

3.3.3. Projektowanie techniczne

Do prac nad projektem technicznym systemu informatycznego można przystąpić dopiero po zatwierdzeniu założeń techniczno-
-ekonomicznych oraz po uzyskaniu koniecznych środków finansowych i zapewnieniu dostawy niezbędnych środków technicznych. Zadaniem tej fazy projektowania jest dostarczenie szczegółowych danych technicznych, technologicznych i organizacyjnych o budowie i działaniu systemu informatycznego. Wynikiem zaś jest projekt techniczny systemu, który w swej warstwie informatycznej (procedury przetwarzane na komputerze) będzie uruchomiony i przetestowany.

Projektowanie techniczne jak to widać na rys.3.4 jest tą częścią całego przedsięwzięcia, w której praca przebiega równolegle i wykonywana jest przez duże zespoły ludzkie (zwłaszcza programistów).

Rozbija się ono na cztery nurty, które zależą od siebie przedmiotowo i czasowo:

- uruchamianie i testowanie sprzętu informatycznego (elementów systemu cyfrowego),
- projektowanie, programowanie, uruchamianie i testowanie oprogramowania,
- projektowanie postaci wejść, wyjść oraz zbiorów danych,
- projektowanie procedur tradycyjnych (ręcznych) oraz szkolenie obsługi całego systemu.

Widać więc, że projektowanie techniczne wymaga harmonijnego współdziałania projektantów systemów, programistów, przedstawicieli użytkownika (projektantów, służb, zaopatrzenia i komórek szkolenia itd.). W związku z tym wymagane jest ustalenie zestawu punktów etapowych i harmonogramów oddawania poszczególnych fragmentów projektu. Ważne jest tutaj, aby te punkty etapowe i harmonogramy ustalone były możliwie wcześnie i aby były one dokładnie sprecyzowane.

Jak pokazuje rys.3.4 częściowe wdrożenie projektowanego systemu informatycznego następuje jeszcze w fazie projektowania technicznego. Polega to na wdrożeniu pewnego podsystemu pilotowego lub jednostki funkcjonalnej, w celu określenia

poprawności zarówno założeń systemu, jak i stopnia ich realizacji.

W tym skrypcie zostaną przedstawione tylko niektóre elementy projektowania technicznego. Pominięte zostaną tutaj zagadnienia szkolenia obsługi systemu i osób bezpośrednio i pośrednio z niego korzystających oraz sprawy związane z uruchamianiem i testowaniem systemu cyfrowego. Ponieważ zagadnienia dotyczące projektowania zbiorów omówione są w części II tego skryptu, dlatego też nie będą one prezentowane w tym miejscu. Przedstawione natomiast zostaną następujące elementy projektowe:

- projektowanie symboli identyfikujących,
- projektowanie dokumentów źródłowych i operacyjnych,
- projektowanie postaci informacji wyjściowych,
- projektowanie technologii procesu przetwarzania,
- wytyczne przy projektowaniu oprogramowania.

3.3.3.1. Projektowanie symboli identyfikujących

Projektowanie systemu symboli identyfikujących lub inaczej kodów rzadko bywa wykonywane od początku do końca, ponieważ zazwyczaj w instytucjach wprowadzających informatykę istnieją już i działają systemy kodów.

Potrzeba projektowania nowego systemu kodów powstaje wtedy, gdy:

- nowy system nakłada się na istniejącą organizację,
- zakres działalności organizacji powiększył się i system kodów jest już zbyt szczupły,
- następuje fuzja dwóch lub więcej organizacji z różnymi systemami kodów i należy opracować jeden kod wspólny.

Celem wprowadzania kodów jest zapewnienie skutecznej identyfikacji lub szybkiego wyszukiwania zakodowanych elementów. Innymi słowy kod jest zastępczą nazwą elementu, którego zwyczajowa nazwa byłaby nieefektywna dla przechowywania i przetwarzania w urządzeniach systemu cyfrowego. Równocześnie kod wskazuje związki z innymi elementami podobnego rodzaju oraz

pewne cechy charakterystyczne związane z danym elementem. Prawidłowo zaprojektowany kod musi spełniać pewne warunki, których listę przytoczono tu zgodnie z pozycją [8] i [10].

Oto one:

- musi być logicznie przystosowany do systemu,
- musi być ścisły i jednoznaczny,
- musi mieć staranny dobór znaków,
- musi mieć jasną i zrozumiałą strukturę,
- musi być oparty na znajomości rozkładu częstości występowania obiektów symbolizacji. Pozwala to na przyporządkowanie najprostszych kodów obiektom o największej częstości występowania,
- znajomość liczności obiektów symbolizacji jest niezbędna dla określenia pojemności informacyjnej kodu,
- znajomość dynamiki wzrostu liczności obiektów symbolizacji pozwala przewidzieć odpowiednią rezerwę pojemności informacyjnej kodu,
- musi być projektowany pod kątem użytkownika. Jeżeli użytkownikiem kodu jest w głównej mierze urządzenie techniczne, to ze zrozumiałych względów najlepszy jest kod binarny. Jeżeli natomiast z kodu tego korzystać będą przede wszystkim ludzie, to wskazane jest stosowanie kodów mnemotechnicznych lub żywych,
- musi uwzględniać możliwość pomyłki, przekłamania. Wymóg ten wyznacza potrzebę rozszerzenia symboli w celu wprowadzenia bitów, cyfr lub znaków kontrolnych, umożliwiających automatyczne wykrywanie i korygowanie błędów.

W informatyce stosowane są następujące rodzaje kodów: sekwencyjne, blokowe, grupowe, pozycyjne, pozycyjno-grupowe, mnemotechniczne oraz żywe.

Kody sekwencyjne stosowane są do identyfikacji zbiorów obiektów o niewielkiej liczności, luźno powiązanych.

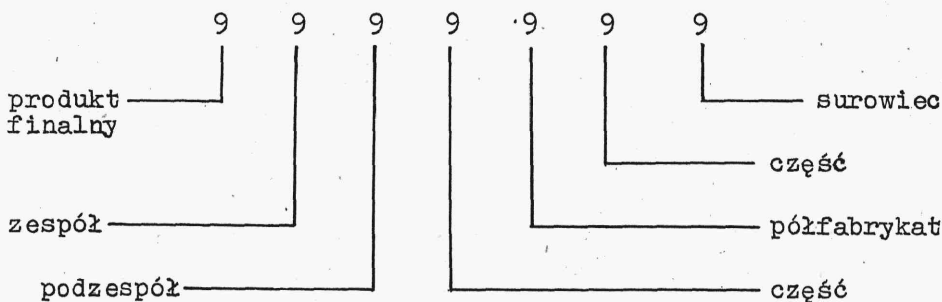
Kodowanie sekwencyjne polega na przydzielaniu kolejnych numerów poszczególnym obiektom zbioru. Na przykład:

- 001 - Adamski A.
- 002 - Arabski W.
- 003 - Barański Z. itd.

Ten typ kodowania jest sztywny, tzn. nie można włączyć nowych obiektów zachowując ustalony na początku porządek ich występowania w zbiorze (np. leksykograficznie). Usunięcie zaś egzystujących obiektów powoduje redundancję (występowanie kodów pustych).

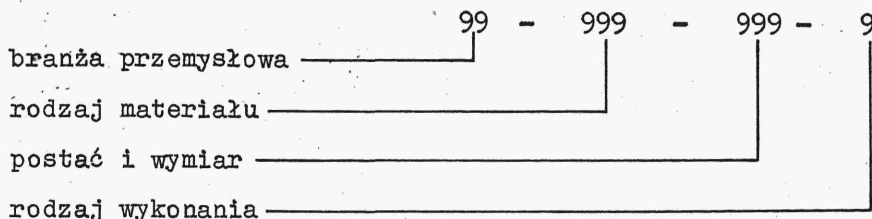
Kody blokowe są modyfikacją kodów sekwencyjnych. Istota tych kodów polega na podzieleniu całego zbioru obiektów informacyjnych na grupy. Według określonej relacji i w ramach grup następuje kodowanie sekwencyjne.

Kody pozycyjne lub inaczej hierarchiczne mogą być stosowane tam, gdzie obiekty informacyjne można sklasyfikować hierarchicznie (np. surowce, materiały, części, podzespoły i zespoły w wyrobie, komórki organizacyjne w przedsiębiorstwie itd.). Warunkiem stosowalności kodów pozycyjnych jest nieprzekraczanie liczności klasyfikowanych podgrup obiektów powyżej podstawy systemu pozycyjnego, np. 10 w klasyfikacji dziesiętnej, czy 26 w klasyfikacji literowej. W tym kodzie każdemu znakowi odpowiada odrębny szczebel klasyfikacji, np.



Najbardziej znanym przykładem kodu pozycyjnego jest Uniwersalny Kod Dziesiętny zaproponowany przez Deweya do klasyfikowania zbiorów bibliotecznych.

Kody pozycyjno-grupowe są uogólnieniem kodów pozycyjnych. Dopuszczają one mianowicie przypadki, w których liczność podgrup przekracza podstawę systemu. Jeżeli na jakiejś pozycji w kodzie liczba klasyfikowanych obiektów przekracza podstawę systemu, to dla tych pozycji rezerwuje się grupę znaków tak liczną, aby grupa ta umożliwiała sklasyfikowanie wszystkich obiektów w podgrupie, np.



Kody mnemotechniczne to takie kody, w których poszczególne znaki reprezentują pewne cechy kodowanych obiektów. Kody te tworzy się w taki sposób, aby wywoływały w użytkowniku określone skojarzenia z obiektem kodowanym np.

TL 670 15 C - Typ lampy, wymiary 670 x 15, czarna,

TL 710 15 B - Typ lampy, wymiary 710 x 15, biała,

ROW-W-BI-NOR

rower				normalny
wyścigowy				biały

Kodowanie żywe polega na tworzeniu symboli identyfikujących, wykorzystując oznaczenia handlowe, nazwy zwyczajowe, parametry techniczne zaczerpnięte z języka naturalnego. Sposób ten pozwala na bezpośrednie (bez słownika) zaczerpnięcie informacji o obiekcie informacyjnym w języku naturalnym. Oczywiście kodowanie za pomocą języka naturalnego powoduje znaczny rozrost objętości zbiorów obiektów informacyjnych.

Projektując system kodowania cyfrowego, trzeba podjąć decyzję, czy należy wprowadzić cyfrę kontrolną. Każdy bowiem kod może być uzupełniony cyfrą kontrolną, która umożliwia wykrywanie błędów powstałych w różnych miejscach obiegu informacji.

Do najczęściej spotykanych błędów zalicza się:

- błąd transkrypcyjny, czyli wprowadzenie na maszynowy nośnik informacji innej cyfry lub znaku niż ten, który był w dokumencie źródłowym, np. zamiast 123 456 wpisano 723 456,
- błąd transpozycyjny polega na przestawieniu cyfr lub znaków między sąsiednimi kolumnami, np. wpisanie 675 431 zamiast 657 431,

- błąd podwójnej transpozycji, np. wpisanie 31 854 zamiast 35 814,

- błędy będące kombinacją wyżej wymienionych.

Konstrukcja cyfry kontrolnej zależy od rodzaju błędu, jakimi chcemy wykryć z jej pomocą.

Założmy, że kod jakiegoś obiektu wynosi 83 597, a cyfrę kontrolną tworzy się przez sumowanie wszystkich cyfr kodu i wzięcie cyfry z pozycji jednostek, to otrzymamy pełny kod z cyfrą kontrolną w postaci 835 971. Jeżeli przy perforacji zmieniona zostanie jedna z cyfr tego kodu, np. 735 971, to test cyfry kontrolnej wykaże błąd, ponieważ $7+3+5+9+7 = 30$.

Tak skonstruowana cyfra kontrolna oczywiście nie wykryje podwójnego błędu transkrypcyjnego typu 745 971.

Tworzenie cyfr kontrolnych oparte na prostym sumowaniu cyfr kodu, choć jest metodą prostą, nie jest często stosowane ze względu na niewielki repertuar wykrywanych błędów. Najczęściej wykorzystywane są metody oparte na stosowaniu wagi i modułu. Waga jest mnożnikiem każdej cyfry kodu wyjściowego, w celu otrzymania iloczynu wag i cyfr kodu. Moduł jest dzielnikiem sumy iloczynów wagowych.

Rozpatrzmy następujący przykład.

Tworzymy cyfrę kontrolną dla pięciocyfrowego kodu, stosując wagi 6, 5, 4, 3, 2 oraz moduł 11. Wyjściowym kodem jest kombinacja cyfr 47 531. Cyfrę kontrolną tworzy się przez:

- pomnożenie każdej cyfry kodu przez odpowiadającą jej wagę,

4	7	5	3	1
x 6	x 5	x 4	x 3	x 2
= 24	= 35	= 20	= 9	= 2

- zsumowanie otrzymanych iloczynów,

$$24 + 35 + 20 + 9 + 2 = 90,$$

- podzielenie sumy przez moduł i zapamiętanie reszty

$$90 : 11 = 8 \text{ reszta } 2,$$

- odjęcie reszty od modułu. Otrzymany wynik jest cyfrą kontrolną kodu

$$11 - 2 = 9 .$$

Stąd cyfrą kontrolną dla tego przykładu jest 9, a pełny kod to 475 319.

Analogiczne sposoby mogą być stosowane do wyznaczania alfanumerycznych i alfabetycznych znaków kontrolnych, ponieważ znakom tym przyporządkowywane są określone wartości liczbowe w kodach cyfrowych (binarnym, oktalnym, dziesiętnym lub szesnastkowym).

3.3.3.2. Projektowanie postaci dokumentów źródłowych i operacyjnych

W procesie projektowania systemów informatycznych zachodzi nieraz konieczność dostosowania istniejących dokumentów do wymogów stawianych przez odpowiednie urządzenia do wprowadzania danych, jak również w celu ułatwienia przenoszenia informacji z tych dokumentów na nośniki maszynowe. Niekiedy trzeba zaprojektować zupełnie nowe dokumenty źródłowe. Oprócz dokumentów źródłowych projektant ma również do czynienia z dokumentami operacyjnymi. Dokumenty te służą do korekty lub eliminacji danych oraz wprowadzania wyników pośrednich do systemu. Wystawione są one przez odpowiednie urządzenie wyjściowe i puszczane w obieg, w trakcie którego następuje dopisanie odpowiednich danych wynikłych z procesu przetwarzania. Oba ww. rodzaje dokumentów istnieją i krążą w systemie w postaci odpowiednich formularzy, których postać, materiał, kolorystyka itd. nie są sprawą obojętną dla jakości projektu systemu.

Projektowanie formularzy uważa się niekiedy za czynność polegającą na logicznym podzieleniu kawałka czystego papieru na poszczególne pola wg informacji, które muszą znaleźć się na tym formularzu. Takie traktowanie tych spraw jest zbyt uproszczeniem problemu. Układ formularzy źródłowych i operacyjnych powinien umożliwiać bezinstrukcyjne przepisywanie da-

nych na maszynowe nośniki informacji (taśmę papierową, karty perforowane, taśmę magnetyczną itd.).

Ważne jest, aby rozmieszczenie danych na formularzu odpowiadało przyzwyczajeniom ich użytkowników, a z drugiej strony umożliwiało łatwe i bezbłędne przenoszenie danych.

Projektując formularze należy przestrzegać pewnych zasad, aby ww wymagania mogły być spełnione. Zasady te podawane są w literaturze w formie wytycznych lub pytań, na które trzeba odpowiedzieć, aby zdać sobie sprawę, czy tok postępowania przy projektowaniu był prawidłowy.

Osobnym zagadnieniem przy projektowaniu wejść do systemu jest stworzenie (zaprojektowanie) metod wykrywania, lokalizacji i ewentualnie korekcji nieuniknionych błędów, powstających przy wypełnianiu dokumentów źródłowych i przenoszeniu ich treści na nośniki maszynowe. O niektórych aspektach tej kontroli była już mowa przy tworzeniu cyfr i znaków kontrolnych.

Prawidłowe funkcjonowanie systemu informatycznego w dużej mierze zależy od poprawności danych wejściowych. Nawet "cudowny" system musi "upaść", jeżeli dane wprowadzane do niego nie będą poprawne. Jeżeli dane wejściowe są błędne, to błędne będą również dane wyjściowe.

Kontrola poprawności danych wejściowych powinna odbywać się w następujących okresach:

- w czasie przygotowywania danych,
- przy wprowadzaniu do maszyny cyfrowej,
- przed rozpoczęciem przetwarzania,
- w czasie działania programu.

W czasie przygotowywania danych do przeniesienia na maszynowe nośniki informacji powinny być dokonane następujące czynności kontrolne:

- sprawdzenie kompletności (czy wszystkie potrzebne dokumenty zostały dostarczone),
- sprawdzenie poprawności formy (właściwe formularze, wypełnione zgodnie z instrukcją itd.),
- sprawdzenie zgodności sum kontrolnych.

Podczas przenoszenia informacji z dokumentów źródłowych na maszynowe nośniki mogą powstać znaczne ilości błędów.

Czynność ta w sposób istotny obniża efekty automatyzacji przetwarzania danych związane z wprowadzeniem systemu informatycznego. W naszych polskich warunkach najczęściej stosowanym urządzeniem do przenoszenia danych są dziurkarki kart. Rzadziej używa się dziurkarek taśmy papierowej. Jak dotychczas nie są u nas stosowane urządzenia typu czytnik pisma odręcznego, czytnik znaków magnetycznych (zapisywanych atramentem magnetycznym) itd. Dopiero ostatnio zaczęto wprowadzać urządzenia do bezpośredniego wprowadzania danych na taśmę magnetyczną (Seecheck).

Sprawdzenie poprawności wyperforowanych kart odbywa się na sprawdzarkach w ten sposób, że treść dokumentu źródłowego zostaje ponownie "wypalcowana" przez operatora sprawdzarki, a następnie w sposób automatyczny porównana z odpowiednimi danymi z kart. Karty błędnie wyperforowane zostają oznaczone i sprawdzone powtórnie, gdyż mógł nastąpić błąd przy sprawdzaniu, i są one następnie poddane ponownej perforacji.

Zanim program użytkowy zajmie się przetwarzaniem wprowadzanych danych wejściowych musi odbyć się kontrola tych danych. Kontrolę tę sprawuje odpowiedni program komputerowy, którego celem jest wykrycie i wyeliminowanie następujących błędów:

- dane mają nieprawidłowe kody (niezgodne z cyfrą kontrolną),
- niekompletność danych lub błędne zapisy,
- dane wykraczają poza ustalone ograniczniki itp.

Sposoby eliminowania nieprawidłowych kodów były już uprzednio omawiane.

Kontrola kompletności danych odbywa się poprzez sprawdzanie utworzonej uprzednio sumy lub sum kontrolnych przy paczkowaniu dokumentów źródłowych.

Błędne dane eliminowane są poprzez sprawdzenie liczby i rodzaju użytych znaków. Trzeba tu pamiętać czy wprowadzona dana jest stałej czy zmiennej długości, np. data może zostać zapisana w postaci 04.04.78 (stała długość pola daty) i 4.4.78 przy zmiennej długości pola.

Określając pola o stałej długości wystarczy podać całkowitą liczbę znaków. Pola o zmiennej długości wymagają okre-

ślenia minimalnej i maksymalnej liczby znaków w polu. Ponadto odbywa się jeszcze sprawdzenie zgodności deklarowanej i faktycznej zawartości pola. Innymi słowy, sprawdza się czy dane pole składa się ze znaków deklarowanych typów: 999 - pole numeryczne, AAA - pole alfabetyczne, XXX - alfanumeryczne, FFF - cyfry stałoprzecinkowe, VVV - zmiennoprzecinkowe.

Przy wprowadzaniu danych, np. na taśmę magnetyczną, bezpośrednio przed przetwarzaniem odbywa się również sprawdzenie, czy dana zawiera się w określonych granicach zmienności.

Sprawdzanie poprawności danych może odbywać się:

- natychmiast po znalezieniu błędu,
- po sprawdzeniu wszystkich danych,
- po sprawdzeniu pewnej liczby danych.

Czasami sprawdzanie poprawności wprowadzanych danych odbywa się podczas przebiegu przetwarzania, czyli podczas wykonywania się programu użytkowego. W programie tworzy się pewne pułapki, które mogą wykryć (choć nie muszą) przeoczone przez poprzednie kontrole błędy.

Reasumując, projekt techniczny wejść systemu informatycznego winien zawierać:

- zasady kodowania i obliczania cyfr kontrolnych,
- zasady wypełniania i znakowania dokumentów źródłowych i operacyjnych,
- wzory prawidłowo wypełnionych dokumentów źródłowych i operacyjnych,
- zasady kontroli formalnej i merytorycznej danych,
- tryb przekazywania dokumentów do ośrodka obliczeniowego,
- wzory wypełniania maszynowych nośników informacji,
- instrukcję dziurkowania kart/taśmy papierowej,
- programy kontroli formalnej i merytorycznej danych przed przystąpieniem do przetwarzania

3.3.3.3. Projektowanie postaci informacji wyjściowych (tabulogramów)

Dane wyjściowe otrzymywane jako produkt funkcjonowania systemu informatycznego są głównym ogniwem łączącym użytkow-

nika z systemem. Prawidłowe, tzn. czytelne i zrozumiałe dla użytkownika, zaprojektowanie formy tych danych jest jednym z istotnych czynników warunkujących powodzenie całego przedsięwzięcia. Główną decyzją projektową jest tutaj prawidłowy wybór urządzenia technicznego do wyprowadzania wyników, np. w zastosowaniach handlowych tym urządzeniem jest najczęściej drukarka wierszowa. W systemach projektowania inżynierskiego lub do prac naukowych najbardziej odpowiednim urządzeniem wyjściowym jest monitor ekranowy sprzęgnięty z drukarką wierszową lub tzw. pisaki X-Y. Czasami wymagane jest, aby wyniki drukowane były na uprzednio przygotowanym formularzu (np. rachunki za energię elektryczną w systemach SAFO, PROGNOZA, ZBYT)

Informacje wyjściowe można podzielić na następujące grupy:

- wyniki podstawowe (tabulogramy wynikowe) stanowią podstawowy cel działania systemu.
- wyniki kontrolne (tabulogramy kontrolne) są niezbędne zarówno dla operatora systemu, jak również dla użytkowników. Operator przy ich pomocy może śledzić przebieg przetwarzania na komputerze, natomiast użytkownik ma możliwość korekcji błędnych danych i orientację czy przetwarzanie odbywa się prawidłowo,
- wyniki wewnętrzne systemu, takie jak statystyki operacyjne, informacje o przetwarzaniu, sprawozdania z wykorzystywanych zasobów systemu (jednostka centralna, kanały, zbiory taśmowe, dyskowe, pamięć operacyjna),
- wyniki do powtórnego przetwarzania.

Projektując postać tabulogramów należy określić:

- sposób identyfikacji,
- zawartość i postać,
- częstotliwość wydawania,
- spodziewaną objętość wydawnictwa.

Postać (format) typowego tabulogramu wynikowego przedstawiona jest w tablicy 3.5. Wszystkie uwagi odnoszące się do projektowania postaci dokumentów źródłowych mają swe zastosowanie również przy projektowaniu formularzy wynikowych.

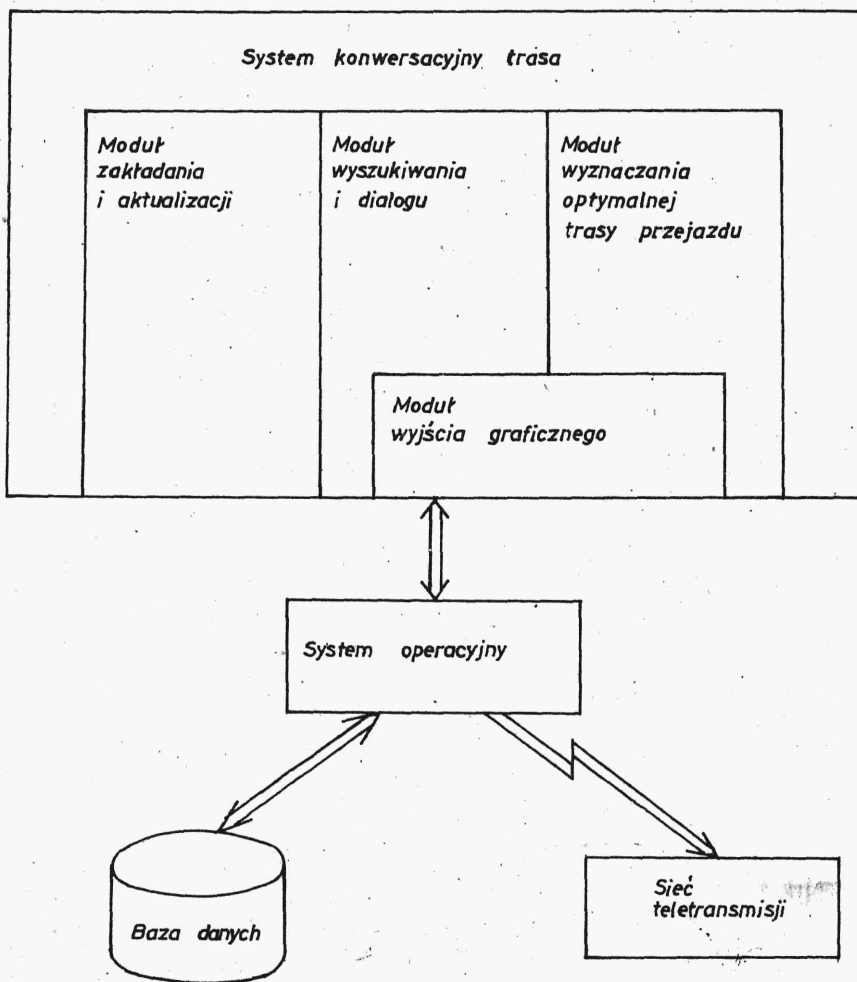
Projekt techniczny wyjść systemu informatycznego musi zawierać:

- wzór tabulogramu określający topologię rozmieszczenia informacji wynikowych,
- opis tabulogramu.

3.3.3.4. Projektowanie technologii procesu przetwarzania informacji

Technologia przetwarzania informacji w systemie informatycznym, to nic innego tylko modele funkcjonowania organiza-

LOGICZNA ORGANIZACJA SYSTEMU TRASA

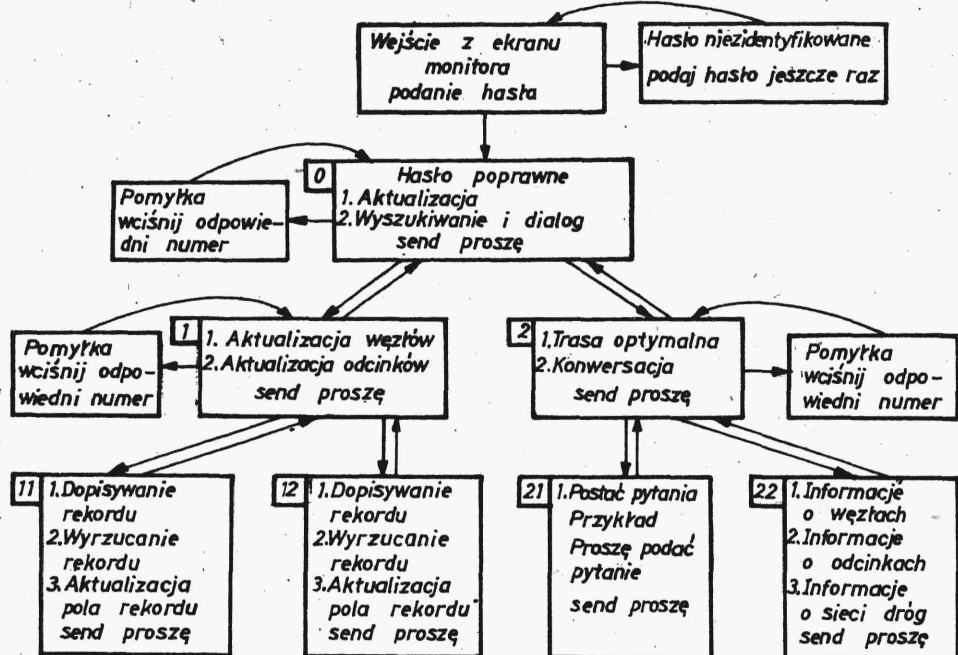


Rys.3.14. Szkic logicznej organizacji systemu

Tablica 3.5

[illegible]

SYSTEM KONWERSACYJNY TRASA



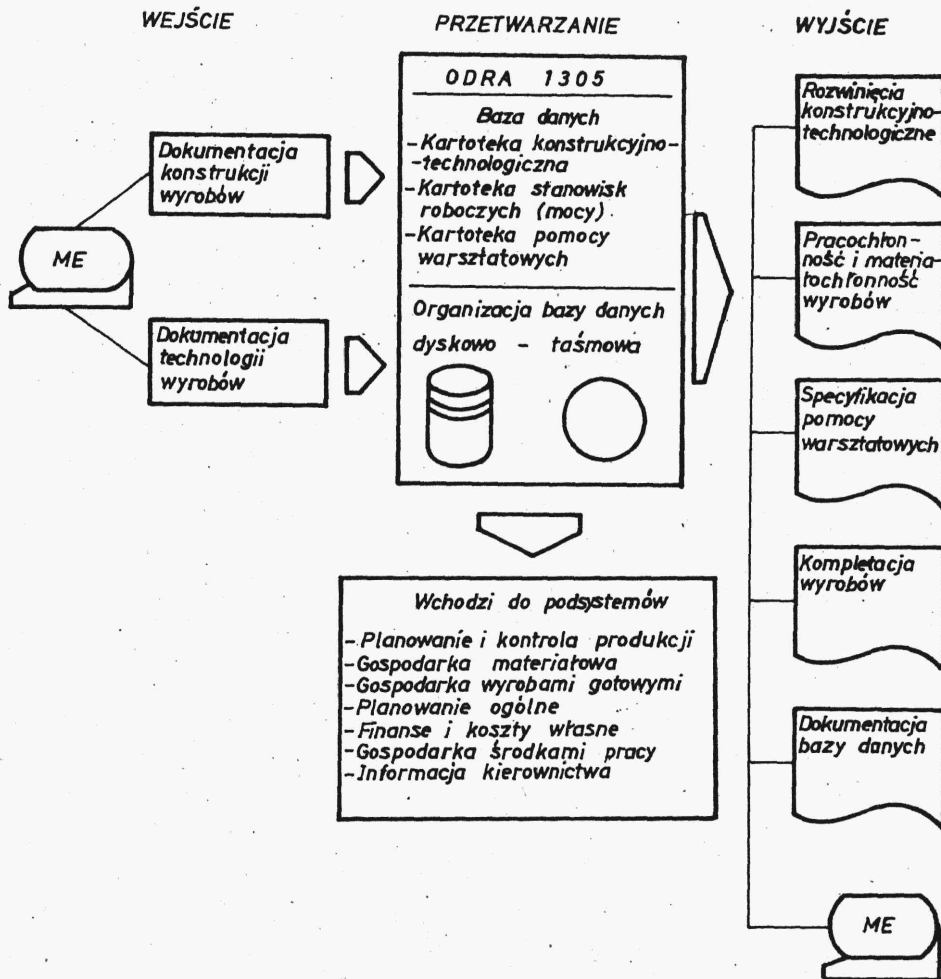
Rys.3.15. Schemat przebiegu systemu

cji jako całości, jej ogniw oraz procedur realizowanych w czasie działania organizacji. Modele te przybierają w projekcie technicznym postać modeli słownych, czyli opisów oraz modeli ikonograficznych. Modele ikonograficzne występują najczęściej jako schematy przepływowe (blokowe) lub inaczej zwane sieciami działań.

Postać tych schematów wynika z organizacji przedsiębiorstwa, poczynionych w projektowaniu wstępnym ustaleń oraz od przyjętego sposobu opisu.

W projekcie technicznym powinny być zamieszczone wszystkie dokumenty dotyczące przetwarzania informacji w systemie, począwszy od schematu ogólnego systemu jako całości, a skończywszy na schematach przebiegów maszyny cyfrowej. Jeżeli dana procedura wykonywana jest w sposób tradycyjny, to oczywiście winna ona mieć swoją reprezentację w zbiorze schematów przetwarzania.

PODSYSTEM TECHNICZNEGO PRZYGOTOWANIA PRODUKCJI

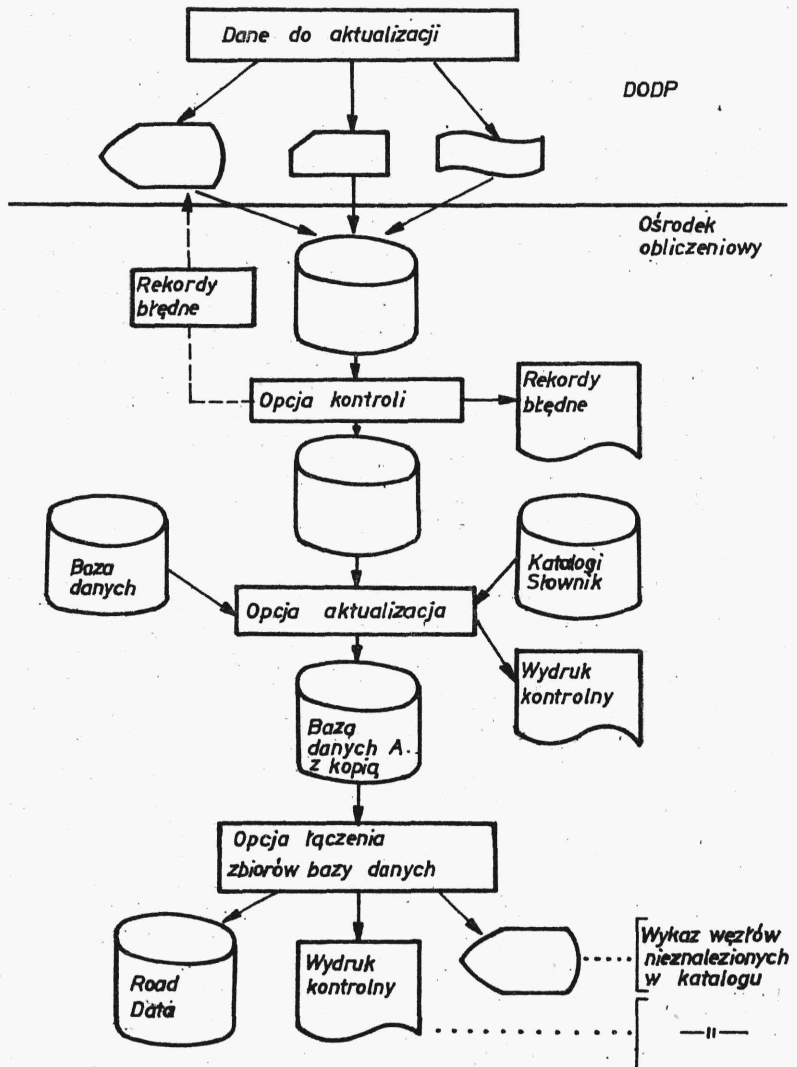


Rys.3.16. Schemat podsystemu

Na poniższych rysunkach pokazano sposoby przedstawienia technologii przetwarzania w porządku wzrastającej ich szczegółowości.

Chociaż zagadnienia dotyczące projektowania struktur zbiorów (logiczna i fizyczna) omawiane są w części II niniejszego skryptu, to warto w rozdziale dotyczącym metodologii projekto-

AKTUALIZACJA ZBIORÓW BAZY DANYCH



Rys.3.17. Schemat przetwarzania

wania systemów informatycznych wspomnieć, że projekt techniczny ww. zagadnień winien zawierać:

- pełny wykaz zbiorów z podaniem ich zawartości merytorycznej i oznakowania (zasady budowy nazw zbiorów),
- zasady gospodarki zbiorami (archiwowane, podstawowe, zmienne itd., numery generacji, daty przedawnienia),

- budowa wewnętrzna poszczególnych zbiorów, w tym: struktura logiczna zbiorów dyskowych, czyli określenie topologii zbiorów i algorytm "randomizacji" adresów w przypadku zbiorów nieśekkencyjnych.

3.3.3.5. Projektowanie oprogramowania

Oprogramowywanie systemów informatycznych, czyli kodowanie procedur przetwarzania z języka (ów) zrozumiałego dla człowieka na język (i) akceptowany przez maszynę cyfrową jest specyficznym etapem projektowym. Jego specyfika polega na tym, że wykonywany jest przez odrębną grupę zawodową, a mianowicie przez programistów. Dotychczasowe fazy i etapy wykonywane były przez analityka (ów) i projektanta (ów) systemu, co pokazuje rys.3.5. W skrajnym przypadku ich wpływ na jakość produktu wyjściowego z tego etapu - na programy użytkowe systemu informatycznego - jest niewielki, ograniczający się do powodowania błędów kodowania.

Programowanie dostarczonych przez projektanta systemów schematów przetwarzania odbywa się według następującej sekwencji czynności:

- sporządzenie schematów blokowych programu. Czynność ta jest czynnością opcjonalną. Niektórzy programiści uważają, iż programy ich są tak jasne i czytelne, że nie wymagają schematów blokowych. W niektórych ośrodkach zajmujących się projektowaniem systemów informatycznych istnieje administracyjny wymóg dokumentowania programów schematami blokowymi,
- kodowanie, czyli układanie listy rozkazów w języku programowania,

- uruchamianie i testowanie. Programy poddawane są najpierw kontroli formalnej dokonywanej przez translatory - poprzez kilkakrotne translacje - po całkowitym wyeliminowaniu błędów formalnych następuje usuwanie błędów natury merytorycznej, tzw. błędów wykonania. Sprawdzanie poprawności programów odbywa się za pomocą tzw. testów, czyli takich zestawów danych i specjalnych programów, które byłyby w stanie spo-

wodować "przejście" programu przez wszystkie pętle i rozgałęzienia decyzyjne. Sprawdzanie wyników testowania z wynikami założonymi przez projektanta (ów) systemu jest następną czynnością tego kroku. Testowanie programów i poprawianie błędów może zająć do 50% czasu przeznaczanego na oprogramowanie systemu informatycznego:

- sporządzenie instrukcji posługiwania się programami.

Instrukcja taka powinna zawierać:

- a) przeznaczenie i zasadnicze funkcje programu,
- b) schemat ideowy,
- c) wymaganą konfigurację sprzętu liczącego,
- d) język programu głównego z wyszczególnieniem użytych podprogramów,
- e) zbiory wejściowe i wyjściowe oraz karty parametryczne,
- f) warianty realizacji programu,
- g) testy,
- h) szacunek czasu realizacji.

Należy tutaj zaznaczyć, że przedstawiony wyżej tok postępowania przy oprogramowywaniu systemu przybiera często charakter procesu iteracyjnego.

Badania niezawodności oprogramowania różnych systemów informatycznych wykazały, że w programach wielkich - a za takie obecnie uważane są programy, przy opracowaniu których pracowało więcej niż jeden programista - najczęściej błędów powstaje na etapie projektowania programów. Błędy te są zwykle trudno wykrywalne poprzez testowanie. Większość z nich wykrywana jest i usuwana dopiero podczas normalnej pracy u użytkownika.

Zasygnalizowane tutaj spostrzeżenia prowadzą do istotnych zaleceń:

- należy w głównej mierze położyć nacisk na kształcenie specjalistów w projektowaniu oprogramowania systemów informatycznych,

- trzeba doskonalić stare i opracowywać nowe narzędzia, czyli metody i języki programowania.

- należy rozwijać metody weryfikacji jakości tworzonego oprogramowania.

Projektant oprogramowania ma teoretycznie możliwość świadomego kształtowania cech produktu wyjściowego w zakresie własności funkcjonalnych i użytkowych. Jednak w chwili obecnej niewiele jest na świecie takich zespołów projektantów - programistów, którzy potrafiliby powiedzieć przed zakończeniem prac, jakie będą na pewno walory funkcjonalne i użytkowe twórnego oprogramowania.

Przeważnie stosowana jest metoda: "najpierw napiszemy programy, a później przekonamy się co z tego wyszło".

Przyczyną tego stanu rzeczy jest zbyt wielkie wyprzedzenie praktyki w stosunku do teorii programowania. Innymi słowy brak jest umiejętności jakościowej - nie mówiąc już o ilościowej - oceny stosowanych metod programowania, języków, technik, tricków itd.

Niżej wymieniono własności systemu programowego (software'u), na które ma wpływ projektant oprogramowania:

- niezawodność działania,
- możliwość rozbudowy,
- możliwość przenoszenia na inny sprzęt,
- czasy działania,
- wykorzystanie dostępnych resursów (zasobów) systemu,
- zabezpieczenie przed niepowołanym dostępem,
- koszty,

Stosunkowo niewielki postęp uzyskiwany na polu zapewniania oprogramowaniu odpowiednich własności użytkowych związany jest z faktem niesłychanie złożonych zależności słabo jeszcze rozeznaczanych między poszczególnymi decyzjami, a uzyskaną własnością oprogramowania.

Nieco lepiej kształtuje się problem nadawania oprogramowaniu odpowiednich własności funkcjonalnych. Stosuje się różne metody ułatwiające projektowanie logiki funkcjonowania algorytmów odzwierciedlających procesy przetwarzania informacji, zachodzące w systemach informatycznych. Można tu wymienić kilka takich metod:

- programowanie modularne,
- programowanie strukturalne,
- metoda góra - dół (top - down),

- specjalne języki i pakiety programowe (ISDOS - Information Systems Design and Optimisation System, PSL - Problem Statement Language, PSA - Problem Statement Analyser, SODA - System Optimisation and Design Algorithm).

Metody te pozwalają na lepszą organizację prac, lepszą komunikację między członkami zespołu, ujednolicenie języka, a nawet na automatyczne generowanie koncepcji systemów informatycznych. Te ostatnie są dopiero na etapie badań, ale wskazują one kierunek prac nad omawianymi zagadnieniami.

Programowanie modularne jest tym sposobem polepszania funkcjonalnych własności programów, który zdobył sobie największą popularność.

Jak należy pisać program modularny? Dotychczas brak jest opracowań, które wskazywałyby na teoretyczne aspekty programowania modularnego. Dlatego też warto tu podać [27] kilka wytycznych wynikających z praktyki, mających pewne cechy metodologiczne.

1. Problem należy rozłożyć na część sterującą i podprogramy. Operacje we/wy powinny być pisane w postaci osobnych podprogramów.

2. Program winien być parametryzowany. Każda zmienna w programie, która używana jest więcej niż dwa razy, powinna być zdefiniowana jako parametr symboliczny.

3. Wszędzie gdzie to konieczne, używać należy tablic decyzyjnych. Jeżeli problem (procedura przetwarzania) jest złożony i posiada wielowarunkowe punkty decyzyjne, to zastosowanie tablic decyzyjnych zdecydowanie poprawia czytelność programu, a odrębne drogi wyjścia z takiego punktu można zapisać jako podprogramy.

4. Należy unikać pisania programów w ten sposób, aby same się modyfikowały.

5. Należy wystrzegać się używania obszarów wspólnych dla kilku podprogramów.

3.3.4. Wdrażanie systemów informatycznych do eksploatacji

Wdrożenie systemu informatycznego jest tą fazą całego procesu, która zasadniczo rozpoczyna się z chwilą zainicjowania