

Inne punkty regulaminu określają szczegółowo terminy i sposoby płatności rachunku, odsetki za zwłokę i sposób reklamowania wysokości rachunku. Pewne kłopoty obu stron powoduje określenie terminu płatności na "15 dzień miesiąca następującego po miesiącu, w którym rachunek został wystawiony". Ze strony TP S.A. lub poczty jako dostarczyciela rachunku czasami występują opóźnienia, a z kolei niektórzy płatnicy nie zauważyli, że zniknął monopol państwowy banków. Termin płatności oznacza, że na konto TP S.A najpóźniej w dniu płatności winny wpłynąć pieniądze płatnika. Jeżeli podpisana została odpowiednia umowa z innym bankiem, tzn. TP S.A ma w nim swoje konto, to data zapłacenia rachunku telefonicznego jest równa dacie wpłaty. Ale takiej umowy nie zawarto z wszystkimi bankami, a w szczególności różne ajencje nie podpisywały takiego porozumienia. Właściwa data zapłaty na rachunku nie oznacza zatem automatycznie dochowania umowy, warto o tym wiedzieć.

3. Struktura taryf.

Usługę po jej wykonaniu trzeba umieć rozliczyć. W tym celu należy określić taryfę, według której rozliczymy usługę i mieć, oczywiście, techniczne możliwości jej rozliczenia.

Opłata taryfowa za każdą pozycję usługi powinna pokrywać koszty tej usługi, natomiast usługi o kosztach niższych od kosztów własnych muszą być subwencjonowane przez inne, jeśli przedsiębiorstwo telefoniczne w swojej ogólnej działalności miałoby nie być narażone na straty.

Każdy, kto zetknął się z problemem ustalenia taryfy za oferowane usługi zdaje sobie sprawę, że nie jest łatwe właściwe ustalenie ceny. Najłatwiej jest zrobić to gdy taka usługa już jest przez kogoś świadczona. Liczymy wtedy, czy my możemy zrobić to taniej lub też ustalamy jej wysokość na takim samym poziomie jak konkurent. Można również, licząc na opanowanie rynku ustalić cenę na niskim poziomie w celu wykluczenia konkurencji, a potem ją podnieść do poziomu nas satysfakcjonującego (i rekompensującego poprzednie jej zaniżenie). Gorzej gdy musimy wymyśleć cenę na coś zupełnie nowego, czego jeszcze na rynku nie było. Zwykle nie wiadomo wtedy ani jak wyglądać będzie popyt na tą nową usługę ani ile wynosić będą nasze

koszty, ani nawet czy całe przedsięwzięcie będzie się opłacało. W takiej sytuacji są np. usługi transmisji danych, jakie zaczynają się pojawiać na naszym rynku. Powszechnie krytykowana jest pierwsza publiczna sieć transmisji danych POLPAK, za, według powszechnego odczucia, zbyt wysokie ceny. W podobnej sytuacji są projektanci sieci transmisji danych typu MAN. Koszty realizacji projektu są stosunkowo duże, popyt jest nieznany, w dodatku postęp techniczny powoduje, że już w trakcie realizacji projektu należy spodziewać się dodatkowych nakładów na przebudowę systemu.

Struktura taryf odzwierciedla politykę cenową operatora telekomunikacyjnego. Kilka składników tej polityki można wyliczyć:

1) Per saldo wpływy muszą przewyższać koszty w określonej perspektywie czasu,
2) Im więcej abonentów tym większy ruch i większe wpływy,
3) Im niższa cena tym więcej abonentów,

ale

4) żeby podłączać nowych abonentów trzeba inwestować,
5) inwestycje zaczynają przynosić dochody dopiero po pewnym czasie.

itd.

Można do tej listy dodać wiele innych składników mogących określać np. preferencje dla określonych grup abonentów (np. biznesu, abonentów wiejskich itp), preferowania pewnych rodzajów połączeń (tańsze zagraniczne niż lokalne lub odwrotnie), niższych taryf w pewnych porach dnia, ustalenia wyższej ceny za połączenie, a niższej za czas jego trwania (do rezygnacji za liczenie czasu trwania rozmowy) itd. /KUB/.

Wydawać by się mogło, że można zastosować znane metody modelowania i symulacji dla sprawdzenia wpływu struktury cen na optymalne tempo rozwoju telekomunikacji w potrzebnych kierunkach np. w celu określenia polityki inwestycyjnej. Nie jest to takie proste. Do takich modeli potrzeba rzetelnych i pełnych informacji, których, nie wdając się w przyczyny, brakuje lub są niewiarygodne. Żeby je zdobyć potrzebne jest wprowadzenie odpowiednich systemów informatycznych zbierania informacji, co jest dopiero planowane w TP S.A.

Opłata za podłączenie abonenta do sieci telefonicznej kształtowana jest bardzo różnie na świecie. Od stosunkowo wysokich opłat jak u nas, do bezpłatnego podłączania przez niektórych operatorów w USA. Za podłączenie abonenta opłata winna równoważyć poniesione wydatki za instalacje Nie powinno więc dziwić, że tam gdzie jest rozwinięta struktura techniczna te opłaty mogą być niższe. Zrozumiałe więc staje się, że akurat te składniki w Polsce są jednymi z najwyższych w świecie, a każdym razie w stosunku do naszych dochodów.

Wysokość abonamentu miesięcznego określa średni poziom wydatków na bieżące potrzeby utrzymania instalacji technicznych w gotowości. Szacuje się je na podstawie średnich wydatków na ten cel na przestrzeni pewnego czasu.

Opłaty za dokonane połączenia można podzielić na trzy kategorie:

a. za połączenia lokalne tj. w obrębie jednej centrali lub w obrębie jednego miasta,
b. za połączenia krajowe,
c. za połączenia międzynarodowe i międzykontynentalne.

W praktyce kategoria połączenia wyznaczana jest na podstawie wybranego numeru kierunkowego. Wielkość opłat za poszczególne kategorie połączeń również jest pochodną polityki cenowej operatora /KUB/.

Polityka cenowa ulega zmianie w czasie. Np. do niedawna rozmowy łączone w obrębie wiejskiej centrali ręcznej nie były rejestrowane - opłata za nie pokrywana była z abonamentu miesięcznego. Nie tak dawno

wprowadzono również opłaty za czas trwania połączenia w rozmowach miejscowych. Te zmiany służą rozwiązaniu bieżących problemów finansowych TP S.A., a ich efekty widoczne są dopiero po pewnym czasie.

4. Sposoby rejestrowania zrealizowanych usług

Wszystkie usługi, jakie na życzenie abonenta zrealizowano są rejestrowane. Rejestracja następuje w tych miejscach sieci telekomunikacyjnej, w której usługa została zrealizowana (z pewnymi wyjątkami), w taki sposób, w jaki dopuszcza to technologia. Różny jest stopień szczegółowości i zakres treściowy rejestrowanej informacji o usłudze. Wprawdzie z tej informacji korzystają różne służby, głównym jej odbiorcą jest dział rozliczeń i obsługi klienta, dlatego niezbędne jest rejestrowanie w taki sposób, aby dało się określić koszt usługi, datę, zleceniodawcę i adresata. Usługi typu zlecenie naprawy, instalacji itp. rejestrowane są ręcznie w postaci wypisanej (gdzieniegdzie z użyciem komputera) faktury. Usługi zlecane telefonistce (np. budzenie, telegramy i inne) rejestrowane są również ręcznie. Tak samo zapisywane są realizowane połączenia przez telefonistki z central międzymiastowych - ręcznie na specjalnych drukach. Abonent podłączony do centrali z obsługą ręczną (większość ośrodków wiejskich, niektóre dzielnice miast) również ma rejestrowane ręcznie wszystkie wykonane zlecenia.

Obecnie przeważająca ilość połączeń dokonywana jest automatycznie. Sposób rejestracji tych połączeń zależny jest od rodzaju centrali, do której przyłączony jest abonent. Łatwo informatykowi wyobrazić sobie algorytm rozliczania rozmów dla central elektronicznych - wystarczy zarejestrować czas trwania połączenia z numerem wybranym i pomnożyć przez odpowiednią pozycję taryfy dla tego połączenia. Nie jest to takie proste dla central analogowych. Klasyczne centrale analogowe (elektromechaniczne) rejestrować potrafią jedynie fakt rozpoczęcia rozmowy (połączenia) i jego czas trwania przemnożony przez cenę. Dokonują tego specjalne układy zaliczające pracujące według prostego algorytmu:

1) po nawiązaniu połączenia wysyłany jest specjalny impuls taryfikacyjny, a następnie
2) na podstawie wybranych numerów kierunkowych określana jest częstość impulsów generowanych przez cały czas trwania połączenia.

Impulsy te zliczane są przez licznik, jaki posiada każdy abonent centrali telefonicznej. Odczytu liczników dokonuje się z fotografii (liczniki ustawione są w regularny sposób w szafach) wykonywanych co miesiąc. Dane te wpisuje się ręcznie do zbiorów i przekazuje do ośrodka przetwarzania. Tak więc w centralach analogowych przy tym sposobie zaliczania gubiona jest większa część informacji o zrealizowanym połączeniu, nawet nie jest znana ich ilość, tylko sumaryczna wartość. Jest to jedna z wad central analogowych na tyle dokuczliwa tak dla operatora jak i dla klientów, że od dawna próbowano zastąpić liczniki elektromechaniczne impulsów taryfikujących rejestratorami komputerowymi. Obecnie TP S.A. rozważa decyzję o powszechnym wyposażeniu istniejących central analogowych w komputerowe rejestratory rozmów. Takim urządzeniem od kilku lat stosowanym przez TP S.A. jest rejestrator BiRT /POL/. Urządzenia te obserwują linię abonenta i zapisują wszystkie zrealizowane połączenia rejestrując datę i czas rozpoczęcia połączenia, czas jego trwania i numer wybranego rozmówcy. Mogą ponadto, korzystając z analogicznego algorytmu co generatory impulsów, obliczać koszt każdej rozmowy. Central analogowych już się w TP S.A. nie instaluje, te które istnieją będą zamieniane na elektroniczne, ale ponieważ centrale analogowe mogą jeszcze przez kilka,

kilkanaście lat spełniać swoją rolę, wyposażenie ich w urządzenie tego typu jest konieczne.

Z interesującego nas w tym artykule punktu widzenia (taryfikacji i rozliczania) centrala analogowa z rejestratorem rozmów daje taki sam poziom usług co elektroniczna tzn. uzyskujemy identyczne informacje o zrealizowanych połączeniach, w taki sam komputerowo czytelny sposób.

5. Przetwarzanie danych

TP S.A. powszechnie używa opracowanego przed laty systemu SART dla komputerów ODRA. Są w użyciu wersje tego systemu na maszyny IBM i RIAD pod nazwą SERAT, KERT, istnieje również wersja na komputery PC pod nazwą mikroSart. Dokonywane są próby ewoluowania tych systemów w kierunku objęcia swoim zakresem innych systemów obsługi klienta np. informacje o numerach telefonicznych, rejestracji wniosków o instalacje i innych. Ponieważ przetwarzanie rachunków wiąże się z rejestracją wpłat, systemy te są również źródłem informacji finansowych.

Jak widać z przedstawionego przeglądu jest znaczna różnorodność dokumentów źródłowych, na których znajdują się dane niezbędne do wystawienia rachunku abonentowi. Dane te grupują się w kilka pozycji:

a) abonament - stała opłata miesięczna
b) opłaty za rozmowy
   - lokalne lub w ruchu automatycznym
   - wg. licznika lub rozliczane z danych rejestrowanych
   - zamawiane - wg specyfikacji
c) inne np. konserwacje, instalacje, naprawy, inne usługi
d) bonifikaty czyli ulgi

Aktualne dane o abonamencie pobiera się z umowy jaką TP S.A. zawiera z abonentem (możliwe są zniżki dla pewnych grup abonentów). Opłaty za rozmowy lokalne odczytywane są z liczników lub z przetworzenia taśm rejestrowych central elektronicznych lub rejestratorów w centralach analogowych. Rozmowy zamawiane np. międzymiastowe lub inne łączone przez telefonistki, są dostarczane w postaci pisanych ręcznie dokumentów i wstępnie wprowadzane na dyskietki lub taśmy. Wszystkie te dane muszą być przekształcone do formatu wymaganego przez system SART. Przetwarzanie, którego efektem jest wystawienie rachunku - faktury odbywa się pod koniec każdego miesiąca. Oznacza to, że na rachunku znajdują się opłaty za rozmowy jakie abonent przeprowadził miesiąc wcześniej, abonament natomiast jest z aktualnego miesiąca. Tak więc rachunek płatny do 15 czerwca zawiera opłatę abonamentową za miesiąc maj, a rozmowy zrealizowane w miesiącu kwietniu.

Do czasu wprowadzenia podatku VAT do rachunku dopisywało się pozycję nadpłata/niedopłata na podstawie analizy salda abonenta. Wtedy po zapłaceniu rachunku bieżącego saldo abonenta się zerowało (z dokładnością do opóźnień w księgowaniu wpłat i ewentualnych pomyłek). Po wprowadzeniu podatku VAT zarówno niedopłaty jak i nadpłaty muszą być traktowane odrębnie jako faktury korygujące VAT. Tak samo traktowane są uwzględnione reklamacje abonentów.

Zapłacone rachunki na poczcie, w bankach lub kasach TP S.A. wracają do ośrodka obliczeniowego i tam służą do uaktualnienia salda klientów. Zasadniczo wpłaty wprowadzane są ręcznie, coraz powszechniej używane są czytniki kodów kreskowych.

6. Plany na najbliższą i dalszą przyszłość

Obecny system rozliczania należności jest

przestarzały. W najbliższej perspektywie TP S.A. planuje stworzenie kilku systemów informatycznych, w których system obsługi klienta zajmie jedno z ważniejszych miejsc. Wzorem niektórych krajów planuje się scentralizowanie przetwarzania danych do wystawiania rachunku w kilku regionach, centralnego zbierania tych danych drogą teletransmisji i centralnego drukowania i rozsyłania rachunków. Celowość takiego rozwiązania nie jest powszechnie akceptowana, ma ona swoje zalety, ale również ma szereg wad. Moim zdaniem ma ona więcej wad niż zalet.

Zanim to nastąpi obecnie działające systemy będą usprawniane przede wszystkim w kierunku eliminacji pracy ręcznej.

7. Wnioski

Brak jasno określonej przez Państwo roli jednego z wielu operatorów jakim w świetle Ustawy o telekomunikacji stała się TP S.A., powoduje niemożność opracowania zarówno strategii działania przedsiębiorstwa jak i taktyki. Odbija się to na sposobie wprowadzania nowych taryf - nie są one wprowadzane w sposób wynikający z praw ekonomii, a interpretowanych ad hoc niejasnych zasad będących mieszaniną socjalnej roli operatora i ekonomicznych uwarunkowań. I, niestety, wygląda na to, że stan taki będzie długotrwały.

8. Literatura

1. Regulamin wykonywania usług telekomunikacyjnych o charakterze powszechnym przez TELEKOMUNIKACJĘ POLSKĄ S.A., Telekomunikacja Polska S.A., sierpień 1993.
2. Krajowa taryfa telekomunikacyjna.
3. H. Babis: Podstawowe problemy zarządzania finansowego telekomunikacją, materiały Krajowego sympozjum telekomunikacji 93, tom A, str. 5-16.
4. F. Kamiński: Prawo o telekomunikacji w Polsce na tle regulacji prawnych w krajach wysoko uprzemysłowionych, materiały Krajowego sympozjum telekomunikacji 93, tom B, str. 218-226.
5. J. Kubasik, J. Kleban: Zasady ustalania cen na usługi telekomunikacyjne, materiały Krajowego sympozjum telekomunikacji 93, tom B, str. 257-266.
6 J. Pol: Bilingowy rejestrator rozmów telefonicznych "BiRT II", materiały Krajowego sympozjum telekomunikacji 93, tom D, str. 322- 338.
7. M. Wierzuchowski, W. Byrski, J.P. Rojkowski: System nadzoru sieci telekomunikacyjnej w Krakowie, materiały Krajowego sympozjum telekomunikacji 93, tom D, str. 280-289.
8. J. Zalewski, J. Fałkowski : Utrzymanie i zarządzanie siecią TPSA, materiały Krajowego sympozjum telekomunikacji 93, tom A, str. 84-90.


PTI
Wydawca:
Polskie Towarzystwo Informatyczne
Oddział Górnośląski
40-014 Katowice, ul. Mariacka 6
Tel./ Fax 153-81-02
Redakcja
i przygotowanie do druku
ZM BUMAR ŁABĘDY S.A.
Centrum Informatyki
44-109 Gliwice, ul. Mechaników 9
Tel. 134-61-49, 61 Fax 134-25-14
M.Rakowska, Cz.Uramowski
Druk i oprawa
Centrum Informatyki
ZM BUMAR ŁABĘDY S.A.
44-109 Gliwice, ul. Mechaników 9
Tel. 134-61-50, 55 Fax 134-25-14
Projekt okładki: / CorelDRAW /
Mirosława Miniich