AKADEMIA OBRONY NARODOWEJ

CENTRUM INFORMATYKI

KOMPUTEROWE WSPOMAGANIE PROJEKTOWANIA SYSTEMÓW INFORMATYCZNYCH

SZAWA 2000

AKADEMIA OBRONY NARODOWEJ

CENTRUM INFORMATYKI

KOMPUTEROWE WSPOMAGANIE PROJEKTOWANIA SYSTEMÓW INFORMATYCZNYCH

płk prof. dr hab. inż., Czesław **FLANEK**
płk dr inż. Zbigniew **KLIMKIEWICZ**
mjr dr Grzegorz **KOTT**

7.4.8.0
INFORM - PRO

WARSZAWA **2000**

Pracę naukowo – badawczą pk. **INFORM – PRO** wykonał zespół pracowników naukowo – badawczych pod kierownictwem naukowym płk prof. dr hab. inż., Czesława **FLANKA** w składzie:

płk dr inż. Zbigniew **KLIMKIEWICZ**

mjr dr Grzegorz **KOTT**

Poszczególni członkowie zespołu opracowali:

- płk prof.. dr hab. inż., Czesław **FLANEK** - rozdział 5, 7, 8 oraz 10.1
- płk dr inż. Zbigniew **KLIMKIEWICZ** - rozdział 1, 3, 4, 9, 10.2, 10.3 oraz opracowanie merytoryczne pracy;
- mjr dr Grzegorz **KOTT** - rozdział 2, 6, 8.1 oraz załącznik 1

PODSTAWY FORMALNO – PRAWNE WYKONANIA PRACY NAUKOWO – BADAWCZEJ

Temat naukowo – badawczy **KOMPUTEROWE WSPOMAGANIE PROJEKTOWANIA SYSTEMÓW INFORMATYCZNYCH** kryptonim **„INFORM - PRO"** , wykonany został zgodnie z **„Planem prac naukowo – badawczych AON na rok 2000"** – pozycja **7.4.8.0.**

SPIS TREŚCI

1. SYSTEMY INFORMATYCZNE I ICH ZŁOŻONOŚĆ

Dynamiczny rozwój metod informatyki, stale poszerzające się pole zastosowań oraz rosnące wymagania stawiane przez użytkowników komputerów powodują,, że coraz częściej informatycy stają, przed zadaniem zaimplementowania (złożonych) systemów informatycznych, a nie, jak to było do niedawna, dostarczenia (mniej złożonych) programów komputerowych. Oznacza to, że dotychczas wystarczające umiejętności algorytmizacji problemów oraz posługiwania się narzędziami implementacji komputerowej (języki programowania, systemy operacyjne, edytory itp.) muszą, być uzupełnione wiedzą dotyczącą, systematycznego projektowania złożonych systemów. Dotyczy to szczególnie sytuacji, w których realizowany system jest rezultatem pracy zespołu (ze względu na rozmiar tej pracy) i jest przewidywane dostarczanie klientom kolejnych, coraz lepszych wersji (co jest celem każdej firmy komputerowej produkującej czy też wdrażającej oprogramowanie). Problem ten występuje nie tylko w wypadku produktów o charakterze komercyjnym, ale również w wypadku implementacji każdego złożonego systemu przygotowywanego przez zespoły informatyków (unikatowe systemy wykonywane na zlecenia przemysłu, banków, urzędów, czy też związane z badaniami naukowymi). Nieznajomość nowoczesnych metod projektowania takich systemów jest z pewności, jednym z najczęstszych powodów niemożności przygotowania produktu spełniającego założone wymagania, konieczności przeznaczenia w trakcie trwania projektu dodatkowych środków finansowych, opóźnień w dostarczeniu produktu. "Najłagodniejszą," konsekwencją, jest skonstruowanie systemu w miarę odpowiadającego specyfikacji, za to kompletnie niezdolnego do dalszej ewolucji.

Celem niniejszej książki jest przedstawienie kilku (spośród kilkudziesięciu) najlepiej znanych i sprawdzonych w praktyce metod analitycznych projektowania systemów informatycznych. Do metod analitycznych projektowania będziemy zaliczać te metody, które umożliwiają, zdefiniowanie formalnego (abstrakcyjnego) modelu systemu informatycznego przez zastosowanie rozbioru (analizy) tego systemu na części składowej[1].

Najbardziej znanymi tego typu metodami są, klasyczne metody projektowania strukturalnego, wywodzące się ze szkoły Yourdona (*structured approach*), oraz metody

---

[1] *Niektórzy autorzy nazywają takie metody metodami strukturalnymi, ale określenia tego w literaturze anglojęzycznej używa się w odniesieniu do klasycznej grupy metod analitycznych (metody Yourdona, DeMarco itp.) do odróżnienia ich od metod obiektowych. Nazwy metody analityczne używa w tym kontekście, w klasycznej w tej dziedzinie monografii (31), Turski w celu odróżnienia takich metod od metod syntetycznych.*

projektowania obiektowego (*object-oriented approach*). Mam nadzieję, że zaznajomienie się z nimi pomoże Czytelnikowi uniknąć wymienionych negatywnych konsekwencji.

Rozpocznijmy od wprowadzenia podstawowych pojęć, które umożliwi, określenie zakresu naszych rozważań.

Algorytmem nazywamy opis czynności, jakie należy wykonać w stosunku do określonych obiektów, aby osiągnąć pewien założony cel.

Program zdefiniujemy jako realizację pewnego algorytmu w konkretnym języku programowania, umożliwiając, wykonanie tego algorytmu za pomoc, komputera.

System oprogramowania określimy jako zbiór programów współpracujących w celu wykonania ogólnego, złożonego oraz wieloaspektowego zadania i realizujących dobrze zdefiniowane operacje (czynności, obliczenia) odnoszące się do poszczególnych aspektów zadania ogólnego.

Typowym przykładem systemów oprogramowania są systemy operacyjne, zintegrowane pakiety korzystania z baz danych (składające się z programów: zarządzania bazą, danych, kalkulacji tabelowej, graficznej prezentacji danych itd.), czy diagnostyczno-sterujące systemy oprogramowania czasu rzeczywistego, stosowane np. w przemyśle (składające się z programów filtrujących monitorowane dane, przetwarzających wstępnie monitorowaną informację, rozpoznających sytuacje alarmowe, wysyłających sygnały sterujące itp.).

Przez ***system informatyczny*** będziemy rozumieć system oprogramowania osadzony na pewnej konfiguracji sprzętowej, której składniki umożliwiają, elementom oprogramowania tego systemu (efektywną,) realizację poszczególnych zadań oraz wzajemną. komunikację i działający w określonym środowisku (np. ludzie - operatorzy systemu, urządzenia, maszyny kontrolowane przez system itp.) zgodnie z dobrze określonymi regułami (np. reguły reakcji systemu na polecenia operatora).

Systemami informatycznymi są, na przykład: sieci komputerowe (oprogramowanie wraz z węzłami sieci (komputerami), okablowaniem, modułami umożliwiającymi transmisję danych (most - *bridge*, ruter - *router*) itp.), systemy monitorujące /diagnostyczne procesy przemysłowe (oprogramowanie wraz z urządzeniami sensorycznymi (czujniki), okablowaniem, specjalizowanymi urządzeniami elektronicznymi wstępnie przetwarzającymi dane, komputerami, terminalami itp..).

Metody analityczne omawiane w niniejszej książce są, często postrzegane jako metody analizy i projektowania systemów oprogramowania. Oznacza to, że projektant nie wykorzystuje w pełni modeli analitycznych w części, w której wymagaj, one definicji

środowiska oraz określenia struktury sprzętowej i sposobu alokacji elementów oprogramowania do składników sprzętowych[2]. W szczególnych wypadkach, np. konstrukcja niewielkiego systemu informatycznego przeznaczonego do pracy na jednym komputerze, czy też systemu, który nie komunikuje się z żadnymi elementami środowiska z wyjątkiem jednego użytkownika, podejście takie może być usprawiedliwione. Gdy projektowany system ma pracować w środowisku rozproszonym, tzn. w sieci komputerowej lub komputerze wieloprocesorowym, ograniczanie metod analitycznych tylko do poziomu oprogramowania jest nieuzasadnione.

Podstawowym powodem problemów występujących w trakcie konstrukcji systemów informatycznych, systemów oprogramowania (a nawet dużych programów) jest ich złożoność. Po przekroczeniu pewnego progu złożoności projektant (programista) przestaje "panować" nad przygotowywanym produktem, ponieważ całościowe zrozumienie funkcjonowania systemu w różnych jego aspektach i na dowolnym poziomie szczegółowości przekracza możliwości człowieka. Rozwiązaniem tego problemu może być strukturalizacja systemu według określonych kryteriów. Jest ona główną ideą, przedstawionych tu metod, a rodzaj przyjętych kryteriów jest tym, co różni poszczególne metody. Wspólną, cechą metod analitycznych projektowania są natomiast mniej więcej te same założenia dotyczące charakteru złożoności systemów informatycznych. Na podstawie tych założeń są definiowane wspomniane kryteria. Przedstawimy teraz te założenia.

1. Złożony system możemy przedstawić w postaci struktury hierarchicznej. Oznacza to, że system możemy zdekomponować na zbiór podsystemów, z których każdy możemy zdekomponować na zbiór podpodsystemów itd., aż dojdziemy do najniższego interesującego nas poziomu szczegółowości.
2. Proces dekompozycji przedstawiony w punkcie 1 nie jest (całkowicie) arbitralny. Kryterium określające, na jakie części należy podzielić w danym kroku system (podsystem), jest oparte na obserwacji dynamiki interakcji między składowymi elementarnymi systemu. Składowe o dużej dynamice wzajemnych interakcji należy grupować w podsystemy. Dynamika interakcji między podsystemami powinna być mała.
3. Systemy informatyczne mają, tendencję do rozrastania się. Zjawisko to może mieć charakter niezamierzony (no. wskutek wzrastających wymagań użytkowników

---

[2] *Większość omawianych tu metod wymaga konstrukcji modeli zorientowanych sprzętowo : Yourdon - model środowiska Procesorowego (Processor Environment Model, PEM), Hatley-Pirbhai - model wymagań*

należy ulepszyć obecną. wersję całego systemu (rozwój jakościowy), czy też dołączyć podsystem dostarczający nową, nieprzewidzianą wcześniej funkcję) lub zamierzony (np. w metodzie prototypowania konstruktor oprogramowania zakłada, że system będzie ewoluował z prostego systemu o podstawowej funkcjonalności w stronę bardziej złożonych form wychodzących naprzeciw życzeniom użytkowników. Zamierzona ewolucja systemu jest typowym założeniem metod obiektowych.

W wypadku metod obiektowych bardzo często wprowadzamy jeszcze jedno założenie dotyczące procesu dekompozycji, a mianowicie, że systemy s~ kombinacjami podsystemów należących do niewielkiej liczby podstawowych klas podsystemów. Założenie to wydaje się jednak prawdziwe jedynie w odniesieniu do niektórych bardzo złożonych "wielopiętrowych" systemów.

Po zdefiniowaniu i krótkim scharakteryzowaniu systemów informatycznych, wyjaśnimy jeszcze, co będziemy rozumieli przez pojęcie projektomanie takich systemów. Konstrukcję systemów informatycznego możemy podzielić na pięć zasadniczych faz:

1. ***Analiza systemu***. Na tę fazę składają się:

a) sformalizowanie zbioru wymagań stawianych przed systemem,

b) zdefiniowanie funkcjonalnego modelu systemu opisującego:

- interakcje systemu ze światem zewnętrznym,
- strukturę systemu,
- przepływ danych i sterowania między elementami systemu,
- abstrakcyjne struktury danych,
- dynamikę (zachowanie się) systemu.

Produktem tej fazy jest tzw. ***model logiczny systemu***.

2. ***Projektowanie systemu***. W tej fazie na podstawie modelu (projektu logicznego) systemu projektujemy fizyczną strukturę systemu, tzn. definiujemy: elementy oprogramowania (*software)* systemu (programy, procedury, funkcje, bloki itp.) oraz interakcje między nimi (np. wywołania), a także składniki sprzętowe (hardware), a następnie określamy sposób przypisania (alokacji) elementów oprogramowania do składników sprzętowych.

---

sPrzgtowych (Hardware Requirements Model) oraz rnodel architektury sprzętowej (Hardware Architecture Model), Booch - diagramy Procesów (Process Diagrams).

3. ***Implementacja systemu.*** Faza ta polega na zakodowaniu algorytmów w konkretnym języku programowania i utworzeniu struktury systemu oprogramowania zgodnie ze specyfikacja struktury fizycznej systemu.

4. ***Instalacja i testowane systemu oraz usuwanie błędów.*** W fazie instalacji dokonujemy fizycznej alokacji elementów oprogramowania do składników sprzętowych zgodnie z projektem struktury fizycznej systemu. Następnie sprawdzamy poprawność działania systemu w środowisku użytkownika i eliminujemy błędy.

5. ***Pielęgnacja i dalszy rozwój systemu.*** Faza ta rozpoczyna się w chwili przekazania systemu użytkownikowi. Jej podstawowymi celami są: ciągła eliminacja błędów, które ujawniają się w trakcie korzystania z systemu przez użytkowników, oraz poszerzanie możliwości i polepszanie parametrów systemu zgodnie z sugestiami użytkowników.

Wymienione fazy mogą, przeplatać się wzajemnie w wielu iteracjach, w przeciwieństwie do tradycyjnego podejścia (*waterfall model*), w którym rozpoczęcie etapu (n + 1) jest uwarunkowane całkowitym zakończeniem n- tego etapu[3].

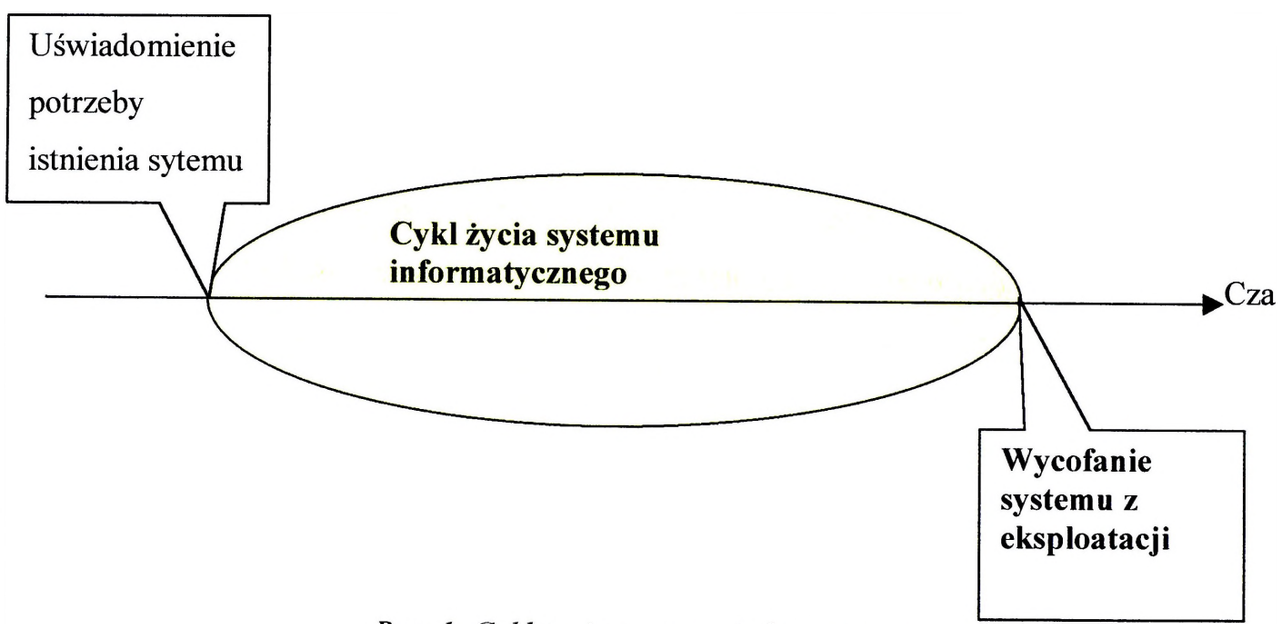
Niniejsza praca jest poświęcona pierwszym dwóm fazom konstrukcji systemów informatycznych. Użyty w jej tytule termin "projektowanie" należy rozumieć szerzej, jako obejmujący analizę systemu oraz projektowanie "właściwe", tzn. projektowanie architektury logicznej i fizycznej warstwy oprogramowania i warstwy sprzętowej systemu informatycznego.

---

[3] *Modele cyklu życia systemu przedstawiłem w rozdz. 2.*

2. CYKL ŻYCIA SYSTEMU INFORMATYCZNEGO I JEGO MODELE

Cykl życia systemu informatycznego (ang. *software life cycle*) jest to proces złożony z ciągu wzajemnie spójnych etapów pozwalających na pełne i skuteczne stworzenie a następnie użytkowanie systemów informatycznych. Zazwyczaj obejmuje on okres (rys. 1) od momentu uświadomienia (zgłoszenia przez przyszłego użytkownika) potrzeby istnienia systemu aż do momentu wycofania systemu z eksploatacji (ang. *retirement phase*).



Rys. 1. Cykl życia systemu informatycznego.

Okres ten dzieli się na następujące fazy:

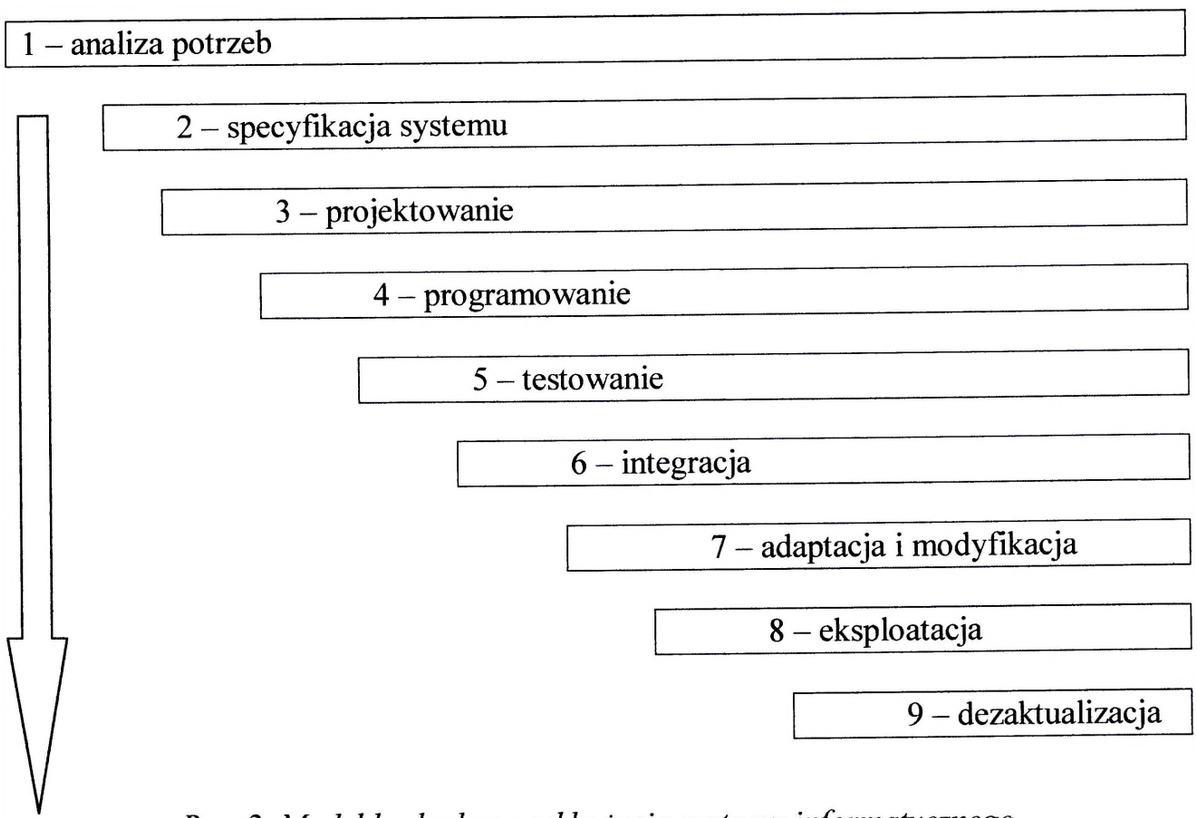
1. analiza wymagań;
2. projekt;
3. implementacja (kodowanie);
4. testowanie;
5. instalacja;
6. eksploatacja;
7. wycofanie.

Systemy informatyczne tworzy się dzisiaj według różnych modeli cyklu życia. Nadrzędną cechą jakiegokolwiek modelu cyklu życia jest jego kompletność, oznaczająca konieczność dokładnego opisania wszystkich faz niezbędnych do stworzenia lub użytkowania elementów systemu. Każda z faz cyklu życia systemu powinna być ściśle określona. Jej opis powinien zawierać specyfikację:

- wejść;
- wyjść,
- składników;
- funkcji;
- dokumentów;
- sprzężeń z innymi fazami.

Założenie to stanowi podstawę do specyfikacji wszystkich systematycznych metod realizacji przedsięwzięć informatycznych.

Najpopularniejszym, tradycyjnym modelem cyklu życia systemu, jest model kaskadowy (rys. 2). Model ten dość naturalnie odpowiada działaniom informatyka tworzącego produkt programistyczny. Model kaskadowy składa się z następujących faz:



Rys. 2. Model kaskadowy cyklu życia systemu informatycznego.

Celem fazy analizy potrzeb jest udzielenie odpowiedzi na pytanie: Co i przy jakich ograniczeniach system ma robić? Wynikiem tej fazy jest zbiór wymagań, które system ma spełnić.

Celem fazy specyfikacji systemu jest ustalenie wszystkich czynników i warunków które mogą wpłynąć na decyzje projektowe, na przebieg procesu projektowego i na realizację wymagań względem systemu. Wynikiem tej fazy jest logiczny model systemu, opisujący sposób realizacji przez system postawionych wymagań, lecz abstrahujących od szczegółów implementacyjnych.

Celem fazy projektowania jest opracowanie opisu sposobu implementacji systemu, zależnego od środowiska implementacji (stosowanego sprzętu i oprogramowania).

Celem fazy programowania (implementacji) jest skonstruowanie zaprojektowanego oprogramowania.

Celem fazy testowania jest wykrycie i usunięcie błędów w systemie oraz ocena jego niezawodności.

Celem fazy integracji jest włączenie systemu do środowiska jego działania (innych systemów).

Celem fazy adaptacji i modyfikacji jest poprawa jakości oprogramowania oraz dostosowywanie systemu do zmian wymagań lub środowiska systemu

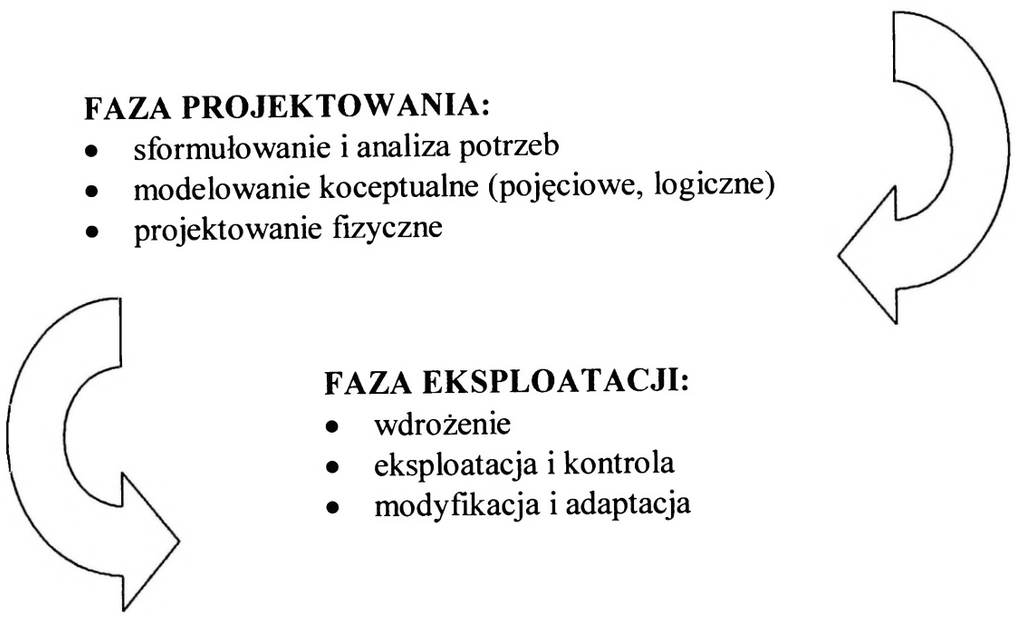
Założenia kaskadowego modelu cyklu życia są następujące:

- na początku każdego projektu istnieje stabilny zestaw potrzeb;
- potrzeby te nie zmieniają się w trakcie życia systemu.

Założenie pierwsze nie sprawdza się w większości przypadków, ponieważ użytkownicy nie mają na wstępie sprecyzowanych potrzeb, a ich specyfikacje nie są spójne. W związku z tym występują częste nieporozumienia między użytkownikami i informatykami, czego konsekwencją jest rozminięcie się funkcji faktycznie przez system realizowanych z oczekiwaniami użytkownika.

Założenie drugie zakłada niezmienność potrzeb użytkownika w trakcie życia systemu. Praktyka dowodzi, że zmiany następują już w fazie projektowania systemu. Na ogół mija długi czas, zanim użytkownik zrozumie cechy i założenia konstruowanego systemu. Także okres między specyfikacją systemu (analizą potrzeb) a testami systemu jest zbyt długi, by potrzeby użytkownika pozostały niezmienne.

Powyższe przesłanki legły u podstaw zmian i udoskonaleń tradycyjnego podejścia do tworzenia systemów informatycznych. Uogólnieniem tradycyjnego cyklu życia jest tak zwany **model Fry'ego**, stosowany w modelach morfologicznych. Model ten ma interesującą strukturę, gdyż składa się tylko z faz projektowania i eksploatacji, przy czym sama implementacja (kodowanie) ma miejsce w fazie eksploatacji. Z góry zakłada się, że system będzie zmieniał się w trakcie eksploatacji. Cykl ten ma następującą postać:



Rys. 3. Model Fry'ego cyklu życia systemu informatycznego.

W fazie analizy potrzeb następuje zebranie potrzeb informacyjnych przyszłych użytkowników (tzn. wymagań dotyczących przyszłego systemu), co ma służyć ustaleniu zgodności celów systemu z celami użytkownika. Jest to zazwyczaj najbardziej czasochłonny, najtrudniejszy i najważniejszy etap, gdyż wynika z niego większość istotnych decyzji dotyczących kolejnych etapów projektowania systemu. Efektem fazy projektowania konceptualnego, zwanego również logicznym, jest opis modelu danych i modelu procesów w systemie. Stadium projektowania fizycznego dotyczy zaprojektowania struktury zbiorów, wzorców dokumentów, technologii przetwarzania specyfikacji programowych. Dokonuje się wyboru struktury pamięci (tzw. Schemat wewnętrzny). Etap wdrożenia dotyczy stworzenia bazy danych i programów zastosowań. Na etapie modyfikacji i adaptacji następuje

udoskonalenie funkcjonowania istniejącego systemu w rezultacie pojawienia się nowych potrzeb.

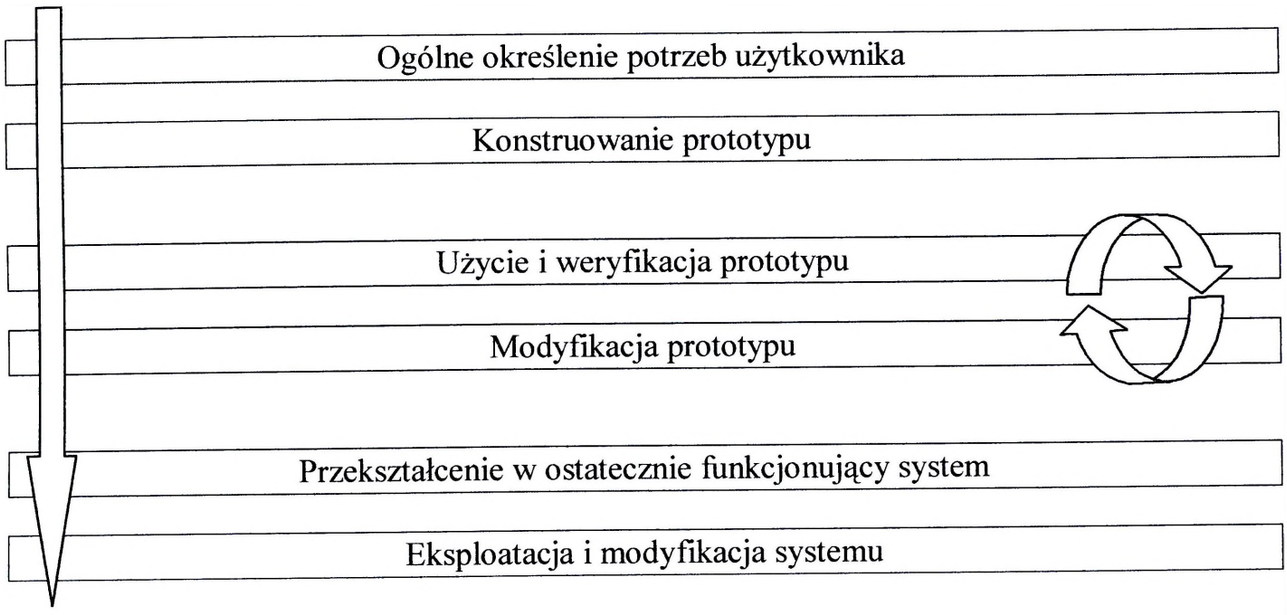
Jeżeli tworzony system informatyczny jest duży i złożony, wymagania którym musi sprostać są niejasne a użytkownik nie ma doświadczenia informatycznego i jest bierny to rozsądnym dla takiego systemu będzie przyjęcie **modelu cyklu życia z prototypem**.

Na rys. 4 przedstawiono model cyklu życia systemu z prototypem:

Prototyp jest wczesną wersją systemu, która wykazuje cechy późniejszego, działającego systemu. W omawianym modelu cyklu życia systemu prototypowanie jest iteracyjne: skonstruowany prototyp jest weryfikowany i na podstawie efektów weryfikacji modyfikowany do czasu uzyskania akceptacji użytkownika.

Konstruowanie prototypu może być:

- poziome – płytkie, niekompletne - opracowanie wszystkich lub większości funkcji systemu,
- pionowe – głębokie, kompletne - realizacja wybranych cech systemu, zwykle krytycznych dla całego projektu.



Rys. 4. Model z prototypem

Decyzja o tym kiedy prototypować zależna jest od tego jaki ma być cel prototypowania:

- wczesne prototypowanie – w fazie analizy wymagań; cel: wyjaśnienie lub sprawdzenie poprawności wymagań
- średnie prototypowanie – na etapie projektowania; cel: potwierdzenie zachowania się systemu w kluczowych obszarach lub sprawdzenie kluczowych aspektów projektu
- późne prototypowanie – na etapie realizacji lub eksploatacji; cel: badanie kluczowych parametrów operacyjnych, zwłaszcza efektywności.

Budowane mogą być dwa podstawowe rodzaje prototypów:

- prototyp odrzucalny – używany do testowania części systemu i niszczony,
- prototyp przyrostowy – przez stopniowe uszczegóławianie przekształca się w część lub całość systemu.

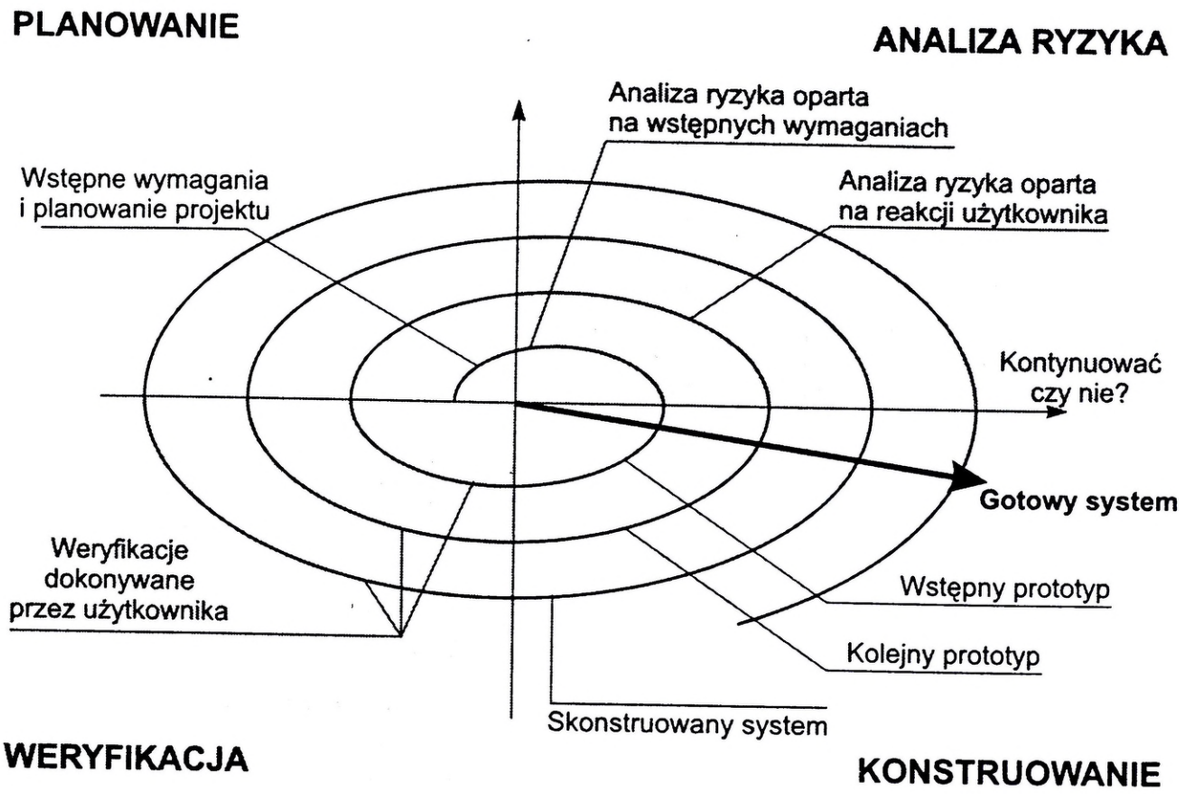
Przyjęcie w pracach nad tworzeniem systemu informatycznego modelu cyklu życia systemu z prototypem niesie ze sobą wiele korzyści. Umożliwia poprawę porozumienia z użytkownikiem, a co za tym idzie opracowanie lepszych wymagań na tworzony system.

Powoduje większe zaangażowanie w projekcie użytkowników oraz ułatwia naukę korzystania z systemu. Prototypowanie ma jednak pewne wady. Do realizacji systemu mogą być wymagane inne, nietradycyjne narzędzia i umiejętności, tym samym może być zwiększony koszt opracowania systemu. Trudniejsze staje się zarządzanie projektem. I na koniec wada podstawowa - prototypy mogą być zbyt obiecujące.

Przy tworzeniu systemu informatycznego zarówno przyszły użytkownik jak i twórca systemu ponoszą pewne ryzyko. Ryzykiem tym jest zaangażowanie sił i środków w tworzenie systemu, który nie będzie spełniał oczekiwań użytkownika, okaże się nieprzydatny. Przykładem modelu cyklu życia systemu w którym analizuje się ryzyko jego tworzenia jest **model spiralny,** przedstawiony na rys. 5. W modelu tym wyróżnia się cztery fazy tworzenia systemu:

1. fazę planowania - w której formułuje się wymagania i planuje ich realizację,
2. fazę analizy ryzyka - w której analizuje się ryzyko kontynuowania prac nad systemem,
3. fazę konstruowania - w której konstruuje się kolejne prototypy systemu,
4. fazę weryfikacji - w której użytkownik weryfikuję zgodność prototypu (systemu) z wymaganiami.

Fazy te wykonywane są cyklicznie, aż do uzyskania oczekiwanej postaci systemu.



Rys.5. Model spiralny

3. METODOLOGICZNE PODSTAWY TWORZENIA SYSTEMÓW INFORMATYCZNYCH

3.1. Zakres i składniki metodyki tworzenia systemów informatycznych

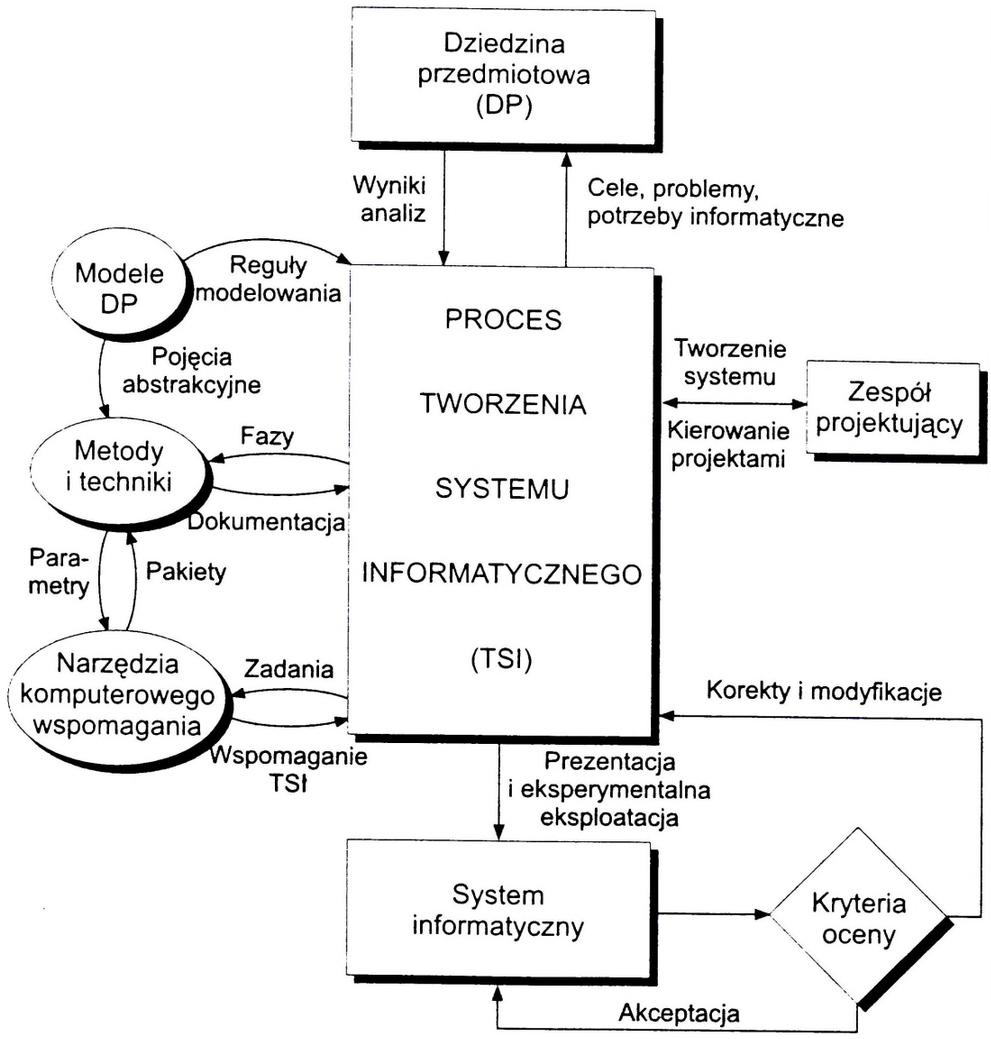
Kluczowym pojęciem w dziedzinie analizy, projektowania i użytkowania systemów jest **metodyka tworzenia systemów informatycznych (TSI).** Można ją zdefiniować jako: spójny, logicznie uporządkowany zestaw metod i procedur o charakterze technicznym i organizatorskim pozwalających zespołowi wykonawczemu realizować cykl życia systemu. Definicja ta nie precyzuje **składników metodyki TSI**. Na obecnym etapie jej rozwoju są to :

- formalizmy, modele opisu rzeczywistości - dziedziny przedmiotowej, jej statyki i dynamiki, zwane modelami konceptualnymi,
- strukturyzacja procesu TSI w postaci odpowiedniej sekwencji etapów, podetapów i poszczególnych zadań, tj. w postaci cyklu życia systemu,
- szczegółowe metody i techniki TSI - jego dokumentowania (w nawiązaniu do teoretycznych konstrukcji formalizmów), wraz z adekwatną symboliką graficzną,
- narzędzia wspomaganego komputerowo TSI, w tym ich prototypowania, określane mianem CASE (ang. *computer-aided systems engineering)* · specyfikacja wymagań merytorycznych wobec poszczególnych twórców oraz interdyscyplinarnego zespołu wykonawczego (projektowego), planującego rozwój systemu i realizującego proces TSI,
- kryteria oceny jakości projektu i systemu oraz mechanizmy jej kontroli.

Powiązania pomiędzy wymienionymi składnikami w postaci uogólnionego schematu przedstawia rys.6.

Rozpoznanie potrzeb informatycznych organizacji, założonych celów wdrożenia systemu informatycznego i występujących problemów inicjuje proces TSI realizowany i sterowany przez zespół projektowy. Analizę dziedziny przedmiotowej przeprowadza się za pomocą modeli dziedziny przedmiotowej, określonych metod i technik oraz pakietów komputerowego wspomagania TSI. Dziedzina przedmiotowa (DP), czyli badany wycinek rzeczywistości, stanowi obiekt, dla którego tworzy się system informatyczny. Przykładami

DP są: działalność organizacji gospodarczej bądź administracyjnej, gospodarka materiałowa firmy, zarządzanie kadrami, obsługa sesji egzaminacyjnej, itp..



Rys. 6. Powiązania między składnikami metodyki TSI

Podstawą opisu dziedziny przedmiotowej są odpowiednie formalizmy, modele DP, zwane modelami konceptualnymi. Pozwalają one na odzwierciedlenie zarówno statyki, jak i dynamiki dziedziny przedmiotowej. Poszczególne elementy modeli mogą być dokumentowane za pomocą odpowiednich **metod i technik**. Mają one zazwyczaj postać diagramów. Analiza ich struktury, parametrów i zasad użytkowania pozwala na dobór właściwych **narzędzi wspomagających proces TSI**, powszechnie nazywanych pakietami

CASE (ang. *computer-aided systems engineering).*

W wyniku powstaje przekazany do oceny, a następnie użytkowania w organizacji gospodarczej lub administracyjnej **system informatyczny** - tj. uporządkowany zestaw wzajemnie powiązanych składników: kadry, danych, procesów, sprzętu, oprogramowania i sieci komputerowej, współpracujących dla wykonania założonych funkcji, pozwalających na rozwiązanie występujących problemów i osiągnięcie założonych celów w danej dziedzinie przedmiotowej. Problematyka zawarta w niniejszej książce dotyczy systemów informatycznych zarządzania, a więc ukierunkowanych na wspomaganie działalności organizacji gospodarczych i administracyjnych na poziomie operacyjnym, taktycznym i strategicznym.

Modele, metody i skomputeryzowane narzędzia tworzą podstawowy **warsztat analityka i projektanta systemów**. Oddziałują one poprzez reguły modelowania, sposoby dokumentowania oraz komputerowego wspomagania, na procedurę i efektywność tworzenia systemu informatycznego. System, opracowany w trakcie procesu TSI przy użyciu instrumentarium warsztatowego, podlega **ocenie według kryteriów spójności i kompletności projektu i systemu**. W zależności od wyników oceny, system jest przyjmowany do eksploatacji bądź ponownie planowany, analizowany, projektowany i wdrażany.

3.2. Klasyfikacja metodyk tworzenia systemów informatycznych

Literatura przedmiotu wskazuje na dużą różnorodność metodyk TSI. Dotyczy to wszystkich składników metodyki: zarówno ich zakresu (zestawu pełnego lub cząstkowego), jak poszczególnych formalizmów, metod, technik i narzędzi. Jednak do tej pory, brak jest uogólnionych powszechnie uznanych podstaw teoretycznych TSI. Jakie są główne przyczyny tego stanu rzeczy? Wydaje się, że najważniejsze to nowość i wczesny okres rozwoju tej dziedziny, a w związku z tym pewna żywiołowość pojawiania się propozycji i podejść. Związane z TSI zagadnienia należą do problemów nieustrukturyzowanych, a więc złożonych zarówno w zakresie ich definiowania, jak i w konsekwencji - rozwiązywania. Na obecny stan mają też zapewne wpływ: specyfika dziedzin przedmiotowych poddawanych procesowi modelowania, szybkie zmiany w sferze inżynierii oprogramowania pozwalające na konstruowanie zautomatyzowanych narzędzi wspomagania procesu TSI. Zastanawiając się

nad przyczynami omawianej sytuacji, Ceri stwierdza[4], iż zagadnienie to jest trudne ze względu na brak dobrze zdefiniowanych teoretycznych podstaw wyrażania potrzeb informatycznych. Występują zatem trudności w stworzeniu zaawansowanego, poprawnego logicznie, precyzyjnego i jednocześnie łatwo zrozumiałego dla użytkownika formalizmu definiowania tych potrzeb.

W związku z próbą analizy metodyk TSI, ich zbiorowość należy sklasyfikować. Z podanych wcześniej względów nie jest to zadanie łatwe. Zaproponowano następujące **kryteria oceny**:

- podejście do procesu TSI,
- definiowanie danych bądź procesów w projekcie,
- oddziaływanie systemu informatycznego na dziedzinę przedmiotową,
- kierunek TSI.

Pierwsze podstawowe kryterium umożliwia wyodrębnienie metodyk technicznych i społecznych . Nazwy są tu umowne, choć oddaj ą one istotę podejść. Metodyki techniczne są ukierunkowane na realizację dobrze ustrukturyzowanego procesu TSI, z pełnymi i sformalizowanymi modelami opisu rzeczywistości. Podejście opiera się na założeniu, że analityk ma neutralny wpływ na organizację w procesie TSI. Z kolei metodyki społeczne akcentują organizacyjne, psychologiczne i socjologiczne problemy związane z TSI. Celem tego podejścia jest zrozumienie roli systemu informacyjnego w ramach systemu społecznego i oddziaływanie na obydwa. Rola analityka jest tu bardziej aktywna. Powodzenie strategii TSI jest w tym wypadku uzależnione od umiejętności specyfikacji podstawowych uwarunkowań organizacyjnych i kadrowych, a także od możliwości oddziaływania na nie. Metodykom technicznym zarzuca się, iż stanowią one swoiste "książki kucharskie", które mogą ominąć istotę problemu ze względu na mechaniczne jego rozwiązywanie, stosownie do zalecanej w konkretnej metodyce procedury. Natomiast podejście społeczne nie pozwala na tak dokładne sprecyzowanie potrzeb informatycznych, aby były one podstawą realizacji kolejnych faz cyklu życia systemu. Dobre rezultaty daje integracja obydwu podejść. System jest wówczas tworzony w powiązaniu z jednym z wymienionych w dalszej części pracy cykli życia systemu, natomiast fazę planowania realizuje się, stosując metody i techniki społeczne.

Kwestia modelowania danych i procesów dziedziny przedmiotowej wiąże się ze strukturalnym podejściem do TSI. Metodyki zorientowane na dane koncentrują analizę i

---

[4] *Ceri S. Requirements Collection and Analysis in Information Systems Design; w Information Processing 86, pod red. Hj. Kuglar, North Holland, Amsterdam 1986, s. 205 – 217.*

projektowanie systemów wokół strukturyzacji danych użytkowanych w organizacji. Z kolei stosując metodyki zorientowane na procesy, dokonuje się dekompozycji procesów gospodarczych w określonej dziedzinie przedmiotowej i określa związki między nimi. Na tej podstawie, uwzględniając potrzeby użytkowników, specyfikuje się dane elementarne.

Metodyki TSI stanowią zazwyczaj kompromis, o różnej proporcji, pomiędzy rozwiązaniami ukierunkowanymi na dane a rozwiązaniami ukierunkowanymi na procesy.

Podstawą kolejnego podziału metodyk TSI są relacje pomiędzy dziedziną przedmiotową (wycinkiem rzeczywistości) a systemem informatycznym. Pierwszy rodzaj związku to **organizacyjne odwzorowanie** zakładające pasywną rolę systemu informatycznego. Decyzje i działania są podejmowane w dziedzinie przedmiotowej. W związku z tym system informatyczny musi być prawdziwym jej odzwierciedleniem, aby mógł funkcjonować efektywnie. Założenie precyzyjnego opisu istniejącej sytuacji ogranicza możliwość wprowadzania innowacyjnych rozwiązań. Przeciwne podejście - **organizacyjnego sterowania** - zakłada wyróżnienie systemu sterowania, w którym podejmuje się decyzje i działania wpływające na dziedzinę przedmiotową. System informatyczny jest jego integralną częścią. Nacisk kładzie się bardziej na określanie potrzeb informatycznych, a mniej na precyzyjny opis świata rzeczywistego. W polskiej literaturze te dwa rodzaje podejść określa się mianem pasywnego i aktywnego .

Wreszcie ostatnie z wymienionych kryteriów umożliwia wyodrębnienie **metodyk zstępujących** (ang. *top-down)* i **wstępujących** (ang. *bottom-up).* Podejście zstępujące oznacza tworzenie systemu poprzez stopniowe, hierarchiczne wyodrębnienie jego składników aż do podstawowego poziomu szczegółowości. Podejście wstępujące z kolei polega na stopniowym budowaniu syntezy systemu poprzez integrację jego elementów, począwszy od poziomu podstawowego. Praktycznie użytkowane metodyki stanowią zazwyczaj swoisty kompromis pomiędzy podejściem technicznym a społecznym, specyfikacją danych i procesów, aktywnego i pasywnego wpływu na dziedzinę przedmiotową oraz wstępującego bądź zstępującego toku projektowania.

Aktualnie dominuje w literaturze i praktyce klasyfikacja opierająca się na połączeniu kryteriów dotyczących opisu dziedziny przedmiotowej oraz doświadczeń praktycznych TSI. Wyróżnia się więc trzy "szkoły", trzy rodzaje podejść metodologicznych do TSI:

- strukturalne,
- obiektowe,
- społeczne.

Historycznie jako pierwsze ukształtowało się podejście strukturalne, zwane również strukturalno-relacyjnym ze względu na ścisłe powiązania z modelem relacyjnym baz danych. Jest to podejście formalne, polegające na tworzeniu uporządkowanego systemu, hierarchicznej strukturze, którego sterowalne składniki stanowią dobrze zdefiniowane moduły funkcji i danych oraz związki między nimi. Cechą charakterystyczną tego podejścia jest oddzielne modelowanie danych i procesów, wykorzystujące diagramowe i macierzowe metody i techniki. W praktycznych zastosowaniach nadal dominuje podejście strukturalno-relacyjne, jednak w literaturze z zakresu informatyki ekonomicznej i w pracach badawczych istotniejszą rolę odgrywa **podejście obiektowe**. Opiera się ono na wyodrębnieniu obiektu, czyli każdego bytu, pojęcia lub rzeczy, mających przypisane znaczenie w kontekście rozwiązywania problemu w danej dziedzinie przedmiotowej. Pojęcie obiektu umożliwia integralne modelowanie danych i procesów.

Podejście społeczne, związane z nazwiskiem Checklanda, akcentuje aspekty ludzkie i społeczne - psychologiczne i socjologiczne - w tworzeniu systemów informatycznych. O ile wcześniej wymienione podejścia dotyczą całego cyklu życia systemu, o tyle podejście społeczne jest użyteczne w fazie planowania systemów informatycznych.

Największą liczbę propozycji metodycznych, zarówno akademickich, jak i komercyj nych, ma podejście strukturalne. Przykładami uznanych metodyk strukturalnych są:

- **SSADM** (ang. *Structured Systems Analysis and Design Method),* standard finansowo wspierany przez rząd brytyjski,
- **IE** (ang. *Information Engineering),* metodyka wykorzystująca prace J. Marlina, stosowana w wielkich korporacjach,
- **YSM** (ang. *Yourdon Systems Method),* metodyka proponowana przez E. Yourdona.

Z kolei najbardziej zalecane metodyki obiektowe to

- **OMT** (ang. *Object Modelling Technique),*
- **RATIONAL** z językiem UML.

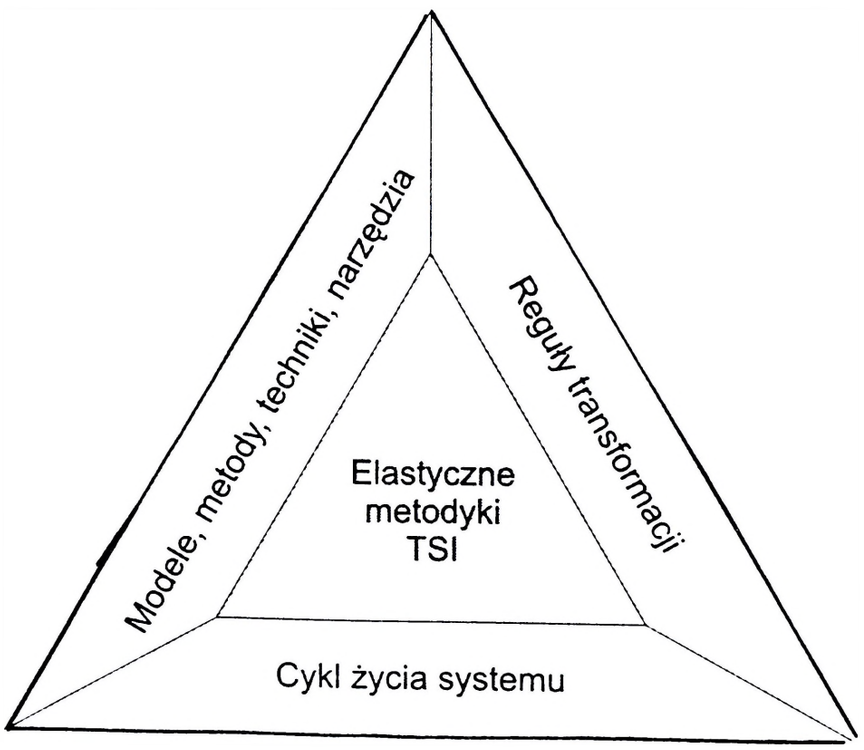
Wreszcie, największym zainteresowaniem jako metodyki społeczne cieszą się:

- **SSM** (ang. *Soft Systems Methodology)* P. Checklanda,
- **ETHICS** (ang. *Effective Technical and Human Implementation of Computer-Based Systems) E.* Mumford.

Komercyjne metodyki, oferowane przez firmy doradcze i komputerowe, wspomagane są często przez związane z nimi pakiety CASE. Przykładem może być

metodyka *Information Engineering* wraz ze związanym z nią pakietem IEWorkbench.

Na podstawie wiedzy o metodykach, technikach i narzędziach tworzenia systemów informatycznych zespół projektowy może zaproponować własną metodykę. W ostatecznym rachunku każda metodyka jest łańcuchem technik i narzędzi, odpowiednich dla różnych faz cyklu życia systemu. W ten sposób powstają elastyczne, kombinowane metodyki TSI]. Podstawowym warunkiem ich efektywnego użytkowania jest opracowanie reguł transformacji pomiędzy często niejednorodnymi technikami proponowanymi dla kolejnych faz TSI. Reguły te definiują ściśle możliwości transformacji poszczególnych kategorii modeli danych i procesów typowych dla kolejnych faz cyklu życia systemu. Przykładem może być transformacja pomiędzy grafami ISAC a diagramami związków encji Składniki elastycznych, kombinowanych metodyk TSI przedstawia rys..7.



Rys. 7. Elementy elastycznej metodyki TSI

Dotychczasowe rozważania skłaniają ku próbie określenia wymagań, które powinna spełnić racjonalna metodyka. Wydaje się, iż najistotniejsze z nich to:

- metodyka powinna objąć cały cykl życia systemu, od analizy do adaptacji i modyfikacji, przy jednoczesnym umożliwieniu płynnych przejść pomiędzy poszczególnymi fazami;
- metodyka TSI winna obejmować różnorodne, dostosowane do specyfiki podejścia, metody, techniki i narzędzia komputerowe ułatwiające zrozumienie

problemów, ich analizę i rozwiązanie;

- metodyka powinna ułatwiać porozumiewanie się pomiędzy różnymi grupami zawodowymi tworzącymi nowy SI - dotyczy to zwłaszcza wstępnych faz procesu TSI, gdy należy zaoferować narzędzie (formalizm) ułatwiające porozumienie informatyków i użytkowników;
- metodyka powinna być stosunkowo łatwa do opanowania i dostosowana do szerokiej klasy problemów oraz zawierać mechanizmy ewolucyjności i modyfikowalności.

Powyższe wymagania stanowią jednocześnie wskazówki przy dobieraniu odpowiedniej metodyki.

W tym aspekcie godny podkreślenia jest wysiłek Wspólnoty Europejskiej w celu stworzenia jednolitej metodyki, którą nazwano EuroMethod. Nad jej szczegółami pracował zespół ekspertów z państw członkowskich. Zespół ten, wywodzący się zarówno z instytucji badawczych, jak i organizacji gospodarczych i doradczych, brał pod uwagę podejścia metodyczne wywodzące się z następujących krajów:

- Wielkiej Brytanii - **SSADM** (ang. *Structured Systems Analysis and Design Method),*
- Francji - **MERISE,**
- Włoch - **DAFNE** (ang. *DAta and Function NEtworking),*
- Holandii - **SDM** (ang. *System Development Methodology),*
- Hiszpanii - **MEIN** (hiszp. *MEthodologica INformatica),*
- Niemiec - **VORGEHENSMODEL**,
- Wielkiej Brytanii i USA - **IE** (ang. *Information Engineering).*

Powstała obszerna dokumentacja metodyczna. Jej zastosowanie było ograniczane sprzecznymi, jak się okazało, interesami organizacji administracyjnych, gospodarczych i badawczych. Aktualnie EuroMethod (nazwa zmieniona na ISPL - ang. *Information Service Procurement Library)* służy jako narzędzie do realizacji przetargów na duże projekty informatyczne całościami, obejmującymi obszerne dziedziny przedmiotowe. Wynika to z samej istoty systemu informatycznego wiążącego ludzi, metody i środki techniczne w celu racjonalnego

4. PLANOWANIE SYSTEMÓW INFORMATYCZNYCH

4.1. Cele planowania systemów informatycznych

Pierwszą fazą cyklu życia systemów informatycznych jest planowanie, zwane również w literaturze przedmiotu strategicznym planowaniem systemu. Faza planowania jest realizowana w odniesieniu do dużych projektów dla złożonych dziedzin przedmiotowych. Samo zadanie planowania jest z natury skomplikowane, interdyscyplinarne. Wymaga obszernej wiedzy nie tylko informatycznej, ale i z zakresu zagadnień ekonomicznych oraz zarządzania firmą. Przedmiotem zainteresowania w tej fazie są m.in.: misja gospodarcza ńrmy, jej cele, biznesplan, otoczenie, konkurencja. Złożoność fazy planowania systemu wynika z konieczności zharmonizowania zagadnień gospodarczych z aktualnymi trendami rozwoju technologii i systemów informatycznych w celu skutecznego, podnoszącego konkurencyjność firmy wspomagania jej funkcjonowania. Poza specjalistami-informatykami w procesie planowania uczestniczy kierownictwo firmy, rozumiejące najlepiej istotę jej funkcjonowania. Zespół projektowy stosuje metody i techniki heurystyczne, stymulujące generowanie twórczych rozwiązań, a następnie ich dokumentowanie. Tak więc, założeniem realizacji fazy planowania systemów jest osiągnięcie dwu celów:

- stworzenie systemu informatycznego skutecznie wspomagającego strategiczne cele firmy,
- zaangażowanie kierownictwa firmy w proces planowania i projektowania, a następnie użytkowania systemu informatycznego.

Planowanie systemu informatycznego wymaga identyfikacji i nadania priorytetów tym technologiom i zastosowaniom, które najbardziej wspomagają misję i cele danej organizacji. Planowanie SI inicjowane jest na podstawie rezultatów i ocen użytkowania istniejących systemów. W związku z wysokim obecnie stopniem informatyzacji zarówno gospodarki, jak i administracji, czynność planowania rzadziej dotyczy systemów nowych, tworzonych od podstaw. W konsekwencji planowanie jest procesem ciągłym, regularnie powtarzanym, stosownie do oceny, czy decyzje kierownictwa oraz czynniki zewnętrzne nie wymagają zmiany planu. Ta funkcja kontrolna winna być wykonywana periodycznie. Cechą odróżniającą planowanie systemów informatycznych od innych faz cyklu życia jest skład zespołu planującego. Obejmuje on analityków-planistów, kierownictwo firmy oraz działu lub

firmy informatycznej. Szczególnie wysokie wymagania stawia się analitykom-planistom. Poza kompetencjami z danej dziedziny przedmiotowej, powinni oni posiadać wiedzę z zakresu organizacji i zarządzania, analizy i projektowania systemów, a także na temat sprzętu, sieci komputerowych oraz baz danych. Ponieważ trudno sprostać takiemu spektrum kompetencyjnemu, analitycy-planiści często współpracują z ekspertami, szczególnie z zakresu sprzętu, sieci oraz baz danych.

4.2. Proces planowania systemu informatycznego

Proces planowania w ujęciu autorów różnych prac z tej dziedziny składa się z trzech kolejnych etapów:

1) studium istniejącego systemu rzeczywistego, będącego przedmiotem informatyzacji,
2) określenia architektury systemu,
3) identyfikacji i oceny obszarów zastosowań.

Pierwszy etap związany jest z analizą i oceną procesów (np. gospodarczych) danej organizacji, a więc takich zagadnień, jak misja gospodarcza, plan strategiczny firmy czy biznesplan. Rozpatrywane są one jednak pod kątem przyszłego systemu informatycznego firmy, który ma być tworzony w dalszych etapach cyklu życia systemu. Wejściem do pierwszego etapu jest, sformułowana zazwyczaj już uprzednio, misja firmy, będąca gospodarczym uzasadnieniem istnienia firmy. Zaś wyjściem tego etapu jest biznesplan, czyli plan funkcjonowania firmy w kategoriach jej misji, celów i ograniczeń, zadań, istotnych czynników powodzenia, klientów, dostawców, konkurencji, wyrobów i usług, kadr, struktury organizacyjnej, materiałów i innych czynników. Toteż na tym etapie szczególnie niezbędne jest zaangażowanie kierownictwa firmy w prace zespołu planistycznego. Podstawowe zadania wykonywane w trakcie studium misji gospodarczej polegają na:

- analizie istotnych czynników powodzenia (omówionej niżej wśród innych metod planowania systemów informatycznych),
- analizie konkurencji firmy,
- analizie zwiększenia wartości (ang. *value chain analysis)* wyrobu lub usługi w cyklu produkcyjnym.

Zaplanowanie architektury systemu informatycznego polega na określeniu zasobów danych, rodzaju zastosowań, sieci komputerowej i infrastruktury technicznej w aspekcie procesów gospodarczych danej organizacji. Podstawowym sposobem definiowania

architektury są macierzowe zależności pomiędzy zasobami danych, funkcjami, komórkami organizacyjnymi, zastosowaniami. Niezwykle przydatna okazuje się tu metoda BSP (ang. *Business Systems Planning).* Plan ten nazywa się również strategicznym bądź głównym planem informatyzacji lub po prostu **infoplanem**.

Ostatnią czynnością w fazie planowania jest identyfikacja i ocena obszarów zastosowań, wynikająca z architektury systemu. Jej celem jest integracja poszczególnych składników systemu w spójną całość, wspomagającą misję gospodarczą w aspekcie:

· procesów gospodarczych i powiązań między nimi,

- bazy danych,
- sieci komputerowej,
- sprzętu i oprogramowania.

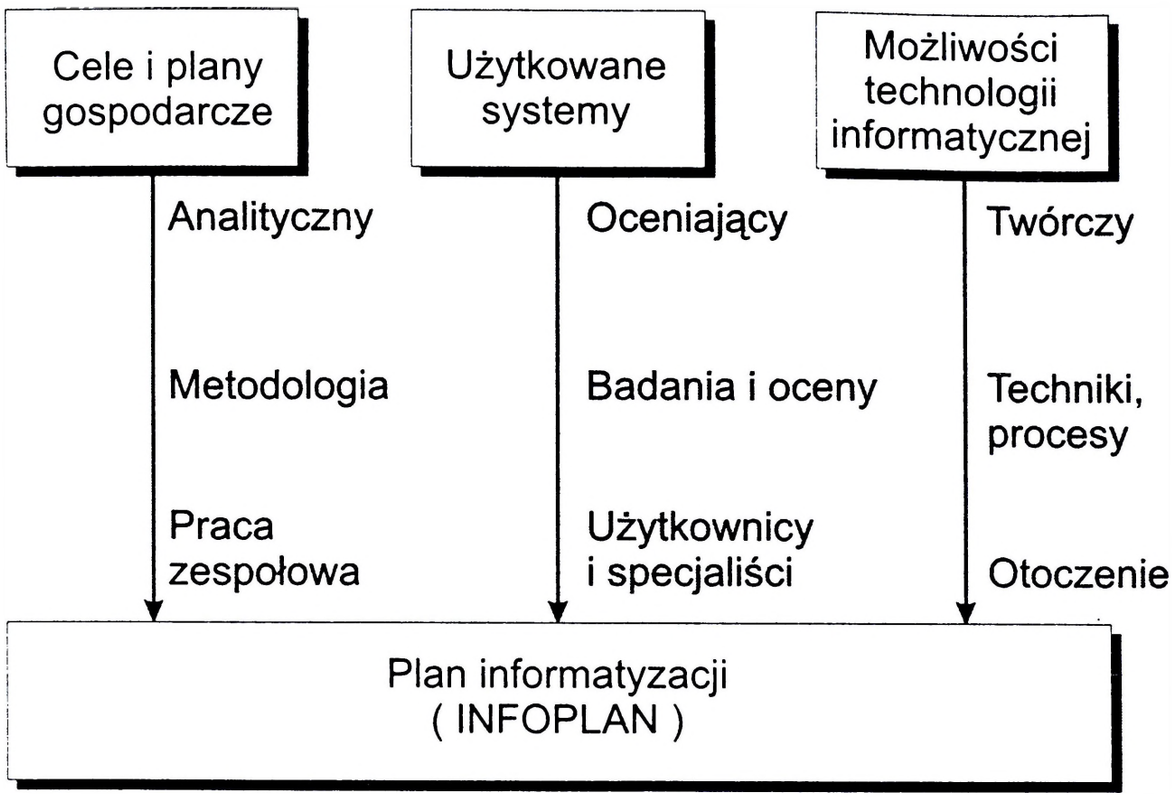
Przeglądu i oceny obszarów zastosowań dokonuje się na podstawie kryteriów jakości i wykonalności.

4.3. Formułowanie strategii informatyzacji (infoplanu)

Autorzy metodologii formułowania strategii informatyzacji organizacji gospodarczych i administracyjnych preferują **podejście sytuacyjne.** Cechuje się ono uzależnieniem zakresu i treści planu strategicznego informatyzacji firmy, czyli infoplanu, od specyfiki działalności firmy, kontekstu technologicznego, stylu zarządzania i struktury organizacyjnej. Pewną syntezą podejść sytuacyjnych jest propozycja Earla, przedstawiona na rys.8. W trakcie **formułowania infoplanu** należy według niego uwzględniać, analizować i określać następujące trzy grupy zagadnień:

- wyrażenie potrzeb, celów i strategii gospodarczej firmy w kategoriach wspomagających systemów informatycznych,
- ocenę aktualnie użytkowanego w firmie systemu informatycznego,
- włączenie nowych, innowacyjnych rozwiązań z dziedziny technologii informatycznej.

Metodologia formułowania infoplanu jest wielostronna i złożona, a rozpoczyna się od badania różnych zjawisk gospodarczych i rozwiązań technologicznych. Trzy podstawowe procesy - analityczny, oceniający i twórczy przebiegają **równolegle,** wywierając na siebie **wzajemny wpływ.**



Rys.8. Formułowanie strategii informatyzacji
Źródło: Earl M.: Managment Strategies for Information Technology: Prentice Hall. New York 1989.

Zróżnicowany jest również **kierunek realizacji** każdego z procesów, a więc odpowiednio: zstępujący (ang. *top-down),* wstępujący (ang. *bottom-up)* i rozprzestrzeniający (ang. *inside-out).* Każdy z procesów jest niezbędny, lecz niewystarczający do opracowania pełnego, spójnego strategicznego planu informatyzacji. W łącznym procesie formułowania infoplanu biorą udział różne grupy zawodowe: od kierownictwa firmy poprzez jej służby informatyczne i użytkowników systemów do konsultantów i specjalistów zewnętrznych. Wnoszą oni różne rodzaje wiedzy do zadania planistycznego: ekonomiczną (finansową, księgową, marketingową), menedżerską i informatyczną.

Rolą pierwszego procesu (analitycznego) jest **identyfikacja planów i celów gospodarczych.** Na tej podstawie definiowane są wspomagające je potrzeby informatyczne, przy użyciu sformalizowanej metodologii o charakterze analitycznym. W realizacji tego procesu współuczestniczą kierownicy szczebla strategicznego, taktycznego i operacyjnego

przy udziale kierownictwa służb informatycznych. Ogólnie proces ten polega na przeprowadzeniu zstępującej **procedury transformacji**, "tłumaczenia" działalności gospodarczej na systemy i technologię informatyczną. Tworzone w ten sposób systemy informatyczne wspomagają działalność gospodarczą firmy. Działalność ta powinna być opisana w postaci zestawu celów gospodarczych, biznesplanów czy gospodarczych planów strategicznych. Często należy jesformułować od podstaw, gdyż są niespójne, źle opracowane czy wręcz nie istnieją. Nawet jeśli zostały opracowane, mogą być nieadekwatne do procesu generowania potrzeb informatycznych.

Istnieje wiele metod pozwalających na identyfikację systemów informatycznych wspomagających mis i cele gospodarcze firm . Generalnie podejście takie powinno być:

- skuteczne w opisie celów, planów i strategii gospodarczych,
- mało czaso- i zasobochłonne,
- powtarzalne wraz ze zmianą strategii i innych warunków otoczenia gospodarczego,
- wskazujące na kierunkowe potrzeby informatyczne, a nie na szczegółowe ' specyfikacje.

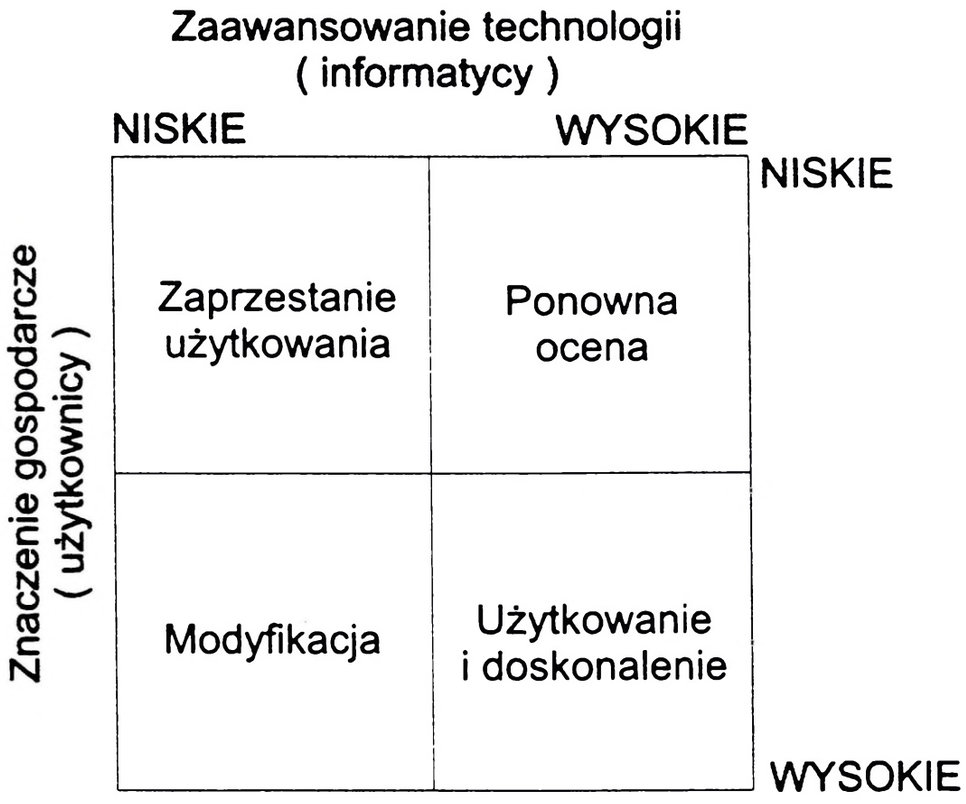
Walory te ma najczęściej stosowana w tym celu, opracowana przez C. Bullem i J. Rockarta metoda Istotnych Czynników Powodzenia (ang. *Critical Success Factors* - CSF).

Zmiana dotychczasowego systemu informatycznego wymaga również **opisu, zrozumienia i oceny istniejących systemów,** jako punktu wyjścia do dalszych decyzji i prac. Jest to druga równoległa ścieżka dochodzenia do planu informatyzacji. System informatyczny firmy ma formę albo zintegrowanej całości, albo zestawu systemów cząstkowych, wspomagających wycinki działalności organizacji gospodarczej lub. administracyjnej. Ten drugi przypadek występuje częściej, ze względu na szybki rozwój sprzętu i oprogramowania. W wyniku opisania funkcjonującego systemu, kierownictwo uzyskuje swoistą "mapę terytorium", rozpoznaje wady i zalety użytkowanego systemu przed rozpoczęciem fazy definiowania infoplanu. Obecnie coraz rzadziej tworzy się system od początku (poza nowymi firmami), toteż dominuje podejście ewolucyjne - stopniowego rozwoju i modyfikacji systemu. Do opisu i zrozumienia architektury i zawartości użytkowanych w firmie systemów informatycznych bardzo często stosowana jest IBM-owska metoda BSP.

Aby natomiast **ocenić eksploatowany system**, należy dobrać odpowiedni **zestaw kryteriów**. Najważniejsze z nich to: zaawansowanie technologiczne systemu oraz jego

znaczenie gospodarcze. Obydwa kryteria mają równorzędne znaczenie. Ich integracja w postaci tablicy oceny systemów tworzy obraz stanu informatyzacji firmy, pozwalający na podjęcie racjonalnych decyzji i działań w tej sferze. Powstają w ten sposób cztery archetypy użytkowanych systemów informatycznych (por. tablica na rys. 9).

Poziom zaawansowania technologicznego określają specjaliści-informatycy, podczas gdy ocena znaczenia gospodarczego eksploatowanych systemów należy do ich użytkowników. Pierwsza grupa oceniających posługuje się ogólnymi pytaniami dotyczącymi niezawodności systemu,



Rys. 9. Tablica oceny systemów

Źródło: Earl M.: Managment Strategies for Information Technology: Prentice Hall. New York 1989.

sprawności technicznej i kosztów jego eksploatacji oraz takich zagadnień szczegółowych, jak parametry sieci, sprzętu i oprogramowania. Wpływ funkcjonowania systemu (lub jego braku) na działalność gospodarczą, łatwość i częstotliwość użytkowania oceniają natomiast sami użytkownicy. Systemy są oceniane według każdego z tych kryteriów zgodnie z określoną

skalą, a następnie kwalifikowane do każdej z czterech grup.

Systemy, których ocena jest wskazaniem do zaprzestania użytkowania, są już zdecydowanie **przestarzałe** technologicznie i merytorycznie. W pewnych przypadkach system od początku mógł słabo wspierać funkcjonowanie firmy bądź istota jej działalności zmieniła się na tyle, iż stał się on bezużyteczny. Jednakże, jeśli mimo przestarzałego poziomu technologicznego, użytkownicy oceniają wysoko jego znaczenie dla sprawnego funkcjonowania firmy, to zaprzestanie eksploatacji systemu mogłoby spowodować negatywne skutki. Kontynuacja eksploatacji takich systemów może być również spowodowana obawami przed skutkami i kosztami modyfikacji. Ostatecznym bodźcem do **modyfikacji** systemu jest zwykle zagrożenie jego upadkiem bądź wdrożenie przez konkurencję systemu zdecydowanie wzmacniającego jej pozycję na rynku. Opracowany w tej sytuacji infoplan sprowadza się do modyfikacji użytkowanego systemu, głównie od strony technologicznej. Jeśli jednak zastosowanie charakteryzuje się wysoką jakością technologiczną przy niskiej ocenie stopnia wspomagania działalności gospodarczej, powinno ono podlegać **ponownej ocenie.** Tego typu przypadek może mieć miejsce wtedy, kiedy świetny profesjonalnie zespół informatyków przygotowuje system przy i minimalnym udziale przyszłych użytkowników. Niezbędna jest wówczas pogłębiona ocena systemu pod kątem jego reorganizacji bądź włączenia dodatkowych funkcji i zastosowań gospodarczych po ponownym sprawdzeniu specyfikacji i analizie systemu. Naturalnie najodpowiedniejsza jest sytuacja, gdy ocena zarówno poziomu zastosowań gospodarczych, jak i technologicznego zaawansowania systemu jest wysoka. Systemy takie są najbardziej użyteczne dla przedsiębiorstwa. Mimo swych walorów, winny one być stale doskonalone - modyfikowane i adaptowane.

Trzecia ścieżka tworzenia strategicznego planu informatyzacji jest oparta na podejściu rozprzestrzeniania (ang. *inside-out).* Jej celem jest identyfikacja szans, czyli stwierdzenie, czy możliwości oferowane przez aktualne trendy rozwoju technologii informatycznej pozwolą podnieść konkurencyjność firmy lub stworzyć nowe opcje strategiczne dla niej. Właśnie podejście rozprzestrzeniania pozwala w głównej mierze odkryć te szanse. Podejście to realizowane jest w trzech taktach, polegających na:

- badaniu (analizowaniu) nowych, innowacyjnych pomysłów i rozwiązań z dziedziny technologii informatycznej,
- specyfikacji przesłanek o charakterze organizacyjnym i kadrowym, sprzyjających wdrożeniu nowatorskich rozwiązań,

- opracowaniu planu informatyzacji opartego na innowacyjnych rozwiązaniach, w wymiarze organizacyjnego i technologicznego środowiska.

Pierwsza czynność oznacza szczegółową analizę potencjalnego wpływu nowatorskich rozwiązań technologicznych na rozwój organizacyjny i gospodarczy firmy. Rozwiązania są generowane z wykorzystaniem metod heurystycznych, takich jak burza mózgów czy metoda delficka. Wypracowane pomysły muszą być ocenione według kryteriów wykonalności oraz strategicznej wartości dla firmy. Innowacyjne użytkowanie technologii informatycznej inspirowane jest przez twórczych menedżerów, specjalistów rozumiejących walory nowych produktów sieciowych, sprzętowych i programistycznych. Wielką rolę w procesie rozprzestrzeniania odgrywa zaangażowanie bezpośrednich użytkowników systemu.

Po wygenerowaniu akceptowalnych rozwiązań innowacyjnych następuje opracowanie przesłanek o charakterze organizacyjnym, uzasadniających celowość innowacji, i planu stworzenia warunków do ich wdrożenia. Oznacza to realizację wstępnych faz cyklu życia systemu. Należy przy tym utrzymać równowagę pomiędzy rodzajem innowacji a naturą potrzeb informatycznych oraz zapewnić możliwość wycofania się z innowacyjnego eksperymentu, narażonego na ryzyko niepowodzenia. Jednakże późniejsze fazy cyklu, po zaaprobowaniu rozwiązania na podstawie jego zalet, realizowane są konsekwentnie, zgodnie z regułami kierowania projektami.

Wykonanie tego zadania wiąże się z decyzjami kadrowymi. Zadanie **rozprzestrzeniania** powierza się młodym profesjonalistom, menedżerom posiadającym dużą wiedzę i umiejętności z zakresu użytkowania określonych narzędzi i technologii. Trzeba ich ulokować w tych obszarach funkcjonowania firmy, gdzie systemy i technologia informatyczna odgrywają rolę strategiczną. Inspiracja do nowatorskich rozwiązań informatycznych podnoszących konkurencyjność firmy pojawia się też w wyniku obserwacji stylu działania klientów, dostawców i konkurentów. Dodatkowo zespół projektowy ocenia już eksploatowane w firmie systemy pod względem wpływu na podniesienie jej konkurencyjności i włącza je do planu informatyzacji. Na plan ten składają się więc ocenione pozytywnie, użytkowane i modyfikowane systemy (w firmie w innych firmach) oraz innowacje technologiczne.

Podstawą twórczego procesu rozprzestrzeniania innowacji informatycznych jest więc zaprojektowanie organizacyjnego i technologicznego środowiska, które umożliwia **wdrażanie innowacji informatycznych**. Podejście to polega zatem na poszukiwaniu wychodzącego z wewnątrz organizacji bodźca, stymulującego procesy innowacyjne i

włączanie ich w formułowanie strategicznego planu informatyzacji firmy.

Po wybraniu wariantu innowacji informatycznych i opracowaniu organizacyjnych i kadrowych przesłanek ich realizacji następuje synteza dotychczas wykonanych działań, tj. opracowanie infoplanu. Uwzględnia się w nim naturalnie rezultaty dwu pozostałych ścieżek formułowania planu informatyzacji - zstępującej i wstępującej. W czynności tej zasadniczą rolę mogą odegrać użytkownicy. Ich udział we współtworzeniu, akceptacji i użytkowaniu innowacyjnych rozwiązań można stymulować przez:

- zastosowanie przyjaznej wobec użytkownika technologii informatycznej, jak menu dostosowanego do specyfiki firmy,
- generatory zastosowań,
- zaadaptowane dla branży systemy ekspertowe,
- wizualizację danych statystycznych.

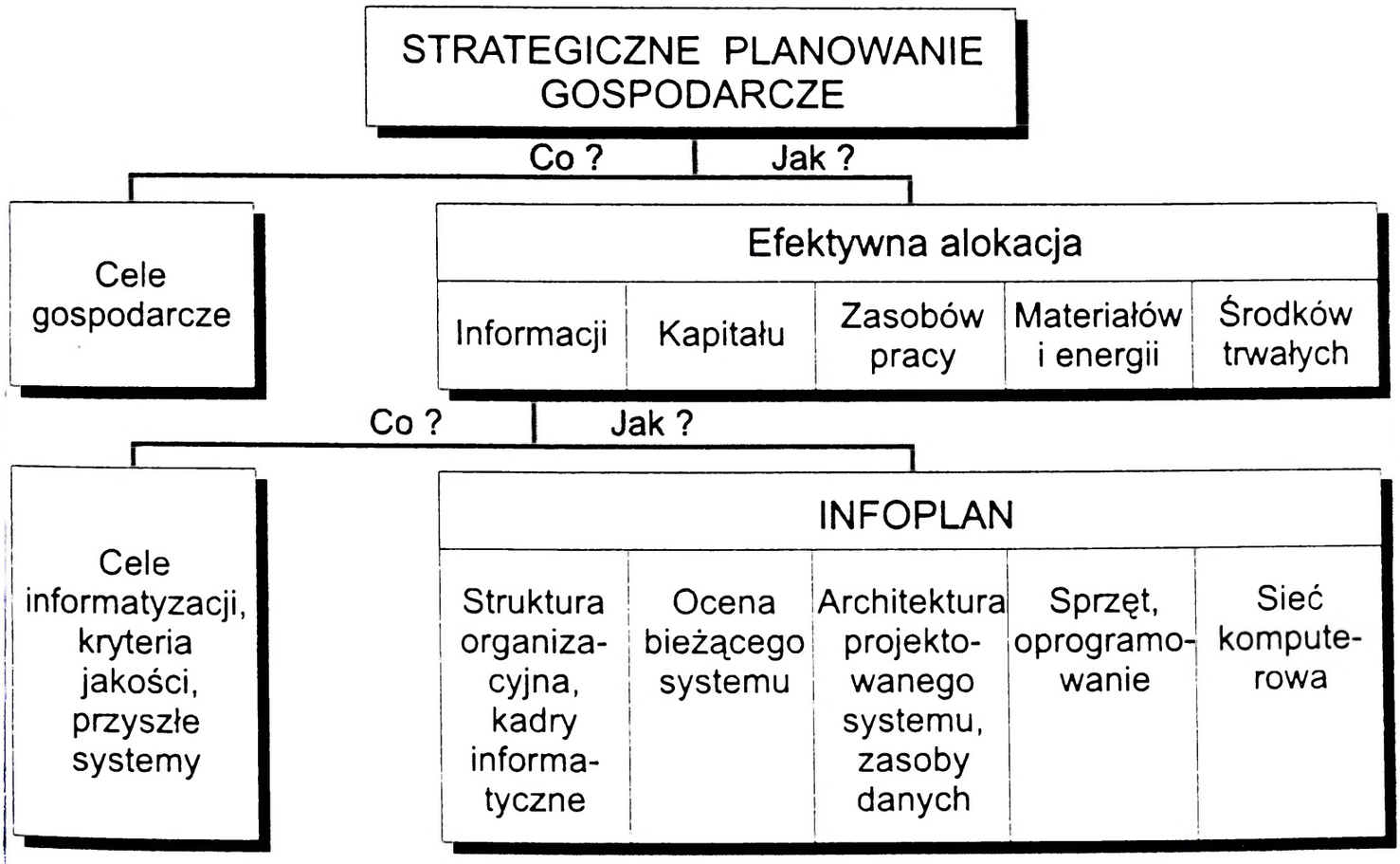
Zastosowanie sieciowych, sprzętowych i programistycznych standardów ukierunkowuje i wspomaga użytkowników w wykorzystywaniu możliwości dostępnych narzędzi i technologii.

Planowanie informatyzacji firmy musi być ściśle związane z planowaniem jej działalności gospodarczej. Każda z tych czynności ma swój rezultat końcowy zgodnie z formułą przedstawioną na rys.10. Zatem planowanie informatyzacji jest częścią strategii gospodarczej firmy i podobnie infoplan jest składnikiem biznesplanu.

Rys.5. Planowanie informatyzacji a strategia gospodarcza firmy

4.4. Infoplan - zawartość i znaczenie

Zasadnicze znaczenie w strategii gospodarczej firmy ma określenie celów firmy w kategoriach podstawowych procesów gospodarczych służących produkcji wyrobów czy usług. Aby je osiągnąć, niezbędna jest efektywna alokacja zasobów firmy: finansowych, kadrowych, materialnych, a także informacyjnych (rys. 11). Zasoby informacyjne są przechowywane w pamięci kadry firmy, dokumentach i opracowaniach papierowych, wreszcie na różnorodnych nośnikach w urządzeniach techniki informatycznej, m.in.



Rys. 11. Strategia gospodarcza a informatyczna

Źródło: Information Managment. Cap Gemini PANDATA. Gdańsk – Utrecht 1991.

w sieciach rozległych. Zasoby informacyjne są niezbędne do efektywnego sterowania funkcjonowaniem organizacji. Ich uporządkowanie, a następnie udostępnienie wspomaga realizację celów firmy oraz potrzeb informatycznych, dzięki planowaniu strategii informatyzacji. Podstawowymi składnikami infoplanu są: struktura służb i kądr informatycznych, architektura systemu, zasoby (bazy) danych, niezbędny sprzęt, oprogramowanie i sieci komputerowe (rys.11).

Treść infoplanu jest uzależniona przede wszystkim od dwu czynników:

- strategii, misji i celów gospodarczych firmy,
- rozwoju technologii informatycznej.

Obydwa te czynniki ulegają szybkim zmianom. Z jednej strony, pozycja firmy na rynku, bieżące zmiany permanentnie dokonujące się na nim, konkurencja (tzw. "ssanie popytu"), a z drugiej strony, coraz szybszy postęp w **technologii informatycznej** (tzw. "ciśnienie technologiczne") wpływają na konieczność stałej aktualizacji planu informatyzacji. Przy obecnym tempie zmian zachowuje on aktualność nie dłużej niż przez 2-3 lata, po czym powinno się opracować nowe wersje infoplanu. Opracowanie oraz aktualizacja tego planu wykonywana jest przez infomenedżera (osobę lub zespół). Infomenedżer jest łucznikiem pomiędzy kierownictwem przedsiębiorstwa a służbami informatycznymi. Transformuje strategię i cele gospodarcze firmy w infoplan w zakresie podanym na rys. 2.2. Aby sprostać temu zadaniu winien mieć rozległe rozeznanie i doświadczenie w funkcjonowaniu firmy, ale i dużą wiedzę fachową z dziedziny ekonomiki przedsiębiorstw, organizacji i zarządzania, marketingu, finansów, księgowości, a przede wszystkim - znać aktualne tendencje w dziedzinie informatyki.

5. ANALIZA STRUKTURALNA SYSTEMÓW INFORMATYCZNYCH

5.1. Analiza systemów informatycznych

Opracowany w pierwszej fazie tworzenia systemów informatycznych infoplan daje podstawę do zainicjowania analizy systemu. Planowanie systemów wiąże się z całościowym spojrzeniem na daną organizację, natomiast przedmiotem analizy jest wybrana dziedzina przedmiotowa, wybrana do informatyzacji. Rezultatem przeprowadzenia analizy jest definicja potrzeb użytkownika, toteż czasem w literaturze używa się zamiennie nazwy: analiza potrzeb użytkownika (ang. *eser requirements analysis).* Nazwa ta podkreśla centralną rolę, jaką w tej fazie - a w konsekwencji w całym procesie tworzenia - odgrywa użytkownik.

Analiza polega w praktyce na przeprowadzeniu studium zagadnień związanych zarówno z działalnością gospodarczą danej organizacji, jak i z wspomagającym ją, planowanym systemem informatycznym. W istocie bada się rozwiązywanie problemów gospodarczych i organizacyjnych firmy w sposób niezależny od technologii informatycznej. Podstawowymi czynnościami w fazie analizy są:

- identyfikacja i charakterystyka problemów i celów,
- studium dziedziny przedmiotowej - opis istniejącego systemu, · studium wykonalności systemu informatycznego,
- definicja i ustalenie priorytetów zidentyfikowanych potrzeb użytkownika.

Po wybraniu, na podstawie wniosków wynikających z infoplanu, dziedziny przedmiotowej przeznaczonej do informatyzacji, następuje identyfikacja problemów gospodarczych i organizacyjnych, a także ich agregacja w większe grupy problemowe. Rzutują one na określenie celów tworzonego systemu, do czego niezbędny jest szczegółowy opis funkcjonującego systemu w postaci studium dziedziny przedmiotowej w kategoriach metodyki strukturalnej, obiektowej bądź społecznej . Ustalone cele i stan bieżący porównuje się z możliwościami i ograniczeniami technicznymi, organizacyjnymi, prawnymi i ekonomicznymi firmy. W takim studium wykonalności systemu do przybliżonego ustalenia kosztochłonności i pracochłonności przedsięwzięcia projektowego w dużym zakresie są wykorzystywane modele procesów i danych, opracowane w trakcie studium dziedziny przedmiotowej . W przypadku akceptacj i wymagań wynikających ze studium wykonalności,

analizę finalizuje definicja potrzeb użytkownika, dokonywana głównie za pomocą metod modelowania danych i procesów i prototypowania. **Dokumentacja** potwierdzająca realizację powyższych czynności analitycznych wraz z infoplanem staje się podstawą zainicjowania projektowania systemu.

Analizę przeprowadza się **metodami:**

- analizy sytuacyjnej,
- modelowania i prototypowania systemów.

W pierwszej grupie wyróżnia się zarówno metody tradycyjne, jak i nowoczesne. Do tradycyjnych należą wywiad, kwestionariusz, obserwacja i analiza dokumentów. Użytkowanie każdej z tych metod od strony merytorycznej, organizacyjnej i psychologicznej zostało szeroko omówione w adekwatnej literaturze. Znajomość tych metod stanowi niezbędny składnik wiedzy i umiejętności praktycznych analityka systemu. Metody nowoczesne wiążą się z podejściem heurystycznym; wykorzystują takie techniki, jak burza mózgów, SWOT, sesja MetaPlanu czy JAD (ang. *Joint Application Development).* Z kolei modelowanie i prototypowanie systemów wiąże się z metodami i technikami, właściwymi dla podejść: strukturalnego, obiektowego i społecznego. Analiza złożonych dziedzin przedmiotowych wymaga komplementarnego stosowania technik analizy sytuacyjnej, modelowania i prototypowania. Metody te często są wspomagane komputerowo przez odpowiednie moduły pakietów CASE.

Pierwszą czynnością w fazie analizy jest studium problemów badanej dziedziny przedmiotowej. Chodzi przede wszystkim o problemy gospodarcze, organizacyjne i menedżerskie. W ich identyfikacji bardzo użyteczne są zwłaszcza metody wywiadów oraz różnego rodzaju warsztaty z udziałem kierownictwa firmy, użytkowników oraz informatyków. Problemy te są następnie agregowane w tematyczne grupy, szczegółowo charakteryzowane w postaci tabel problemów lub wzbogaconych wizerunków. Zidentyfikowane w trakcie tego studium problemy pozwalaj ą na określenie celów systemu. Tak więc, tworzony system powinien wspomagać realizację ustalonych celów i przyczyniać się do rozwiązywania bądź ograniczania występujących trudności.

Zdefiniowanie potrzeb użytkownika jest możliwe, dzięki opracowaniu **studium i charakterystyki dziedziny przedmiotowej**. Służy temu całe spektrum metod. Naturalnie dominują diagramy przepływów danych oraz diagramy związków encji. Użyteczne ~ mogą okazać się wzbogacone wizerunki, grafy ISAC-u, słowniki /skorowidze danych, relacyjne i obiektowe modele danych, techniki decyzyjne, diagramy struktury czy diagramy Jacksona.

Punktem wyjścia opisu systemu jest diagram kontekstowy, rozwijany następnie do szczegółowej prezentacji danych i procesów.

Ustalenie problemów występujących w przedsiębiorstwie, identyfikacja celów informatyzacji oraz opis dziedziny przedmiotowej stają się podstawą opracowania **studium wykonalności systemu** w aspekcie zasobów kadrowych, materialnych i finansowych organizacji. Studium składa się z czterech części, oceniających wykonalność systemu pod względem:

- technicznym,
- organizacyjnym,
- prawnym,
- ekonomicznym.

Zagadnienia możliwości realizacji systemu z zastosowaniem dostępnej technologii informatycznej - sprzętu, oprogramowania i sieci komputerowych - oraz dokonywanie niezbędnych analiz porównawczych parametrów technologii stanowią treść **technicznej** części studium. Z kolei wykonalność **organizacyjna** polega głównie na ocenie potrzeb zmian struktur organizacyjnych i kwalifikacji pracowników. Zgodność zasad działania systemu z obowiązującymi przepisami **prawnymi** oraz konieczne jego modyfikacje dotyczą wykonalności prawnej .

Zazwyczaj wiele dyskusji, emocji i kontrowersji wywołuje kwestia **wykonalności ekonomicznej.** Nie brak propozycji i formuł liczenia ekonomicznej efektywności komputeryzacji firmy, z uwzględnieniem różnych rodzajów kosztów i wyników finansowych. Praktyka dowodzi, iż trudno przeprowadzić tego typu rachunek ekonomiczny. O ile bowiem stosunkowo łatwo określić koszty systemu, o tyle oszacowanie efektów sprawia poważne trudności. Nie bez znaczenia jest też wpływ szybko zmieniającej się, już w trakcie tworzenia systemu, technologii informatycznej. Skuteczność przedsięwzięć informatycznych należy oceniać przede wszystkim przez pryzmat podniesienia konkurencyjności firmy na rynku, a także utrzymania lub podniesienia jej zyskowności.

Z całą pewnością użyteczne są natomiast metody szacowania **pracochłonności** i w konsekwencji **kosztochłonności** projektowania i wdrażania systemu. Najbardziej znane są tu dwie metody:

- analiza punktów funkcjonalnych **FPA** (ang: *Function Point Analysis),*
- **COCOMO** (ang. *COnstructive COst MOdel).*

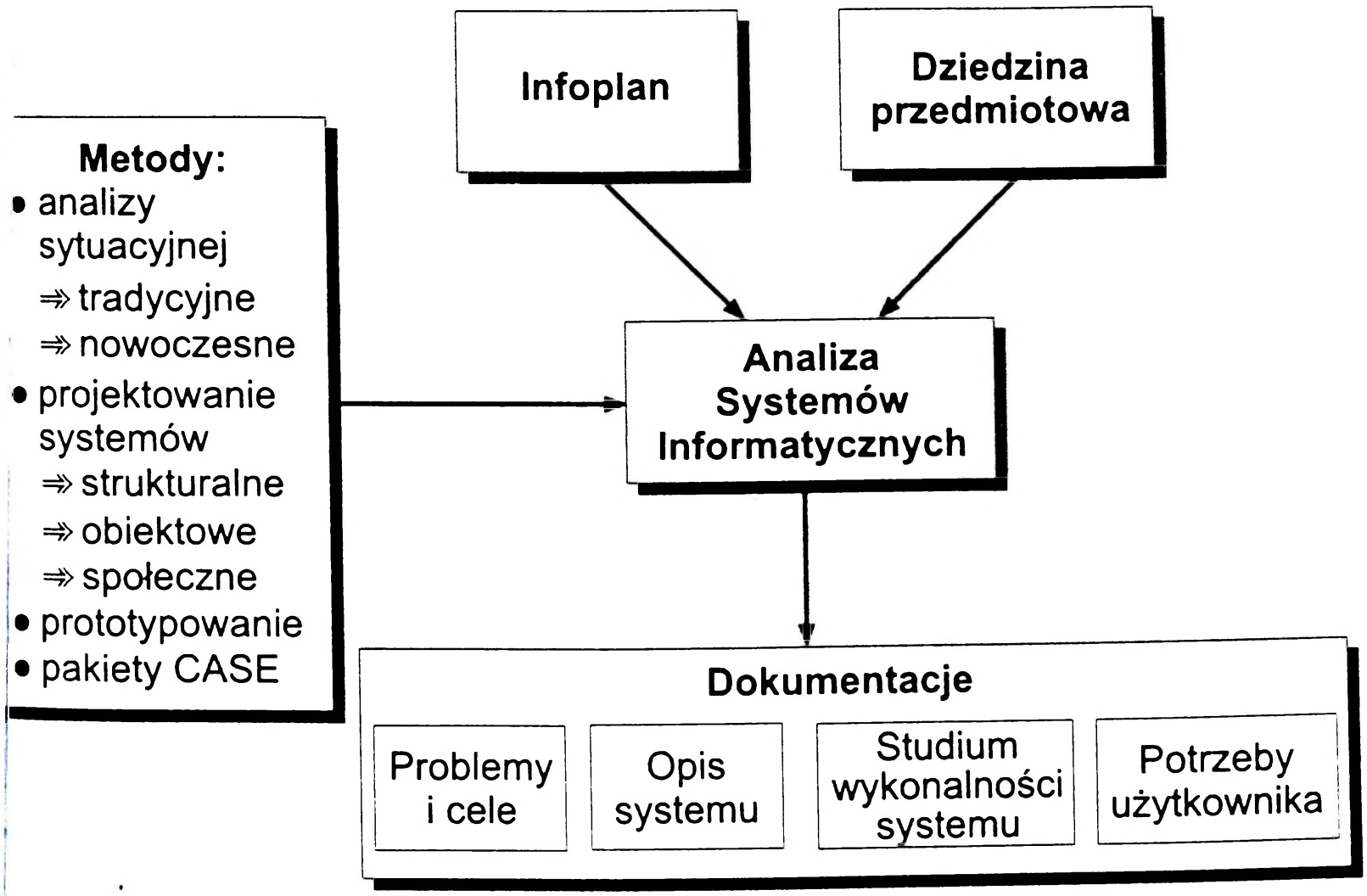
Bardziej zaawansowana jest metoda **analizy punktów funkcjonalnych**. Jej założeniem jest określenie liczonej w osobodniach pracochłonności tworzonego systemu na podstawie - w przeliczeniu na punkty - jego zakresu (wielkości) oraz złożoności. Zgodnie z dokonanym uprzednio szczegółowym opisem systemu wyodrębnia się pięć elementów: jego wejścia, wyjścia, zapytania, interfejsy i logiczne pliki danych. Pomocniczą rolę w ich identyfikacji mogą odegrać diagramy przepływu danych. Ocenie punktowej podlega również stopień złożoności poszczególnych składników opisu systemu, z podziałem na proste, średnio złożone i złożone. Każdy z wyodrębnionych składników oceniany jest według zawartej w odpowiednich tabelach skali punktów funkcjonalnych, uwzględniających liczbę pól lub atrybutów oraz liczbę odwołań do tych grup danych. Określona w ten sposób liczba punktów funkcjonalnych danego składnika jest dodatkowo korygowana o różne parametry wpływu, takie jak wpływ urządzeń komunikacyjnych, rozproszenia danych, częstotliwości wykonywania transakcji i innych. Skorygowane punkty funkcjonalne netto, dotyczące poszczególnych składników zostają zsumowane. Na podstawie doświadczeń praktycznych zakłada się, iż jeden punkt funkcjonalny netto odpowiada 8 godzinom pracy w technologii języków trzeciej generacji (3GL) lub 1,5 godziny pracy w technologii języków czwartej generacji (4GL): Wyliczenia te pozwalają więc na **określenie nakładów czasu pracy**, a w konsekwencji kosztów wdrożenia systemu. Należy zwrócić uwagę na znaczną pracochłonność dokonanego szacunku, który mimo dokładnych miar oraz współczynników korekty ma charakter przybliżony. Wydaje się, iż może on mieć zastosowanie w przypadku tworzenia dużych systemów, związanych z poważnymi nakładami finansowymi.

Znacznie prostsza w swej konstrukcji jest metoda COCOMO. Daje ona jednak na ogół gorsze wyniki, a stopień przybliżenia jest znacznie gorszy. Zasada obliczania pracochłonności opiera się na **oszacowaniu liczby linii kodu** niezbędnych do wdrożenia projektu. Ich liczba pozwala z kolei na określenie pracochłonności realizowanego projektu.

Finalne rezultaty fazy analizy to identyfikacja i **modelowanie potrzeb użytkownika**. Ich określeniu służy cała gama metod tradycyjnych i heurystycznych. Wyniki wywiadów, kwestionariuszy, obserwacji, analizy dokumentacji, warsztatów, burz mózgów skonfrontowane z celami projektu i jego opisem pozwalają zdefiniować potrzeby użytkownika. Definicja potrzeb może być oparta na analizie procesów bądź danych. Pierwsze z podejść zbudowane jest na modelu: **wejście-proces-wyjście**. Formalnym ujęciem tego typu definicji potrzeb jest IBM - owska metoda HIPO. Modele wejście – proces - wyjście są przyjazne wobec użytkownika w związku z łatwością i przystępnością ich tworzenia.

Ponieważ zarówno zakres, jak i specyfika procesów w organizacjach gospodarczych zmieniają się z upływem czasu, to bardziej stabilne wydaje się stworzenie **modelu danych,** którego zasoby mogłyby być wykorzystywane przez zmieniający się zestaw procesów.

Modelowanie potrzeb może wykorzystywać metody scharakteryzowane powyżej. Jednak niezwykle użyteczne może okazać **prototypowanie** systemów. Poprzez definiowanie i ocenę formatek, zestawień wynikowych czy dialogu następuje określenie prototypu, użytecznego w dalszych fazach cyklu życia systemu. Wyniki modelowania strukturalnego czy obiektowego oraz prototypowania mogą się wzajemnie uzupełniać. Tak zdefiniowane potrzeby użytkownika są uporządkowane, tj. ustala się priorytet oraz syntetyzuje w postaci spójnej specyfikacji analizy systemu, służącej do opracowania jego projektu. Syntezę zasad i rezultatów analizy systemów przedstawia rys.12.



Rys. 12. Zasady i rezultaty analizy systemu

6. PRZEGLĄD WYBRANYCH TECHNIK

6.1. Diagramy DFD (Data Flow Diagram)

Modele używane w analizie i projektowaniu systemów informatycznych umożliwiają uwypuklenie istotnych cech systemu przy pominięciu mniej ważnych szczegółów. Wszystkie modele zawierają element graficzny, przedstawiający w sposób obrazowy i jasny podstawowe elementy systemu oraz element tekstowy w postaci odpowiednio precyzyjnej definicji poszczególnych elementów modelu.

Podstawowym narzędziem w strukturalnej analizie systemów jest diagram przepływu danych (ang. Data Flow Diagram - **DFD**), stanowiący model funkcjonalny systemu. Na diagramie tym przedstawia się wszystkie procesy zachodzące w analizowanym systemie - odwzorowujące funkcje systemu, przepływy informacji między procesami, miejsca składowania danych (składnice danych), oraz źródła i miejsca odbierania informacji (terminatory). Celem techniki modelowania procesów (funkcji) jest ich graficzna reprezentacja umożliwiająca lepsze i głębsze zrozumienie natury i specyfiki tych procesów oraz odpowiednie ich zmodyfikowanie w zależności od stawianych systemowi wymagań. Procesy są tu rozumiane bardzo szeroko. Mogą być one częściowo lub całkowicie zautomatyzowane, mogą to być również czynności wykonywane ręcznie.

Diagramy DFD obrazują procesy zachodzące w systemie oraz wymianę danych między nimi, jak również sposób w jaki dane te są wprowadzane i wyprowadzane z samego systemu. Zespół diagramów DFD dla systemu wraz z opisem elementów występujących na diagramie w słowniku danych (repozytorium) stanowi model procesów w systemie. Jest to graficzna „mapa" procesów, ukazująca przepływ danych między procesami w systemie oraz między światem zewnętrznym a systemem.

Zazwyczaj pierwszym tworzonym diagramem DFD jest diagram kontekstowy systemu, następnym diagram poziomu zerowego, później zaś diagramy niższych poziomów, nazywane diagramami szczegółowymi.

Diagram kontekstowy przedstawia podstawowe cechy systemu:

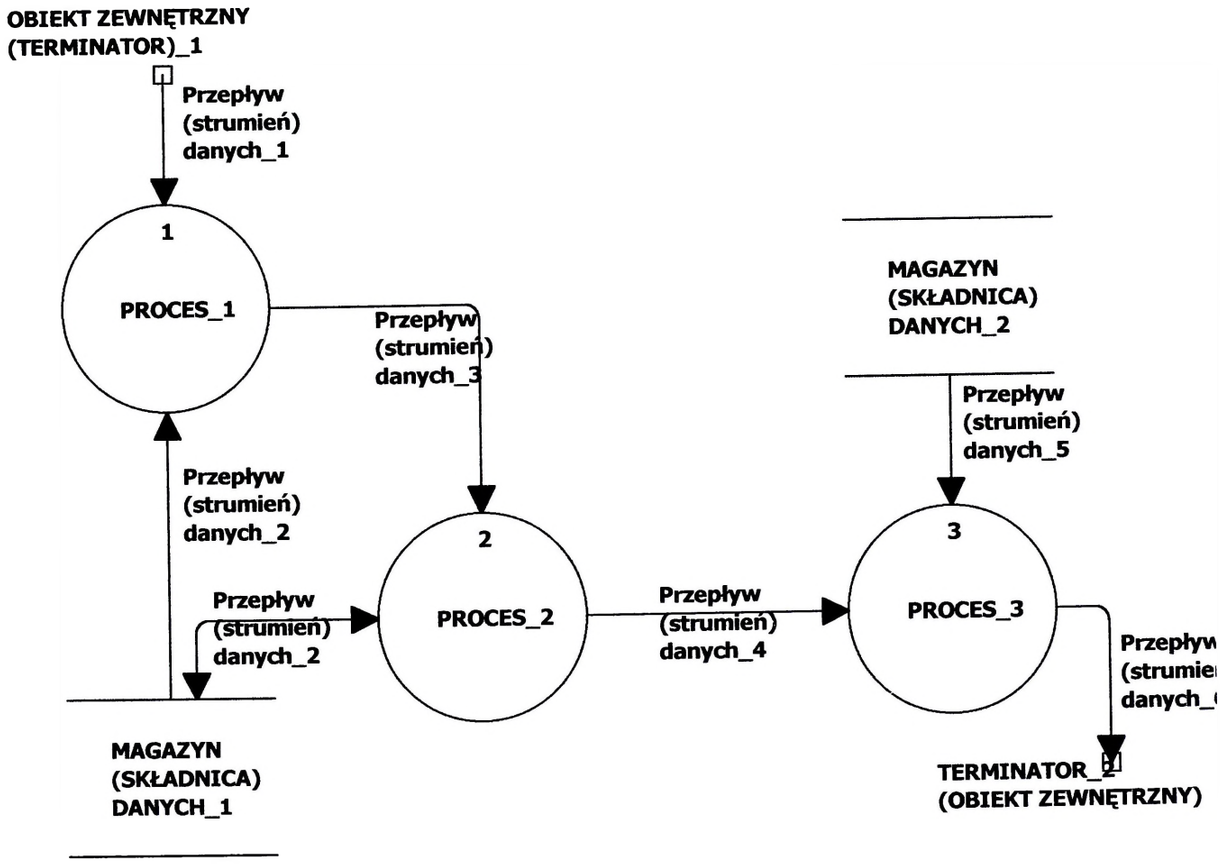
- granice systemu;
- źródła i odbiorców informacji w systemie;
- główne wejścia i wyjścia z systemu czyli informacje płynące między światem zewnętrznym a systemem.

Diagram przepływu danych poziomu zerowego przedstawia główne funkcje realizowane przez system. Każdy diagram (oprócz diagramu kontekstowego) ma diagram rodzica i może mieć wiele diagramów potomków. Każdy diagram potomek ukazuje szczegóły jednego z procesów na diagramie rodzicu. Tworzenie diagramów potomków dla procesów diagramu rodzica nosi nazwę dekompozycji.

Diagramy DFD zbudowane są z czterech podstawowych symboli reprezentujących:

- obiekty zewnętrzne względem systemu (terminatory);
- składnice (magazyny) danych;
- procesy;
- przepływy (strumienie) danych.

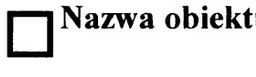
Na rysunku 13 przedstawiono przykładowy diagram DFD zawierający wszystkie symbole, mogące wystąpić na diagramie przepływu danych.



Rys. 13. Przykładowy diagram przepływu danych

Obiekty zewnętrzne (terminatory) reprezentują źródła lub miejsca przeznaczenia informacji, które są zewnętrzne w stosunku do systemu lub dekomponowanego procesu. Obiekty zewnętrzne dostarczają informacji, która powoduje wykonanie pewnych procesów w systemie, względnie odbierają informacje produkowane przez system lub dekomponowany proces. Takim obiektem może być na przykład użytkownik, instytucja nadrzędna lub systemy informatyczne współpracujące z projektowanym systemem.

Obiekty zewnętrzne reprezentowane są na diagramie kontekstowym w postaci kwadratu, wewnątrz którego umieszczona jest unikalna nazwa identyfikująca obiekt.



W diagramie poziomu zerowego i diagramach szczegółowych obiekty zewnętrzne reprezentowane są w postaci małego kwadratu, obok którego umieszczana jest nazwa obiektu. Obiekty te są powtórzeniem terminatorów poziomu diagramu kontekstowego lub reprezentują procesy zewnętrzne względem dekomponowanego procesu.

Składnice (magazyny) danych jest nazwaną kolekcją encji[5], atrybutów, związków i innych jeszcze nie sformalizowanych informacji (danych), które powinny być przechowywane przez pewien czas. Dane mają charakter stały i mogą być usunięte jedynie przez specyficzne działanie procesu. Składnice reprezentują miejsca gdzie dane są składowane i skąd są udostępniane procesom, które na nich operują - szczególnie jeśli procesy działają asynchronicznie tzn. może wystąpić przesunięcie w czasie między działaniem tych procesów. Zawartość składnic danych jest udostępniana jedynie procesom. Jeżeli z zawartości składnicy powinna być udostępniona obiektowi terminalnemu, to musi istnieć proces pośredniczący, który pobierze odpowiednie dane ze składnicy i przekaże je obiektowi terminalnemu. Dla programisty lub projektanta systemu składnica zwykle reprezentuje tablicę lub kolekcję tablic bazy danych, ale znaczenie

[5] *Encja jest obiektem mającym znaczenie, o którym informacje powinny być znane lub przechowywane. Encja jest definiowana zbiorem atrybutów - opisem znaczących cech, tzn. dowolnych szczegółów służących do klasyfikacji, rozróżnień, kwalifikacji lub wyrażających stan encji.*

symbolu jest bardziej ogólne. Składnica może być teczką z dokumentami, archiwum czy nawet książką telefoniczną.

Magazyn danych na diagramach DFD jest przedstawiany w postaci dwóch równoległych, poziomych odcinków („okładek kondensatora"), z zawartą między nimi unikalną nazwą opisową charakteryzującą zawartość składnicy danych.

Procesy odpowiadają tym składnikom systemu, które przetwarzają dane. Procesy otrzymują i przesyłają dane za pośrednictwem przepływów danych. Analogicznie jak w przypadku składnic, proces kojarzony jest zwykle z programem komputerowym lub wręcz z procedurą, jednak w przypadku analizy systemu może to być na przykład „ręczne" przygotowanie dokumentu.

Proces to definicja sposobu wykonywania przez system jednej lub więcej funkcji. Jest to czynność, działanie na danych w systemie; coś, co jest robione w systemie w celu zaspokojenia:

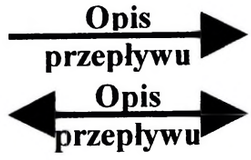
- jednej lub większej liczby funkcji przedsięwzięcia stanowiącego podmiot systemu;
- administracyjnych, kontrolnych lub zarządczych potrzeb samego systemu.

Tym samym pojęć procesu i funkcji podmiotu nie można utożsamiać. Różnica polega na tym, że proces nie modeluje tego co robi podmiot systemu lub co powinien robić; proces modeluje to, co powinien robić system. System jest więc zbiorem procesów, z których pewne modelują działanie służące do realizacji funkcji podmiotu zaś inne modelują działania niezbędne dla implementacji[6] systemu.

Proces jest reprezentowany przez okrąg, wewnątrz którego znajduje się nazwa charakteryzująca realizowane przez proces funkcje oraz unikalny numer procesu. Numery procesów na diagramie potomku powstają z połączenia numerów: procesu na diagramie rodzicu oraz unikatowego numeru w obrębie danego diagramu. Procesy na diagramie poziomu zerowego mają numery kolejne. Proces, reprezentujący system na diagramie kontekstowym, nie jest numerowany.

---

[6] ***Implementacja*** *(realizacja,* implementation*), urzeczywistnienie informatycznego projektu programu, bazy danych, interfejsu, systemu itd. Ten sam projekt może mieć wiele różnych implementacji lepszych lub gorszych, starszych i nowszych.*



Przepływ (strumień) danych jest nazwaną kolekcją encji, atrybutów, związków i innych nieznormalizowanych informacji (danych) przekazywanych między dwoma procesami, procesem a obiektem terminalnym lub między procesem a składnicą danych. Istnienie przepływu danych między dwoma procesami oznacza, że jeśli jest wykonywany proces stanowiący źródło przepływu, to w pewnym momencie jego wykonywania będzie następowało przekazywanie danych. Strumień danych może być traktowany jak rurociąg, którym podróżują dane. W fizycznej reprezentacji przepływu danych, jak linia telefoniczna lub kanał satelitarny, dane będą całkiem dosłownie przepływały strumieniem przez połączenie komunikacyjne.

Strumień danych, łączący dwa procesy, reprezentuje chwilowe przeniesienie informacji. Kiedy dotrą one do procesu przyjmującego, decyzja o tym co się z nimi dalej stanie zależy od tego procesu. Jeśli proces - odbiorca zignoruje nadchodzące dane, to zostaną one na zawsze utracone. Gdy strumień danych wejdzie do magazynu danych, zawartość magazynu jest modyfikowana zawartością strumienia. Może to oznaczać dodawanie, modyfikację lub usuwanie danych znajdujących się już w składnicy. Jedynie magazyn danych może służyć do przechowania na stałe chwilowego przepływu danych.

Strumień danych jest reprezentowany linią ze strzałką wskazującą kierunek przepływu. Nazwa obok niej określa przepływające dane. W diagramach DFD używa się także strzałek dwustronnych. Jest to notacja skrócona, obrazująca przepływ dwóch strumieni danych przenoszących te same informacje w obu kierunkach. Dwukierunkowe strumienie danych rzadko występują między dwoma procesami; częściej spotykane są między procesem i magazynem danych.

W przeciwieństwie do przepływu, którego adresatem jest proces, dane wpływające do składnicy danych są w niej umieszczane na stałe. Jeśli nie są już dłużej potrzebne, to musi być zdefiniowany specjalny proces powodujący ich zniszczenie. Zniszczenie danych pokazywane jest jako strumień prowadzący do magazynu, o nazwie określającej niszczone dane. Wszelkie przepływy wychodzące z magazynu danych nie reprezentują operacji destrukcyjnych.

Umieszczenie składnicy danych na diagramie DFD nie jest związane z żadną decyzją dotyczącą jej przyszłej implementacji. Dopiero w projekcie systemu dla każdej składnicy danych będzie wybierana jej fizyczna implementacja. Można zdecydować się na realizację pięciu składnic w postaci plików papierowych, jednej jako pliku komputerowego, a pozostałych dziesięciu jako komputerowej bazy danych korzystającej z systemu zarządzania bazą danych. Na diagramie przepływu danych magazyn danych nie określa żadnej

szczególnej fizycznej implementacji. Jego istnienie oznacza po prostu, że dane muszą być w jakiś sposób permanentnie przechowywane przez system.

Jak już wspomniano proces dekompozycji przebiega w układzie hierarchicznym, począwszy od diagramu kontekstowego do specyfikacji funkcji elementarnych, dalej niedekomponowalnych. Opracowane diagramu kontekstowego staje się podstawa diagramu zerowego zwanego również systemowym. Umożliwia on rozpoczęcie procesu wyodrębniania diagramów na niższych poziomach hierarchii aż do specyfikacji procesów elementarnych. Dekompozycja jest narzędziem opanowania złożoności systemu i jego opisu. Stopień złożoności systemu jest zróżnicowany i w jakimś zakresie można go mierzyć ilością poziomów hierarchii dekompozycyjnej (wyłączywszy poziom kontekstowy).

Diagram kontekstowy definiuje zakres i granice systemu. Przedstawia on powiązania systemu z jego otoczeniem, czyli kontekstem, w którym system funkcjonuje. System przedstawiany jest jako pojedynczy proces. Na jego obwodzie przedstawia się terminatory a w szczególnych przypadkach zewnętrzne składnice danych, powiązane bezpośrednio przepływami danych. Ze względów technicznych często znajduje tu zastosowanie zasada kilkakrotnego umieszczania tych samych terminatorów, zróżnicowanych oznaczeniami . Przepływy do i z terminatorów przekraczają granice systemu. W trakcie tworzenia diagramu kontekstowego analityk wykonuje następujące czynności:

1. przedstawienie systemu w postaci jednego procesu,
2. ustalenie wspólnie z użytkownikami zorientowanymi w specyfice dziedziny przedmiotowej, listy podstawowych zdarzeń - zapytań (operacji wejściowych) i związanych z nimi odpowiedzi (operacji wynikowych); stanowią one przepływy danych wiążące system z otoczeniem,
3. określenie źródeł i przeznaczenia danych - są to terminatory oraz zewnętrzne składnice danych.

Dokładna realizacja tych czynności pozwala na opracowanie diagramu kontekstowego, zrozumiałego dla użytkownika i będącego punktem wyjścia procesu dekompozycji w postaci hierarchii DFD. Tak przygotowany diagram stanowi swoista „mapę terytorium" dziedziny przedmiotowej. Dekompozycja procesu w diagramie kontekstowym rozpoczyna się od stworzenia diagramu poziomu zerowego (zwanego również systemowym), wynikającego bezpośrednio z diagramu kontekstowego. Decydująca jest tu dekompozycja jednego procesu z diagramu kontekstowego na kilka procesów. Są one agregatami dekomponowanymi dalej aż do funkcji elementarnych. Na poziomie zerowym wprowadza się też wewnętrzne składnice

danych oraz przenosi terminatory z diagramu kontekstowego. Procesy, składnice danych oraz terminatory powiązane są przepływami danych z diagramu kontekstowego.

6.2. Model danych - Diagram związków obiektów (ERD)

Wprawdzie dane w każdej organizacji zmieniają się na bieżąco, jednak rodzaje zbieranych, przechowywanych i przetwarzanych danych pozostają stabilne w dłuższej perspektywie czasu. Poprzez uogólnienie ich typów, cech i zależności między nimi można tworzyć modele danych. Model danych służy do wyrażenia struktury danych projektowanego lub istniejącego systemu. Przez strukturę danych rozumie się typy danych występujących w systemie, wzajemne powiązania między danymi oraz pewne ograniczenia nałożone na dane. Stosownie do pojęć, jakimi opisywane są dane, istnieje wiele modeli danych. Modele konceptualne (pojęciowe) opisują dane za pomocą pojęć, z których korzystają użytkownicy. Taki opis jest całkowicie niezależny od rozważań na temat pamięci, efektywności i innych szczegółów implementacyjnych. Modele fizyczne z kolei opisują dane w kategoriach sposobu ich przechowywania w pamięci komputera i nie są zrozumiałe dla użytkowników. Pośrednią grupę stanowią modele implementacyjne, które operują pojęciami w zasadzie zrozumiałymi dla użytkowników, ale bliżej związanymi z konkretną strukturą (logiczną) danych w pamięci komputera. Do ważniejszych powodów różnicowania modeli i tworzenia modeli pojęciowych należą: ograniczenie ryzyka utraty kompletnego opisu funkcjonalnego dziedziny przedmiotowej, poprzez pominiecie kwestii wdrożeniowych oraz stworzenie możliwości komunikowania się kierownictwa, użytkowników i informatyków, we wspólnym, zrozumiałym języku, bez uciekania się do technicznej terminologii informatyki.

Model związków obiektów (ang. entity relationship, ER), od momentu jego zaproponowania przez Chena, stał się najbardziej popularnym formalizmem modelowania danych na poziomie logicznym, początkowo w odniesieniu do baz danych a następnie również innych klas systemów informatycznych. Podstawowymi pojęciami omawianego formalizmu są obiekt (encja), atrybut i związek.

Przyjmuje się, iż obiektem (encją) (ang. entity) jest jednoznacznie identyfikowalny składnik badanej rzeczywistości (dziedziny przedmiotowej), o którym informacja jest, lub może być, zbierana i przechowywana. Obiektem jest każdy składnik badanej dziedziny przedmiotowej, który można nazwać a więc osoba, miejsce, rzecz, zdarzenie, stan, plan czy pojęcie. Obiekt to coś, co może być postrzegane i jest odróżnialne. Obiekt może być rzeczywisty lub abstrakcyjny. Obiekty mają swoje własności (cechy charakterystyczne) zwane atrybutami,

takie jak nazwa, waga, cena, wiek itp. Atrybut jest funkcją przypisującą obiektowi wartość cechy ze zbioru wartości tej cechy czyli dziedziny atrybutu. Obiekty, mające te same atrybuty łączy się w typy obiektów. Typy obiektów utożsamia się z reprezentującymi je obiektami - tym samym nazwy obiekt (encja) używa się do określenia typu obiektu (encji). Opisuje się je rzeczownikami lub wyrażeniami rzeczownikowymi w liczbie pojedynczej. W danym modelu logicznym należy używać różnych nazw dla tych samych lub podobnych atrybutów, przypisanych różnym encjom.

W podziale rodzajowym, określającym zależność atrybutu encji lub związków od innych ich atrybutów, wyróżnia się atrybuty pierwotne i pochodne. Atrybuty pierwotne to podstawowe cechy w danej encji lub związku. Nie oblicza się ich, czy wyprowadza, na podstawie wartości innych atrybutów. Jest to natomiast właściwością atrybutów pochodnych, włączanych do zestawu atrybutów danej encji czy związku, w nawiązaniu do wymagań konkretnych funkcji realizowanych w danej organizacji. Rola atrybutu w procedurach przetwarzania informacji jest kolejnym kryterium ich podziału. Występują tu atrybuty:

- selekcyjne, służące identyfikacji encji i związków,
- opisowe, charakteryzujące werbalnie encje i związki,
- proceduralne, stanowiące cechy ilościowo-wartościowe, wykorzystywane w obliczeniach.

Atrybut (lub zestaw atrybutów), którego wartość w sposób jednoznaczny identyfikuje każdy obiekt w zbiorze obiektów (typie) nazywa się kluczem. Pozwala on na jednoznaczne określenie wystąpienia obiektu w danym obiekcie. Jeśli używa się jednego atrybutu dla określenia encji, mamy do czynienia z kluczem prostym. Natomiast, jeśli w tym celu używa się więcej aniżeli jednego atrybutu, z kluczem złożonym. Z każdym atrybutem związana jest dziedzina atrybutu, stanowiąca zbiór wartości danego atrybutu, które są jemu przyporządkowane. Atrybut jest funkcją, wiążącą encję lub związki z dziedzinami wartości. Dana dziedzina może przybrać postać przedziału wartości, skończonego zestawu wartości, lub wartości binarnych (T lub N, 0 lub 1). Każdemu atrybutowi w danym wystąpieniu encji lub związku przyporządkowana jest tylko jedna wartość z jego dziedziny. Jeśli dziedzina atrybutu jest zbiorem jednoelementowym, to mamy do czynienia nie z atrybutem a stałą.

Związek stanowi naturalne powiązanie pomiędzy dwu lub więcej encjami w badanej dziedzinie przedmiotowej. Związki między obiektami przedstawiają powiązania w świecie rzeczywistym. Zbiór powiązań łączących obiekty należące do określonych typów stanowi określony typ powiązania. (np. *SKŁADA_SIĘ* jest zbiorem „konkretnych" powiązań pomiędzy konkretnymi obiektami typu *PRODUKT* i *CZĘŚĆ*).

W identyfikowaniu i modelowaniu związków należy brać pod uwagę następujące ich cechy:

- stopień (liczebność) związku,
- opcjonalność lub obligatoryjność związku.

Stopień związku, oznacza stosunek ilościowy pomiędzy liczebnościami wystąpień encji, uczestniczących w danym związku. Stopień związku mówi zatem o tym, ile wystąpień encji jednego rodzaju jest powiązanych z iloma wystąpieniami encji innego rodzaju. Istnieją następujące stopnie związku:

- 1 : 1 jeden do jednego np. *KIEROWNIK - WYDZIAŁ*,
- 1 : N jeden do wielu np. *WYDZIAŁ- PRACOWNIK*,
- N : M wiele do wielu np. *MASZYNA - DETAL*.

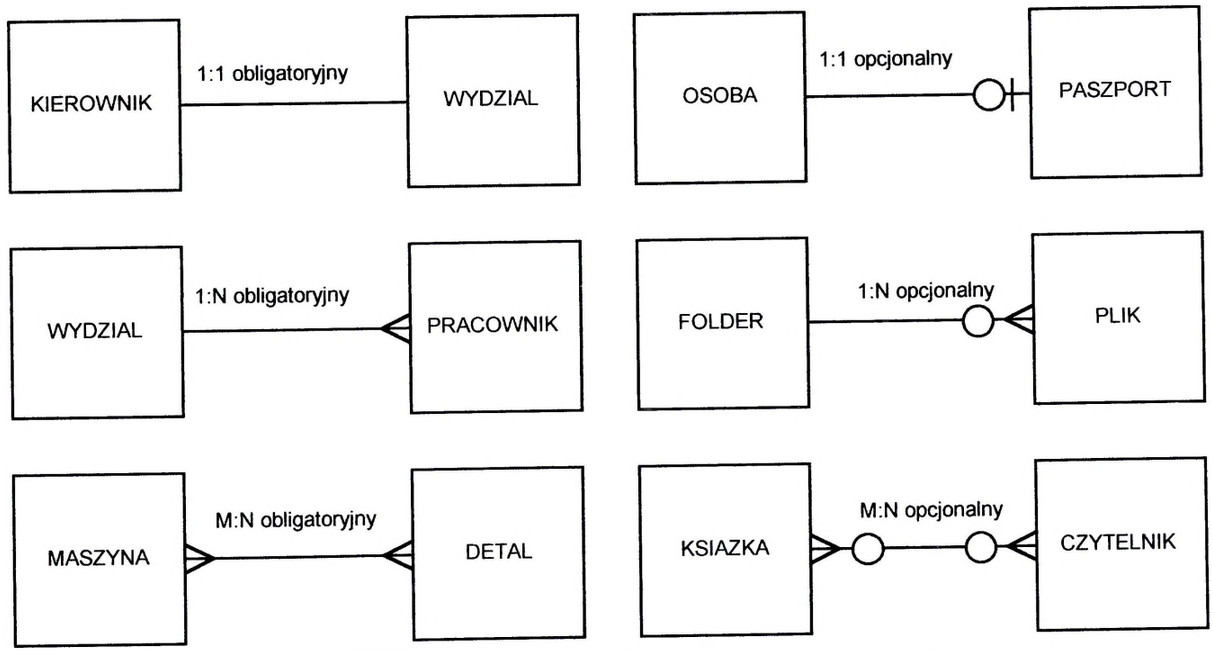
Stopień związków 1:1 oznacza, iż każde wystąpienie encji *KIEROWNIK* jest powiązane z tylko jednym wystąpieniem encji *WYDZIAŁ*. Inaczej - jeden *KIEROWNIK* kieruje jednym *WYZIAŁEM*. W drugim przypadku - każde wystąpienie encji *WYDZIAŁ* powiązane jest z jednym lub wieloma wystąpieniami encji *PRACOWNIK*, przy czym każde wystąpienie encji *PRACOWNIK* powiązane jest z tylko jednym wystąpieniem encji *WYDZIAŁ*. A zatem *WYDZIAŁ* zatrudnia wielu *PRACOWNIKÓW*, natomiast *PRACOWNIK* zatrudniony jest wyłącznie na jednym *WYDZIALE*. Trzecia z kolei opcja oznacza, iż każde wystąpienie encji *MASZYNA* jest powiązane z jednym lub wieloma wystąpieniami encji *DETAL* i odwrotnie każde wystąpienie encji *DETAL* powiązane jest z wieloma wystąpieniami encji *MASZYNA*. Jest to uogólnienie praktycznej sytuacji, w której na każdej *MASZYNIE* obrabia się jeden lub wiele *DETALI* a każdy *DETAL* jest obrabiany na jednej lub wielu *MASZYNACH*.

Rozważane dotąd związki mają charakter obligatoryjny. Analiza sytuacji rzeczywistych, występujących w danej dziedzinie przedmiotowej, wykazuje, że poza obligatoryjnymi, istnieją opcjonalne związki pomiędzy encjami. Związki opcjonalne opisuje słowo „może".

Graficzną reprezentacją modelu danych ER jest diagram ERD (Entity Relationship Diagram) - diagram związków obiektów. Diagram ten posiada różne notacje graficzne. Przyjęta w tym dokumencie notacja zwana jest notacją „kurzych łapek". Typy obiektów są przedstawione za pomocą prostokątów, a linie łączące te prostokąty symbolizują istniejące

powiązanie między konkretnymi obiektami należącymi do łączonych typów. Powiązanie między obiektami w zależności od swojej charakterystyki przedstawiane jest linią z pewnymi „dodatkami" - na przykład z „kurzymi łapkami" po stronie wiele. Inną cechą powiązania może być jego opcjonalność, zaznaczana w postaci małego kółka. Powiązanie między obiektami A i B jest opcjonalne (kółko od strony obiektu B) jeśli obiekt A może wystąpić bez powiązanych z nim obiektów typu B. W przykładzie z OSOBĄ i PASZPORTEM, przy założeniu, że każda osoba może mieć (ale nie musi) dokładnie jeden paszport, powiązanie jest opcjonalne. W innych warunkach, gdyby założenie powyższe nie było prawdziwe, powiązanie nie miałoby tej cechy opcjonalności.

Ilustrację związków o różnych stopniach, opcjonalnych lub obligatoryjnych przedstawia rys 14:



Rys.14. Notacja ERD - przykłady najczęściej używanych rodzajów powiązań między obiektami

Metoda budowania modelu danych systemu prezentowanego graficznie za pomocą ERD jest procesem składającym się z pięciu kroków:

1. identyfikacja (wydzielenie) zbioru obiektów w systemie wraz z ich atrybutami kluczowymi,
2. identyfikacja powiązań bezpośrednich między obiektami oraz ich stopnia,

3. przekształcenie tablicy powiązań w pojęciowy model danych i identyfikacja pozostałych atrybutów obiektów,
4. przekształcenie powiązań M:N na dwa powiązania typu 1:N i identyfikacja dodatkowych atrybutów charakterystycznych dla nowo powstałych obiektów,
5. sprawdzenie poprawności otrzymanej struktury poprzez porównania z wymaganiami systemu.

Prześledźmy sposób budowy modelu danych na przykładzie prostego systemu przetwarzania danych pewnej firmy[7]:

System przetwarzania zamówień w firmie X

Założenia:

- firma ma około 25 000 klientów;
- klienci zamawiają towary składając zamówienie;
- pojedyncze zamówienie może dotyczyć wielu wyrobów;
- każdy klient należy do jednego z 600 rejonów;
- każdy klient zaopatrywany jest z jednego z 20 magazynów;
- klient ma przypisany konkretny magazyn w zależności od rejonu, do którego należy;
- wyroby znajdują się w magazynach.

Aby wymagania użytkownika były spełnione, system musi zapewnić:

- wyszukanie wszystkich zamówień klienta z równoczesnym pokazaniem zamówionych wyrobów oraz magazynów, z których one pochodzą;
- wyszukanie wszystkich klientów, którzy zamówili konkretny wyrób;
- określenie poziomu zapasów każdego wyrobu w konkretnym magazynie;
- określenie poziomu zapasów konkretnego wyrobu w każdym magazynie;
- wyszukanie wszystkich klientów należących do danej grupy.

Krok 1 – identyfikacja obiektów w systemie

Wyróżnienie zespołu oczywistych identyfikatorów (atrybutów identyfikujących w sposób jednoznaczny obiekty):

- numer klienta

---

[7] *Przykład i jego rozwiązanie zostało zaczerpnięte z książki P.Fuglewicza i innych: "CASE dla ludzi", Wydawnictwo LUPUS, Warszawa 1995*

- kod wyrobu
- kod magazynu
- kod rejonu
- numer zamówienia

Identyfikatory umożliwią wyróżnienie pięciu typów obiektów:

- klient
- wyrób
- magazyn
- rejon
- zamówienie

Krok 2 – identyfikacja bezpośrednich zależności między obiektami

	klient	wyrób	magazyn	rejon	zamówienie
klient				x	x
wyrób			x		x
magazyn					
rejon			x		
zamówienie					

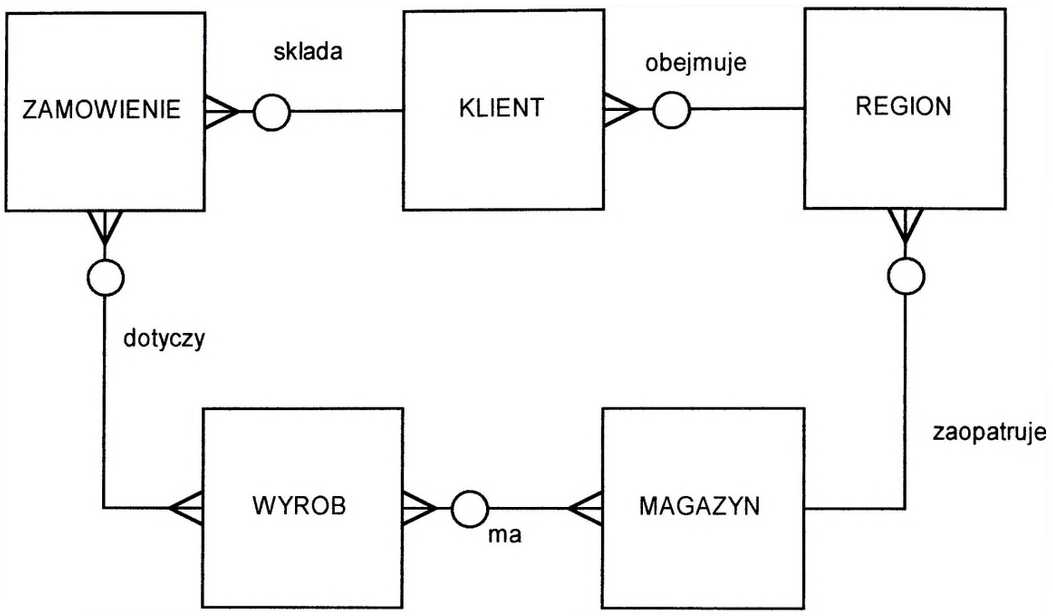
Ustalenie stopnia i opcjonalności związku:

dla związku między obiektami typu A i B należy stwierdzić:

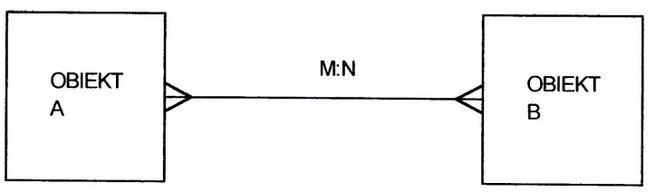
- czy każde wystąpienie obiektu typu A może być powiązane z więcej niż jednym wystąpieniem obiektu typu B (tzn. czy A „posiada" B albo A jest nadrzędne w stosunku do B)
- czy każde wystąpienie obiektu typu B może być powiązane z więcej niż jednym wystąpieniem obiektu typu A (tzn. czy B „posiada" A albo B jest nadrzędne w stosunku do A)

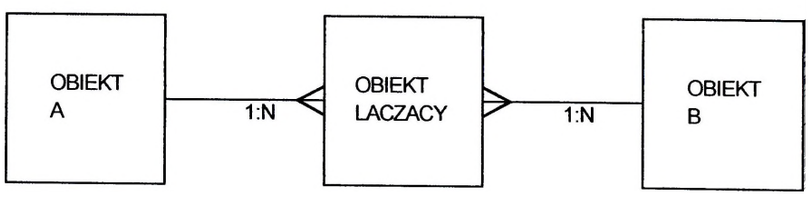
Liczebność powiązania	**A „posiada" B**	**B „posiada" A**
Powiązanie 1:N (A nadrzędny)	T	N
Powiązanie 1:N (B nadrzędny)	N	T
Powiązanie M:N	T	T
Powiązanie 1:1	N	N

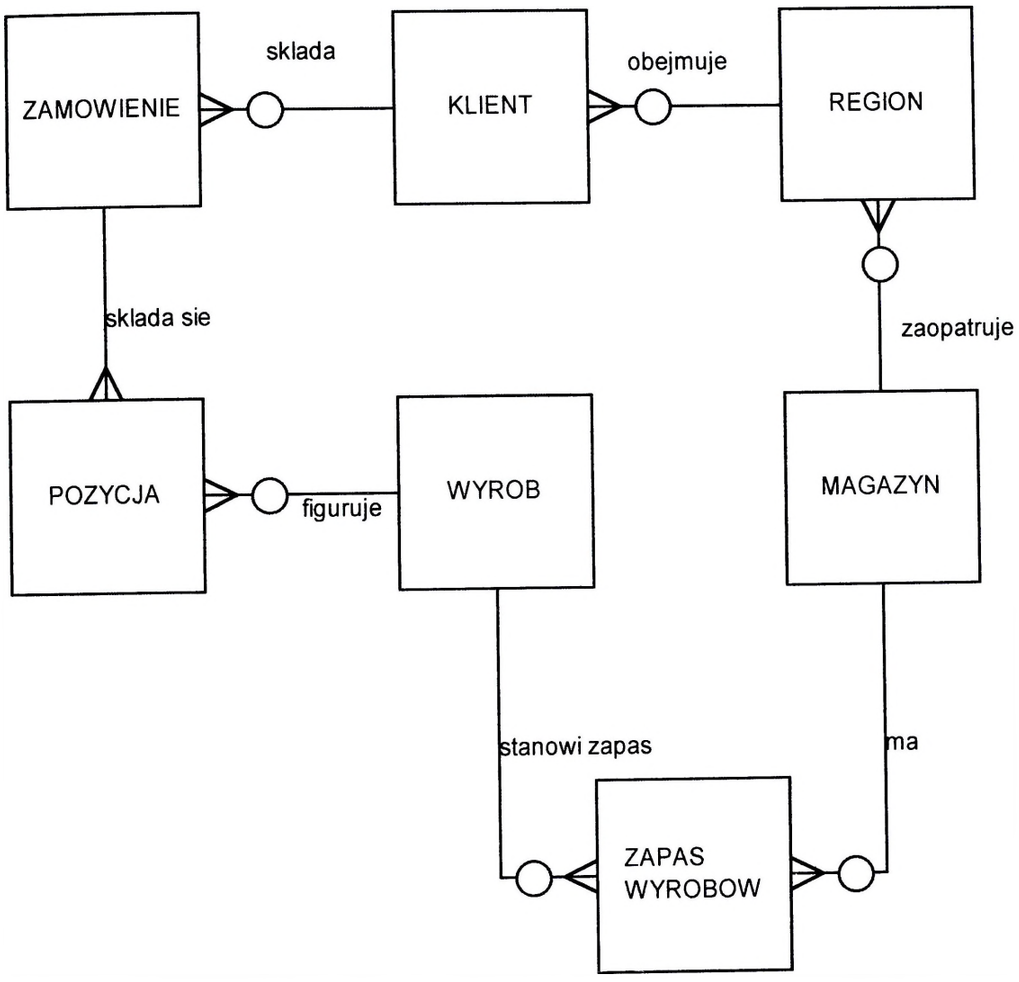
Krok 3 – utworzenie pojęciowego modelu danych



Krok 4 – przekształcenie powiązań typu wiele do wielu







Krok 5 – weryfikacja uzyskanego modelu danych

Sprawdzamy:

- dostęp do danych (atrybutów) danego obiektu jest możliwy, jeśli znany jest identyfikator obiektu
- jeśli znane są atrybuty dla danego wystąpienia obiektu, to dostęp do atrybutów obiektów podrzędnych w stosunku do tego obiektu jest możliwy dzięki powiązaniom stanowiącym ścieżki dostępu.

Diagram DFD przedstawiał funkcjonalny model systemu, odpowiadał więc na podstawowe pytanie, co system ma robić. Model danych w systemie przedstawiony za pomocą ERD służy do zilustrowania statycznej struktury danych w systemie w sposób niezależny od konkretnego systemu zarządzania bazą danych. W wielu pakietach CASE taki

schemat można odwzorować automatycznie w model używany w docelowym systemie zarządzania bazą danych.

Modelowanie związków encji, tworzenie ich diagramów jest techniką przystępną, zrozumiałą dla menedżerów i użytkowników nie mających specjalistycznego przygotowania informatycznego. Jakkolwiek kategorie pojęciowe modelowania związków encji są stosunkowo proste, to sam proces modelowania jest procedurą twórczą, wymagającą dużej znajomości zarówno dziedziny przedmiotowej jak i techniki modelowania. Problemem jest też utrzymanie spójności konstruowanego obrazu danych. O ile modelowanie procesów może podlegać dekompozycji, pozwalającej utrzymywać zależności hierarchiczne, a przez to zmniejszenie stopnia złożoności opisu, to modelowanie danych jest bardziej całościowe. Opracowany model związków encji dotyczy całej dziedziny przedmiotowej i wymaga opanowania wszystkich zależności miedzy elementami składowymi modeli. Bez użytkowania pakietów CASE byłoby to niemożliwe w odniesieniu do bardziej złożonych dziedzin przedmiotowych.

6.3. Diagram historii życia obiektu - Entity Life History (ELH)

Diagram DFD przedstawia funkcjonalny model systemu, precyzuje więc, co system ma robić. Procesy, realizujące funkcje systemu, komunikują się z innymi procesami, składnicami danych i obiektami terminalnymi za pomocą przepływów danych. Przepływy te inicjują działanie procesów – mogą być więc utożsamione ze zdarzeniami uruchamiającymi proces. Zdarzenia te mogą być zdarzeniami mającymi źródło poza systemem. Takie zdarzenia, nazywane zdarzeniami zewnętrznymi, reprezentowane są na diagramie kontekstowym systemu jako przepływy danych od obiektów terminalnych do systemu. Na kolejnych poziomach dekompozycji diagramu DFD występują przepływy danych (zdarzenia) nie wychodzące poza granice systemu, niewidoczne przez otoczenie (środowisko) systemu. Takie zdarzenia, generowane przez system, noszą nazwę zdarzeń generowanych wewnętrznie. Niekiedy zdarzenie w systemie nie jest reprezentowane przepływem danych, jest ukryte w opisie procesu nie podlegającego dekompozycji. Przykładem takiego zdarzenia może być rejestrowany przez system upływ kwantu czasu (godziny, doby, miesiąca).

Model danych w systemie przedstawiony za pomocą diagramu związków obiektów służy do zilustrowania statycznej struktury danych w systemie w sposób niezależny od konkretnego systemu zarządzania bazą danych. Składnice danych, zdefiniowane w modelu funkcjonalnym systemu zawierają obiekty zdefiniowane w modelu danych. Model danych

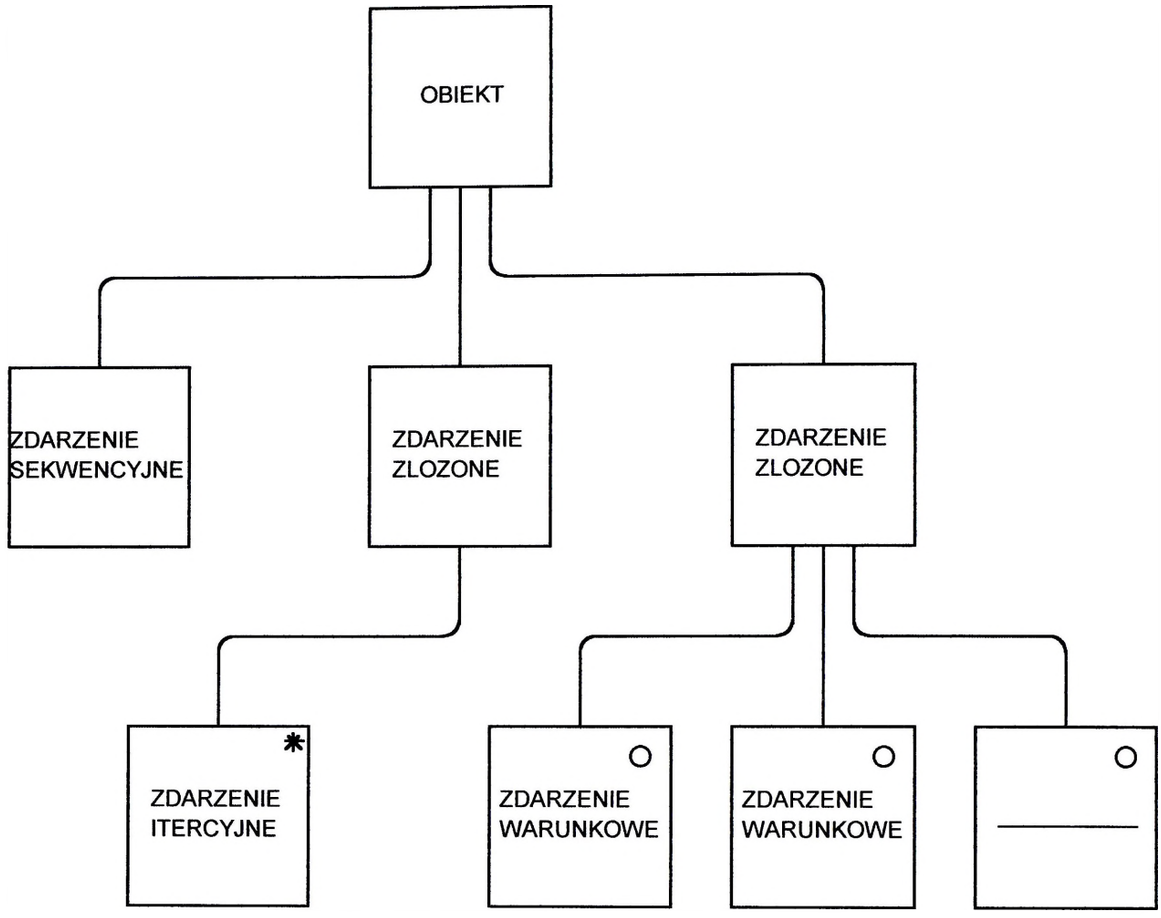
opisuje statyczną strukturę danych, ale zawartość tej struktury, stan obiektów zawartych w systemie ulega dynamicznym zmianom. Powodem tych zmian są zdarzenia, zaś rodzaj tych zmian określają procesy działające na składnicach danych.

Diagramy historii życia obiektów (ELH) są formalizmem modelowania zmian stanu obiektów w czasie. Celem budowanego modelu jest:

- identyfikacja każdego pojedynczego zdarzenia, które oddziałuje na obiekt oraz sekwencji tych zdarzeń w życiu obiektu,
- weryfikacja poprawności konstrukcji DFD – reprezentacji każdego zdarzenia w przepływie danych odpowiedniego procesu,
- zidentyfikowanie sytuacji błędnych oraz warunków wyjątkowych, mogących oddziaływać na obiekt.

Diagram historii życia obiektu ma strukturę hierarchiczną, reprezentowaną przez drzewo, którego korzeń stanowi dany obiekt, zaś węzły tego drzewa odpowiadają zdarzeniom oddziałującym na obiekt. W celu identyfikacji zdarzeń oddziałujących na dany obiekt rozpatruje się jedynie procesy aktualizujące składnicę danych reprezentującą (zawierającą) dany obiekt. Dla każdego takiego procesu rozważa się przepływy aktualizujące składnicę oraz identyfikuje wszystkie zdarzenia, których rezultatem jest dany przepływ. Wyróżnia się trzy podstawowe rodzaje zdarzeń:

- sekwencyjne,
- warunkowe,
- iteracyjne.



Rys.15. Notacja ELH

Diagram ELH ma strukturę drzewa. Korzeniem drzewa jest obiekt, reprezentowany na diagramie (patrz rys15) za pomocą za pomocą prostokąta, w który wpisuje się nazwę obiektu. Zdarzenia sekwencyjne, ustalające kolejność zdarzeń, są reprezentowane jako drzewa składające się albo z sekwencji, albo z selekcji, albo z iteracji zdarzeń. Każdy składnik takiej struktury sekwencji zdarzeń jest reprezentowany jako prostokąt z odpowiednią nazwą zdarzenia. Sekwencja zdarzeń jest czytana od lewej do prawej.

Zdarzenia warunkowe (selekcyjne) reprezentuje grupę opcji, z których może zostać wybrana tylko jedna. Tym która z opcji zostanie wybrana, steruje warunek wyboru. Komponent ELH typu poszczególne zdarzenie warunkowe jest reprezentowany w postaci prostokąta z symbolem opcjonalności w postaci kółka w prawym górnym rogu prostokąta. Dopuszczenie możliwości zaniechania wyboru opcji jest reprezentowane przez wybór tak zwanej opcji zerowej. Symbolem opcji zerowej jest prostokąt bez nazwy zdarzenia (z poziomą kreską).

Iteracja zdarzeń definiuje zdarzenie powtarzające się pewną liczbę razy (możliwe jest zero wystąpień). Reprezentowane jest na diagramie jako prostokąt z symbolem powtarzalności w postaci gwiazdki w prawym górnym rogu.
Podstawowe komponenty ELH mogą tworzyć strukturę o dowolnym stopniu złożoności; grupowane są w sekwencję zdarzeń złożonych, reprezentowanych na diagramie za pomocą za pomocą prostokąta, w który wpisuje się nazwę zdarzenia złożonego.

Tworzenie diagramu historii życia obiektu następuje w dwóch etapach:

1. utworzenie pomocniczej tablicy krzyżowej obiekt/zdarzenia, poprzez:
   - wybranie obiektów z diagramu ERD,
   - identyfikację na podstawie DFD zdarzeń dotyczących danego obiektu;
2. rozważenie dla każdego obiektu z ERD:
   - normalnego cyklu życia,
   - zdarzeń specjalnych (wyjątkowych),
   - sytuacji błędnych.

Dla identyfikacji zdarzeń oddziałujących na dany obiekt na podstawie diagramu DFD bierze się pod uwagę jedynie procesy aktualizujące składnicę reprezentującą dany obiekt. Dla każdego takiego procesu rozważa się przepływy aktualizujące składnice oraz identyfikuje wszystkie zdarzenia, których rezultatem jest dany przepływ. Uzyskana w ten sposób wstępna lista zdarzeń służy do utworzenia tablicy krzyżowej wiążącej obiekty w systemie ze zdarzeniami oddziałującymi na te obiekty. Struktura tablicy krzyżowej jest następująca:

Identyfikator procesu	**Identyfikator składnicy danych**	**Identyfikator przepływu danych**	**Rodzaj aktualizacji:** **I -tworzenie** **R -odczyt** **M –modyfikacja** **D – usunięcie**	**Nazwa zdarzenia**

Tablice krzyżowe są dodatkowym sposobem sprawdzenia poprawności identyfikacji wszystkich zdarzeń dla każdego obiektu. W kolumnie „Rodzaj aktualizacji" tej tablicy muszą znaleźć się litery I (każdy obiekt musi najpierw powstać) oraz co najmniej jedna M (w przeciwnym wypadku świadczy to o braku zdarzeń oddziałujących na ten obiekt). W przypadku większości obiektów powinna się znaleźć litera D (każdy system powinien zapewnić możliwość usunięcia obiektu).

Tworzenie diagramu ELH dla obiektów zaczynamy od rozważania tylko zdarzeń występujących w normalnym cyklu życia wg następującego harmonogramu:

1. wybranie zdarzeń oddziałujących na dany obiekt (z tablicy),
2. ustalenie sekwencji zdarzeń,
3. sprawdzenie, czy pewne zdarzenia mogą zachodzić warunkowo (czy są możliwe selekcje zdarzeń),
4. sprawdzenie, czy pewne zdarzenia mogą powtarzać się wielokrotnie,
5. jeśli występują iteracje, to czy zmieniają one sekwencję zdarzeń,
6. sprawdzenie, czy system traktuje jednakowo wszystkie iteracje zdarzeń danego typu.

Mając jako punkt startowy tablicę wszystkich zdarzeń należy upewnić się, że dla każdego obiektu zidentyfikowano wszystkie zdarzenia, które na niego oddziałują, w czym pomocny może być następujący zestaw pytań:

Jakie zdarzenia powodują, że obiekt:

- pojawia się w systemie?
- opuszcza system?
- zmienia „właściciela"?
- nabywa obiekty podrzędne lub je traci?
- zmienia opcjonalne powiązania?
- zmienia wartości swoich atrybutów?

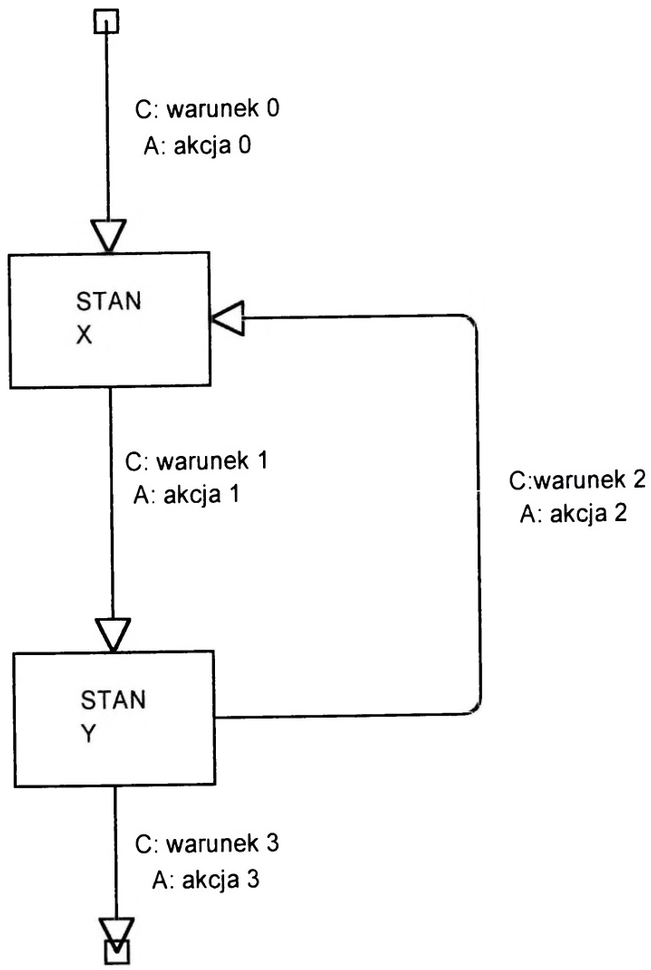
6. 4. Diagram zmian stanów - State Transition Diagram (STD)

Diagram STD, podobnie jak diagram ELH, przedstawia zależności czasowe w projektowanym systemie. W notacji diagramów STD wykorzystywane są trzy typy obiektów (rys. 16). Są to:

- stan systemu; stan w danym momencie czasowym jest przedstawiony w postaci prostokąta; stanem takim może być na przykład „oczekiwanie na naciśnięcie klawisza", „zapisywanie transakcji";
- przejście lub zmiana stanu przedstawiona jest na schemacie w postaci strzałki skierowanej od jednego stanu X do innego stanu Y. Każde przejście pomiędzy stanami musi być opisane przez warunek powodujący przejście (jest to akcja, którą system wykrywa i na którą reaguje, np. naciśnięcie klawisza) oraz akcję, którą system

podejmuje po wystąpieniu warunku, np. wyświetlenie menu. Opis warunku i akcji, najczęściej w postaci równoważnika zdania, umieszcza się obok strzałki symbolizującej przejście. Opis warunku powinien znajdować się nad opisem akcji, oddzielony od niego poziomą linią. Przed tekstem opisującym warunek piszemy często C: (od ang. Condition), a przed tekstem opisującym akcję piszemy A: (od ang. Action).

- interfejs wejścia i wyjścia (ang. interface). Interfejs taki jest rysowany jako mały prostokąt i oznacza, że następuje przejście do stanu na innym diagramie, lub że nastąpiło przejście z innego diagramu. W szczególności dla jednego diagramu sprzęgi reprezentują stan początkowy (sprzęg wejściowy) i końcowy (sprzęg wyjściowy) systemu.



Rys.16. Notacja STD

Diagram STD może być używany jako dobre uzupełnienie DFD, ponieważ można za jego pomocą pokazać następstwo czasowe poszczególnych procesów ze schematu DFD.

Przepływy danych wejściowych mogą tu być warunkami powodującymi zmianę stanu, a dane wyjściowe z procesu można traktować jako akcje.

Ogólne zasady sporządzania diagramów STD można określić następująco:

- system zawsze musi znajdować się w jakimś stanie,
- na diagramie musi znajdować się jeden i tylko jeden interfejs wejściowy, zwykle oznaczany jako START,
- na diagramie musi znajdować się jeden i tylko jeden interfejs wyjściowy, zwykle oznaczany jako STOP,
- każdy stan powinien być dostępny ze stanu początkowego,
- stan końcowy powinien być dostępny dla każdego stanu,
- dany warunek powinien powodować przejście z każdego stanu tylko do jednego innego stanu.

Diagram STD jest najczęściej stosowany w technikach prototypowania (symulacji) różnego rodzaju systemów lub przy modelowaniu dialogu człowiek-maszyna.

7. CHARAKTERYSTYKA ANALITYCZNYCH METOD PROJEKTOWANIA

Wprowadzenie metod analitycznych było odpowiedzią, na problem złożoności systemów informatycznych. Jakkolwiek metody te są bardzo zróżnicowane tak co do podstawowych koncepcji, jak i sposobu ich wyrażania, możemy wyróżnić trzy wspólne ich cechy. Są to:

1. Przywiązanie dużej wagi do etapu analizy systemu, w szczególności do budowy abstrakcyjnego modelu systemu.
2. Zastosowanie zasady rozbioru (dekompozycji) modelu systemu jako podstawowej koncepcji i co się z tym wiąże, zdefiniowanie szczegółowych kryteriów i sposobów dokonywania tej dekompozycji.
3. Zdefiniowanie modelu logicznego oraz modelu fizycznego (tak dla warstwy oprogramowania, jak i dla warstwy sprzętowej) w postaci grafopodobnych struktur przedstawianych za pomoc, ściśle określonych konwencji notacyjnych.

Przedstawione zostaną teraz bardziej szczegółowo w/w cechy w odniesieniu do dwóch najbardziej reprezentatywnych grup metod analitycznych, tj.

- metod strukturalnych;
- metod obiektowych.

W **metodach strukturalnych** występuje bardzo wyraźny podział między etapem analizy a etapem projektowania fizycznego, będącego podstawą, implementacji systemu. W monografiach poświęconych tym metodom wielokrotnie jest powtarzana zasada rozdziału obu etapów, polegająca na tym, że w trakcie budowy modelu logicznego systemu nie powinniśmy brać pod uwagę żadnych czynników związanych z projektowaniem architektury systemu. Może to bowiem niekorzystnie wpływać na proces fizycznego projektowania, ograniczając projektanta systemu w podejmowaniu decyzji leżących w zakresie jego kompetencji. Dlatego też często sugeruje się, aby etapy te rozdzielić personalnie, tzn. aby analityk systemu i jego projektant były dwiema różnymi osobami. Model logiczny systemu w metodach strukturalnych powinien być wyabstrahowany nie tylko z decyzji projektowych ale, tym bardziej, z decyzji implementacyjnych. Powodem takiego ścisłego rozdziału obu etapów jest fakt, że w modelu strukturalnym dokonuje się tzw. funkcjonalnej dekompozycji systemu na procesy, będące logicznymi elementami o charakterze "czarnych skrzynek" przekształcających wejściowe strumienie danych w wyjściowe strumienie danych. Procesy nie mają odpowiednika ani w elementach oprogramowania, ani w składnikach sprzętowych (tzn. nie mają.

odpowiedników na poziomie konstrukcji implementacyjnych), gdyż podstawowym celem jest tutaj formalny (i strukturalny) opis funkcji systemu, a zatem tego, co system ma robić, aby zaspokoić wymagania użytkownika. Pytanie "Jak system ma zrealizować te funkcje ?", zadajemy dopiero w trakcie drugiego etapu, tzn. etapu projektowania, na którym system dekomponujemy na elementy mające implementacyjne interpretacje w postaci programów, procedur, funkcji itp., po czym określamy ich przypisanie do składników sprzętowych. Podsumowując, całkowicie różne cele obu etapów metod strukturalnych powodują przyjęcie różnych kryteriów. Stąd też wspomniany postulat autonomii etapów analizy i projektowania strukturalnego.

Nieco inaczej ta sprawa wygląda w wypadku **metod obiektowych**. Jest to spowodowane tym, że języki obiektowe programowania zawierają, konstrukcje odpowiadające (mniej lub więcej) elementom używanym do budowy modelu (klasy i obiekty). W rezultacie etap analizy w metodach obiektowych pokrywa całkowicie etap analizy w metodach strukturalnych oraz wstępną, fazę projektowania logicznej struktury warstwy oprogramowania w metodach strukturalnych (definicje klas i obiektów). Na etapie projektowania w metodach obiektowych uzupełniamy logiczną strukturę warstwy oprogramowania specyfikacją sposobu realizacji pewnych wymagań narzuconych przez tę strukturę oraz dokonujemy "alokacji" klas i obiektów do modułów oprogramowania (warstwa oprogramowania), a następnie dokonujemy dystrybucji tych ostatnich między składniki sprzętowe, zwane tutaj procesorami (warstwa sprzętowa). Ogólnie jednak w metodach obiektowych większy akcent niż w wypadku metod strukturalnych jest położony na etap analizy zwanej tu projektowaniem logicznym.

Zasada rozbioru modelu systemu jest stosowana w metodach analitycznych dwojako.

Po pierwsze, opis systemu jest rozwarstwiony pod kątem trzech podstawowych aspektów. Tak więc zamiast jednego opisu mamy zwykle do czynienia z trzema jego "rzutami" przedstawiającymi:

- architekturę systemu (w warstwach oprogramowania i sprzętowej),
- jego zachowanie (opis dynamiki),
- jego strukturę logiczną. (tzn. strukturę danych i strukturę funkcjonalną).

Po drugie, dokonujemy dekompozycji systemu w każdym z wyżej wymienionych wymiarów. Dekompozycji tej dokonujemy na zasadzie poziomów abstrakcji, wprowadzonej przez Dijkstrę. Zasada ta polega na podziale systemu na poziomy tworzące hierarchię, z których każdy stanowi uogólniony model systemu w postaci powiązanych ze sobą.

elementów składowych, opisujący jedynie istotne (dla tego poziomu) aspekty systemu i pomijający nieistotne.

W metodach analitycznych wykorzystujemy dwa rodzaje operacji abstrakcji prowadzące do dwóch typów hierarchii:

a) hierarchii kompozycyjnej),

b) hierarchii uogólniającej.

Hierarchia kompozycyjna opisuje rozbiór złożonych obiektów (procesów, struktur danych itp.) na prostsze elementy składowe.

Hierarchia uogólniająca opisuje, jak pewne bardziej szczegółowe kategorie zawierają się pojęciowo (semantycznie) w bardziej ogólnych kategoriach. Dekompozycji systemu w metodach strukturalnych dokonujemy przede wszystkim na podstawie hierarchii kompozycyjnej[8], a w metodach obiektowych na podstawie obu typów hierarchii.

Jakkolwiek szczegółowe kryteria dekompozycji (i ich szczegółowa interpretacja) są. dla każdej z obu grup metod specyficzne, to wspólne jest przyjęcie dwóch podstawowych kryteriów, a mianowicie:

a) kryterium spójności elementów składowych systemu ,

b) kryterium więzi między elementami składowymi systemu.

Przez **spójność elementu składowego systemu** rozumiemy charakterystykę określającą jak mocno są wzajemnie związane lub w jak dużym stopniu odnoszą. się do siebie wewnętrzne składniki pierwotne (procesy, obiekty, procedury itp.), z których ten element jest zbudowany. Na charakterystykę tę składa się wiele czynników, które omówimy szczegółowo w rozdz. 4. Samo kryterium mówi, że powinniśmy tak dekomponować system, aby uzyskać jak największą, spójność poszczególnych jego elementów na wszystkich poziomach abstrakcji.

Kryterium więzi między elementami składowymi stanowi, że dekompozycji systemu należy dokonywać w taki sposób, aby zminimalizować siłę powiązań między tymi elementami na każdym poziomie abstrakcji. Im mniejsza bowiem jest siła tych powiązań, tym bardziej elementy są niezależne, co z kolei powoduje, że cały system jest mniej złożony i dzięki temu bardziej niezawodny.

Na zakończenie tego rozdziału krótko przedstawimy trzy warstwy metod analitycznych projektowania. **Pierwszą i zdecydowanie kluczową„ warstwę** stanowią.

---

[8] *W metodach strukturalnych architekturę systemu dekomponujemy na podstawie hierarchii kompozycyjnej; do dekompozycji struktur danych systemu możemy wykorzystywać również hierarchię uogólniającą*

podstawowe koncepcje ogólnej "filozofii" leżącej u podstaw danej metody. Na koncepcje te maję, wpływ następujące czynniki:

- rodzaj systemów oprogramowania, którymi są, szczególnie zainteresowani twórcy metody,
- uwypuklenie najważniejszych (w opinii twórców metody) źródeł złożoności systemów i problemów związanych z projektowaniem tych systemów.

Spostrzeżenie to jest bardzo ważne, gdyż prowadzi do wniosku, że:

- nie istnieją, uniwersalne metody projektowania,
- nawet jeśli udało nam się dobrać odpowiednią metodę do typu projektowanego przez nas systemu, to na pryncypia metody powinniśmy spojrzeć poprzez specyfikę naszego systemu i traktować je raczej jako wskazówkę i pomoc, a nie jak na absolutne zasady, krępujące nas w trakcie projektowania.

Drugą warstwą, są, praktyczne wskazówki dotyczące procesu projektowania. Są. one zwykle rezultatem wieloletnich doświadczeń związanych z posługiwaniem się daną, metodą. i dlatego mogą. być stosowane z mniejszą, doza krytycyzmu.

I wreszcie **trzecia warstwa** - konwencje rotacyjne, czyli to, co czyni z metod analitycznych "krainę chaosu" i powoduje wiele nieporozumień, zwłaszcza wśród początkujących użytkowników. Najczęściej spotykanym błędem jest redukowanie metod strukturalnych i obiektowych do poziomu graficznych konwencji rotacyjnych, przy równoczesnym braku znajomości dwóch wcześniej wymienionych warstw. Działalność taka jest mało sensowna (moc metod leży w koncepcjach i pragmatyce ich używania, a nie w sposobie wyrażania tego w postaci graficznej), a w dodatku może prowadzić do groźnych w skutkach nieporozumień (gdy np. ktoś "przejmuje" takie "obrazki" i traktuje je jako rzeczywisty opis wymodelowanego i zaprojektowanego systemu).

Mniejszym problemem jest fakt, że niektóre metody to (mniej lub bardziej udane) warianty tematów podstawowych. W pracach, w których są one wprowadzane, często bardzo pobieżnie są omawiane podstawowe koncepcje i pragmatyka, za to uderza (szczególnie w wypadku osób początkujących w tej dziedzinie) odmienność i egzotyka konwencji rotacyjnych oraz terminologii. Omawiając w niniejszej książce metody "klasyczne" (które mogą być układem odniesienia dla innych pochodnych metod) mam nadzieję pomóc Czytelnikowi w odróżnieniu istotnego od nieistotnego oraz w swobodnym poruszaniu się w tej, z pewności, interesującej gałęzi informatyki.

7.1. Metody strukturalne

W procesie kształtowania się metod strukturalnych ogromne znaczenie miała metoda uściślania krokowego zdefiniowana przez Dijkstrę i rozwinięta przez Wirtha. Zasada poziomów abstrakcji Dijkstry, opisuje proces rozpoczynający się od najniższego poziomu systemu, w trakcie którego tworzymy coraz wyższe poziomy abstrakcji - odpowiada zatem metodzie wstępującej; zasada uściślania traktuje proces analizy systemu w sposób zstępujący. Najpierw definiujemy ogólni strukturę systemu docelowego na poziomie 0, a następnie w rekurencyjny sposób uszczegóławiamy definicje elementów poziomu n przez zbudowanie dla każdego z nich podstruktury na poziomie (n +1).

Drugą, niemniej ważna koncepcją., która wpłynęła na rozwój metod strukturalnych, była idea projektowania składanego wprowadzona przez Myersa w 1975 roku). Metoda ta dostarczała konkretnych wskazówek dotyczących przebiegu procesu projektowania. Postępowanie według tych wskazówek prowadziło do dekompozycji systemu w taki sposób, by były spełnione kryteria spójności elementów składowych systemu oraz powiązań między tymi elementami. Ogólną idea projektowania składanego jest rekurencyjna dekompozycja problemu na podproblemy na podstawie analizy struktury problemu oraz sposobu, w jaki struktura ta przekształca dane przepływające przez nią,. Tak więc metoda ta umożliwia efektywne zastosowanie zasady poziomów abstrakcji Dijkstry, której wykorzystanie w procesie konstruowania hierarchii kompozycyjnej omówiliśmy w poprzednim rozdziale, oraz wspomaga koncepcję uściślania krokowego.

Praktyczną realizacją, wprowadzonych powyżej idei (w sensie dostarczenia "symbolicznego języka" oraz sformalizowanych reguł wykorzystania do projektowania systemów informatycznych) była metoda Yourdona - Constantina zaproponowana w 1979 roku. W tej metodzie główny nacisk jest położony na fazę projektowania, dla której wprowadzono diagramy strukturalne reprezentujące w postaci hierarchicznej struktury drzewowej programy, procedury itp. oraz wywołania między nimi. Dla fazy analizy zostały zdefiniowane: diagramy przepływu danych (*data fiow diagrams*) opisujące dekompozycję architektury logicznej systemu w postaci zagnieżdżających się rekurencyjnie procesów oraz diagramy hierarchiczne (*hierarchy diagrams*) reprezentujące hierarchię struktur danych.

Z kolei DeMarco w swoim modelu koncentruje się na fazie analizy, dla której opracował wskazówki metodyczne. Dotyczą one przede wszystkim sytuacji, w których mamy

do czynienia z istniejącym już zaimplementowanym systemem, który ma podlegać dalszej ewolucji. DeMarco zaproponował metodę umożliwiającą, wymodelowanie istniejącego systemu (który nie został zaprojektowany zgodnie z podejściem strukturalnym), a następnie przekształcenie tego "tymczasowego" niestrukturalnego modelu w finalny - strukturalny. Monografia DeMarco z pewnością, przyczyniła się do znacznego upowszechnienia się metod strukturalnych, gdyż opisywała metodę przygotowania *post factum* dokumentacji strukturalnej dla systemów istniejących i wykorzystywania jej w dalszym rozwoju tych systemów.

W modelu, relacyjnym danych, zaproponowanym przez Chena w 1976 roku, dekompozycji systemu dokonuje się nie w odniesieniu do jego funkcji, ale względem przetwarzanych przez niego (złożonych) struktur danych. Diagramy relacyjne danych (*Entity Relationship Diagrams, ERD*) zostały włączone do metod strukturalnych projektowania do opisu struktur danych; sama koncepcja postrzegania problemu nie tyle w kategoriach funkcjonalnych, ile jako zbioru pewnych obiektów, ich atrybutów oraz relacji między tymi obiektami odegrała ważną rolę w kształtowaniu się metod obiektowych.

Kolejne dwie metody, które pojawiły się w połowie lat osiemdziesiątych, były przeznaczone do projektowania systemów czasu rzeczywistego. Ward i Mellor rozszerzyli model Yourdona - Constantina o dwa nowe aspekty, tj. modelowanie struktur danych za pomocą diagramów relacyjnych danych oraz modelowanie zachowania się (dynamiki) systemu, które zdefiniowali za pomocą. automatów o skończonej liczbie stanów, zwanych tutaj diagramami przejść stanowych (*State Transition Diagrarrcs, STD*). W metodzie Hatleya-Pirbhaiego dokonano wyraźnego rozdziału przepływu danych od przepływu sterowania (tzn. przepływu sygnałów sterujących). Dla tego ostatniego wykorzystano diagramy przepływu sterowania (*Control Flow Diagmms, CFD*) oraz specyfikacje sterowania (*Controd Specifications, CSPEC*) jako narzędzie modelowania. Równocześnie bardzo znacznie wzmocniono modelowanie dynamiki systemu przez wprowadzenie wielopoziomowych konstrukcji złożonych z automatów tak sekwencyjnych, jak i kombinatorycznych oraz umożliwienie opisu zachowania się systemu nie tylko w jakościowych kategoriach dyskretnych (stany), ale i charakterystyk sparametryzowanych czasem. W tym celu zdefiniowano diagramy czasowe które obecnie są stosowane również w niektórych metodach obiektowych (np. w metodzie Boocha).

W 1989 roku Yourdon zaproponował syntezę dotychczasowego dorobku metodycznego podejścia strukturalnego w postaci tzw. nowoczesnej analizy strukturalnej Poza jej kompleksowym, w stosunku do omówionych wyżej metod, charakterem, nowym elementem jest wykorzystanie metody prototypowania systemu. W stosunku do

"klasycznego" modelu DeMarco główną różnicą, jest to, że nowoczesna analiza strukturalna opiera się na podziale etapu analizy strukturalnej na dwie fazy: fazę konstrukcji modelu podstawowego odpowiadającego dotychczasowemu modelowi budowanemu na etapie analizy strukturalnej i fazę konstrukcji modelu implementacyjnego, w czasie której dokonuje się logicznej dystrybucji elementów modelu podstawowego między zadania (elementy oprogramowania) oraz procesory (elementy sprzętowe).

7.2. Metody projektowania operacyjnego

Główna ideą, projektowania operacyjnego jest modelowanie struktury systemu oprogramowania na podstawie struktur danych, z jakimi system ma do czynienia. System jest tutaj traktowany jako odwzorowanie (transformacja) danych wejściowych na dane wyjściowe.

W metodzie Jacksona bardzo wyraźnie jest oddzielone modelowanie istotnych właściwości obiektów dziedziny problemu, który system ma obsługiwać, od modelowania samego systemu. To skoncentrowanie się na modelowaniu obiektów - struktur danych sprawia, że metoda ta jest (podobnie jak wspomniany poprzednio model Chena) bliska metodom obiektowym. Proces projektowania rozpoczynamy od etapu analizy, tzn. od identyfikacji obiektów przestrzeni problemu oraz elementarnych akcji, jakie sę, wykonywane przez te obiekty. Każdy taki obiekt jest charakteryzowany jako proces sekwencyjny za pomocy diagramów struktur danych (*Data Structure Charts, DSG*). W kolejnym etapie, zwanym etapem specyfikacji, definiujemy system, który jest modelowany jako tzw. sieć specyfikacji systemu (*System Specificatio - Network, SSN*), składająca się z obiektów - procesów sekwencyjnych. W końcowym etapie, nazywanym etapem implementacji, sieć ta jest uzupełniana pewnymi pomocniczymi procesami o charakterze implementacyjnym (np. generator raportów) i transformowana do dwóch typów diagramów. Diagramy implementacji systemu (*System Implementation Diagram, SID*) reprezentują, dystrybucję elementów systemu do procesorów i procesów fizycznych. Diagramy struktur programu (*Program Structnre Chart, PSC*) opisują organizację kodu dla poszczególnych procesów. Etap implementacji w metodzie Jacksona odpowiada więc etapowi projektowania w metodach strukturalnych.

Tym co, na pierwszy rzut oka, wyróżnia metodę Warniera - Orra spośród innych metod analitycznych jest wykorzystanie języka tekstowego opisu modelu zamiast języka graficznego. Diagramy Warniera - Orra mają postać "tekstu strukturalnego", tzn. poszczególne grupy są wyodrębniane przez ujęcie ich elementów w klamry, a kolejne

poziomy abstrakcji w strukturze wizualizujemy przez "wcinanie" tekstu. Dodatkowo wprowadzamy specjalne oznaczenia dla konstrukcji składniowych typu "wybór", "iteracja". Łatwość w posługiwaniu się diagramami w połączeniu z ich czytelności, powoduje, że są one bardzo często używane do celów dokumentacyjnych w wypadku niewielkich systemów informatycznych. Sama idea projektowania systemu jest bardzo zbliżona do idei metody Jacksona i może być uważana za jej wariant o innych konwencjach notacyjnych i mniej precyzyjnie określonej semantyce.

8. PODSTAWOWE KONCEPCJE PROJEKTOWANIA STRUKTURALNEGO

Szkoła strukturalna wyraźnie rozróżnia etap analizy, w trakcie którego jest definiowany tzw. model logiczny systemu od etapu właściwego projektowania, w trakcie którego jest tworzony tzw. model fizyczny systemu. W niniejszym rozdziale omówimy podstawowe koncepcje, które wpłynęły na ukształtowanie się metodyki modelowania obu etapów. Wprowadzimy też podstawowe pojęcia zdefiniowane w teorii projektowania systemów informatycznych (pojęcia te są używane nie tylko w podejściu strukturalnym, ale też w obiektowym).

Model logiczny jest definiowany przede wszystkim, aby sformalizować wymagania przyszłego użytkownika systemu. Model ten jest więc pewnego rodzaju "pomostem" między użytkownikiem (dla którego projekt fizyczny systemu, budowany pod kątem implementacji, byłby niezrozumiały) a projektantem systemu (dla którego, z kolei, wymagania użytkownika, nieformalne i wyrażone za pomocą, języka potocznego, mogłyby stanowić źródło nieporozumień i wieloznaczności). Innym ważnym celem jest zrozumienie przez konstruktorów specyfiki środowiska użytkownika, tzn. środowiska, w którym system ma działać. Poza tym model taki służy celom dokumentacji projektu. Jak więc widać, stanowi on (zwykle jedyną) podstawę projektowania systemu i w konsekwencji wpływa zasadniczo na końcowy kształt produktu.

W literaturze przedmiotu model logiczny jest zdefiniowany jako model określający, jakie funkcje system musi wykonywać, aby spełnić wymagania użytkownika. Jest to więc opis tego, co system musi robić, bez określania tego, jak system będzie to robił (tzn. bez określania, jak będzie zaprojektowana jego fizyczna struktura, czy jak będzie on zaimplementowany).

Model fizyczny systemu konstruowany w fazie projektowania strukturalnego ma określić, w jaki sposób system powinien być zrealizowany, aby wykonywać funkcje zdefiniowane przez model logiczny. Koncepcje leżące u podstaw modelowania logicznego i fizycznego systemu mają swoje źródło w kryteriach, pod których kątem charakteryzuje się tak pożądane cechy konstruowanego systemu informatycznego, jak i pożądane cechy samego procesu realizacji projektu konstrukcji systemu. Omówimy je teraz pokrótce.

1. ***Zgodność z funkcjonalną specyfikację wymagań).***

Kryterium to wydaje się tak oczywiste, że można pomyśleć, iż nie warto o nim wspominać. Każdy przecież zgodzi się, że system należy skonstruować tak, by realizował zdefiniowane zadania. Problem leży tutaj nie w tym, że analitycy, projektanci czy programiści nie zgadzają się z tym postulatem, ale w tym, że miejsce definiowania tych zadań znajduje się na granicy między przyszłym użytkownikiem systemu (który bądź to definiuje explicite żądania, bądź też, np. w wypadku konstrukcji kolejnych wersji oprogramowania produkowanego na skalę masową, oczekuje spełnienia pewnych wymagań, którym poprzednie wersje nie mogły sprostać) a zespołem konstruującym system. Jest to więc miejsce szczególnie narażone na wystąpienie niejednoznaczności w procesie konstruowania systemu. Konsekwencje nieprawidłowej specyfikacji wymagań są, wręcz katastrofalne dla projektu i końcowego produktu, gdyż, jak wynika z badań statystycznych przeprowadzonych dla dużych projektów informatycznych odpowiadają one ok. 56% błędów w produkcie, a koszt znalezienia takich błędów stanowi aż 82% kosztów związanych z fazą usuwania błędów w gotowym produkcie. Sporządzenie w początkowej fazie projektu specyfikacji produktu adekwatnej do stawianych przez użytkownika zadań umożliwia realizację następnych pięciu postulatów.

2. ***Wydajność).***

Wydajność systemu jest charakteryzowana przez zbiór parametrów odpowiadających "osiągom" systemu (np. czas reakcji na wystąpienie pewnych sytuacji). Pożądana wydajność jako specyfikacja parametryczna systemu jest określana w fazie analizy strukturalnej, a rzeczywista wydajność jest sprawdzana w fazie testów zaimplementowanego systemu.

3. ***Koszty projektu.***

Wydatki poniesione w związku z projektem muszą, zmieścić się w budżecie przeznaczonym na budowę i pielęgnację systemu.

4. ***Czas trwania projektu.***

Cecha ta jest jednym z najsłabszych punktów projektów informatycznych. Czas trwania projektu nie powinien przekroczyć założonego terminu zakończenia prac, a poszczególne etapy powinny być realizowane zgodnie z przyjętym harmonogramem.

5. ***Bezpieczeństwo.***

Kryterium brane pod uwagę szczególnie w wypadku systemów czasu rzeczywistego. Cecha powodująca, że w razie wystąpienia błędu w systemie spadek jego wydajności jest "łagodny" i nie stanowi zagrożenia dla środowiska albo wręcz pełna funkcjonalność i wydajność są. zachowane.

6. ***Przyjazność.***

Cecha określająca łatwość posługiwania się systemem przez użytkownika. Bardzo ważna cecha przy posługiwaniu się systemem przez nieprofesjonalnych użytkowników.

7. ***Niezawodność***

Ta cecha jest zwykle mierzona za pomocą, wskaźnika określającego średni czas między awariami, oznaczanego w literaturze przedmiotu przez MTBF (skrót od angielskiego terminu *Mean Time Between Failures*). Jest cechą, krytyczną w systemach czasu rzeczywistego.

8. ***Pielęgnowalność***

Cecha mierzona za pomoc, wskaźnika, określającego średni czas potrzebny na usunięcie błędu będącego przyczyną. awarii i przywrócenie systemu do stanu używalności. W literaturze wskaźnik ten jest oznaczany przez MTTR (skrót od angielskiego terminu *Mean Time To Repair*).

9. ***Efektywność***

System powinien efektywnie wykorzystywać dostępne (i zwykle niewystarczające całkowicie) zasoby. Przez zasoby nie należy rozumieć tu jedynie zasobów sprzętowych, tj. procesora, pamięci operacyjnej, pamięci zewnętrznej itp., ale również zasoby ludzkie. System wymagający nadmiernego "asystowania" przez operatorów, wyspecjalizowanych fachowców zajmujących się w trybie *on-line* interpretacją, jego zachowania, czy też współpracujący z użytkownikiem w sposób marnujący jego czas jest równie (a może nawet bardziej) nieefektywny co system nieoptymalnie gospodarujący dostępnym sprzętem.

10. ***Modyfikowalność***

Cecha określającą, jak łatwe jest przystosowywanie systemu do zmieniających się wymagań użytkownika (np. rozbudowa systemu).

11. ***Elastyczność***

Cecha określająca, czy i w jakim stopniu system może wykonywać zadania będące nieznacznymi odmianami zadania, dla którego został zaprojektowany i zaimplementowany, bez dokonywania modyfikacji jego implementacji. Do cechy tej przywiązuje się ostatnio coraz więcej wagi.

Pierwsze sześć kryteriów ma silny związek z etapem analizy strukturalnej, którego pierwszą fazą, jest sformalizowanie zbioru wymagań. Wymagania te można podzielić w następujący sposób.

A. Wymagania dotyczące konstruowanego systemu:

- **funkcjonalne** - określające, co system powinien robić, tzn. jakie funkcje powinien realizować (w tym, co system powinien robić w wypadku wystąpienia błędów),
- **parametryczne** (wydajność) - określające, "jak dobrze" system powinien realizować te funkcje,
- **komunikacyjne** - definiujące sposób komunikacji systemu z użytkownikiem (przyjazność), a także z innymi systemami.

B. Wymagania dotyczące samego projektu, w ramach którego system jest konstruowany:

- koszty realizacji,
- czas realizacji.

Omówione w poprzednim rozdziale metody analityczne wspomagają proces specyfikacji wymagań należących do pierwszej grupy i dlatego teraz przedstawione zostaną one nieco dokładniej.

W fazie formalizacji wymagań użytkownika podstawowe zadanie analityka możemy zdefiniować jako opisanie, jak system powinien reagować (np. jakie wyniki podawać, jakie akcje sterujące podejmować w wypadku wystąpienia pewnych zdarzeń (np. w wypadku podania systemowi określonych danych z pewnego zakresu liczbowego, przekazania ;y mu sygnału o awarii jednego z kontrolowanych urządzeń). Oczywiście ten schemat typu bodziec - reakcja jest określany pierwotnie przez przyszłego użytkownika systemu, który dla swojego problemu chce uzyskać rozwiązanie st w postaci systemu informatycznego. Kłopot polega na tym, że użytkownik {a nie potrafi w większości wypadków w sposób jednoznaczny oraz kompletny scharakteryzować nie tylko rozwiązania (czyli wymagań stawianych systemowi), ale i samego problemu. Zauważmy zatem, że zadaniem analityka systemów informatycznych jest nie tylko upewnienie się, iż zaproponowany model logiczny systemu:

- stanowi rzeczywiście rozwiązanie problemu opisanego przez użytkownika (adekwatność modelu w stosunku do pierwotnego opisu problemu) oraz że
- rozwiązanie to obejmuje wszystkie aspekty opisanego problemu, włączając warunki brzegowe, sytuacje awaryjne itp. (kompletność modelu),

ale również doprowadzenie do sytuacji (przez dialog z przyszłym użytkownikiem), w której sam taki opis w stosunku do rzeczywistego problemu jest adekwatny i kompletny.

Dlatego **pierwszym celem konstrukcji modelu logicznego** jest formalizacja samego problemu użytkownika (oczywiście z wykorzystaniem specyfikacji sporządzonej wstępnie

przez użytkownika, ale jednocześnie przy jej weryfikacji), a następnie wymagań odnośnie do rozwiązania tego problemu (tzn. wymagań odnośnie do systemu). Formalizacja służy tutaj właśnie ujednoznacznieniu specyfikacji tak ze względu na adekwatność, jak i kompletność modelu w stosunku do rzeczywistego problemu.

Drugim celem budowy modelu logicznego jest sprawdzenie, czy (i w jakim stopniu) można rozwiązać problem użytkownika przez dostarczenie mu systemu informatycznego w kontekście pozostałych wymagań (tj. parametrycznych, komunikacyjnych, a także dotyczących samego projektu - tzn. kosztów i czasu przeznaczonego na konstrukcję systemu). Wymagania parametryczne i komunikacyjne mogą spowodować z kolei zdefiniowanie dodatkowych warunków realizacji systemu te przez zespół informatyków. Takimi warunkami mogą być:

- konieczność dostarczenia systemowi dodatkowych informacji, które są niezbędne do rozwiązania problemu (warunek niezbędny do spełnienia, wymagania funkcjonalnego),
- zwiększenie kosztów projektu spowodowane koniecznością zakupu narzędziowych pakietów grafiki (warunek niezbędny do spełnienia wysokich wymagań jakości komunikacji z użytkownikiem) lub lepszej platformy sprzętowej (do spełnienia wymagań parametrycznych).

W tej fazie w drodze negocjacji ustalamy kompromis między wymaganiami użytkownika w stosunku do systemu a środkami, jakie chce on zainwestować w projekt. W wypadku dużych projektów, o wysokich kosztach i długim czasie realizacji na tym etapie sugerujemy jedną, z metod tzw. szybkiego prototypowania. Z powyższych rozważań wynika, że aby zweryfikować możliwości realizacji wymagań tak funkcjonalnych, jak i niefunkcjonalnych, analityk systemu powinien zbudować taki model logiczny, który spełnia następujące cztery postulaty. Przede wszystkim jest szczegółowy i oczywiście dobrze sformalizowany oraz poprawny w ramach przyjętego formalizmu. Niemniej ważne jest, aby model logiczny adekwatnie reprezentował rozwiązanie rzeczywistego problemu.

Przedstawione powyżej postulaty warunkują cechy narzędzia (języka) modelowania oraz samego modelu. Postulat szczegółowości modelu oznacza, że skonstruowany model będzie w większości wypadków bardzo złożony. Dlatego też język modelowania musi umożliwiać opis funkcji systemu na różnych poziomach szczegółowości. Jeśli język ma służyć również modelowaniu systemów, w których funkcjonalność jest sparametryzowana czasem (systemy czasu rzeczywistego) lub zmienia się wskutek występowania pewnych zdarzeń, to powinien umożliwiać modelowanie zmian stanów systemu. W wypadku

systemów przetwarzających złożone struktury danych język powinien dawać możliwość definiowania abstrakcyjnego modelu takich struktur. Z postulatu formalizacji modelu wynika, że język modelowania powinien charakteryzować się również wysokim stopniem sformalizowania, gdyż tylko wtedy będzie wystarczająco jednoznaczny, aby reprezentować adekwatnie model. W wypadku postulatu poprawności formalnej modelu wymagamy, aby narzędzie modelowania umożliwiało precyzyjne sprawdzenie, czy skonstruowany model logiczny:

- jest kompletny, tzn. czy na każdym poziomie szczegółowości modelu funkcjonalnego każdy jego element (funkcja) ma wystarczającą informację wejściową, aby zrealizować zdefiniowane dla niego zadanie (wymaganie);
- jest spójny, tzn., po pierwsze, czy na każdym poziomie szczegółowości możemy złożyć model całego systemu z jego elementów, oraz, po drugie, czy przy przechodzeniu z poziomu na poziom zachowujemy jednoznaczne odwzorowanie elementów jednego poziomu w elementy drugiego poziomu,
- nie jest redundantny, tzn. czy wszystkie funkcje są niezbędne do realizacji wymagań oraz czy wszystkie dane wejściowe i wyniki są wykorzystywane w modelu.

Zauważmy, że powyższe postulaty mogą, prowadzić do zdefiniowania narzędzia formalnego, którego wykorzystanie do modelowania daje w rezultacie opis problemu i jego rozwiązania trudny do percepcji przez przyszłego użytkownika systemu (jak również dla analityka i projektanta systemu). Nie byłoby to zgodne z postulatem adekwatności, gdyż wiarygodnego potwierdzenia adekwatności modelu systemu do problemu może udzielić tylko przyszły użytkownik systemu. Dlatego też jako podstawowego narzędzia modelowania logicznego w metodach strukturalnych używamy graficznych języków, które są zrozumiałe nawet dla osób nie przywykłych do sformalizowanych opisów.

Na koncepcje projektowania strukturalnego w największym stopniu wpłynęło pięć kryteriów, tzn. kryteria:

- niezawodności,
- pielęgnowalności,
- efektywności,
- modyfikowalności
- elastyczności.

Teoria i praktyka konstruowania systemów oprogramowania wskazują zgodnie na złożoność systemów jako na główną, przyczynę ich zawodności. Złożoności systemu można uzyskać przez jego dekompozycję na składniki o dużym (tak, jak to tylko jest możliwe) stopniu niezależności. Do podobnych wniosków dochodzimy rozważając pozostałe kryteria. Zauważmy, że pielęgnacja systemu będzie łatwiejsza, jeśli podzielimy go na w miarę małe fragmenty wykonujące w całości pewne (pod)zadania. (W wypadku wystąpienia błędu łatwiej go znaleźć w małym fragmencie. Łatwiej go również znaleźć, jeśli za wykonane zadanie, w którym wystąpił błąd, odpowiedzialny jest jeden składnik, a nie np. pięć składników.) Jeśli system zdekomponujemy na fragmenty, które mogą, być implementowane i modyfikowane oddzielnie (tzn. modyfikacja pewnego składnika nie powoduje konieczności wprowadzania zmian w innych składnikach), to koszty implementacji i modyfikacji będą, stosunkowo niewielkie. Problem konstrukcji modelu fizycznego możemy zatem sprowadzić do problemu jego dekompozycji na podstawowe składniki w sposób, który zapewni jego dalsza optymalną implementację w sensie wyżej wymienionych kryteriów.

Zanim omówimy szczegółowo kryteria dekompozycji systemu fizycznego, wprowadzimy podstawowe w projektowaniu strukturalnym pojęcie modelu (module), używane do określenia podstawowego składnika systemu oprogramowania.

Modułem nazywamy sekwencję instrukcji programu wyodrębniona z tego programu za pomocy ograniczników i mającą identyfikator, przez który możemy odwołać się do niej jako do całości[9]. Teraz postaramy się odpowiedzieć na pytanie: jak dekomponować system, tzn. jak wyodrębniać moduły ?

W początkowej części pracy przedstawione zostały dwa podstawowe kryteria dekompozycji:

- kryterium spójności modelów,
- kryterium więzi międzymodelowych,

które są używane w metodach strukturalnych oraz obiektowych. Mówimy, że dwa moduły są mocno związane, jeśli zachodzą między nimi silne wzajemne relacje, uniemożliwiające zrozumienie działania jednego z nich bez znajomości drugiego. Moduły niezwiązane są niezależne w powyższym sensie, co oznacza, że mogą. być implementowane i modyfikowane osobno.

---

[9] *W metodach strukturalnych sekwencję instrukcji programu wyodrębnioną z programu za pomocą ograniczników i nie mającą identyfikatora nazywamy w metodach strukturalnych segmentem.*

W literaturze definiuje się cztery główne czynniki wpływające na siłę więzi. Omówimy je teraz charakteryzując w punktach możliwe sytuacje powstałe po modularyzacji systemu (używając numeracji - od najlepszej do najgorszej sytuacji).

A. Ze względu na rodzaje odwołań między modelami wyróżniamy trzy rodzaje systemów.

1. ***System połączony minimalnie*** - to system, w którym wszystkie wywołania modułów (i powroty z nich) ograniczaj się do pojedynczych (unikatowych) punktów przekazywania sterowania, a wszystkie dane są przekazywane przez parametry (wejściowe i wyjściowe).

2. ***Systemem połączonym normalnie*** nazywamy system, w którym dopuszczamy:

- więcej niż jeden punkt wejścia do wywoływanego modułu (przy dalszym zachowaniu parametrycznego przekazywania danych),
- więcej niż jeden punkt powrotu do wywołującego modułu pod warunkiem, że alternatywne punkty powrotu są jawnie określone w tym module.

3. Jeśli system nie jest połączony ani minimalnie, ani normalnie, to jest to ***system połączony patologicznie*** [10].

B. Przepływ komunikacji między modułami w systemie może mieć charakter:

1. Jednokierunkowego przepływu danych.
2. Dwukierunkowego przepływu danych.
3. Jednokierunkowego przepływu sterowania.
4. Dwukierunkowego przepływu sterowania.

Im wyższego poziomu jest typ komunikacji (przy typach będących kombinacjami powyższych typów bierzemy pod uwagę typ o największym indeksie), tym silniejsza więź występuje między modułami.

C, Wyróżniamy cztery podstawowe sposoby wiązania modułów.

1. Moduły łączy ***więź danych***, jeśli wszystkie dane w trakcie ich komunikacji są przekazywane przez parametry (wejściowe i wyjściowe). W tym wypadku wyróżniamy jeszcze: bardziej preferowane przekazywanie parametrów przez wartość (przekazujemy "kopię" danych do modułu wywoływanego) oraz to

---

[10] *Oczywiście należy unikać odwołań "patologicznych", ale praktyka inżynierii oprogramowania pokazuje wiele wypadków, gdzie dla celów optymalizacji takie odwołania stosuje się. (Autor przytacza takie stwierdzenie jako fakt, nie jako zachętę do konstrukcji systemów z połączeniami patologicznymi.) Dlatego Yourdon wprowadza nawet do konwencji notacyjnych używanych w fazie projektowania oznaczenie połączenia patologicznego.*

przekazywanie parametrów przez adres (przekazujemy rzeczywisty adres miejsca, gdzie dane się znajdują i wywoływany moduł operuje bezpośrednio na tych danych).

2. Moduły są, połączone ***więzią cechy***, jeśli odwołują, się do wspólnej dla nich (ale nie globalnej) współdzielonej struktury danych.
3. Moduły są, połączone ***więzią wspólną*** jeśli odwołuj, się do tej samej globalnej struktury danych.
4. Moduły są związane treściowo, jeśli jeden z nich odwołuje się bezpośrednio do treści drugiego.

D. Ze względu na złożoność interfejsu między modelami mamy do czynienia z:

1. Interfejsem prostym (niewielka liczba parametrów)
2. Interfejsem złożonym (duża liczba parametrów).

Złożoności interfejsu nie należy mylić, oczywiście, z ilością danych przekazywanych między modułami.

Przejdźmy teraz do kryterium spójności modelów. W metodyce projektowania strukturalnego wyróżniamy osiem stopni "mocy" spójności modułów (omówimy je od najlepszej do najgorszej sytuacji).

1. ***Moc funkcjonalna*** występuje, gdy każdy element składowy modułu jest niezbędnym elementem realizacji dobrze określonej (pojedynczej, niezłożonej) funkcji przypisanej temu modułowi.

2. ***Moc informacyjna*** występuje, gdy jeden moduł wykonuje kilka dobrze określonych (pojedynczych, niezłożonych) wzajemnie niezależnych funkcji, z których każda odnosi się do tej samej struktury danych. Moduł o mocy informacyjnej jest ekwiwalentem zbioru modułów o mocy funkcjonalnej operujących na "ukrytej" strukturze danych. Dekompozycja systemu na takie moduły jest celem metod obiektowych, omawianych w części III książki.

3. ***Moc sekwencyjna*** modułu oznacza, że zgrupowano w nim sekwencję elementów realizujących kolejne etapy przetwarzania pewnych struktur danych, tzn. dane wyjściowe - wyniki poprzedniego etapu są, danymi wejściowymi następnego etapu.

4. ***Moc komunikacyjna*** modułu występuje, gdy moduł składa się z elementów realizujących pewien algorytm, przy czym elementy te operuj, na tej samej strukturze danych lub produkują te same dane wyjściowe.

5. ***Moc proceduralna*** (algorytmiczna) modułu oznacza, iż składa się on z elementów, które realizuj, w całości pewien algorytm (procedurę) wykorzystywany do rozwiązania

określonego zadania w systemie (przy czym poszczególne elementy nie muszą, być powiązane ze względu na przetwarzane struktury danych).

6. ***Moc klasyczna*** w terminologii anglojęzycznej jest określana tzn. moc związana z faza (określonym okresem czasu) procesu przetwarzania danych. Moduł o tego rodzaju mocy zawiera funkcje, które są wykonywane na pewnym etapie (lub pewnych etapach) działania systemu. Do tej klasy należą,, przykładowo, wszystkie moduły "inicjujące" lub "kończące" działanie systemu.

7. ***Moc logiczna*** modułu jest rezultatem zgrupowania w nim wielu, w istocie różnych, funkcji należących do tej samej klasy ze względu na rodzaj procesu przetwarzania danych. Przykładem takiego modułu może być moduł wprowadź dane odpowiedzialny za wprowadzanie wszystkich danych do systemu (niezależnie od źródła tych danych, ich znaczenia i roli, jaki odgrywają, w systemie).

8. ***Moc przypadkowa*** modułu oznacza, że między jego elementami składowymi nie występuj, znaczące relacje. Z sytuacją~ taki. mamy często do czynienia, gdy projektant lub programista "modularyzuje" system przez zbieranie w modułach identycznych sekwencji rozkazów występujących w różnych miejscach w systemie oprogramowania.

Spójność modułowa jest jednym z najbardziej "wymiernych" kryteriów używanych przy ocenie modularyzacji systemów informatycznych. W literaturze możemy spotkać nawet liczbowe skale, które mogą być używane do mierzenia jakości modularyzacji.

Na przykład Yourdon i Constantine sugerują następującą. skalę;

10 - moc funkcjonalna,

9 - moc sekwencyjna,

7 - moc komunikacyjna,

5 - moc proceduralna (algorytmiczna),

3 - moc klasyczna,

1 - moc logiczna,

0 - moc przypadkowa.

Mocy informacyjnej - najbardziej pożądanej w metodach obiektowych należałoby tu przypisać wskaźnik 10.

W metodach strukturalnych do dekompozycji systemu na moduły, tzn. do budowy modelu fizycznego systemu, używamy, podobnie jak w wypadku przedyskutowanej wcześniej konstrukcji modelu logicznego, języków graficznych modelowania.

8.1. Schematy strukturalne - Structure Charts (STC)

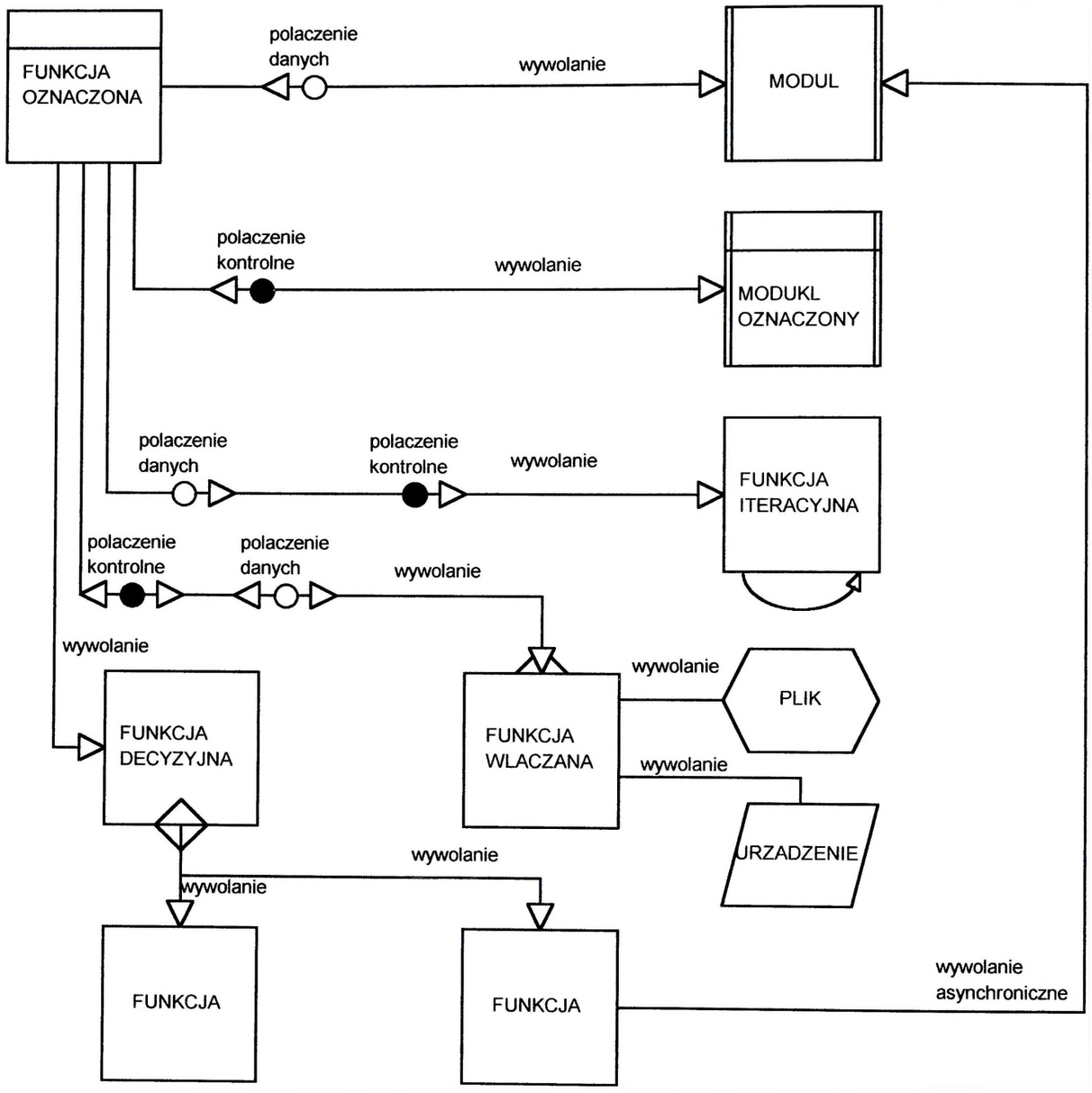
Diagram STC (ang. Structure Charts) przedstawia strukturę systemu w podziale na moduły (procedury), funkcje, hierarchię wywołań funkcji i modułów oraz zależności pomiędzy nimi. Użyteczność diagramu STC implikowana jest posiadaniem przez niego następujących cech:

- reprezentuje hierarchię programu, pokazując nadrzędność i podrzędność funkcji oraz sekwencję wywołań,
- moduły na diagramie odpowiadają już zrealizowanym modułom (funkcjom) bibliotecznym, bibliotekom DLL lub pakietom,
- pokazana jest komunikacja między funkcjami i (lub) modułami, poprzez zaznaczenie przekazywanych informacji i danych sterujących (flag).

Zwykle funkcje (moduły) diagramu STC są realizacją procesów z diagramu DFD zaś przepływy danych odpowiadają połączenia. Przyjęta w schematach strukturalnych notacja jest następująca (rys.17):

- funkcje są przedstawione w postaci prostokątów,
- funkcje specjalne (oznaczone, iteracyjne, decyzyjne, dołączane) są reprezentowane prostokątami ze specjalnymi symbolami.
- wywołania oznaczane są strzałką skierowaną w stronę funkcji wywoływanej,
- wywołania asynchroniczne (np. wywołanie funkcji przez przerwanie od urządzenia zewnętrznego lub od innego procesu) oznaczane jest urządzenia zewnętrzne linią przerywaną.

Bardzo istotnym elementem są połączenia. Istnieją dwa rodzaje połączeń: połączenia danych i połączenia kontrolne. Połączenie danych pokazuje przepływ danych (rekordów, parametrów) od funkcji do funkcji i jest zaznaczone jako strzałka wychodząca z pustego kółka. Połączenie kontrolne jest przedstawiane za pomocą strzałki wychodzącej z zapełnionego kółka. Połączenie kontrolne pokazuje przepływ wszelkich informacji kontrolnych, takich jak na przykład sygnał końca zbioru (EOF) lub braku rekordu.

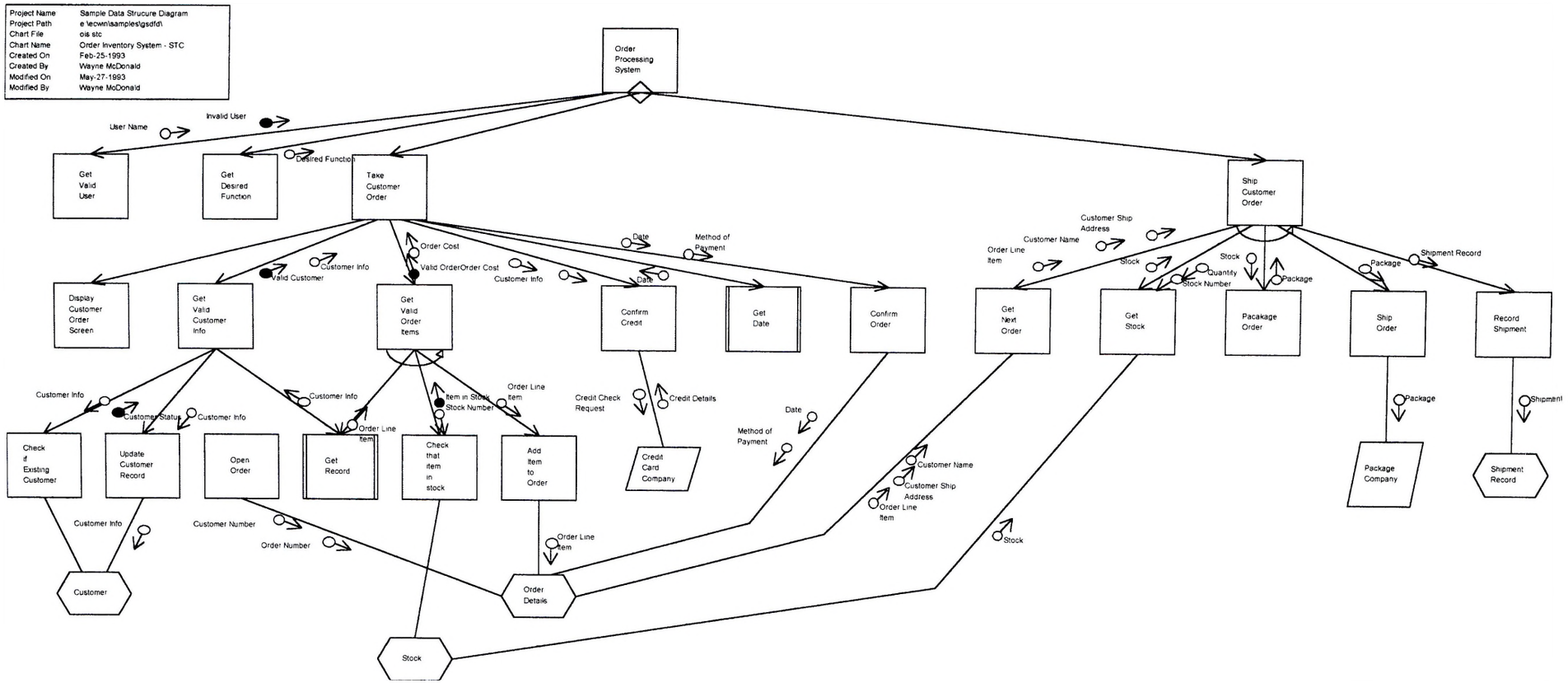


Rys. 17. Notacja STC.

Na rys. 18 przedstawiono przykładowy diagram STC systemu obsługi sprzedaży wysyłkowej.

Za pomocą diagramów STC można generować tak zwane szkielety programów. Generowanie szkieletów wymaga specjalnego narzędzia, które tworzy niekompletne funkcjonalnie, ale poprawne syntaktycznie programy w zadanym języku programowania. Szkielety te mogą zawierać wszystkie odwołania do podprogramów, komentarze przenoszone z diagramów, odwołania do modułów bibliotecznych itp. Tak przygotowane programy wystarczy tylko wypełnić docelowym kodem i już mamy gotowy system. Oczywiście w

praktyce sprawa jest bardziej skomplikowana, niemniej warto od razu wygenerować sobie listę procedur do napisania, chociażby w celu oszacowania czasu ich zakodowania.



Rys. 18. Przykładowy diagram STC

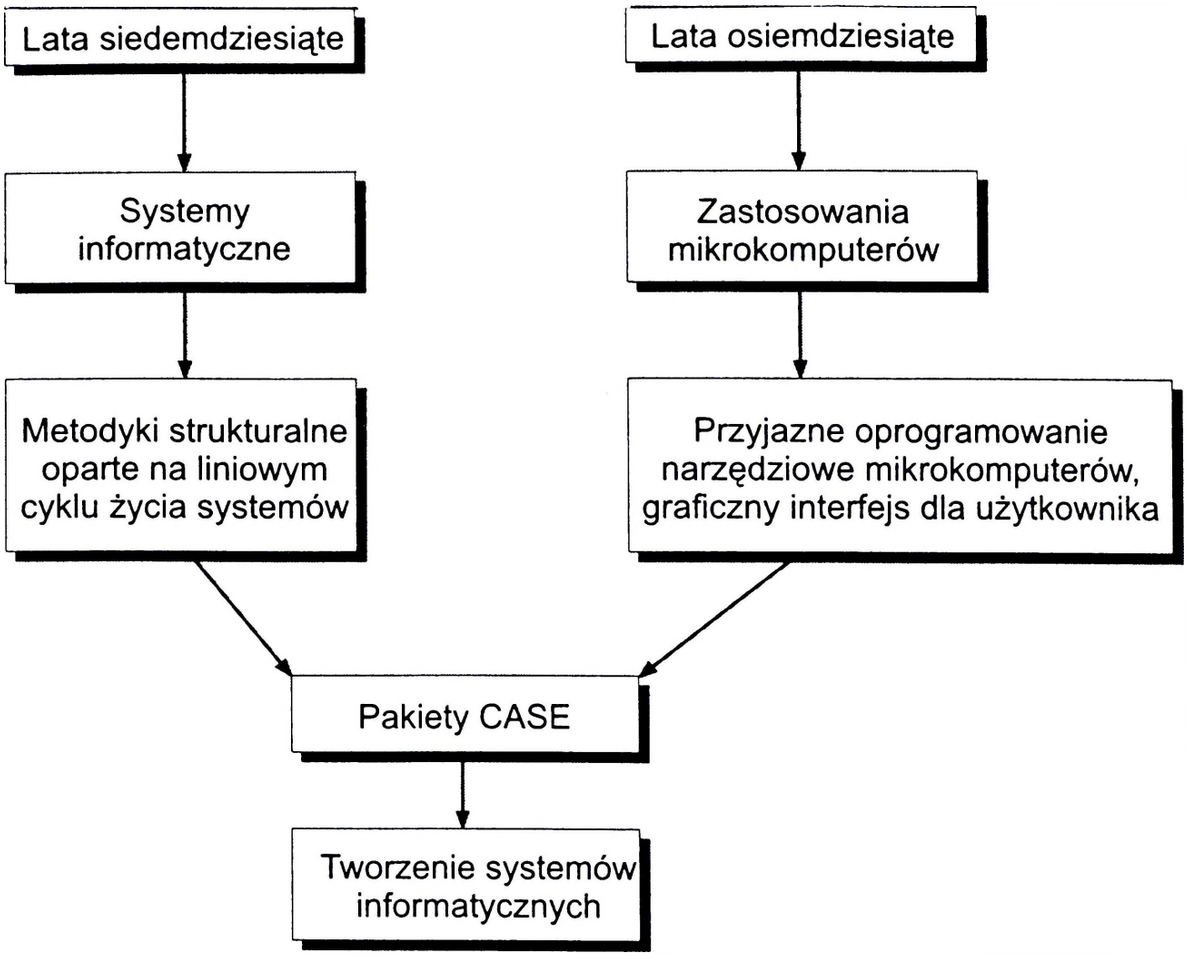
9. KOMPUTEROWO WSPOMAGANE TWORZENIE SYSTEMÓW INFORMATYCZNYCH. PAKIETY CASE

9.1. Pakiety CASE - istota i generacje

Idea wspomaganego komputerem tworzenia systemów informatycznych (TSI) nie jest nowa. Już na początku lat siedemdziesiątych przeprowadzono w tej dziedzinie pewne badania, eksperymenty i wdrożenia. Chodzi tu przede wszystkim o pakiet PSL/PSA, opracowany na Uniwersytecie Michigan. Prawdziwy jednak przełom w tej dziedzinie nastąpił w latach osiemdziesiątych, wraz z masowym użytkowaniem mikrokomputerów. Stworzyły one możliwość powiązania dotychczasowego bogatego dorobku w sferze metodyk TSI z cechą przyjazności oprogramowania. Powstają w ten sposób liczne pakiety wspomagające proces TSI, nazywane powszechnie w świecie CASE, (od ang. *C - computer, A - aided, assisted, S - software, system, E - engineering).*

Sformułowanie zostało po raz pierwszy użyte przez prof. J. Manleya z Carnegie Mellon University w 1981 r. Określa on tym pojęciem narzędzia, które wspomagają zautomatyzowane tworzenie systemów informatycznych w trakcie cyklu życia systemu. Pakiety CASE stanowią zastosowanie technologii komputerowej w odniesieniu do procesów, technik i metodyk tworzenia systemów informatycznych. Spotyka się również zamiennie skrót IPSE (ang. *integrated project supporting environment).* Niektórzy autorzy podkreślają, że chodzi w tym przypadku o pakiety wspomagające przede wszystkim inżynierię oprogramowania lub kierowanie projektami. Nazwa CASE jest skrótem powszechnie stosowanym w różnych językach. Do powstania technologii CASE przyczyniła się wiedza metodologiczna o tworzeniu systemów informatycznych, a następnie rozwój mikrokomputerów. Drogę rozwoju pakietów CASE przedstawia rys..

Jak wynika z definicji, pojęcie technologii CASE jest bardzo pojemne, toteż włączono do tej kategorii różnorodne narzędzia i pakiety z dziedziny inżynierii oprogramowania. Była to oczywiście reakcja twórców oprogramowania na impulsy rynkowe. Jednakże genezy pakietów CASE należy upatrywać w próbach komputerowego wspomagania użytkowania bądź automatyzowania faz analizy i projektowania systemów informatycznych, a więc w podstawach metodologicznych TSI. Odtąd pakiety CASE są nierozłącznie związane z metodykami TSI. Powszechnie toruje sobie drogę przekonanie, iż bez swoistego przewodnika - metodyki TSI - wykorzystanie pakietu CASE może mieć ograniczone bądź wręcz negatywne oddziaływanie.



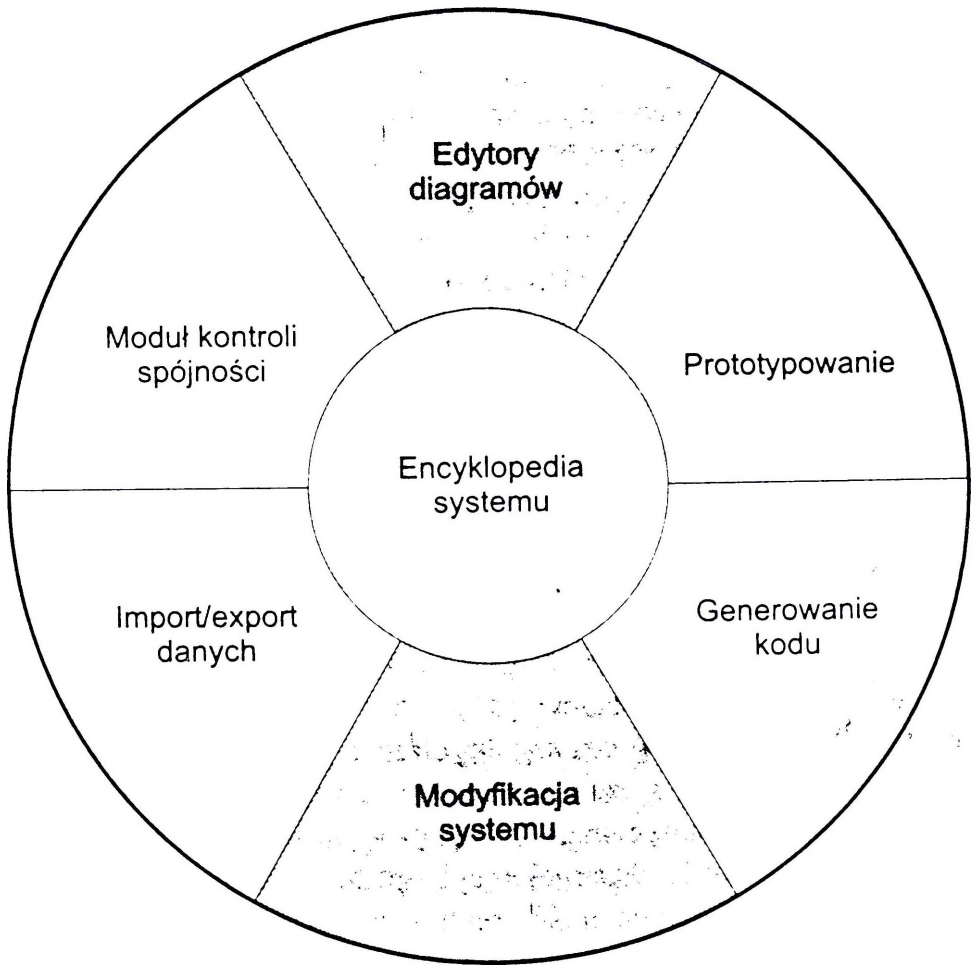
Rys.19. Rozwój pakietów CASE

Wprawdzie historia technologii CASE jest krótka, lecz można już mówić o dwu jej generacjach. Pierwsze narzędzia wycinkowo wspomagały użytkowanie poszczególnych technik bądź realizację pewnych fragmentów cyklu życia systemu. Zadania te realizowano autonomicznie, bez zapewnienia automatycznego przepływu opisów pomiędzy kolejnymi fazami, lub wykorzystania wspólnej encyklopedii systemu. Pierwsze produkty CASE eliminowały konieczność realizacji rudymentarnych czynności analityków, projektantów i programistów, zwłaszcza poprzez komputeryzację generowania, aktualizacji i modyfikacji różnego rodzaju diagramów.

Doświadczenie w użytkowaniu pakietów CASE oraz logika metodyk tworzenia systemów skłaniały ku ich integracji w pakiety wspomagające cały proces TSI. W ten sposób, począwszy od 1986 roku powstawały kolejne pakiety CASE (drugiej generacji) zwane **pakietami zintegrowanymi**, czyli I-CASE. Integracja przebiegała wokół pełnej realizacji cyklu życia systemu oraz encyklopedii zawierającej wszystkie opisy systemu (diagramy, macierze, tablice, opisy słowne) oraz powiązania między nimi.

Podstawowym składnikiem pakietu CASE (rys.20) jest encyklopedia systemu, zwana

również repozytorium. Można ją określić jako bazę danych twórców systemu - zbiornicę wszystkich obiektów występujących w danym systemie oraz zależności między nimi wraz z modułem zarządzania obiektami. Obiekty te to różnego rodzaju specyfikacje systemowe, jak diagramy przepływu danych, związków encji, Jacksona, diagramy struktury, schematy baz danych, prototypy systemów, formatki I ekranów czy zestawień wynikowych, definicje dialogu, menu i programy zastosowań. W ten sposób encyklopedia systemu jest składnikiem integrującym poszczególne pakiety czy środowiska CASE, wiąże się z wszystkimi fazami cyklu życia systemu, a poza tym z czynnościami kierowania projektami, adaptacji i modyfikacji systemów oraz bieżącego doskonalenia ich jakości.



Rys.20. Podstawowe składniki pakietu CASE

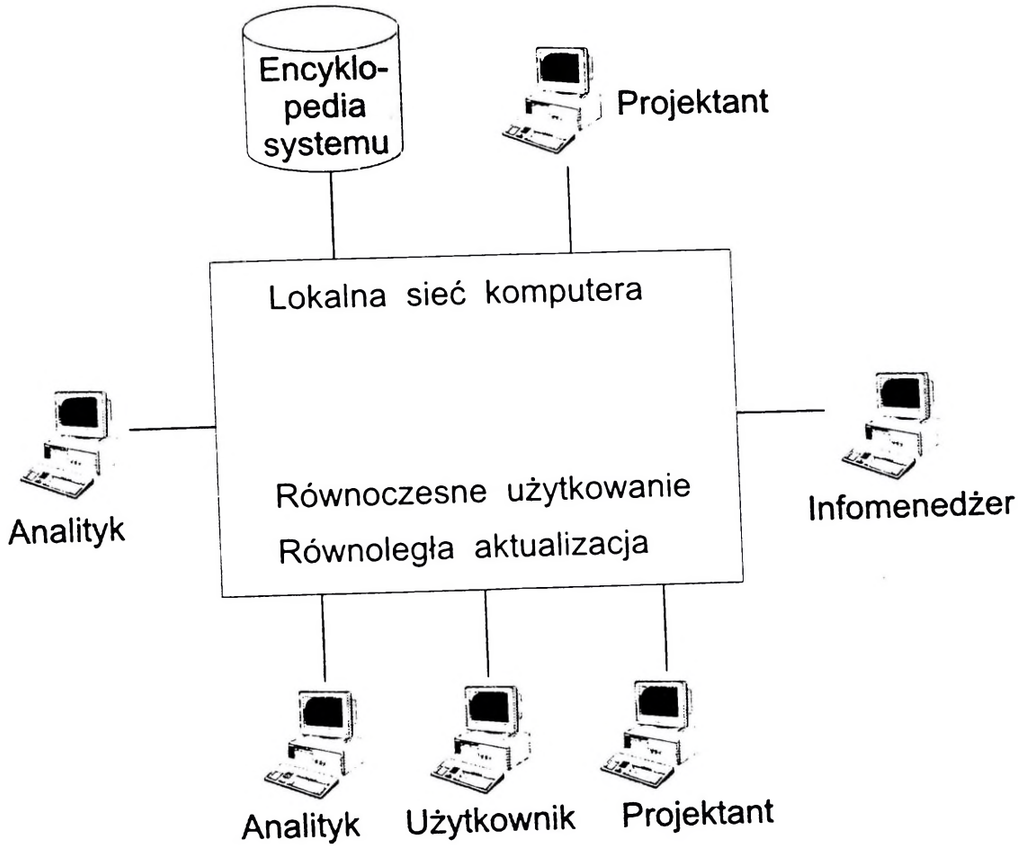
Pierwszym czynnikiem, który przyciąga uwagę twórców systemów do pakietów CASE, są ich **możliwości graficzne.** Prawdziwa jednak ich wartość przejawia się w możliwościach encyklopedii systemu. Poza encyklopedią systemu składnikami pakietu CASE najczęściej są:

- edytory diagramów różnorodnych technik TSI,

- generator kodu programowego,
- moduł prototypowania systemów,
- moduł modyfikacji i adaptacji systemów,
- moduł eksportu / importu danych,
- moduł kontroli spójności systemu.

Dodatkowe moduły występujące w różnych pakietach to: edytory werbalnych opisów systemu (powiązane często logicznie z edytorami graficznymi), moduły zapytań i generowania zestawień, przygotowanie dokumentacji systemu, kontroli wersji projektu i poufności danych, samouczki. Pakiety obejmujące wszystkie lub większość wymienionych modułów, charakteryzują się na ogół wysokim stopniem integracji, wzajemnej współzależności i sekwencji użytkowania. Coraz większego znaczenia nabiera jednak kwestia autonomiczności modułów. Na przykład generator kodu APS w pakiecie Excelerator może być użytkowany niezależnie, z pominięciem odpowiedniego modułu wysokiego poziomu. Specyfikacje do generatora przygotowuje się autonomicznie na zasadzie prototypowania. Oprogramowanie CASE ma wówczas charakter **modularny** i określa się go mianem M-CASE.

Pakiety CASE użytkowane są najczęściej przez pojedynczego twórcę w odniesieniu do danego projektu. Tworzy on lokalną encyklopedię tego projektu, mieszczącą się w ramach centralnej encyklopedii, zawierającej pełne informacje o zrealizowanych, aktualnych i planowanych systemach informatycznych danej organizacji. Systemy informatyczne powstają jednak w **zespołach.** Oczywiście możliwa jest równoległa praca projektantów na różnych modułach danego pakietu CASE. Bardziej pożądane jest jednak jednoczesne modelowanie danych czy procesów przez zespół projektantów. Rozwiązanie to jest możliwe dzięki **sieciom** i oprogramowaniu sieciowemu. Lokalna encyklopedia systemu może być aktualizowana przez cały zespół równolegle, dzięki serwerom sieciowym. Stawia to określone zadania przed modułem zarządzającym encyklopedią systemu, który powinien zapobiegać wprowadzaniu zmian do danej specyfikacji przez dwu lub kilku twórców dokładnie w tym samym czasie. Zespołowe tworzenie projektu systemu informatycznego w warunkach użytkowania pakietu CASE w sieci przedstawia rys.21.



Rys. 21. Użytkowanie pakietu CASE w sieci

Aktualne udoskonalenia pakietów CASE drugiej generacji zmierzają ku:

- uzupełnieniu o możliwości technik projektowania i programowania systemów w czasie rzeczywistym,
- zastosowaniu baz wiedzy i systemów ekspertowych dla doradztwa w zakresie alternatywnych rozwiązań.

9.2. Rodzaje pakietów CASE

Duża różnorodność pakietów CASE na rynku oprogramowania (jest ich aktualnie kilkaset) stwarza pewne problemy z ich klasyfikacją. Czynnikami, które wpływają na brak precyzji tej klasyfikacji są: szybki rozwój i doskonalenie pakietów, zmieniająca się oferta rynkowa, tj. proponowanie nowych pakietów przy jednoczesnym zanikaniu już wprowadzonych, różnorodność metodyczna, brak standardów oraz niejednolitość pakietów obejmujących zmienny obszar tematyczny. Podstawowe kryteria klasyfikacyjne to:

- kompleksowość,
- otwartość,

- elastyczność metodologiczna,
- rodzaj wspomaganego modelu dziedziny przedmiotowej.

Najczęściej przyjmowanym kryterium podziału jest kompleksowość narzędzi CASE. Z tego punktu widzenia dzieli się je na:

- cząstkowe,
- pośrednie,
- zintegrowane.

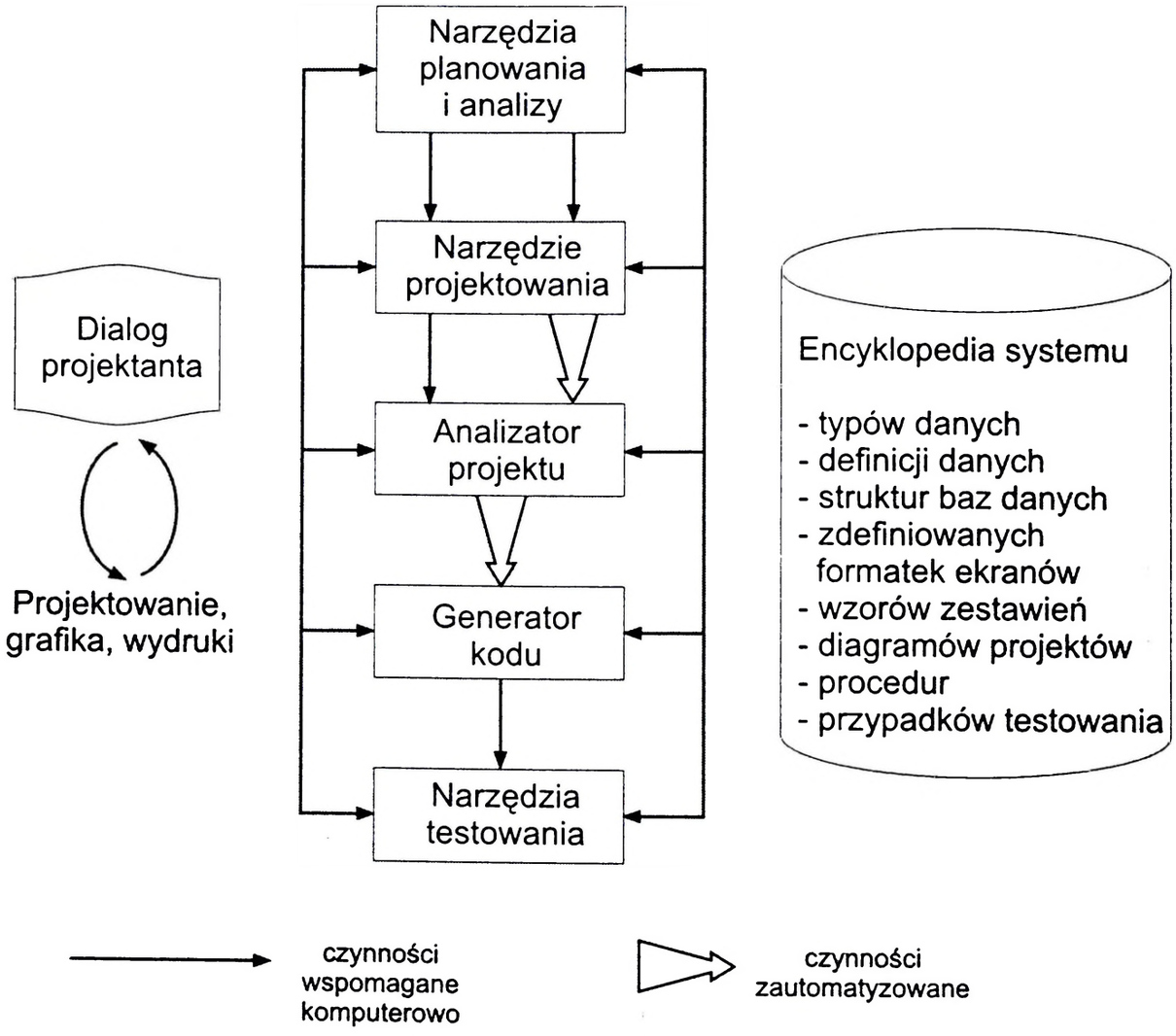
Cząstkowy pakiet CASE wspomaga użytkowanie **wybranej metody** czy techniki. Przykładami takiego pakietu są np. GraphDoc, pozwalający na generowanie A-grafów, czy I-grafów w metodyce ISAC bądź edytory graficzne diagramów przepływów danych, diagramów związków encji lub innych technik diagramowych. Z kolei pakiety pośrednie pozwalają na wspomaganie jednego lub kilku etapów cyklu życia procesu TSI w określonej metodyce. Pakiety te wykorzystują repozytorium (encyklopedię systemu). Z bardziej znanych produktów CASE warto wymienić tu: Excelerator czy Teamwork. Celem użytkowania **pakietów zintegrowanych** jest wspomaganie całego cyklu życia systemu, począwszy od analizy strategicznej działalności gospodarczej firmy i definiowania założeń systemu, poprzez jego projekt, do wdrożenia przy użyciu generatorów kodu i schematu bazy danych. Pakiety te na ogół powiązane są ściśle z pewną metodyką tworzenia systemów informatycznych oraz relacyjną bazą danych. Egzemplifikacją takiego pakietu jest Information Engineering Workbench firmy Knowledgware. Narzędzia cząstkowe mogą być użytkowane niemal *ad hoc,* podczas gdy efektywne stosowanie pakietów zintegrowanych wymaga specjalistycznego przeszkolenia.

Funkcjonalny zakres typowego zintegrowanego pakietu CASE przedstawia rys. 7.4 Poszczególne narzędzia odpowiadają głównym fazom cyklu życia systemu, a więc planowaniu i analizie (narzędziom analizy), projektowaniu (narzędziom planowania i projektowania i analizatorowi projektu), wdrażania (generatorowi kodu) oraz eksploatacji, modyfikacji i adaptacji (narzędziom testowania). Pogrubione strzałki na rys. 22 oznaczają automatyczny lub półautomatyczny przepływ specyfikacji projektowych lub programistycznych pomiędzy poszczególnymi fazami. W pierwszej fazie następuje opracowanie ogólnego opisu, modelu, szkicu badanej dziedziny przedmiotowej. Mają one postać diagramów i macierzy. Przygotowanie różnorodnych graficznych opisów w tej fazie jest czasochłonne. Wydajność twórcy systemu może być zwielokrotniona dzięki grafice komputerowej - generowaniu i edycji różnorodnych grafów i schematów, badaniu ich i ocenie

rozwiązań alternatywnych w celu modyfikacji i udoskonalenia opisu. Technika konstruowania tych grafów powinna być stosunkowo prosta, aby użytkownicy mogli ją zrozumieć i stosować.

Model konceptualny (szkic) systemu powinien być łatwo transformowany w projekt systemu, który z kolei powinien mieć cechy jednoznaczności, ścisłości i dokładności, aby mógł być automatycznie generowany na kod programowy. Projekt jest kontrolowany przez analizatora projektu, wskazującego błędy składniowe i semantyczne, jak również brak spójności i zupełności projektu. Wygenerowany w następnej fazie kod programowy podlega testowaniu dla sprawdzenia stopnia realizacji celów systemu. Podobna procedura choć prowadzona innymi metodami, dotyczy generowania schematu - najczęściej relacyjnej - bazy danych. Wyniki testu mogą oddziaływać na zmiany w projekcie bądź w modelu systemu. Należy zaznaczyć, iż pakiety CASE nie mają charakteru "automatu" umożliwiającego płynne przejście z opisu dziedziny przedmiotowej lub specyfikacji potrzeb informatycznych do kodu wynikowego i schematu bazy danych. Z całą pewnością takie rozwiązanie stanowi cel, do którego dążą producenci pakietów CASE. Na ogół dominują pakiety otwarte na różne opcje metodyczne powiązane z sobą logicznie, lecz nie automatycznie. Stąd generatory kodów poszczególnych języków projektowania, będące składnikami pakietu CASE, często użytkowane są jako oddzielne pakiety oprogramowania.

Kwestia bezpośredniego związku pakietu z określoną metodyką jest podstawą kryterium otwartości, tj. podziału na pakiety otwarte i dedykowane. Założeniem pakietów otwartych jest stworzenie elastycznego narzędzia, które może być wykorzystane w całości lub w części w odniesieniu do różnych metodyk TSI, stosownie do preferencji użytkownika. Pakiety te skoncentrowane są wokół encyklopedii systemu i posiadają budowę modularną. Użytkownik decyduje o doborze modułów właściwych dla danego projektu. Pakiety mają najczęściej



Rys.23. Zakres funkcjonalny zintegrowanego pakietu CASE

poszerzającą się liczbę sprzęgów do najbardziej popularnych systemów zarządzania bazami danych oraz generatorów kodu. Rozwiązanie takie znakomicie poszerza zakres użytkowania pakietu. Natomiast narzędzia dedykowane są bardzo ściśle związane z określoną metodyką. Integracja metodyki i narzędzia jest elementem określonej strategii rynkowej producenta oprogramowania bądź firmy doradczej. Istnieje zatem ścisły związek między fazami, zadaniami czy technikami danej metodyki tworzenia systemów informatycznych a modułami danego pakietu..

Otóż wyróżnia on pakiety:

- kierujące zespołem projektowym i narzucające sposób tworzenia systemu, np. poprzez komunikaty błędów,
- doradcze - zachęcające do określonej procedury metodycznej, np poprzez komunikaty ostrzegawcze,
- elastyczne, pozostawiające twórcy swobodę doboru metodyki.

W odniesieniu do trzeciego kryterium można wyodrębnić:

· pakiety bez reguł metodycznych,

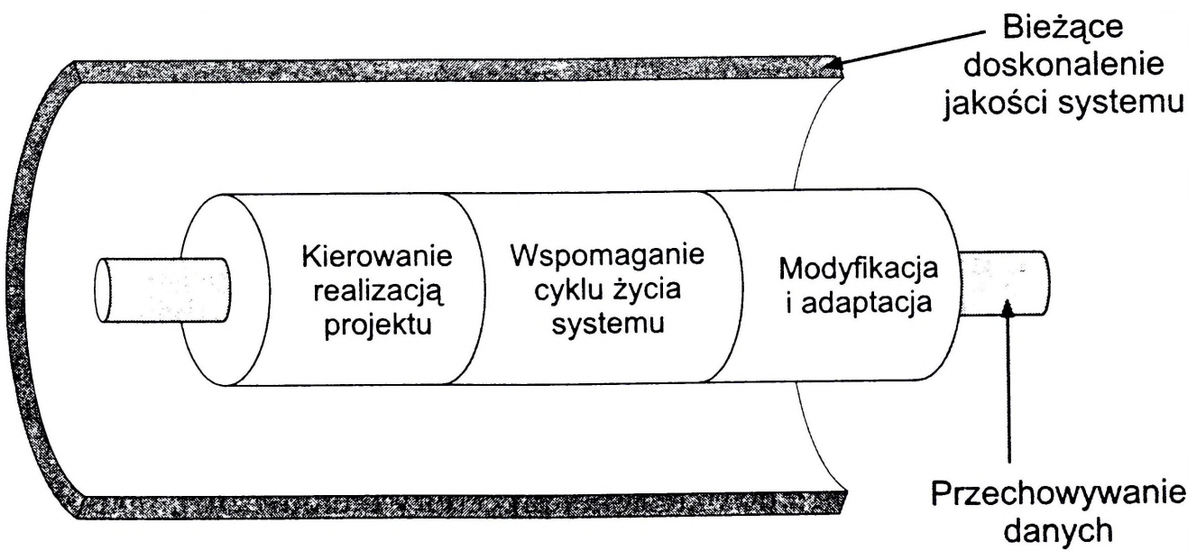
- pakiety z wbudowanymi regułami metodycznymi,
- pakiety z możliwością własnego rozbudowania reguł metodycznych.

Pierwszy rodzaj narzędzi CASE stanowią proste edytory poszczególnych diagramów czy macierzy. Ich celem jest szkicowanie grafiki systemu. Utrzymana jest kontrola spójności i zupełności opisów. Reguły metodyczne zapisane są przede wszystkim w zintegrowanych i dedykowanych narzędziach CASE. Złamanie reguł jest sygnalizowane. Podejmowane są próby umożliwienia twórcom systemu i użytkownikom zapisu własnych nowych zasad. Pakiety takie uwzględniają elementy sztucznej inteligencji.

Ostatnim kryterium podziału jest wybrany model dziedziny przedmiotowej. Pod tym względem wyróżnia się pakiety CASE wspomagające strukturalne, obiektowe bądź społeczne metodyki tworzenia systemów informatycznych.

Klasyfikacje pakietów CASE zdominowane są przez odniesienie do cyklu życia systemu. Tymczasem stosowane są one do szerszej klasy zagadnień, jak przechowywanie danych, kierowanie realizacją projektu, modyfikacja i adaptacja systemów. Z tego punktu widzenia syntetyczną, obszerną a jednocześnie precyzyjną klasyfikację opracowała firma doradcza CASE Research. Uwzględnia się w niej pięć następujących klas pakietów CASE:

- przechowywania danych,
- wspomagania cyklu życia systemu,
- adaptacji i modyfikacji systemów,
- kierowania realizacją projektów,
- bieżącego doskonalenia jakości systemu.



Rys. 24. Model klasyfikacji pakietu CASE

Podaną klasyfikację graficznie przedstawia rys. 24. W kolejnych czterech punktach scharakteryzowano wymienione klasy pakietów, począwszy od narzędzi wspomagających cykl życia systemu. W praktyce pakiety CASE przechowywania danych w postaci encyklopedii systemu stały się integrującym składnikiem większości pakietów i środowisk CASE, toteż i nie omawia się ich oddzielnie.

W świetle najnowszych tendencji wyłania się klasyfikacja na:

- pakiety CASE,
- środowiska CASE.

Pakiety zostały już obszernie zaprezentowane jako wycinkowe, pośrednie, zintegrowane, otwarte, dedykowane czy też o różnym stopniu uwzględniania reguł metodycznych.

Środowiska CASE tworzone są przez znaczących producentów sprzętu i oprogramowania. Zmierzają oni do opracowania swoistych standardów CASE, umożliwiających kompatybilność tworzonego i użytkowego oprogramowania.

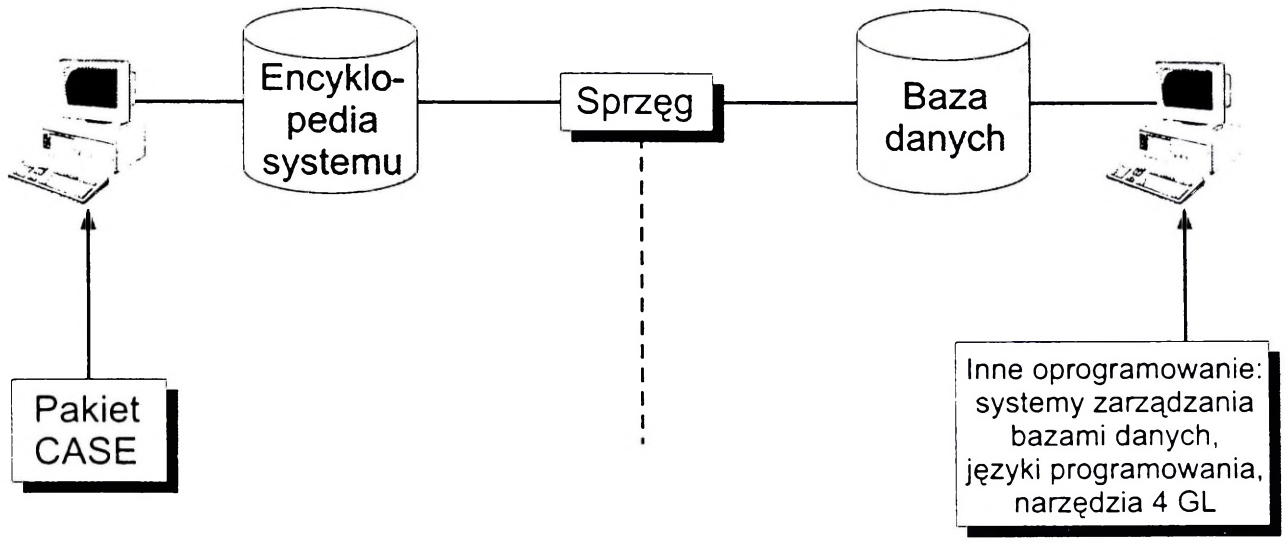
Podane klasyfikacje dotyczą przede wszystkim narzędzi CASE związanych z podejściem strukturalnym. Zintegrowane pakiety w coraz większym stopniu obejmują jednak komputerowe wspomaganie modeli obiektowych - pakiet SDW. Należy zaznaczyć, że w związku z szybko postępującymi zmianami na rynku oprogramowania, pozycja i znaczenie wymienionych w niniejszym rozdziale konkretnych produktów może się zmieniać.

9.3. Integracja i standaryzacja pakietów CASE

Złożoność zadań tworzenia systemów informatycznych w naturalny sposób skłaniała do integracji pakietów CASE. W praktyce realizują się na rynku oprogramowania dwie koncepcje integracji pionowej i poziomej. Pierwsze rozwiązanie przyjęło postać zintegrowanych pakietów CASE (I-CASE). Rozumie się przez nie narzędzia wspomagające cały cykl życia systemu od planowania strategicznego, poprzez narzędzia wysokiego poziomu, do narzędzi niskiego poziomu. Przykładem takiego pakietu jest Information Engineering Facility lub System Development Workbench (wraz ze sprzęgami do baz danych i generatorów kodu). Znaczącym wzbogaceniem potencjału integracyjnego pakietu CASE są sprzęgi (ang. *bridges),* które łączą go z popularnymi bazami danych oraz generatorami programów. Ideę tego typu integracji przedstawia rys. 25.

- Oprócz integracji pionowej pojawiła się koncepcja integracji poziomej C-CASE (ang: *component CASE),* czyli **integracji składnikowej.** Została ona

zaproponowana przez duże firmy komputerowe, między innymi IBM i DEC. Koncepcja ta odpowiada pojęciu środowiska CASE i polega na opracowaniu standardu, do którego dostosowują się producenci oprogramowania, pragnący wejść na rynek danego producenta sprzętu. Integracja typu C-CASE stwarza znacznie większe możliwości integracyjne.



Rys. 25. Integracja narzędzi CASE z oprogramowanie narzędziowym

W związku z przedstawionymi uprzednio rodzajami pakietów CASE pożądana jest również integracja szersza, która pozwalałaby na elastyczność w przesyłaniu, a następnie użytkowaniu wyników pracy różnych zespołów, stosujących w różnych fazach cyklu życia systemu I różne narzędzia CASE. Wyodrębnić można następujące sytuacje wymagające integracji wychodzącej poza ramy narzędzia CASE danego producenta: i · powiązanie pakietów wspomagania cyklu życia systemu z narzędziami kierowania projektami, modyfikacji i adaptacji oraz przechowywania danych,

- przesyłanie plików - analiz, projektów, baz danych czy programów , między

pakietami CASE (wycinkowymi, zintegrowanymi) na tym samym ; poziomie cyklu życia systemu,

- integracja pomiędzy pakietami lub modułami CASE wysokiego poziomu a oferowanymi na rynku systemami zarządzania bazami danych i generatorami kodu, dokonywana poprzez opracowanie odpowiednich sprzęgów.

Spośród opracowanych **technik transferu** danych pomiędzy pakietami CASE wymienić należy przede wszystkim:

- dostosowanie parametrów eksportu plików jednego narzędzia do j encyklopedii systemu drugiego lub adaptację wzajemną struktury przesyłanych plików i encyklopedii,
- transfer plików w ramach standardu, opracowanego przez danego producenta oprogramowania w jego środowisku C-CASE,
- przyjęcie standardu międzynarodowego, wypracowanego przez producentów oprogramowania CASE i ekspertów.

Podejmowane są również próby **standaryzacyjne** w dziedzinie pakietów CASE w aspekcie międzynarodowym. Największe znaczenie przypisuje się: ;

- **CDIF** (ang. *CASE Data Interchange Format),* zaproponowany przez i Electronic Industries Association,
- **ANSI/ISO IRDS** (ang. *Information Resource Dictionary System)* w zakresie modelowania danych.

9.4. Tendencje rozwoju pakietów CASE

Rynek oprogramowania pakietów CASE jest bardzo zmienny. Niektóre produkty utrzymują się na nim przez dłuższy czas, podczas gdy inne, mimo agresywnych kampanii reklamowych, nie znajdują większego uznania. Niektóre produkty zmieniają nazwę. Producenci poszczególnych pakietów łączą się, modyfikując nazwy produktów. Podana w podrozdziale 1 klasyfikacja cechuje się trwałością, ale rynek oprogramowania CASE należy oceniać według aktualnej sytuacji, której obiektywnym odzwierciedleniem są przede wszystkim dane zawarte na następujących stronach Internetu:

- http://osiris.sunderland.ac.uk/sst/casehome.html
- http://osiris.sunderland.ac.uk/rif/metacase/metacase.home.html
- http://osiris.suderland.ac.uk/rif/moose.home.html
- http://www.qucis.queensu.ca/Software-Engineering/case.html

- · http://www.astarte.ch/mt/
- http://www.yahoo.com/Computers_and_Internet/Software/development_Tools/
- http://www.yahoo.com/Computers-and-Internet/Software/Software Engineering/

Bieżącą informację rynkową o charakterze porównawczym opracowały różne firmy i zespoły CASE Reports czy CASE Industry Directories oraz wydawnictwo periodyczne CASE Trends.

W pierwszym okresie pakiety CASE automatyzowały podejście strukturalne. Gama tych metod była stopniowo wzbogacana o strategiczne planowanie systemów informatycznych i restrukturyzację procesów gospodarczch (BPR). Moduły wspomagające te metody wkrótce stały się składnikiem pakietów CASE. Podobne zjawisko pojawiło się wraz ze wzrostem znaczenia pakietów ERP (MRP II). Wprowadzano lub modyfikowano moduły, ukierunkowane na modelowanie procesów, w tym zwłaszcza przebiegu pracy i przepływu dokumentów. Niektóre pakiety są powiązane z konkretnym oprogramowaniem klasy ERP, np. pakiet ARIS jest związany z systemem R/3. Znaczącą zmianę stanowiło pojawienie się podejścia obiektowego. Oznacza ono konstruowanie modułów włączanych do dotychczasowych pakietów, lecz częściej tworzenie niezależnych obiektowych pakietów CASE. Swoistym przeglądem bieżącego znaczenia i doświadczeń wynikających z użytkowania pakietów CASE jest lista dyskusyjna CASE-L. Na podstawie powyższych źródeł można przyjąć, że przykładami aktualnie efektywnie użytkowanych pakietów CASE są:

- **strukturalne** (z wyżej wymienionymi modyfikacjami): Visible Analyst Workbench (Visible Systems Corporation), Methods Workbench (ISDE Metasoft Ltd.), ERWin (Logic Works), Excelerator II (Intersolv), System Engineer (LBMS), Case/Designer 2000 (ORACLE), Software through Pictures (Interactive Development Environments), MAESTRO (Softlab), LOCANA (Rapid Prototyping Laboratory);
- **obiektowe**: Object Maker (Mark V Systems Ltd.), Objectory Support Environments (Objectory Corporation), Rational Rose (Rational), OOATooI (Object International, Inc.), Bridge Point (Project Technology Inc.), System Architect (Popkin Software and Systems), CADRE Teamwork (CADRE Technologies Inc.).

Lista ta naturalnie zmienia się z upływem czasu. Zatem wyrobienie sobie poglądów na temat aktualnych zastosowań i znaczenia pakietów CASE wymaga skorzystania ze wskazanych wyżej źródeł, głównie elektronicznych.

10. WDRAŻANIE I UŻYTKOWANIE SYSTEMÓW INFORMATYCZNYCH

10.1. Wdrażanie systemów informatycznych

Złożone i kosztochłonne czynności planowania, analizy i projektowania systemów informatycznych syntetyzowane są w technicznym projekcie systemu, zwanym również specyfikacją systemu. Dokument ten służy całemu zespołowi projektowemu, tj. analitykom, projektantom, programistom, specjalistom baz danych, profesjonalnych pakietów oprogramowania, sprzętu i sieci komputerowych, do rozpoczęcia kolejnej fazy w cyklu życia systemu - **wdrażania** systemu **informatycznego.** Polega ona na zbudowaniu nowego systemu informatycznego i przekazaniu go do eksploatacji.

W fazie wdrażania systemu informatycznego następuje istotna jakościowa **zmiana** w funkcjonowaniu organizacji gospodarczej, w wymiarze zarówno technicznym, jak i - przede wszystkim - psychologiczno-socjologicznym. W związku z tym należy dążyć do zminimalizowania oporu przed dokonywaniem tej zmiany. Rozważyć trzeba następujące możliwości zwiększenia stopnia akceptacji wdrażanego systemu przez kierownictwo firmy i jego przyszłych użytkowników:

- pozyskanie zaangażowania kierownictwa firmy, · przyjęcie zasady stopniowego rozwoju,
- elastyczność w doborze parametrów projektu, · uwzględnienie aspektów socjotechnicznych.

Osiągnięcie skutecznego poziomu **zainteresowania** i zaangażowania kierownictwa firmy w trakcie wdrażania zaprojektowanego systemu jest podstawowym warunkiem powodzenia podjętego przedsięwzięcia. Trudno liczyć na efektywne poparcie kierownictwa bez podjętej znacznie wcześniej, współpracy w realizacji fazy planowania metodami analizy sytuacyjnej. Wpływ systemu na sposób wykonywania indywidualnej bądź zespołowej pracy jest oceniany i ostatecznie akceptowany przez kierownictwo. Zasadniczym argumentem za wdrożeniem nowego systemu jest artykułowany przez zespół projektowy fakt likwidacji lub znaczącego ograniczenia trudności w funkcjonowaniu firmy, mających wpływ na jej efektywność i konkurencyjność. Niekiedy **stopniowa,** przeprowadzana w małych zakresach zmiana może być łatwiejsza do akceptacji zarówno dla kierownictwa, jak i użytkowników, wziąwszy pod uwagę, iż procesy związane z tworzeniem systemów informatycznych mają charakter długoterminowy. Należy jednak zaznaczyć, iż zasada ta nie ma zastosowania w przypadku

dokonywania radykalnej zmiany np. poprzez restrukturyzację procesów gospodarczych z wykorzystaniem technologii informatycznej.

Zarówno natura makroekonomicznych i mikroekonomicznych procesów gospodarczych, jak i technologia informatyczna zmieniają się. Tempo tych zmian jest coraz szybsze. Twórcy systemów muszą uwzględniać aspekt **elastyczności** systemu. Oznacza to w praktyce tworzenie systemów w taki sposób, aby procedury i technologie mogły być wymieniane na coraz nowocześniejsze. W praktyce jest to zadanie niełatwe, zważywszy trudność przewidywania trendów rozwoju w informatyce. Jednak samo odczucie elastyczności projektowanego systemu winno towarzyszyć analitykom i projektantom w trakcie procesu jego tworzenia. Proces tworzenia i sam system zawierają składniki techniczne i społeczne: sprzęt, oprogramowanie, procedury, kadry.

Istotne jest, by wdrażanie systemu przebiegało na zasadzie równoległego, zharmonizownego uwzględniania tych składników. Traktowanie każdego z nich bądź indywidualnie, bądź jako priorytetowego, bez przewidywania wzajemnych interakcji, w znacznej mierze może przyczynić się do ostatecznego niepowodzenia przedsięwzięcia projektowego.

Wdrażanie systemu informatycznego obejmuje takie czynności i zadania, jak: zakup sprzętu, instalacja bądź rozwój sieci komputerowej, zainstalowanie bazy danych, wprowadzenie do niej danych, opracowanie i testowanie programów, zakup pakietów oprogramowania, przygotowanie dokumentacji systemu oraz przeszkolenie jego użytkowników. Dopiero wówczas system ten można przekazać do eksploatacji. **Czynności** te można uporządkować w następujący ciąg:

- zakup bądź uzupełnienie oraz instalacja sprzętu informatycznego wraz z siecią komputerową,
- wdrożenie i przetestowanie bazy danych,
- opracowanie i przetestowanie programów, zakup pakietów oprogramowania, integracja oprogramowania systemu,
- testowanie systemu i jego zainstalowanie.

Projekt techniczny systemu decyduje o sposobie jego wdrażania. Pierwsze dwie czynności inicjowane są w podanej kolejności, chociaż późniejsza ich realizacja może mieć charakter równoległy. Zainstalowanie sprzętu i sieci komputerowych umożliwia podjęcie prac nad oprogramowaniem systemu. Parametry sprzętu i sieci ulegają aktualnie bardzo dynamicznym zmianom. Wynikają z tego wnioski o potrzebie z jednej strony szybkiego

wdrażania systemu, a z drugiej - zapewnienia dokonywania unowocześnień. Naturalnie niemal każda współczesna organizacja gospodarcza posiada komputerową infrastrukturę sprzętowo-sieciową, toteż wdrożenie systemu komputerowego wiąże się raczej z jej unowocześnianiem, tj. modyfikacją i adaptacją bądź poszerzaniem. Przypadki pełnej wymiany należą raczej do wyjątków.

Inna sytuacja dotyczy bazy danych, której struktury danych opracowane zostały we wcześniejszych etapach cyklu życia systemu. Struktury te należy dostosować do parametrów oprogramowania obsługującego bazę danych, czyli systemu zarządzania bazą danych (SZBD). Aktualnie jest to najczęściej system relacyjny, bardzo rzadko sieciowy. Dużego znaczenia nabiera podejście obiektowe, toteż na rynku oprogramowania oferowane są tzw. uniwersalne serwery baz danych (proponowane np. przez firmy INFORMIX i ORACLE), mające charakter relacyjno-obiektowy. Wdrażając i użytkując bazy danych, można posługiwać się istniejącym oprogramowaniem, skoncentrowanym wokół zakupionego i zainstalowanego wcześniej SZBD (bądź jego kolejną ulepszoną wersją). Wdrożenie bazy danych może oznaczać jednak konieczność zakupu nowego systemu zarządzania bazą danych wraz towarzyszącym oprogramowaniem (narzędzia czwartej generacji, język zapytań SQL, pakiety CASE).

Kolejnym istotnym składnikiem wdrażania systemu informatycznego jest jego oprogramowanie. W trakcie tworzenia i testowania programów wykorzystuje się zarówno zakupione pakiety oprogramowania, jak i oprogramowanie dedykowane, przygotowane przez zespół programistów, a ukierunkowane na specyfikę danego zastosowania. Wykorzystuje się algorytmy wyspecyfikowane w fazie projektowania systemu, opisane przy użyciu diagramów struktury, Jacksona, Nassi-Schneidermana czy Warniera-Orra, a także za pomocą tablic decyzyjnych. W związku z tym pojawia się często potrzeba opracowania programów integrujących pakiety oprogramowania z programami i podprogramami dedykowanymi. Całość oprogramowania w firmie tworzy jej bibliotekę oprogramowania. Czynności programistyczne mogą być w znaczący sposób przyspieszone, dzięki:

- wykorzystaniu zasad prototypowania systemu; kolejne iteracje zawierają w coraz większym wymiarze oprogramowanie systemu, do wersji finalnej włącznie;
- językom czwartej generacji, ograniczającym w znacznej mierze liczbę linii kodu w programie;
- generatorom kodu, które w sposób automatyczny generują na podstawie specyfikacji systemu znaczące partie kodu programu; generatory kodu są

składnikami zintegrowanych pakietów CASE;

- zakupieniu i wdrożeniu zintegrowanego pakietu zastosowań klasy MRPII, którego oprogramowane moduły wymagają jedynie adaptacji do specyfiki określonej firmy.

Nieodłącznym elementem wdrażania oprogramowania jest jego testowanie, zwłaszcza programów dedykowanych. Rozróżnić tu można testy:

- indywidualne, dotyczące sprawdzania poprawności poszczególnych modułów oprogramowania,
- zintegrowane, zwane testami akcentacyjnymi, a związane z kontrolą i korektą całości oprogramowania systemu, harmonijnego współdziałania wszystkich jego modułów.

Ostatnimi zadaniami w fazie wdrażania systemu są: testowanie i zainstalowanie. W przeciwieństwie do poprzednich testów na tym etapie chodzi o całościową, zintegrowaną kontrolę, a następnie korektę funkcjonowania wszystkich modułów i całości wdrażanego systemu. Testowi podlega spójność, sprawność i efektywność współdziałania kadry, sprzętu, sieci komputerowych, baz danych i oprogramowania w osiąganiu celów założonych w fazie planowania. Testowanie może opierać się na ściśle sformalizowanych zasadach kontroli jakości QA (ang. *quality assurance)* lub TQM (ang. *total quality management).* Aby test mógł odbyć się na danych rzeczywistych, wprowadza się stosownie do przyjętej struktury danych i parametrów SZBD odpowiednie dane rzeczywiste z organizacji gospodarczej do plików i bazy danych. Poszczególne pliki oraz baza danych mogą cechować się dużą liczebnością rekordów. Ich wprowadzenie jest więc czynnością czasochłonną. Zazwyczaj przynajmniej część tych danych była wcześniej zarejestrowana i przechowywana w postaci elektronicznej w dotychczasowym systemie lub systemach użytkowanych w firmie. Istnieje zatem możliwość konwersji tych plików do bazy danych wdrażanego systemu przy użyciu specjalistycznego oprogramowania.

Ściśle związane z sobą są zagadnienia opracowania dokumentacji dla użytkowników oraz ich szkolenia. W sposób naturalny, stosownie do wybranej metodyki, tworzona jest dokumentacja systemu w całym cyklu życia. Zawiera ona opis modeli w kolejnych fazach cyklu, przygotowana może być w postaci papierowej i elektronicznej, z wykorzystaniem odpowiednich metod, technik i narzędzi. Choć część dokumentacji, zwłaszcza w fazie analizy, powstaje przy współudziale użytkowników systemu, znakomita jej większość byłaby dla nich niezrozumiała ze względu na swój profesjonalny charakter. Toteż istnieje konieczność opracowania dokumentacji i materiałów szkoleniowych, ukierunkowanych na

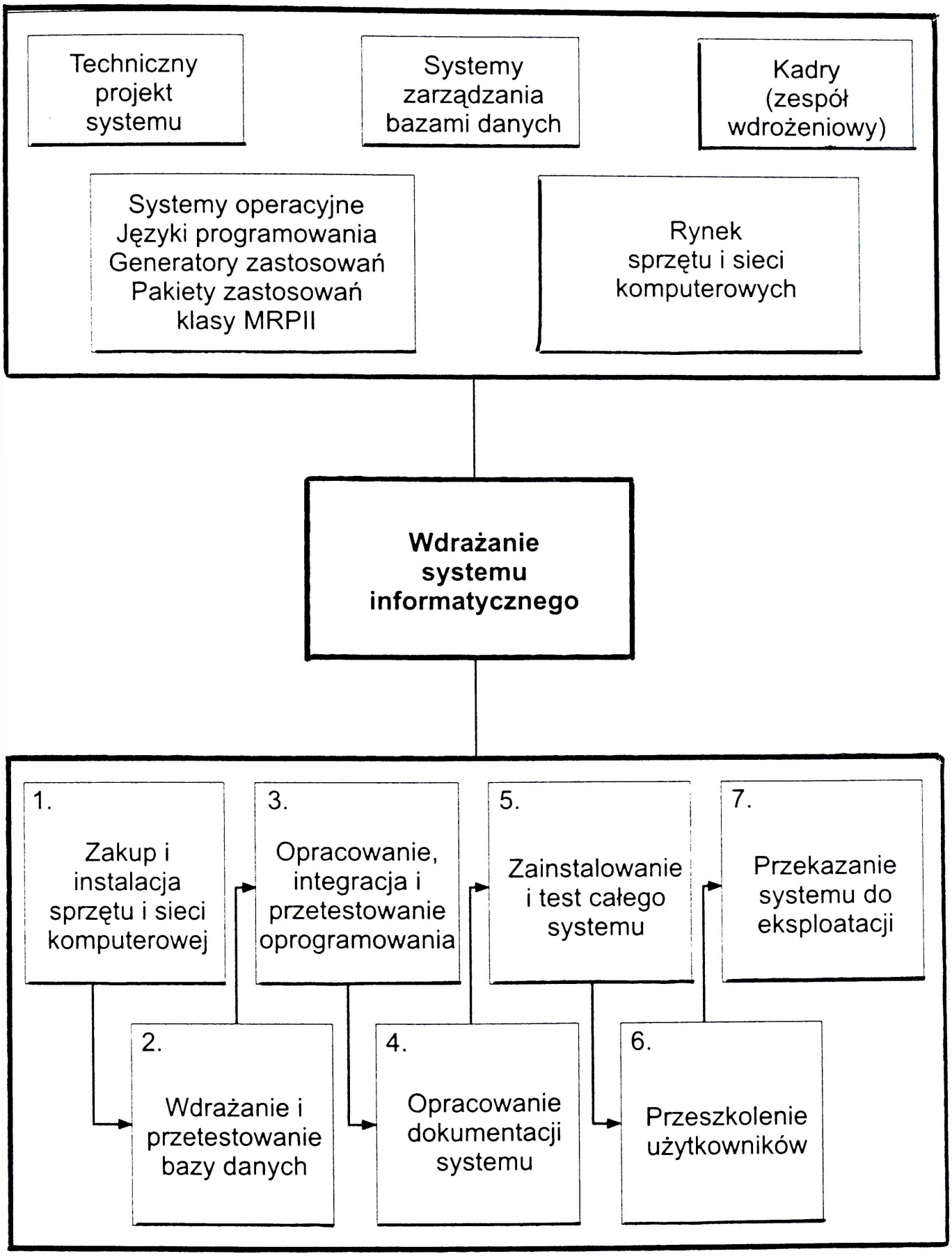
przygotowanie użytkowników do sprawnej eksploatacji systemu.

Szkolenia winny przygotować pracowników w pierwszej fazie do sprawnego użytkowania zainstalowanego sprzętu, działającego zgodnie z odpowiednim systemem operacyjnym. Następnie pracownicy powinni opanować funkcje wdrażanego systemu, stosownie do specyfiki komputeryzowanego stanowiska pracy. Przeszkolenie pracowników dużych instytucji ubezpieczeniowych, banków czy urzędów administracji w związku z wdrażaniem systemu informatycznego jest poważnym zadaniem, realizowanym w dłuższym okresie, wymagającym często zatrudnienia kadry specjalistycznych firm doradczych i szkoleniowych. Dobry program szkoleniowy wykorzystuje zestawy zróżnicowanych metod i form nauczania, jak: wykłady, rozwiązywanie przypadków, dyskusje seminaryjne, ćwiczenia laboratoryjne, korzystanie z komputerowych samouczków, studiowanie materiałów szkoleniowych oraz biuletynów lub materiałów przesyłanych Intranetem.

Wykonanie wszystkich czynności i uzyskanie pozytywnych rezultatów testu oznacza zainstalowanie wdrażanego systemu i przekazanie go do eksploatacji użytkownikom. Kompleksowo proces wdrażania systemu informatycznego przedstawiony jest na rys. 26

10.2. Użytkowanie, modyfikacja i adaptacja systemu

Ostatnią fazą cyklu życia systemu jest jego użytkowanie (eksploatacja), powiązane z wprowadzeniem koniecznej modyfikacji i adaptacji. Faza eksploatacji polega na stałym wykorzystywaniu systemu informatycznego do wspomagania bieżącej działalności organizacji, przez generowanie i dostarczanie użytecznej informacji. Jest to więc faza zbierania owoców realizacji cyklu życia systemu dzięki współdziałaniu kadry, sprzętu, sieci, oprogramowania i bazy danych. Na podstawie bieżących transakcji wprowadza się dane do poszczególnych plików bazy danych, w wyniku czego następuje jej aktualizacja; przesyłanie i przetwarzanie danych odbywa się stosownie do przyjętych algorytmów generowania formatek i zestawień końcowych, odpowiedzi na zapytania dotyczące podejmowania bieżących i perspektywicznych decyzji gospodarczych. Administratorzy systemu, w miarę pojawiania się potrzeb, doradzają lub przeprowadzają szkolenia kadry i użytkowników. Opierając się na przyjętym projekcie systemu, opracowuje się harmonogram eksploatacji, określający sekwencję czynności, wykonywanych bieżąco i okresowo.



Rys. 26. Proces wdrażania systemu informatycznego

Elementem tego harmonogramu jest kontrola jakości użytkowanego systemu. Jej rezultatem może być decyzja o podjęciu modyfikacji lub adaptacji systemu.

Eksploatacja systemu rzadko przebiega płynnie. Podstawowe interwencje związane z użytkowaniem systemu dotyczą:

- usuwania błędów w funkcjonowaniu, zwłaszcza oprogramowania, bazy danych i sieci komputerowej,
- odzyskiwania systemu po jego zawieszeniu lub upadku. Czynności powyższe, polegające na stałym ulepszaniu technologicznym systemu, jego pielęgnowaniu, w miarę pojawiania się braków i błędów, służą modyfikacji systemu.

Modyfikacja ma charakter kosmetycznych, drobnych, choć ciągłych korekt, nie zmieniających w zasadniczy sposób zakresu funkcjonalnego i technologicznego użytkowanego systemu. Jednakże zarówno organizacja gospodarcza, jak i jej otoczenie mogą podlegać **zmianom** skokowym. W związku z tym system ulega stopniowej dezaktualizacji, która może podważyć sensowność użytkowania wdrożonego systemu. W skrajnym przypadku eksploatowany system może firmie szkodzić zamiast wspomagać realizację założonych celów i zadań. Są trzy podstawowe czynniki wpływające na dezaktualizację systemu:

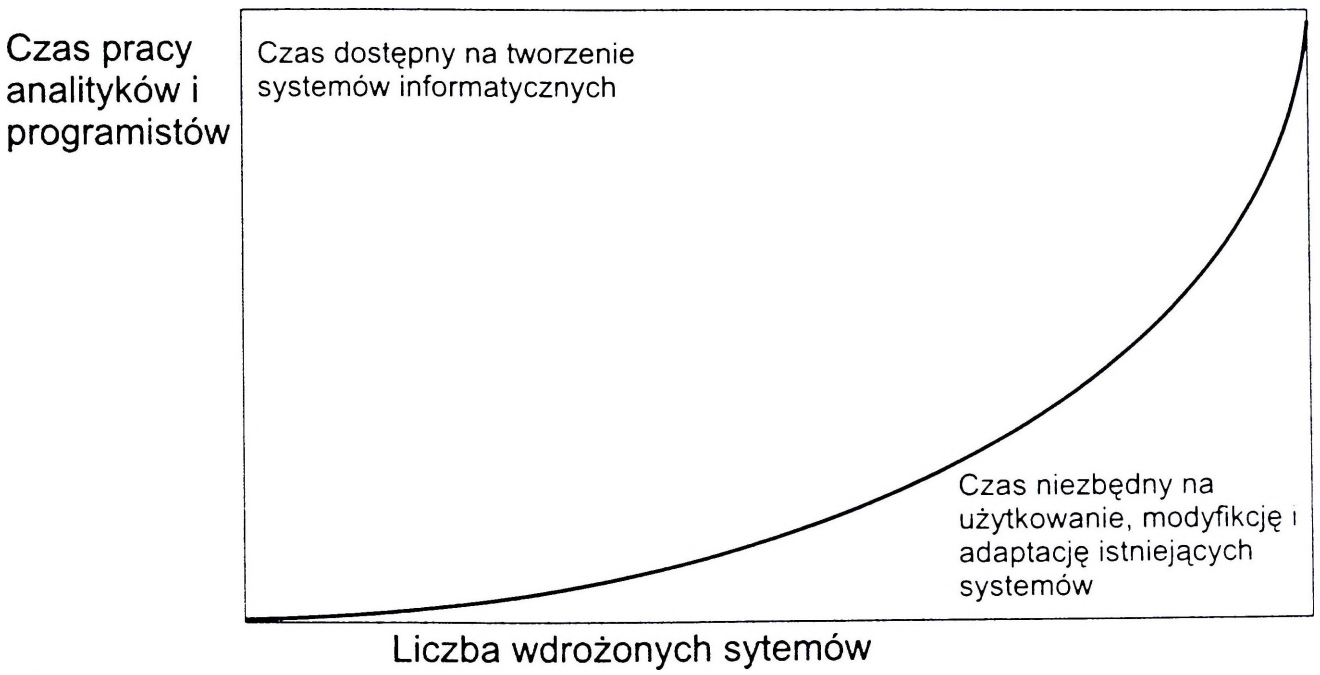
- wewnątrzorganizacyjne transformacje, wynikające ze zmiany strategii gospodarczej firmy,
- zmiany w otoczeniu organizacji gospodarczej, związane przede wszystkim ze zmianą popytu na produkowane wyroby lub usługi oraz ze wzmocnieniem roli konkurencji,
- zmiany w technologii informatycznej, które mogą znacznie podnieść sprawność i nowoczesność posiadanej infrastruktury sprzętowo-programistycznej firmy.

Dwa ostatnie czynniki często stymulują zainicjowanie przemian wewnątrzorganizacyjnych. We wszystkich tych przypadkach należy wprowadzić radykalne zmiany użytkowanego systemu informatycznego, czyli jego **adaptację.** Jej narzędziem może być restrukturyzacja procesów gospodarczych BPR.

Proces adaptacji oznacza w praktyce **powtórzenie cyklu** życia systemu w odniesieniu do określonego modułu lub modułów wdrożonego systemu. W fazie planowania adaptacji określa się więc problemy, cele i potrzeby, sporządza infoplan. Może on dotyczyć wybranego modułu lub większej liczby procedur systemu. Po dokonaniu analizy, opracowuje się projekt

nowego systemu, który zostaje wdrożony. Najbardziej kosztowne zmiany dotyczą sprzętu, podczas gdy najbardziej czasochłonna jest adaptacja oprogramowania. Mniejszym problemem jest restrukturyzacja czy zmiana formatów bazy danych, dokonywana za pomocą programów związanych z SZBD.

Użytkowanie systemu, a zwłaszcza jego modyfikacja i adaptacja, wiąże się z poważnymi **nakładami** materialnymi i kadrowymi. Wraz z rosnącą liczbą użytkowanych systemów w organizacji gospodarczej coraz więcej czasu pracy analityków i projektantów przeznacza się na eksploatację, modyfikację i adaptację wdrożonych systemów kosztem planowania, analizy i projektowania nowych systemów. Zależność tę przedstawia rys. 27



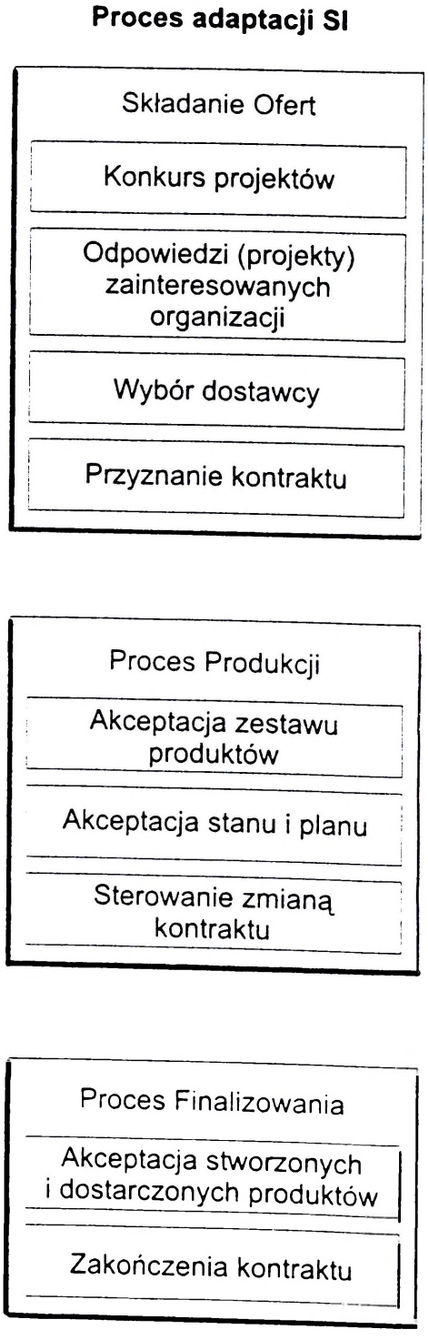
Rys. 27. Wdrożone systemy a czas eksploatacji

Wspomniana wyżej kontrola użytkowanego systemu może mieć postać audytu informatycznego wykonywanego przez wyspecjalizowaną firmę doradczą. Audyt funkcjonowania systemu oznacza zebranie i ocenę informacji nt.: odpowiedniego poziomu przeszkolenia kadry, nowoczesności i adekwatności stosowanej metodologii i technologii, racjonalności wykorzystania zasobów kadrowych i komputerowych, poziomu satysfakcji użytkownika z eksploatowanego systemu, posiadanego i stosowanego infoplanu oraz pełnej dokumentacji systemu, istnienia procedur udoskonalania i odzyskiwania systemu. Wyniki audytu stają się podstawą do podejmowania decyzji co do wprowadzenia zmian w systemie

informatycznym lub decyzji o jego dezaktualizacji i wycofaniu z użytkowania.

10.3. Adaptacja systemu na podstawie standardu EuroMethod

EuroMethod (EM) kojarzy się przede wszystkim z propozycją europejskiego standardu metodyki TSI. Najnowsza wersja, znana również pod nazwą ISPL (ang. *Information Service Procurement Library),* ukierunkowana jest przede wszystkim na efektywne kierowanie zakupem systemu informatycznego w drodze przetargu.



Rys. 28. Proces adaptacji SI,

. W związku z ogromnymi środkami przeznaczanymi na zakup systemów i technologii informatycznej w państwach Unii, narzędzie typu EM ułatwia kontrolę przebiegu przedsięwzięć związanych z informatyzacją. EM, użytkowana prawidłowo, pozwala na osiągnięcie potencjalnych korzyści w zarządzaniu ryzykiem, związanym z przejściem do zmienionej sytuacji. EuroMethod została skonstruowana więc w celu wspomagania organizacji przy **zakupie efektywnych SI,** w różnorodnych sytuacjach, poprzez skłanianie klientów i dostawców do kontroli kosztów i czasu realizacji, zarządzania ryzykiem i polepszenia wzajemnego zrozumienia. Szczegółowe **cele** EuroMethod są następujące:

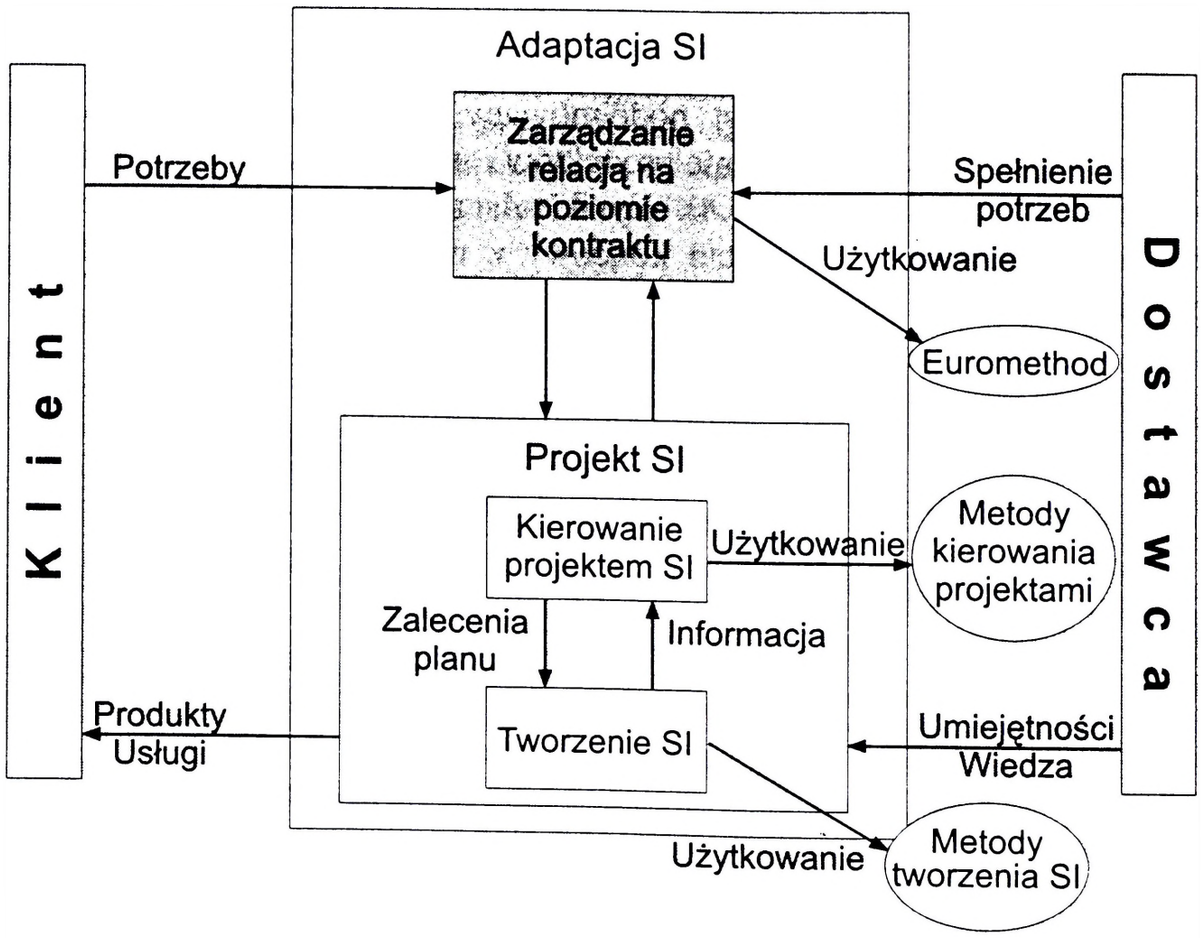
- wspomaganie wzajemnego zrozumienia między klientami i dostawcami **SI** na otwartym międzynarodowym rynku poprzez zapewnienie przewodnika, uzupełnionego zestawem pojęć i terminologii, użytkowanych w trakcie zawierania transakcji,
- poprawienie trafności zakupów SI poprzez branie pod uwagę pełni sytuacji problemowej i związanego z nią ryzyka,
- zapewnienie podstawy dla harmonizacji różnych metod.

Aby sprostać szybkim zmianom, współczesne organizacje potrzebują jeszcze bardziej skutecznych i elastycznych systemów informatycznych (SI). **Adaptacja** SI danej organizacji, mająca sprostać jej nowym potxzebom, jest podstawowym zadaniem, które winno być zrealizowane tak szybko i tak efektywnie, jak to możliwe. Ten właśnie proces jest wspomagany przez EM. EuroMethod może być użytkowana w całym procesie zakupu, począwszy od pojawienia się potrzeb na nowy lub zmodyfikowany SI do sfinalizowania kontraktu, który spełnia określone wcześniej wymagania. Z tego powodu wprowadzono pojęcie zakupu SI, które opisuje ten proces. EuroMethod porządkuje i wspomaga proces zakupu, w szczególności strategię zakupu, relację klient-dostawca i zarządzanie realizacją kontraktu dla wzrostu możliwości osiągnięcia pożądanego rozwiązania w ramach limitu czasu i kosztów. W EuroMethod koncepcja adaptacji SI została wprowadzona, aby uwzględnić różne typy projektów, które mogą być powiązane z systemami informatycznymi. Adaptacja systemu informatycznego może więc przybrać następujące postaci:

- tworzenie systemu,
- użytkowanie systemu, · projektowanie systemu,
- modyfikacja i adaptacja zwrotna,
- restrukturyzacja procesów gospodarczych,
- instalacja systemu,

- różnego rodzaju studia systemów.

Zakres EuroMethod obejmuje każdy projekt adaptacji SI, w którym dowolny **klient** zamawia dowolną pracę (dzieło) u dowolnego **dostawcy.** Klient może pochodzić zarówno z sektora publicznego, jak i prywatnego, a dostawca może być organizacją zewnętrzną lub działem w organizacji klienta. Klient i dostawca mogą funkcjonować w różnych krajach. Do zastosowania EuroMethod wystarczą dwie strony powiązane formalnym porozumieniem o podjęciu adaptacji SI.



Rys. 29. Projekt adaptacji SI

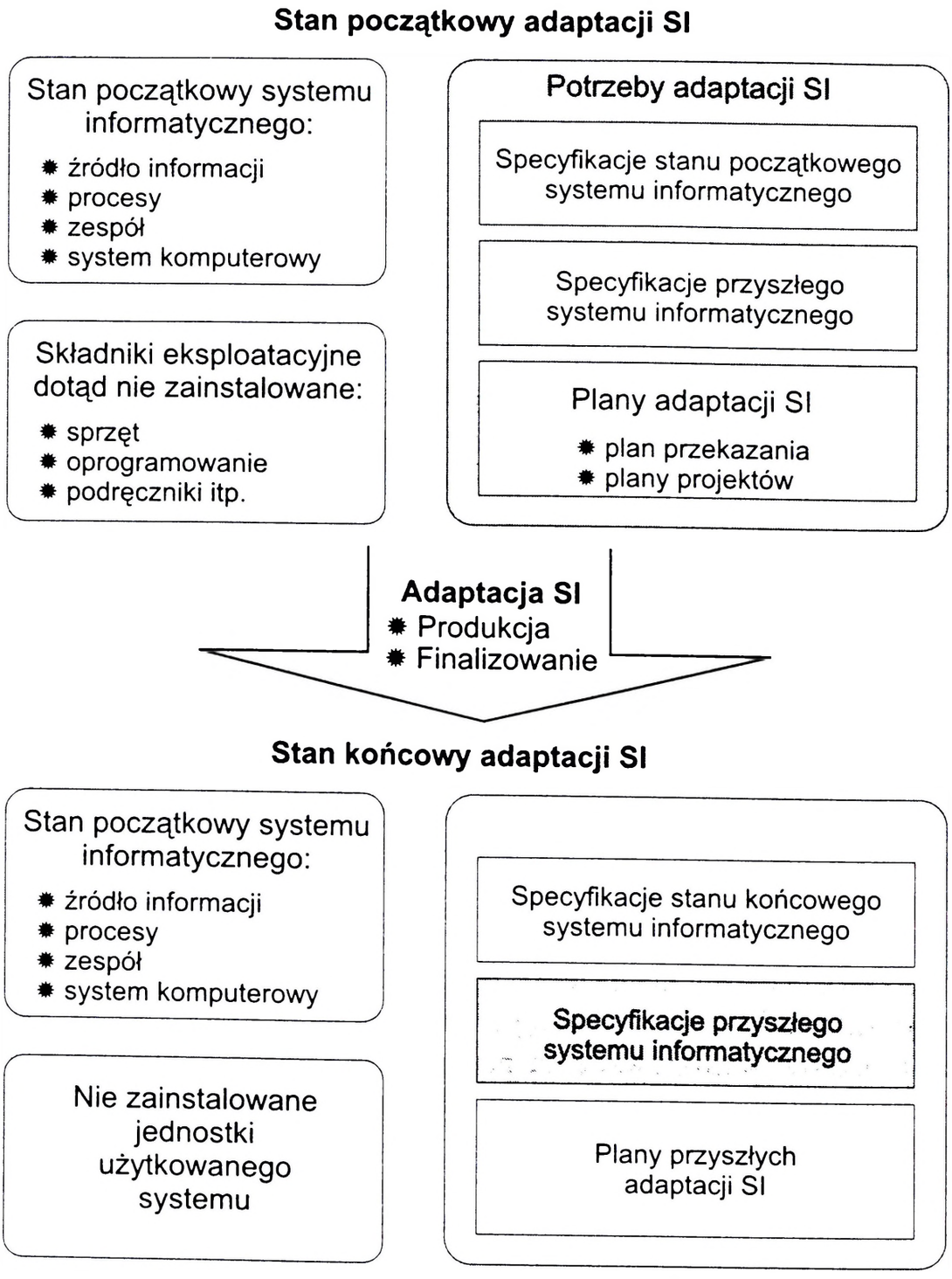
Kontrakt może być prawnym porozumieniem, łączącym różne organizacje, lub wewnętrznym porozumieniem pomiędzy dwoma działami tej samej organizacji. W celu osiągnięcia pełnych korzyści, EuroMethod winna być stosowana zarówno przez klienta, jak i dostawcę. Jednakże nawet jeśli jest ona użytkowana tylko przez jedną stronę, to opłaca się dzięki lepszemu planowaniu i zarządzaniu projektami.

Dla każdej postaci adaptacji SI, EuroMethod definiuje trzy podstawowe procesy: składania ofert, produkcji i finalizowania (rys.28). W każdym z tych procesów wykonywane

są pewne typy transakcji klient-dostawca. Transakcja obejmuje wymianę wytworów pomiędzy klientem i dostawcą. W prostych procesach zakupu, transakcje składania zamówień przebiegają w sposób zbliżony do kolejności przedstawionej na rys. 29. W złożonych procesach zakupu może być więcej transakcji, takich jak:

- ogłoszenie zaproszenia do zainteresowanych, odpowiedzi zainteresowanych organizacji, wstępny wybór potencjalnych dostawców;
- ogłoszenie konkursu projektów, odpowiedzi zainteresowanych organizacji, wybór dostawcy, przyznanie kontraktu.

Projekt adaptacji SI jest podejmowany przez zespół wykonawczy. Celem projektu jest dostarczenie produktów i usług, które spełniają potrzeby zgłoszone przez klienta (rys.29). Wyniki adaptacji systemu informatycznego ze stanu początkowego w stan końcowy z wykorzystaniem EuroMethod przedstawia rys.30.



Rys. 30. Stany początkowy i końcowy adaptacji SI

Załącznik 1

PROJEKT SYSTEMU PRZETWARZANIA INFORMACJI WSPOMAGAJĄCEGO PRACE KOMISJI PRZETARGOWEJ

1 Zamówienia publiczne

Zaprojektowanie systemu przetwarzania informacji wspomagającego pracę komisji przetargowej wymaga scharakteryzowania procedury przetargowej, będącej podstawą istnienia i działania komisji przetargowej. Znaczące wydaje mi się przybliżenie trybów, w jakich zamówienia publiczne mogą być udzielane, wpływu szacunkowej wartości zamówienia na sposób udzielenia zamówienia publicznego, dokumentu - specyfikacji istotnych warunków zamówienia oraz procedury protestacyjno-odwoławczej.

Udzielenie zamówienia musi odbywać się zgodnie z przepisami ustawy o zamówieniach publicznych w sytuacji, gdy zachodzą łącznie następujące przesłanki:

- podczas realizacji zamówienia wydatkowane są środki publiczne,
- zamówienie udzielane jest przez podmiot zobowiązany do stosowania ustawy o zamówieniach publicznych (podmioty te to m.in. państwowe jednostki budżetowe i zakłady budżetowe, jednostki samorządu terytorialnego i sejmiki samorządowe, ale także spółdzielnie, fundacje i stowarzyszenia w zakresie, w jakim dysponują środkami publicznymi),
- przedmiotem zamówienia jest dostawa sprzętu, wykonanie usługi lub realizacja roboty budowlanej.

1.1 Tryby udzielania zamówień publicznych

Podstawowym trybem udzielania zamówienia publicznego jest przetarg nieograniczony, który może być stosowany zawsze, bez względu na okoliczności. Zastosowanie trybu innego niż przetarg nieograniczony, bez względu na wartość zamówienia, możliwe jest tylko w sytuacjach ściśle określonych w ustawie. Zamówienia publicznego można udzielić, poza przetargiem nieograniczonym, w następujących trybach:

1. **przetargu ograniczonego** - w sytuacji, gdy

- ze względu na specjalistyczny charakter zamówienia istnieje ograniczona liczba dostawców lub wykonawców mogących wykonać dane zamówienie lub
- koszty przeprowadzenia przetargu nieograniczonego byłyby niewspółmiernie wysokie w stosunku do wartości zamówienia;

2. **przetargu dwustopniowego** - w sytuacji, gdy
   - nie można z góry określić szczegółowych cech technicznych i jakościowych zamówienia lub
   - z powodu specjalistycznego charakteru zamówienia konieczne są negocjacje z dostawcami, wykonawcami lub
   - przedmiotem zamówienia jest przeprowadzenie badań, eksperymentu, sporządzenie opinii naukowej albo świadczenie innych wyspecjalizowanych usług lub
   - przedmiotem zamówienia jest zaprojektowanie i wykonanie robót budowlanych;

3. **negocjacji z zachowaniem konkurencji** - w sytuacji, gdy
   - zachodzi pilna potrzeba udzielenia zamówienia, której nie można było wcześniej przewidzieć i nie wynikała ona z winy zamawiającego lub
   - wszczęto postępowanie przetargowe i nie wpłynęła wymagana liczba ofert lub wszystkie oferty odrzucono, a można w sposób uzasadniony przypuszczać, że powtórny przetarg nie doprowadzi do zawarcia umowy;

4. **zapytania o cenę** - w sytuacji, gdy przedmiotem zamówienia są dostawy rzeczy lub usługi powszechnie dostępne o ustalonych standardach jakościowych;

5. **zamówienia z wolnej ręki** - w sytuacji, gdy
   - zamawiający dokonuje zamówień dodatkowych nie przekraczających 20% wartości uprzedniego zamówienia, a zachodzi konieczność zachowania tych samych norm, parametrów i standardów lub

- zawiera umowę o przeprowadzenie badań, eksperymentu lub sporządzenie opinii naukowej lub
- ze względu na szczególny rodzaj zamówienia można je uzyskać tylko od jednego dostawcy, wykonawcy lub
- dokonuje zamówienia publicznego na twórcze prace projektowe oraz na prace z zakresu działalności twórczej w dziedzinie kultury i sztuki lub
- można skorzystać tylko z jednego dostawcy, wykonawcy w wyniku zastosowania dozwolonego zakresu preferencji krajowych lub
- ze względu na szczególne okoliczności gospodarcze lub społeczne, których zamawiający nie mógł przewidzieć, wymagane jest natychmiastowe wykonanie zamówienia lub
- wartość zamówienia nie przekracza równowartości 3000 euro.

1.2 Wartość szacunkowa zamówienia a procedura przetargowa

Przy stosowaniu przetargu nieograniczonego zachodzą istotne różnice w procedurze, w zależności od wartości szacunkowej zamówienia - kwotą graniczną jest 30 000 euro. Zgodnie z art. 15 ust. 1 w przetargu nieograniczonym, w którym wartość zamówienia przekracza 30 000 euro, zamawiający zobowiązany jest stosować przepisy ustawy dotyczące publikacji ogłoszeń w Biuletynie Zamówień Publicznych, pisemności postępowania, protokołu postępowania, specyfikacji istotnych warunków zamówienia, terminów, wadium, protestów i odwołań. Zamawiający, który ustalił wartość szacunkową zamówienia na kwotę niższą, nie musi stosować wyżej wymienionych przepisów, udziela zamówienia na zasadach uproszczonych (prowadzi tylko dokumentację podstawowych czynności związanych z postępowaniem). Dokumentacja podstawowych czynności, podobnie jak protokół sporządzany, gdy wartość zamówienia przekracza 30 000 euro, jest jawna dla dostawców i wykonawców, którzy ubiegali się o udzielenie zamówienia.

Kiedy szacunkowa wartość zamówienia jest wyższa od kwoty 20 000 euro, a zamawiający chce zastosować tryb zamówienia z wolnej ręki, musi wcześniej uzyskać na to zgodę prezesa Urzędu Zamówień Publicznych. Zamawiający przy wartości szacunkowej zamówienia powyżej kwoty 200 000 euro musi uzyskać

zgodę prezesa Urzędu Zamówień Publicznych na zastosowanie trybu innego niż przetarg nieograniczony (bez możliwości zastosowania zamówienia z wolnej ręki).

1.3 Specyfikacja istotnych warunków zamówienia

Specyfikacja istotnych warunków zamówienia jest podstawowym dokumentem przygotowanym przez zamawiającego jeszcze przed rozpoczęciem postępowania o zamówienie publiczne o wartości powyżej 30 000 euro. Zgodnie z przepisami ustawy o zamówieniach publicznych specyfikacja powinna zawierać informacje dotyczące przedmiotu zamówienia, warunków sporządzenia i złożenia ofert przez wykonawców oraz wstępnych warunków realizacji zamówienia. Informacje powinny być przedstawione jasno, zwięźle i wyczerpująco. Szczegółową zawartość specyfikacji istotnych warunków zamówienia reguluje art. 35 ust. 1. Wynika z niego, że w specyfikacji muszą się znaleźć następujące dane:

- szczegółowy opis sposobu przygotowania ofert;
- opis kryteriów i sposobów oceny ofert;
- informacje ze wskazaniem, jakie dokumenty mają dostarczyć wykonawcy;
- szczegółowe określenie przedmiotu zamówienia;
- wskazanie pożądanego terminu wykonania umowy;
- istotne dla zamawiającego postanowienia, które zostaną wprowadzone do treści zawieranej umowy;
- opis odpowiednich części zamówienia, jeżeli dopuszczalne jest składanie ofert częściowych;
- opis sposobu obliczenia ceny oferty;
- wskazanie miejsca i terminu składania ofert;
- opis sposobu udzielania wyjaśnień dotyczących specyfikacji istotnych warunków zamówienia publicznego;
- termin, do którego wykonawca będzie związany złożoną ofertą;
- wskazanie miejsca i terminu otwarcia ofert;
- informacje o trybie otwarcia i oceny ofert;
- nazwisko, stanowisko służbowe osoby wyznaczonej do bezpośredniego kontaktowania się z wykonawcami;
- pouczenie o środkach odwoławczych przysługującym wykonawcom w toku postępowania o udzielenie zamówienia publicznego;

- ogólne warunki umowy - wzór albo regulamin.

W sytuacji, kiedy specyfikacja jest niezgodna z przepisami prawa, przedsiębiorca ma prawo złożyć na tym etapie protest do zamawiającego ze wskazaniem konkretnych zastrzeżeń. Jeżeli po zapoznaniu się ze specyfikacją złoży swoją ofertę, oznacza to akceptację treści specyfikacji istotnych warunków zamówienia.

Ustawa o zamówieniach publicznych w art. 18 ust. 4 wskazuje na możliwość stosowania preferencji krajowych, natomiast nie ma w niej mowy o preferencjach lokalnych, z czego wynika, że stosowanie preferencji lokalnych jest niezgodne z przepisami prawa. Preferencje lokalne w specyfikacji istotnych warunków zamówienia są zatem podstawą do złożenia protestu.

Kiedy specyfikacja istotnych warunków zamówienia jest zgodna z obowiązującymi przepisami, ale niezbyt jasna czy źle sformułowana, to na podstawie art. 36 ust. 1 ustawy o zamówieniach publicznych wykonawcy mogą zwracać się do zamawiającego o wyjaśnienie specyfikacji. Zamawiający jest zobowiązany do udzielenia wyjaśnień (chyba że prośba o wyjaśnienie wpłynęła na mniej niż 6 dni przed otwarciem ofert). Następnie przesyła treść wyjaśnienia wszystkim wykonawcom, którym doręczono specyfikację istotnych warunków zamówienia (bez ujawniania źródła zapytania). Zapytanie do specyfikacji może spowodować wydłużenie terminu składania ofert.

1.4 Procedura protestacyjno - odwoławcza

W specyfikacjach istotnych warunków zamówienia zamawiający rzadko informuje o trybie przetargu nieograniczonego. W sytuacji takiej należy bardzo dokładnie przeanalizować przesłanki, które zostały wcześniej wymienione odnośnie do obydwu trybów, ponieważ w procedurze przetargowej, w której wartość szacunkowa zamówienia nie przekracza 30 000 euro, nie stosuje się przepisów o protestach i odwołaniach.

Przejdźmy teraz do postępowania powyżej 30 000 euro. W toku postępowania o zamówienie publiczne wykonawca ma prawo zaskarżyć czynności zamawiającego przez wniesienie protestu. Od rozstrzygnięcia protestu (braku rozstrzygnięcia) wykonawca może złożyć odwołanie do prezesa Urzędu Zamówień Publicznych.

Należy pamiętać, że postępowaniu odwoławczemu nie podlega wybór trybu postępowania o udzielenie zamówienia publicznego, zastosowanie preferencji krajowych, odrzucenie wszystkich ofert.

Wykonawca może wnieść protest w ciągu 7 dni od dnia, w którym otrzymał wiadomość o okolicznościach stanowiących podstawę jego wniesienia. Zamawiający rozpatruje protest najpóźniej w ciągu 7 dni od jego wniesienia. Jeżeli protest zostanie uwzględniony, zamawiający powtarza oprotestowaną czynność. W sytuacji braku rozstrzygnięcia protestu w terminie należy uznać, że został on oddalony.

W przypadku rozstrzygnięcia, odrzucenia jak też nierozpoznania protestu w terminie zainteresowanemu wykonawcy przysługuje odwołanie do prezesa Urzędu Zamówień Publicznych. Należy pamiętać, że wykonawca musi wyczerpać tok instancji określony w ustawie o zamówieniach publicznych, a więc przed wniesieniem odwołania do prezesa UZP powinien uprzednio wnieść protest do zamawiającego. Odwołanie zainteresowany wykonawca wnosi w terminie 3 dni od dnia doręczenia rozstrzygnięcia protestu lub 7 dni w przypadku braku rozstrzygnięcia protestu bezpośrednio do prezesa Urzędu Zamówień Publicznych.

Podstawowe elementy formalne odwołania zostały określone w Regulaminie postępowania przy rozpatrywaniu odwołań w sprawach o udzielenie zamówień publicznych z 20 sierpnia 1999 r. (DzU nr 73 z 1999 r., poz. 815). Regulamin § 2 ust. 1 wymaga zamieszczenia w odwołaniu następujących danych:

1. nazwa (firma) odwołującego się,
2. wskazanie nazwy i adresu zamawiającego,
3. określenie przedmiotu zamówienia,
4. określenie daty wniesienia protestu,
5. wskazanie podstaw prawnych i okoliczności uzasadniających wniesienie odwołania,
6. podpis osoby uprawnionej na podstawie odrębnych przepisów do składania oświadczeń woli w imieniu odwołującego się,
7. załączniki - kopię protestu oraz kopię rozstrzygnięcia protestu, jeżeli został rozpatrzony.

Dodatkowym wymogiem formalnym wynikającym z przepisów rozporządzenia Rady Ministrów z 10 marca 1998 r. w sprawie wysokości oraz szczegółowych zasad pobierania wpisu od odwołań wnoszonych w postępowaniu o

udzielenie zamówień publicznych jest konieczność opłacenia wpisu. Opłacenie wpisu musi nastąpić najpóźniej w terminie 7 dni od daty otrzymania wezwania. Nieuiszczenie wpisu lub nieuiszczenie wpisu w terminie powoduje, że wniesione odwołanie pozostaje bez rozpoznania. Wysokość wpisu jest zmienna, została określona jako czterokrotna kwota najniższego miesięcznego wynagrodzenia, która jest określana przez ministra pracy i polityki socjalnej. Po spełnieniu wymagań formalnych przez zainteresowanego wykonawcę, odwołanie jest rozpatrzone przez zespół trzech arbitrów w terminie do 14 dni. W sytuacji uwzględnienia odwołania przez zespół arbitrów może on nakazać w orzeczeniu kończącym postępowanie powtórzenie czynności przez zamawiającego lub tę czynność unieważnić.

Przekroczenie któregokolwiek terminu w procedurze protestacyjno-odwoławczej powoduje oddalenie odwołania ze względów formalnych (bez merytorycznego rozpoznania zarzutów).

W sytuacji, kiedy odwołanie zostało oddalone ze względów formalnych, zainteresowany wykonawca może na podstawie art. 9 ust. 2 pkt. 2 ustawy o zamówieniach publicznych złożyć do prezesa Urzędu Zamówień Publicznych skargę na zamawiającego.

Kontrolą w trybie art. 9 ust. 2 pkt. 2 mogą być objęte wszystkie postępowania o zamówienia publiczne bez względu na wartość szacunkową zamówienia.

2 MODEL ŚRODOWISKA PROJEKTOWANEGO SYSTEMU

Nowoczesna analiza strukturalna zakłada zbudowanie tak zwanego **modelu zasadniczego (ang. essential model),** składającego się z dwóch części:

- modelu środowiska (ang. environmental model);
- modelu zachowań (ang. behavioral model).

Podstawowym założeniem przy budowie modelu zasadniczego jest skoncentrowanie się na wymaganiach użytkownika (lub wymaganiach wynikających z logiki realizowanych procesów), bez wdawania się w szczegóły implementacyjne i bez opisywania (modelowania) istniejącego systemu, czyli bez budowania aktualnego modelu fizycznego.

Głównym celem budowy **modelu środowiska** jest odwzorowanie otoczenia systemu - środowiska otaczającego system, jego kontekstu. Nie chodzi tu głównie o oddziaływanie środowiska na system lecz o wyznaczenie granic projektowanego systemu. Użytkownik i projektant muszą być zgodni co do tego, jakiego obszaru ma dotyczyć system - gdzie się zaczyna i kończy.

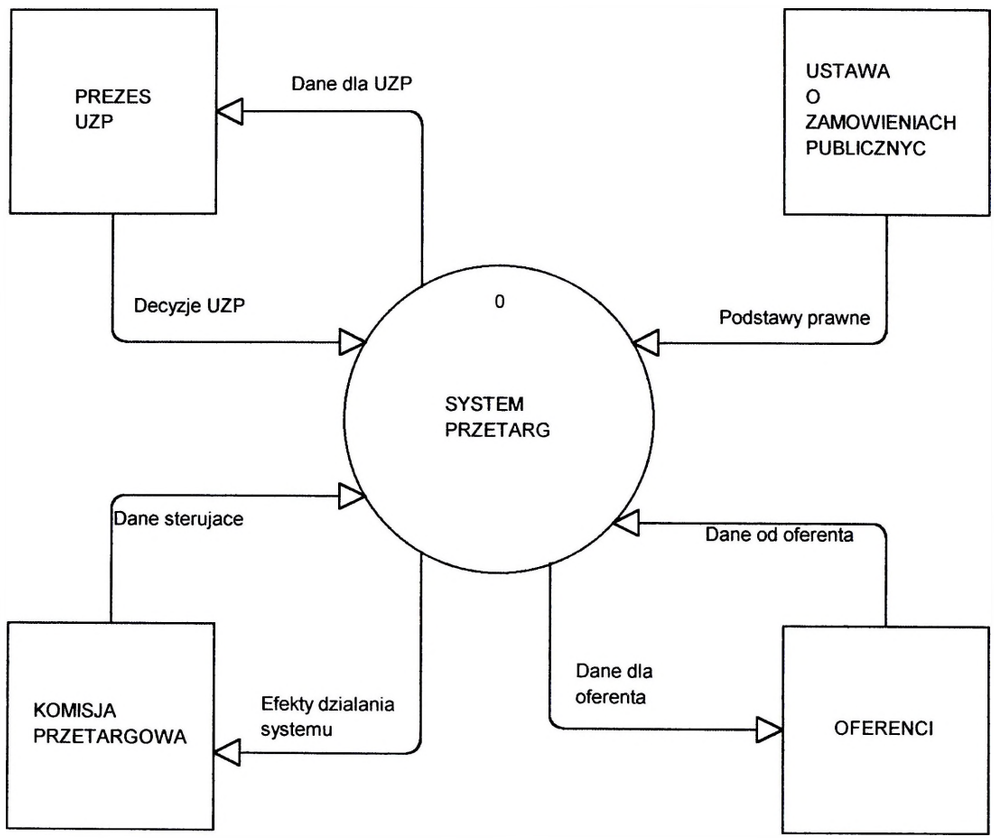
Model środowiska składa się z trzech elementów, którymi są:

- określenie celu;
- diagram kontekstowy;
- lista zdarzeń.

Określenie celu to kilka zwięzłych zdań opisujących cel systemu. Diagram kontekstowy przedstawia system, obiekty zewnętrzne (w stosunku do systemu) oddziałujące na system oraz przepływy danych łączące obiekty zewnętrzne z systemem. Diagram kontekstowy jest szczególną postacią diagramu przepływu danych, służącym do wyznaczenia granic systemu.

Lista zdarzeń to specyfikacja zdarzeń, które oddziałują na system i na które system musi odpowiadać. Zdarzenia przedstawione są na diagramie kontekstowym w postaci przepływów danych. Ważne jest, aby listę zdarzeń sporządzać z punktu widzenia otoczenia systemu, a nie „ze środka systemu".

Poniżej przedstawiono diagram kontekstowy systemu przetwarzania informacji wspomagającego prace komisji przetargowej nazwanego w skrócie systemem **PRZETARG**. Na diagramie tym zobrazowane są obiekty zewnętrzne w stosunku do systemu oraz dane, jakie płyną między systemem a tymi obiektami. Poniższy diagram wraz z opisem znajdujących się na nim elementów stanowi model środowiska projektowanego systemu informatycznego.



Rys. 2-1. Diagram kontekstowy systemu przetwarzania informacji wspomagającego prace komisji przetargowej

Na diagramie kontekstowym wyróżniono cztery obiekty zewnętrzne (terminatory):

- **PREZES UZP** – terminator reprezentuje Prezesa Urzędu Zamówień Publicznych wraz z podległym mu Urzędem Zamówień Publicznych jako organizacją, z którą współpracuje komisja przetargowa w toku realizacji procedury przetargowej.

- **USTAWA O ZAMÓWIENIACH PUBLICZNYCH** – terminator reprezentuje uwarunkowania prawne i definicję procedury przetargowej, stanowiące podstawę prawną działań komisji przetargowej i systemu PRZETARG.
- **OFERENCI** – terminator reprezentuje firmy biorące udział w procedurze przetargowej.
- **KOMISJA PRZETARGOWA** – terminator reprezentuje komisję przetargową, powołaną do realizacji procedury przetargowej.

Obiekty zewnętrzne (terminalne) dostarczają informacji, która powoduje wykonanie lub zaniechanie wykonania określonych procesów w systemie oraz odbierają informacje będące efektem działania systemu.

Zadania, funkcje i możliwości poszczególnych obiektów zewnętrznych oraz podstawowe dane otrzymywane i wytwarzane przez projektowany system są następujące:

- **PREZES UZP** – decyduje o przyjęciu trybu udzielenia zamówienia publicznego, publikuje informacje o ogłoszeniu i rozstrzygnięciu przetargu w Biuletynie Zamówień Publicznych, steruje procedurą odwoławczą.

  Strumienie danych:

  → *Dane dla UZP* – wnioski do UZP dotyczące ustalenia trybu udzielenia zamówienia publicznego, skrócenia ustawowego terminu trwania procedury przetargowej. Specyfikacja istotnych warunków zamówienia, stanowiąca podstawę ogłoszenia przetargu. Informacja o rozstrzygnięciu przetargu.

  ← *Decyzje UZP* – decyzje Prezesa UZP będące efektem rozpatrzenia wniosków komisji przetargowej. W szczególności ogłoszenia w Biuletynie Zamówień Publicznych wszczęcia procedury przetargowej i jej rozstrzygnięcia.

- **USTAWA O ZAMÓWIENIACH PUBLICZNYCH** – jest zbiorem dokumentów prawnych (ustawy, rozporządzenia) definiujących uwarunkowania prawne i określających zakres, sekwencję działań i sposób

organizacji procedury przetargowej. Ustalenia dostarczane przez ten terminator stanowią podstawę prawną działań komisji przetargowej i systemu PRZETARG.

Strumienie danych:

→ *Podstawy prawne* – ustalenia zawarte w ustawie o zamówieniach publicznych z 10 czerwca 1994 r. (tekst jednolity DzU Nr 119 z 1998 r. poz. 773). Dodatkowe ustalenia wynikające z rozporządzeń i porozumień międzyresortowych dotyczące cytowanej ustawy. Także komentarze i opinie prawne do ustawy i rozporządzeń.

- **OFERENCI** – terminator reprezentuje firmy biorące udział w procedurze przetargowej, w jakimkolwiek momencie jej trwania. W przypadku wyboru trybu przetargu ograniczonego lub dwustopniowego firmy brane pod uwagę przez komisję przetargową jako potencjalni oferenci. W toku trwania procedury przetargowej – firmy, które pobrały SIWZ. W toku rozstrzygania przetargu – firmy które złożyły oferty (oferenci).

  Strumienie danych:

  → *Dane dla oferenta* – wszystkie informacje przekazywane oferentowi w trakcie trwania procedury systemowej. Między innymi informacje zawarte w:
  - ogłoszeniu przetargu,
  - specyfikacji istotnych warunków zamówienia lub zapytaniu o cenę,
  - odpowiedzi na zapytania firm,
  - odpowiedzi na zgłoszone protesty.

  → *Dane od oferenta* – informacje przekazywane przez oferentów (firmy biorące udział w procedurze przetargowej) projektowanemu systemowi. Między innymi informacje zawarte w:
  - zgłoszeniu się do przetargu (fakcie pobrania SIWZ),

- zapytaniach firm,
- ofertach firm,
- zgłoszonych protestach.

- **KOMISJA PRZETARGOWA** – zespół powołany przez kierownictwo spośród pracowników, odpowiedzialny za realizację procedury przetargowej. Składa się zwykle z przewodniczącego komisji, jej sekretarza i członków. Na użytek niniejszej analizy osoby funkcyjne komisji będą utożsamiane z komisją. Komisja przetargowa steruje realizacją procedury przetargowej wspomaganej projektowanym systemem; jest jego użytkownikiem.

  Strumienie danych:

  → *Efekty działania systemu* – informacje dostarczane przez system komisji przetargowej. Informacje te mogą być monitami systemu do komisji o terminowe podjęcie stosownych decyzji lub odpowiedzią systemu na żądania komisji.

  ← *Dane sterujące* – informacje dotyczące procedury przetargowej zasilające system, dostarczane przez komisję, decyzje komisji lub żądania danych od systemu.

Uściślenie i konkretyzacja zawartości wyróżnionych przepływów danych nastąpi w opisie diagramów DFD, będących rozwinięciem diagramu kontekstowego.

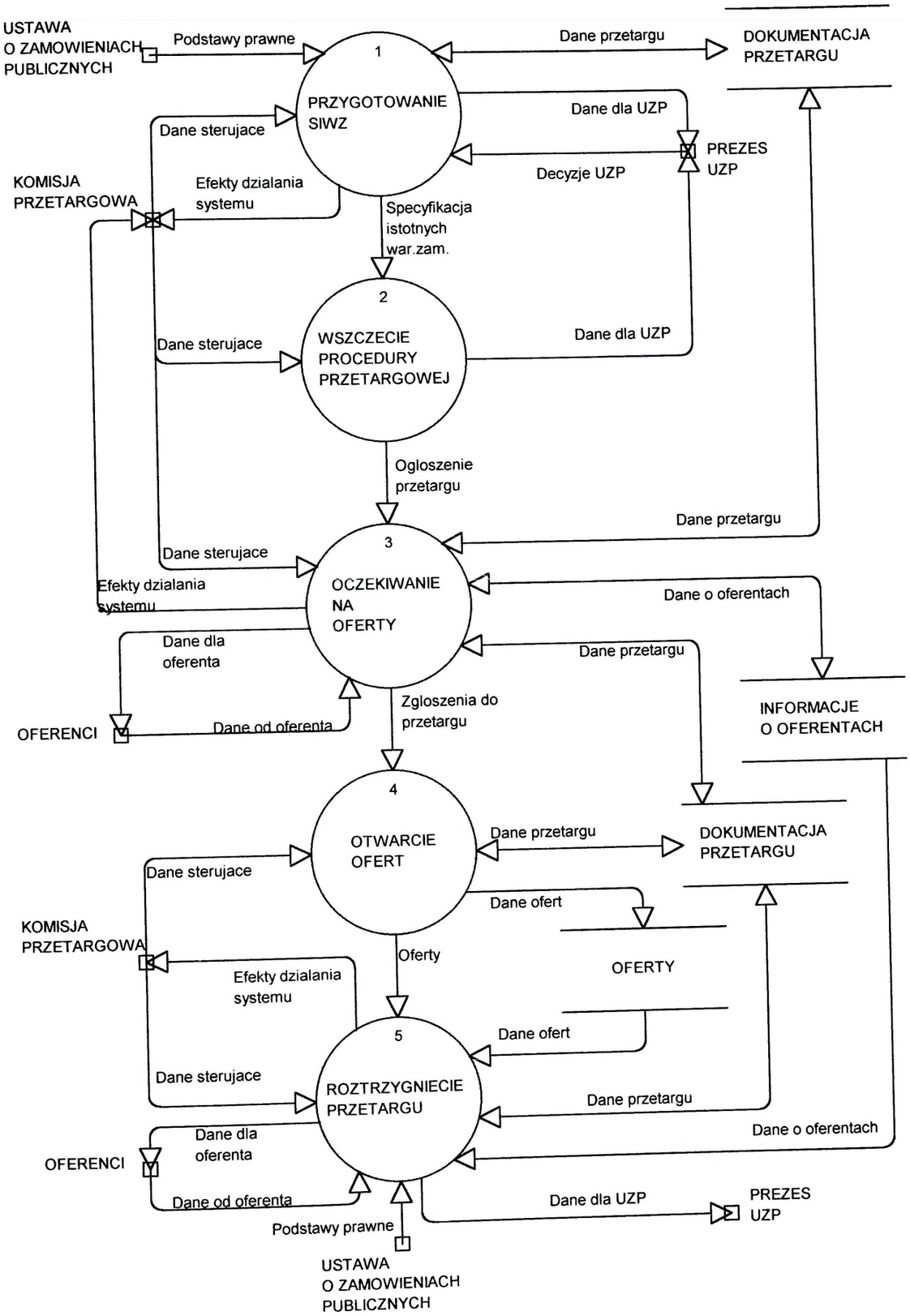
3 MODEL ZACHOWAŃ PROJEKTOWANEGO SYSTEMU

Celem budowy **modelu zachowań** jest identyfikacja odpowiedzi (reakcji) systemu na zdarzenia wyspecyfikowane na liście zdarzeń. Odpowiedzi są to procesy zachodzące w systemie. Ilość procesów w systemie implikowana jest ilością zdarzeń. W modelu zachowań definiowane są także dane. Specyfikowane są w specjalnych miejscach systemu, zwanych magazynami lub składnicami danych, ponieważ procesy w systemie zachodzą w sposób asynchroniczny i często nieciągły. Ogólną reprezentację modelu zachowań stanowi diagram poziomu zerowego.

Diagram poziomu zerowego projektowanego systemu informatycznego wspomagającego prace komisji przetargowej, przedstawiono na rysunku 3-1. Na diagramie tym wyróżniono pięć podstawowych funkcji systemu:

1. Przygotowanie specyfikacji istotnych warunków zamówienia
2. Wszczęcie procedury przetargowej
3. Oczekiwanie na oferty
4. Otwarcie ofert
5. Rozstrzygniecie przetargu

będących odzwierciedleniem podstawowych faz procedury przetargowej i ich sekwencji w jej realizacji.



Rys. 3-1 Diagram poziomu zerowego projektowanego systemu przetwarzania informacji wspomagającego prace komisji przetargowej.

3.1 Przygotowanie specyfikacji istotnych warunków zamówienia (proces 1)

◊ OPIS PRZEZNACZENIA: proces ma za zadanie wspomóc prace komisji przetargowej w tworzeniu specyfikacji istotnych warunków zamówienia (SIWZ). Podstawą do tworzenia SIWZ jest ustalenie trybu udzielenia zamówienia publicznego, opracowanie harmonogramu procedury przetargowej oraz edycja i wydanie SIWZ jako dokumentu.

◊ OBIEKTY ZEWNĘTRZNE:

- PREZES UZP – adresat wniosków o zmianę trybu udzielenia zamówienia publicznego, skrócenie procedury przetargowej (o ile takie wnioski komisja będzie składać).
- USTAWA O ZAMÓWIENIACH PUBLICZNYCH – dostarcza podstaw prawnych ustalania trybu udzielenia zamówienia publicznego, ustalania harmonogramu procedury przetargowej i wymogów dotyczących obligatoryjnej zawartości SIWZ.
- KOMISJA PRZETARGOWA – sterująca działaniem procesu tworzenia SIWZ.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Podstawy prawne – artykuły ustawy o zamówieniach publicznych.
- Dane sterujące – informacje od komisji przetargowej niezbędne do skonstruowania SIWZ; także żądania dostarczenia przez proces informacji o dotychczasowych dokonaniach w tworzeniu SIWZ.
- Dane przetargu – informacje o przebiegu procedury przetargowej, stanie jej zaawansowania, przyjętych ustaleniach.
- Decyzje UZP – informacje zawarte w odpowiedziach Urzędu Zamówień Publicznych na formalne wnioski komisji przetargowej, wystosowane do Prezesa UZP.

→ Dane wyjściowe:

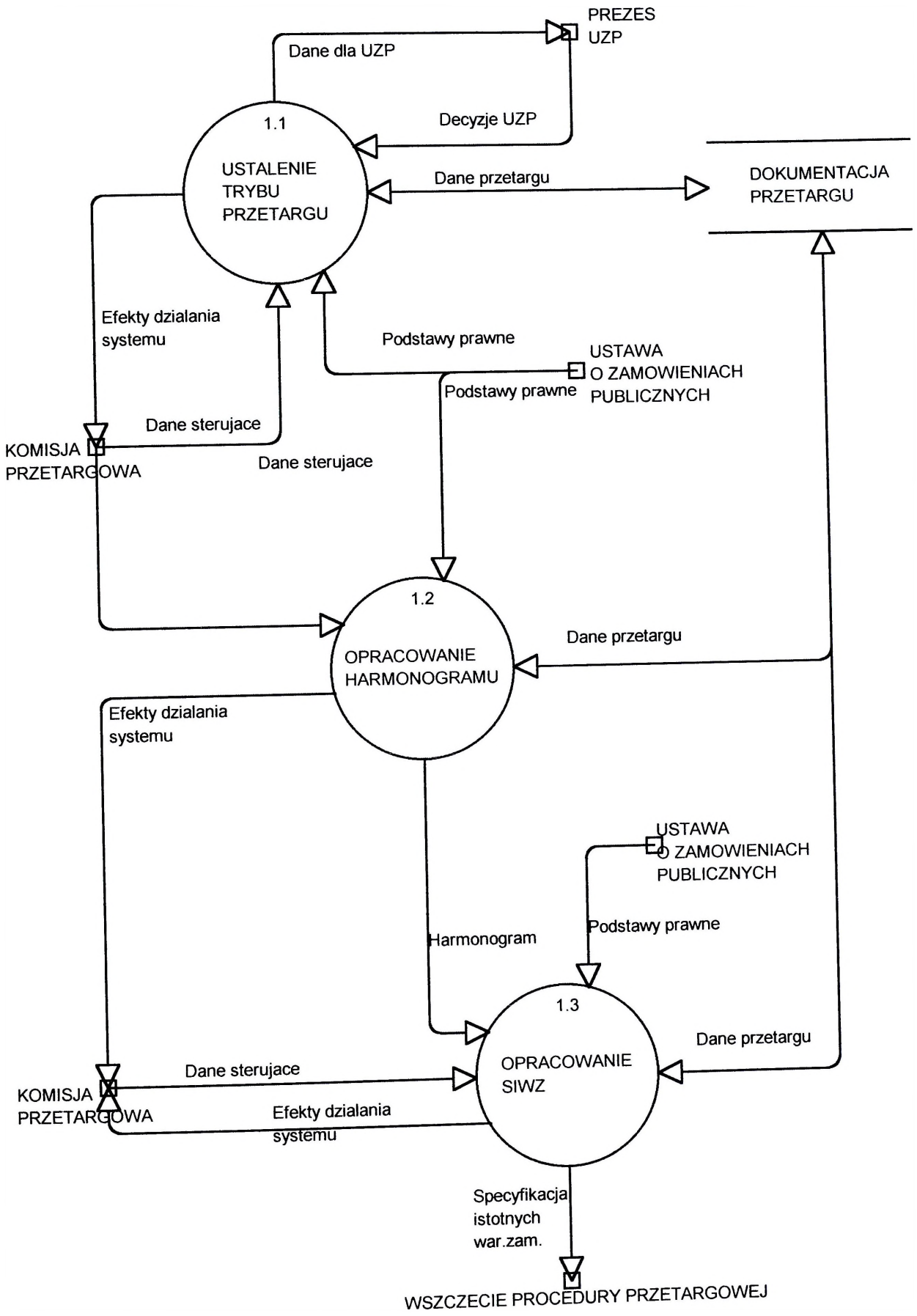
- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji żądań komisji skierowanych do procesu. Komunikaty wynikające z kontroli formalnej i logicznej zgromadzonej dokumentacji przetargu, kolejne wersje SIWZ.
- Dane dla UZP – informacje zawarte we wnioskach komisji przetargowej wystosowanych do Prezesa UZP.
- Dane przetargu – informacje, będące efektem działania procesu, warte zapisania w repozytorium systemu.
- Specyfikacja istotnych warunków zamówienia – informacje precyzujące między innymi:
  - szczegółowy opis sposobu przygotowania ofert;
  - opis kryteriów i sposobów oceny ofert;
  - informacje ze wskazaniem, jakie dokumenty mają dostarczyć wykonawcy;
  - szczegółowe określenie przedmiotu zamówienia;
  - wskazanie pożądanego terminu wykonania umowy;
  - istotne dla zamawiającego postanowienia, które zostaną wprowadzone do treści zawieranej umowy;
  - opis odpowiednich części zamówienia, jeżeli dopuszczalne jest składanie ofert częściowych;
  - opis sposobu obliczenia ceny oferty;
  - wskazanie miejsca i terminu składania ofert;
  - opis sposobu udzielania wyjaśnień dotyczących specyfikacji istotnych warunków zamówienia publicznego;
  - termin, do którego wykonawca będzie związany złożoną ofertą;
  - wskazanie miejsca i terminu otwarcia ofert;
  - informacje o trybie otwarcia i oceny ofert;
  - nazwisko, stanowisko służbowe osoby wyznaczonej do bezpośredniego kontaktowania się z wykonawcami;
  - pouczenie o środkach odwoławczych przysługującym wykonawcom w toku postępowania o udzielenie zamówienia publicznego;

– ogólne warunki umowy - wzór albo regulamin

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – składnica danych, pełniąca rolę repozytorium systemu. Zapisywane są w niej wszelkie informacje przydatne w realizacji procedury przetargowej. Podczas realizacji bieżącego procesu w składnicy zapisywane są dokumenty będące efektem korespondencji z UZP, ustaleń dotyczących tworzenia harmonogramu realizacji procedury przetargowej, kolejne wersje tworzonej SIWZ.

◊ ELEMENTY SKŁADOWE PROCESU: rysunek 3-2 przedstawia diagram DFD dla procesu 1 przygotowania SIWZ. Wyróżniono na nim 3 podprocesy.



Rys. 3-2 Diagram DFD dla procesu przygotowania specyfikacji istotnych warunków zamówienia (SIWZ)

3.1.1 Ustalenie trybu przetargu (proces 1.1)

◊ OPIS PRZEZNACZENIA: proces ma wspomóc komisję przetargową w określeniu wartości szacunkowej zamówienia i ustaleniu trybu przetargu (w porozumieniu z UZP). Zgodnie z art. 15 ust. 1 w przetargu nieograniczonym, w którym wartość zamówienia przekracza 30 000 euro, zamawiający zobowiązany jest stosować przepisy ustawy dotyczące publikacji ogłoszeń w Biuletynie Zamówień Publicznych, pisemności postępowania, protokołu postępowania, specyfikacji istotnych warunków zamówienia, terminów, wadium, protestów i odwołań. Kiedy szacunkowa wartość zamówienia jest wyższa od kwoty 20 000 euro, a zamawiający chce zastosować tryb zamówienia z wolnej ręki, musi wcześniej uzyskać na to zgodę prezesa Urzędu Zamówień Publicznych. Zamawiający przy wartości szacunkowej zamówienia powyżej kwoty 200 000 euro musi uzyskać zgodę prezesa Urzędu Zamówień Publicznych na zastosowanie trybu innego niż przetarg nieograniczony (bez możliwości zastosowania zamówienia z wolnej ręki).

◊ OBIEKTY ZEWNĘTRZNE:

- PREZES UZP – adresat wniosków o zmianę trybu udzielenia zamówienia publicznego, skrócenie procedury przetargowej (o ile takie wnioski komisja będzie składać).
- USTAWA O ZAMÓWIENIACH PUBLICZNYCH – dostarcza podstaw prawnych ustalania trybu udzielenia zamówienia publicznego.
- KOMISJA PRZETARGOWA – określająca szacunkową wartość zamówienia i występująca do Prezesa UZP.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Podstawy prawne – fragmenty ustawy o zamówieniach publicznych dotyczące określeniu wartości szacunkowej zamówienia i ustaleniu trybu przetargu.

- Dane sterujące – informacje od komisji przetargowej niezbędne do kalkulacji wartości szacunkowej zamówienia i decydujące o ustaleniu trybu przetargu.
- Dane przetargu – dotychczasowe ustalenia dotyczące wartości porównywalnych zamówień i korespondencji z UZP.
- Decyzje UZP – informacje zawarte w odpowiedziach Urzędu Zamówień Publicznych na formalne wnioski komisji przetargowej, wystosowane do Prezesa UZP.

→ Dane wyjściowe:

- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji żądań komisji skierowanych do procesu. Wyniki kalkulacji wartości zamówienia, oznaczenia oraz treść dokumentów będących wynikiem korespondencji z UZP.
- Dane dla UZP – informacje zawarte we wnioskach komisji przetargowej wystosowanych do Prezesa UZP.
- Dane przetargu – informacje, będące efektem działania procesu, warte zapisania w repozytorium systemu.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – składnica modyfikowana jest informacjami będącymi efektem korespondencji z UZP i ustaleń wartości szacunkowej zamówienia.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.1.2 Opracowanie harmonogramu przetargu (proces 1.2)

◊ OPIS PRZEZNACZENIA: proces ma za zadanie wspomóc komisję przetargową w ustaleniu harmonogramu realizacji kolejnych faz procedury przetargowej. Ustalana jest orientacyjna data ogłoszenia przetargu i zależne od niej daty:

składania ofert, składania zapytań, otwarcia ofert, rozstrzygnięcia przetargu, oczekiwania na odwołania, podpisania umów i realizacji zamówienia.

◊ OBIEKTY ZEWNĘTRZNE:

- USTAWA O ZAMÓWIENIACH PUBLICZNYCH – dostarcza podstaw prawnych ustalania harmonogramu procedury przetargowej.
- KOMISJA PRZETARGOWA – określająca Harmonogram procedury przetargowej.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Podstawy prawne – fragmenty ustawy o zamówieniach publicznych dotyczące terminów procedury przetargowej.
- Dane sterujące – informacje od komisji przetargowej niezbędne do ustalenia harmonogramu procedury przetargowej.
- Dane przetargu – dotychczasowe ustalenia mające wpływ na ustalenie harmonogramu procedury przetargowej.

→ Dane wyjściowe:

- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji żądań komisji skierowanych do procesu. Wyniki ustalania harmonogramu procedury przetargowej.
- Dane przetargu – informacje, będące efektem działania procesu, warte zapisania w repozytorium systemu.
- Harmonogram przetargu – ustalenia dotyczące orientacyjnej daty ogłoszenia przetargu i zależnych od niej dat: składania ofert, składania zapytań, otwarcia ofert, rozstrzygnięcia przetargu, oczekiwania na odwołania, podpisania umów i realizacji zamówienia.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – składnica modyfikowana jest informacjami będącymi efektem ustalenia harmonogramu przetargu.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.1.3 Opracowanie specyfikacji istotnych warunków zamówienia (proces 1.3)

◊ OPIS PRZEZNACZENIA: zadaniem procesu jest wspomaganie komisji przetargowej w tworzeniu specyfikacji istotnych warunków zamówienia (SIWZ). Efektem pracy procesu jest dokument określający:

- szczegółowy opis sposobu przygotowania ofert;
- opis kryteriów i sposobów oceny ofert;
- informacje ze wskazaniem, jakie dokumenty mają dostarczyć wykonawcy;
- szczegółowe określenie przedmiotu zamówienia;
- wskazanie pożądanego terminu wykonania umowy;
- istotne dla zamawiającego postanowienia, które zostaną wprowadzone do treści zawieranej umowy;
- opis odpowiednich części zamówienia, jeżeli dopuszczalne jest składanie ofert częściowych;
- opis sposobu obliczenia ceny oferty;
- wskazanie miejsca i terminu składania ofert;
- opis sposobu udzielania wyjaśnień dotyczących specyfikacji istotnych warunków zamówienia publicznego;
- termin, do którego wykonawca będzie związany złożoną ofertą;
- wskazanie miejsca i terminu otwarcia ofert;
- informacje o trybie otwarcia i oceny ofert;
- nazwisko, stanowisko służbowe osoby wyznaczonej do bezpośredniego kontaktowania się z wykonawcami;
- pouczenie o środkach odwoławczych przysługującym wykonawcom w toku postępowania o udzielenie zamówienia publicznego;
- ogólne warunki umowy - wzór albo regulamin

◊ OBIEKTY ZEWNĘTRZNE:

- USTAWA O ZAMÓWIENIACH PUBLICZNYCH – dostarcza podstaw prawnych tworzenia SIWZ.
- KOMISJA PRZETARGOWA – tworzy SIWZ.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Podstawy prawne – fragmenty ustawy o zamówieniach publicznych dotyczące SIWZ.
- Dane sterujące – informacje od komisji przetargowej niezbędne do tworzenia SIWZ.
- Dane przetargu – dotychczasowe ustalenia mające wpływ na tworzenia SIWZ
- Harmonogram przetargu – ustalenia dotyczące orientacyjnej daty ogłoszenia przetargu i zależnych od niej dat: składania ofert, składania zapytań, otwarcia ofert, rozstrzygnięcia przetargu, oczekiwania na odwołania, podpisania umów i realizacji zamówienia.

→ Dane wyjściowe:

- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji żądań komisji skierowanych do procesu. Wyniki ustalania SIWZ.
- Dane przetargu – informacje, będące efektem działania procesu, warte zapisania w repozytorium systemu.
- Specyfikacja istotnych warunków zamówienia – dokument stanowiący podstawę konstrukcji ofert, ich oceny i przebiegu procedur przetargowych.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – składnica modyfikowana jest informacjami zawartymi w SIWZ.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.2 Wszczęcie procedury przetargowej (proces 2)

◊ OPIS PRZEZNACZENIA: zadaniem procesu jest wspomaganie komisji przetargowej w pracach związanych z przygotowaniem wniosku o ogłoszenie informacji o przetargu w Biuletynie Zamówień Publicznych.

◊ OBIEKTY ZEWNĘTRZNE:

- PREZES UZP – ogłaszający informacje o przetargu w Biuletynie Zamówień Publicznych.
- KOMISJA PRZETARGOWA – tworzy wniosek o ogłoszenie informacji o przetargu w Biuletynie Zamówień Publicznych.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Specyfikacja istotnych warunków zamówienia – dokument stanowiący podstawę konstrukcji wniosku o ogłoszenie informacji o przetargu w Biuletynie Zamówień Publicznych.
- Dane sterujące – informacje od komisji przetargowej sterujące tworzeniem wniosku.

→ Dane wyjściowe:

- Ogłoszenie przetargu – informacje zawarte w Biuletynie Zamówień Publicznych, dotyczące przetargu.
- Dane dla UZP - wniosek o ogłoszenie informacji o przetargu w Biuletynie Zamówień Publicznych

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.3 Oczekiwanie na oferty (proces 3)

◊ OPIS PRZEZNACZENIA: proces wspomaga komisję przetargową w pracach związanych z dystrybucją SIWZ, przyjmowaniem zapytań, opracowaniem odpowiedzi na zapytania i ich dystrybucją.

◊ OBIEKTY ZEWNĘTRZNE:

- OFERENCI – firmy biorące udział w procedurze przetargowej.
- KOMISJA PRZETARGOWA – dystrybuująca SIWZ, przyjmująca zapytania, opracowująca odpowiedzi na zapytania.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Ogłoszenie przetargu – informacje zawarte w Biuletynie Zamówień Publicznych , dotyczące przetargu.
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje od komisji przetargowej sterujące działaniem procesu.
- Dane od oferenta – informacje o firmach, które pobrały SIWZ; zapytania oferentów dotyczące SIWZ.
- Dane o oferentach – informacje o firmach, które pobrały SIWZ; dotychczasowa korespondencja z oferentami.

→ Dane wyjściowe:

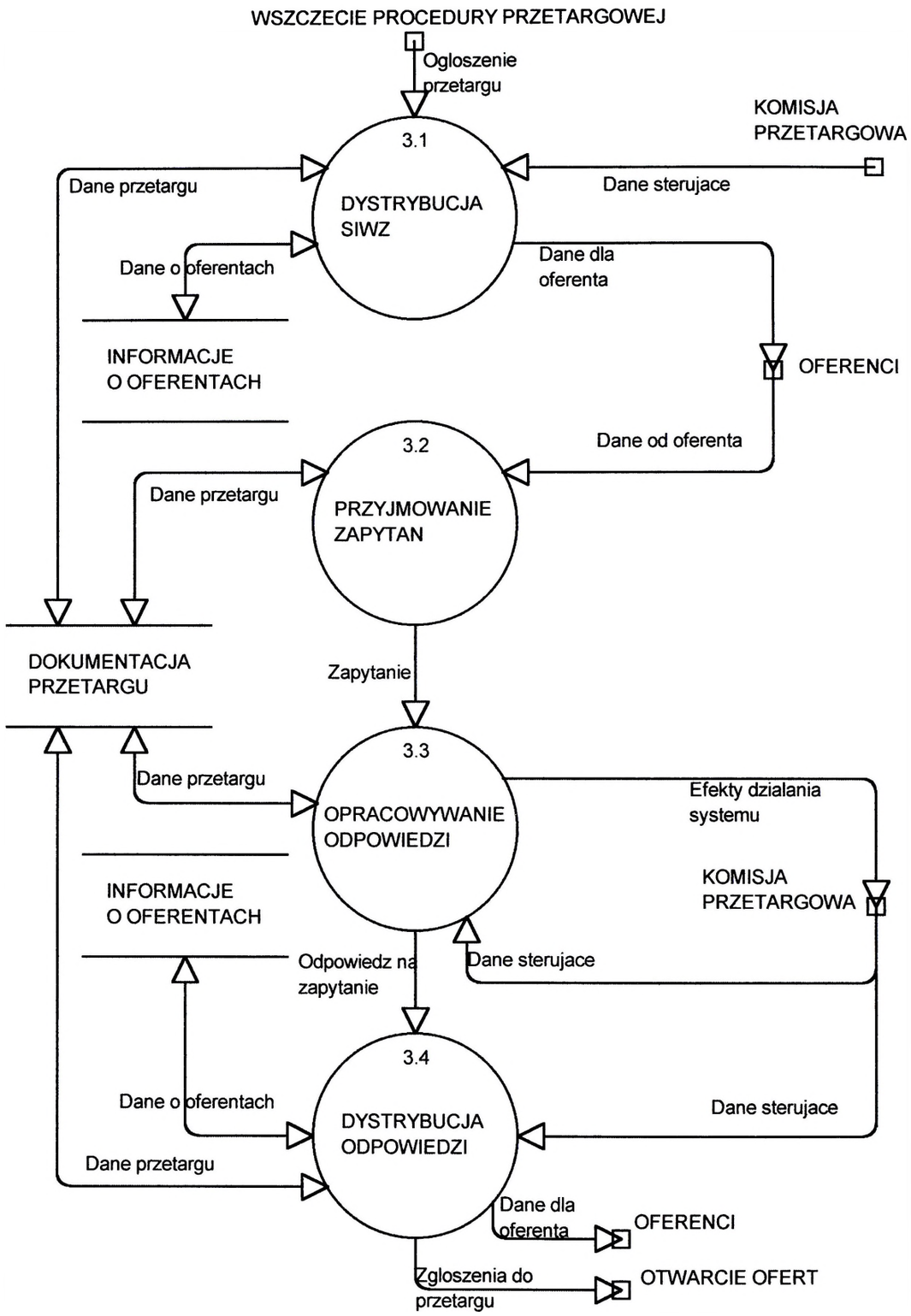
- Dane o oferentach - informacje o firmach, które pobrały SIWZ; efekty korespondencji z oferentami.
- Dane przetargu – informacje modyfikujące dokumentację dotyczące przetargu.
- Dane dla oferenta – SIWZ oraz odpowiedzi na zapytania dotyczące SIWZ.
- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji żądań komisji skierowanych do procesu, dotyczące dystrybucji SIWZ, przyjmowania zapytań, opracowania odpowiedzi na zapytania i ich rozsyłaniem.

- Zgłoszenia do przetargu – informacje o złożonych ofertach i wniesieniu wadium przez firmy

◊ SKŁADNICE DANYCH:

- INFORMACJE O OFERENTACH – w składnicy gromadzone są informacje, umożliwiające kontakt z oferentami oraz obrazujące efekty tych kontaktów (korespondencję dotyczącą przetargu, fakt złożenia oferty).
- DOKUMENTACJA PRZETARGU – w składnicy danych zapisywane są wszelkie informacje związane z efektami dystrybucji SIWZ, przyjmowania zapytań, opracowania odpowiedzi na zapytania i ich dystrybucją.

◊ ELEMENTY SKŁADOWE PROCESU: rysunek 3-3 przedstawia diagram DFD dla procesu 3 oczekiwania na oferty. Wyróżniono na nim 4 podprocesy:



Rys. 3-3 Diagram DFD dla procesu przygotowania specyfikacji istotnych warunków zamówienia (SIWZ)

3.3.1 Dystrybucja SIWZ (proces 3.1)

◊ OPIS PRZEZNACZENIA: proces wspomaga prace komisji przetargowej związane z wydawaniem (sprzedażą, rozsyłaniem) SIWZ oraz tworzeniem bazy danych informacji o potencjalnych oferentach.

◊ OBIEKTY ZEWNĘTRZNE:

- OFERENCI – firmy pobierające SIWZ.
- KOMISJA PRZETARGOWA – dystrybuująca SIWZ.
- WSZCZĘCIE PROCEDURY PRZETARGOWEJ – proces wspomagający przygotowaniem wniosku o ogłoszenie informacji o przetargu w Biuletynie Zamówień Publicznych.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Ogłoszenie przetargu – informacje zawarte w Biuletynie Zamówień Publicznych, dotyczące przetargu.
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje od komisji przetargowej sterujące działaniem procesu.
- Dane o oferentach – informacje o firmach, które pobrały SIWZ.

→ Dane wyjściowe:

- Dane przetargu – informacje o dystrybucji SIWZ.
- Dane o oferentach – informacje o firmach, które pobrały SIWZ.
- Dane dla oferenta – SIWZ pobierany przez oferenta.

◊ SKŁADNICE DANYCH:

- INFORMACJE O OFERENTACH – w składnicy gromadzone są informacje, umożliwiające kontakt z oferentami.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.3.2 Przyjmowanie zapytań (proces 3.2)

◊ OPIS PRZEZNACZENIA: proces wspomaga przyjmowanie zapytań od potencjalnych oferentów, dotyczących informacji zawartych w SIWZ.

◊ OBIEKTY ZEWNĘTRZNE:

- OFERENCI – firmy, które wystosowały zapytania.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane od oferenta – zapytania oferentów dotyczące SIWZ.

→ Dane wyjściowe:

- Dane przetargu – informacje o zadanych pytaniach dotyczących SIWZ.
- Zapytanie – informacje zawarte w zapytaniach od potencjalnych oferentów.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – w składnicy danych zapisywane są informacje związane z przyjmowania zapytań od potencjalnych oferentów.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.3.3 Opracowanie odpowiedzi na zapytania (proces 3.3)

◊ OPIS PRZEZNACZENIA: proces wspomaga opracowanie odpowiedzi na zadane przez potencjalnych oferentów zapytania, sygnalizuje zbliżanie się terminu udzielenia odpowiedzi, blokuje odpowiadanie na zapytania zadane po ustalonym w SIWZ terminie.

◊ OBIEKTY ZEWNĘTRZNE:

- KOMISJA PRZETARGOWA – opracowująca odpowiedzi na zapytania.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Zapytanie – informacje zawarte w zapytaniach od potencjalnych oferentów.
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje od komisji przetargowej sterujące działaniem procesu.

→ Dane wyjściowe:

- Dane przetargu – informacje modyfikujące dokumentację dotyczące przetargu.
- Efekty działania systemu – informacje dla komisji przetargowej, sygnalizujące zbliżanie się terminu udzielenia odpowiedzi.
- Odpowiedź na zapytanie – informacje zawarte w odpowiedzi na zapytanie zadane przez potencjalnego oferenta.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.3.4 Dystrybucja odpowiedzi na zapytania (proces 3.4)

◊ OPIS PRZEZNACZENIA: proces wspomaga ustalenie listy oferentów i zgodne z nią rozsyłanie odpowiedzi na zapytania wszystkim potencjalnym oferentom.

◊ OBIEKTY ZEWNĘTRZNE:

- OFERENCI – firmy biorące udział w procedurze przetargowej.
- KOMISJA PRZETARGOWA – dystrybuująca odpowiedzi na zapytania, tworząca listę oferentów.

- OTWARCIE OFERT – proces wspomagający przebieg posiedzenia komisji przetargowej, związanego z otwarciem ofert.

← Dane wejściowe:

- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje od komisji przetargowej sterujące działaniem procesu.
- Dane o oferentach – informacje o firmach, które pobrały SIWZ; dotychczasowa korespondencja z oferentami.
- Odpowiedź na zapytanie – informacje zawarte w odpowiedzi na zapytanie zadane przez potencjalnego oferenta.

→ Dane wyjściowe:

- Dane o oferentach - efekty korespondencji z oferentami.
- Dane przetargu – informacje modyfikujące dokumentację dotyczące przetargu.
- Dane dla oferenta – odpowiedzi na zapytania dotyczące SIWZ.
- Zgłoszenia do przetargu – informacje o złożonych ofertach i wniesieniu wadium przez firmy

◊ SKŁADNICE DANYCH:

- INFORMACJE O OFERENTACH – w składnicy gromadzone są informacje, umożliwiające kontakt z oferentami oraz obrazujące efekty tych kontaktów (korespondencję dotyczącą przetargu, fakt złożenia oferty).

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.4 Otwarcie ofert (proces 4)

◊ OPIS PRZEZNACZENIA: proces wspomagający przebieg posiedzenia komisji przetargowej, związanego z otwarciem ofert.

◊ OBIEKTY ZEWNĘTRZNE:

- KOMISJA PRZETARGOWA – kontrolująca wniesienie wadium przez oferenta, ogłaszająca wartość oferty.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Zgłoszenia do przetargu – informacje o złożonych ofertach i wniesieniu wadium przez firmy
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje od komisji przetargowej sterujące działaniem procesu.

→ Dane wyjściowe:

- Dane przetargu – informacje zawarte w ofertach, modyfikujące dokumentację dotyczące przetargu .
- Dane ofert – informacje zawarte w ofertach, niezbędne do ich oceny.
- Oferty – zawartość informacyjna ofert.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – w składnicy danych zapisywane są informacje będące wynikiem otwarcia ofert przez komisję przetargową.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.5 Rozstrzygnięcie przetargu (proces 5)

◊ OPIS PRZEZNACZENIA: proces wspomaga komisję przetargową w wyborze oferenta poprzez:

- ustalenie zgodności złożonych ofert z SIWZ,
- ocenę ofert zgodnie z funkcją wyboru oferenta.

Dodatkowo proces ma za zadanie wspomóc ogłoszenie wyników przetargu i rozpatrzenie ewentualnych protestów i odwołań.

◊ OBIEKTY ZEWNĘTRZNE:

- OFERENCI – firmy biorące udział w procedurze przetargowej.
- KOMISJA PRZETARGOWA – sterująca rozstrzygnięciem przetargu.
- PREZES UZP – powiadamiany o rozstrzygnięciu przetargu.
- USTAWA O ZAMÓWIENIACH PUBLICZNYCH – dostarcza podstaw prawnych, które definiują procedurę rozstrzygania przetargu i procedury odwoławcze.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Oferty – zawartość informacyjna przyjętych ofert.
- Dane ofert – informacje zawarte w ofertach, niezbędne do ich oceny.
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane o oferentach – informacje o firmach, które złożyły oferty.
- Dane od oferenta – ewentualne protesty oferentów.
- Podstawy prawne – ustalenia definiujące procedurę rozstrzygania przetargu i procedury odwoławcze.
- Dane sterujące – informacje sterujące realizacją procedury rozstrzygania przetargu i procedur odwoławczych.

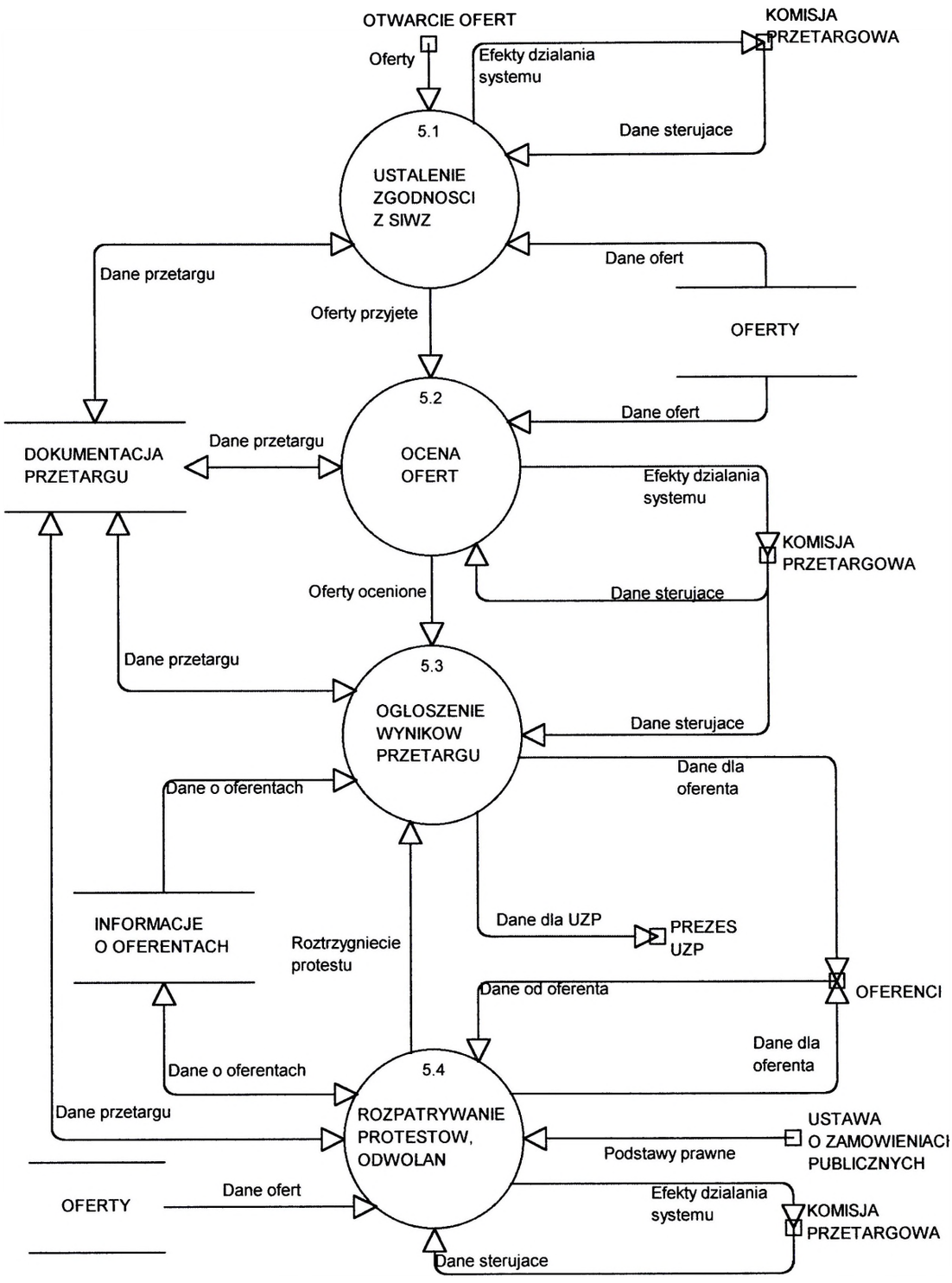
→ Dane wyjściowe:

- Dane przetargu – informacje modyfikujące dokumentację przetargu, będące wynikiem oceny ofert, wyboru oferenta oraz ewentualnych odwołań oferentów.
- Dane dla oferenta – efekty prac komisji przetargowej udostępniane oferentom, zawiadomienia o złożeniu protestów, odpowiedzi na protesty.
- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji zadań procesu
- Dane dla UZP – informacja o rozstrzygnięciu przetargu.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – w składnicy zapisywane są dane będące wynikiem oceny ofert, wyboru oferty, rozpatrywania protestów i odwołań.
- INFORMACJE O OFERENTACH – w składnicy gromadzone są informacje, umożliwiające kontakt z oferentami oraz obrazujące efekty tych kontaktów.

◊ ELEMENTY SKŁADOWE PROCESU: rysunek 3-4 przedstawia diagram DFD dla procesu 5 rozstrzygnięcia przetargu. Wyróżniono na nim 4 podprocesy:



Rys. 3-4 Diagram DFD dla procesu rozstrzygnięcia przetargu

3.5.1 Ustalenie zgodności z SIWZ (proces 5.1)

◊ OPIS PRZEZNACZENIA: proces wspomaga prace komisji przetargowej przy ustaleniu zgodności złożonych ofert z zapisami SIWZ. Sprawdzane są:

- sposób przygotowania ofert,
- dostarczenie wymaganych dokumentów,
- spełnienie istotnych dla zamawiającego postanowień,
- opis odpowiednich części zamówienia, jeżeli dopuszczalne jest składanie ofert częściowych,
- sposób obliczenia ceny oferty.

◊ OBIEKTY ZEWNĘTRZNE:

- KOMISJA PRZETARGOWA – ustalająca zgodność oferty z SIWZ.
- OTWARCIE OFERT - proces wspomagający przebieg posiedzenia komisji przetargowej, związanego z otwarciem ofert.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Oferty – zawartość informacyjna przyjętych ofert.
- Dane ofert – informacje zawarte w ofertach, niezbędne do ich ustalenia ich zgodności z SIWZ.
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje sterujące realizacją procedury ustalenia zgodności ofert z SIWZ.

→ Dane wyjściowe:

- Dane przetargu – informacje modyfikujące dokumentację przetargu, będące wynikiem ustalenia zgodności ofert z SIWZ.
- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji zadań procesu
- Oferty przyjęte - zawartość informacyjna ofert zgodnych z SIWZ.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – w składnicy zapisywane są dane będące wynikiem ustalania zgodności ofert z SIWZ.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.5.2 Ocena ofert (proces 5.2)

◊ OPIS PRZEZNACZENIA: proces wspomaga ocenę i wybór najlepszej oferty. Ocena jest przeprowadzana według kryteriów oceny ofert zapisanych w SIWZ. Wybór najlepszej oferty dokonywany jest według funkcji wyboru oferenta zawartej w SIWZ.

◊ OBIEKTY ZEWNĘTRZNE:

- KOMISJA PRZETARGOWA – oceniająca oferty i dokonująca wyboru najlepszej oferty.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Oferty przyjęte - zawartość informacyjna ofert zgodnych z SIWZ.
- Dane ofert – informacje zawarte w ofertach, niezbędne do ich oceny.
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje sterujące realizacją procedury oceny ofert.

→ Dane wyjściowe:

- Dane przetargu – informacje modyfikujące dokumentację przetargu, będące wynikiem oceny ofert i wynikiem stosowania funkcji wyboru oferenta.
- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji zadań procesu

- Oferty ocenione – oferty wraz z ich punktową oceną.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – w składnicy zapisywane są dane będące wynikiem oceny ofert.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.5.3 Ogłoszenie wyników przetargu (proces 5.3)

◊ OPIS PRZEZNACZENIA: proces ma za zadanie wspomóc komisję przetargową w ogłoszeniu wyniku przetargu, niezależnie od tego, czy przetarg został rozstrzygnięty czy unieważniony. Dodatkowo, proces wspomaga zawiadamianie oferentów o złożonych protestach i ich rozstrzygnięciu.

◊ OBIEKTY ZEWNĘTRZNE:

- OFERENCI – firmy biorące udział w procedurze przetargowej.
- KOMISJA PRZETARGOWA – sterująca rozstrzygnięciem przetargu.
- PREZES UZP – powiadamiany o rozstrzygnięciu przetargu.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Oferty ocenione – oferty wraz z ich punktową oceną.
- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje sterujące realizacją procedury ogłoszenia wyników przetargu.
- Dane o oferentach – informacje o firmach, które złożyły oferty.
- Rozstrzygnięcie protestu – informacje o złożonych protestach i ich rozstrzygnięciu.

→ Dane wyjściowe:

- Dane przetargu – informacje modyfikujące dokumentację przetargu, będące wynikiem ogłoszenia wyników przetargu, informacji o protestach i ich rozstrzygnięciu.
- Dane dla oferenta – zawiadomienia do oferentów o rozstrzygnięciu przetargu, proteście lub jego rozstrzygnięciu.
- Dane dla UZP – informacja o rozstrzygnięciu przetargu.

◊ SKŁADNICE DANYCH:

- INFORMACJE O OFERENTACH – w składnicy zawarte są informacje, umożliwiające kontakt z oferentami oraz obrazujące efekty tych kontaktów.

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

3.5.4 Rozpatrywanie protestów i odwołań (proces 5.4)

◊ OPIS PRZEZNACZENIA: proces wspomaga realizację postępowania odwoławczego, jeśli oferent zaskarżył czynności zamawiającego przez wniesienie protestu.

◊ OBIEKTY ZEWNĘTRZNE:

- OFERENCI – firmy wnoszące protest.
- KOMISJA PRZETARGOWA – sterująca rozpatrzeniem protestu.
- USTAWA O ZAMÓWIENIACH PUBLICZNYCH – dostarcza podstaw prawnych, które definiują procedury odwoławcze.

◊ PRZEPŁYWY DANYCH:

← Dane wejściowe:

- Dane przetargu – dotychczasowe ustalenia dotyczące przetargu.
- Dane sterujące – informacje sterujące realizacją procedur odwoławczych.
- Dane o oferentach – informacje o firmach, które złożyły protesty.

- Dane ofert – informacje zawarte w ofertach, niezbędne do ich oceny.
- Dane od oferenta – protesty oferentów.

→ Dane wyjściowe:

- Rozstrzygnięcie protestu – informacje o złożonych protestach i ich rozstrzygnięciu.
- Dane przetargu – informacje modyfikujące dokumentację przetargu, będące wynikiem realizacji procedur odwoławczych.
- Dane dla oferenta – zawiadomienia o złożeniu protestów, odpowiedzi na protesty.
- Efekty działania systemu – informacje dla komisji przetargowej, będące efektem realizacji zadań procesu.

◊ SKŁADNICE DANYCH:

- DOKUMENTACJA PRZETARGU – w składnicy zapisywane są dane będące wynikiem rozpatrywania protestów i odwołań.
- INFORMACJE O OFERENTACH – w składnicy gromadzone są informacje obrazujące efekty rozpatrywania protestów i odwołań..

◊ ELEMENTY SKŁADOWE PROCESU: proces nie podlega dalszej dekompozycji na tym etapie analizy.

BIBLIOGRAFIA

1. Flasiński M. - Wstęp do analitycznych metod projektowania systemów informatycznych. Inżynieria oprogramowania. WNT. Warszawa 1997 r.
2. Fuglewicz P. Stąpor K. Trojnar A. - CASE dla ludzi. Inżynieria oprogramowania. Wyd. LUPUS. Warszawa 1995 r.
3. Gibson M.L.: - The CASE Philosophy. *BYTE* 1989 nr 4, s. 209-218.
4. Kozielski S.: - Projektowanie Struktury Logicznej Relacyjnych Baz Danych Metodą Syntezy przy uwzględnieniu zasad wyszukiwania danych. *Zeszyty Nnukowe Politechniki Śląskiej,* 1993, Z. 11.
5. Martin J., McClure C.: - Structured Techniques: The Basis for CASE. Englewood Cliffs, Prentice-Hall 1989.
6. Ochman J. - Integracja w systemach informatycznych zarządzania. Podstawy teorii i metodologii.. Wyd. PWE. Warszawa 1992 r.
7. Pressman R.: - Software Engineering & CASE: Puradigms, Methods and Tools. Materiały z wykładu, Warszawa 1991.
8. Przewodnik Do Tworzenia Specyfikacji Wymagań Na Oprogramowanie w CSBI SA. Dokument wewnętrzny, Katowice 1991.
9. Sage A. P. - Systems managment for Information Technology and Sowtware Enginneering. John Wiley & Sans. New York 1995.
10. Wrycza S.: - Współczesne metodyki tworzenia systemów informatycznych zarządzania. Gdańsk, ORG-SERVICE 1988.
11. Wrycza S. - Analiza i projektowanie systemów informatycznych zarządzania. Metodyki, techniki, narzędzia. Wydawnictwo Naukowe PWN. Warszawa 1999 r.
12. Wrycza S. - Aktualne trendy komputerowo wspomaganego tworzenia systemów informatycznych. PTI. Świnoujście 1990 r.
13. Wrycza S. - Pakiety CASE – wyzwania, możliwości, bariery. Warszawa 1991 r.
14. Yourdon E.: - Modern Structured Analysis. Englewood Cliffs, Prentice-Hall 1989.