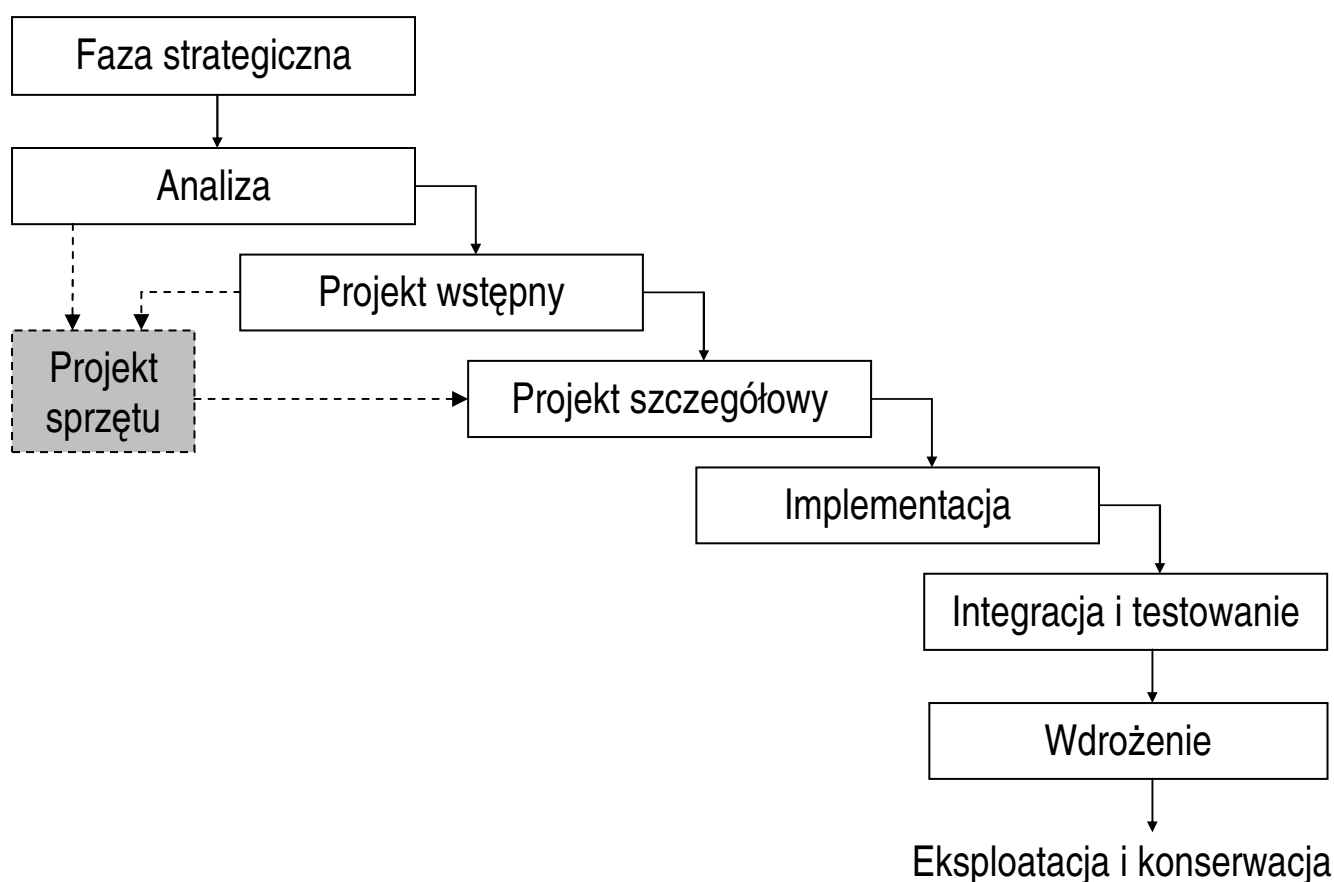


Metody strukturalne

- Dekompozycja funkcjonalna (funkcyjna)
- Odrębny opis działań (funkcji) i danych
- Zachowanie systemu jest wynikiem wykonania funkcji

Kaskadowy model procesu projektowego



Proces kaskadowy

1. Faza strategiczna

2. Faza analizy (modelowanie)

- poznanie dziedziny i organizacji przedsiębiorstwa
- analiza wymagań, modelowanie funkcji i danych

Wynik: model analityczny, plan testów akceptacyjnych

3. Faza projektu

- określenie architektury programu
- określenie algorytmów i struktur danych
- zaprojektowanie bazy danych

Wynik: projekt, plan testów

4. Faza implementacji

- implementacja programów
- opracowanie przypadków testowych

Wynik: kod i listing programów

5. Faza integracji i testowania

- łączenie i testowanie programów, usuwanie błędów
- opracowanie podręczników użytkownika
- instalacja i strojenie
- testowanie akceptacyjne

Wynik: kod, listing, dokumentacja użytkownika

6. Wdrożenie i eksploatacja

Diagram przepływu danych

Tworzy model przetwarzania danych w systemie (*data flow diagram* — *DFD*). Elementy modelu:

- **Proces**
element aktywny, przetwarzający dane.
- **Zbiór**
element pasywny, magazynujący dane.
- **Przepływ**
dane (wiadomości), przekazywane między procesami oraz między procesami i zbiorami.

Przepływy wyrażają relację producent-konsument

→ przepływ nie określa mechaniki realizacji (kto kogo wywołuje).

Model jest abstrakcyjny i może ilustrować:

- fizyczną organizację przetwarzania lub firmy,
- logiczne zależności między procesami przetwarzającymi.

Reprezentacja graficzna — diagram przepływu danych

- procesy (kółka),
- zbiory (domknięte lub niedomknięte prostokąty),
- przepływy (strzałki).

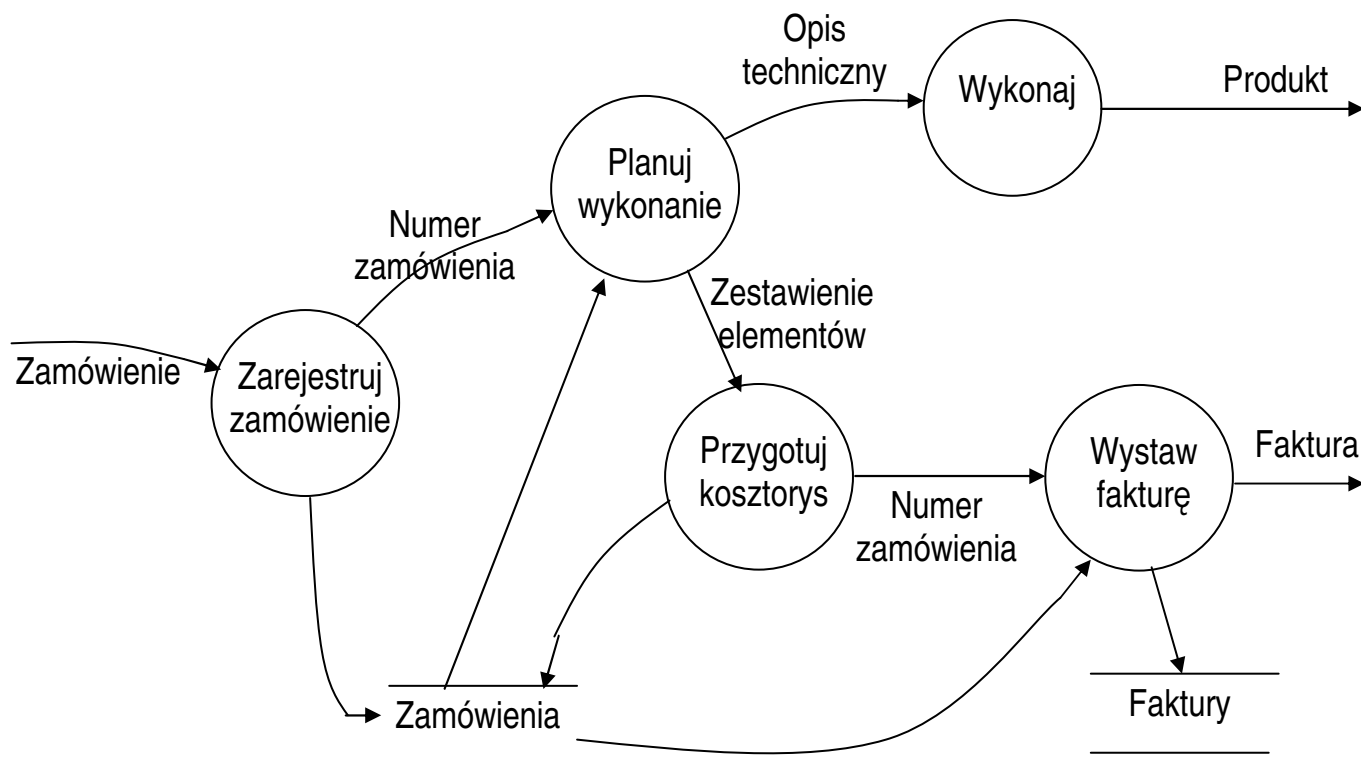


Diagram przepływu danych nie zawiera:

- informacji proceduralnej o sposobie przetwarzania,
- informacji o przepływie sterowania między procesami.

Zadaniem modelu jest:

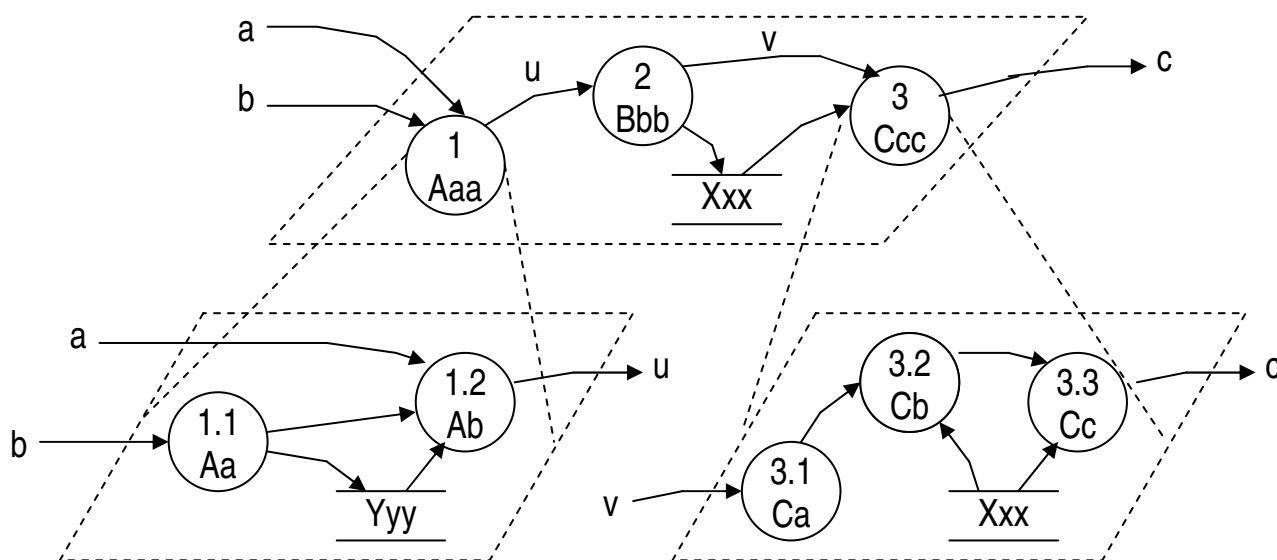
- dekompozycja całości na mniejsze elementy (procesy)
- pokazanie interfejsu tych elementów.

Specyfikacje procesów (minispecyfikacje):

- opis tekstowy ,
- zdania pseudokodu,
- tablice lub drzewa decyzyjne.

Reprezentacja hierarchiczna

- w rozwinięciu procesu mogą wystąpić zbiory,
- konieczna zgodność przepływów.



Implementacyjne konotacje diagramu danych:

- procesy → moduły programu,
 przepływy → przepływy danych między modułami programu,
 zbiory → struktury danych i zbiory komputerowe.

Słownik danych

Zbiór definicji elementów diagramu przepływu danych:

- procesów (nazwy i minispecyfikacje),
 - danych, przechowywanych w zbiorach,
 - przepływów,
 - elementów danych zawartych w przepływach i zbiorach (atrybuty).
- plus procedury związane z utrzymaniem spójności danych.

- Proste elementy danych:

Identyfikator = [PESEL | NIP]

Nr telefonu = * numer telefonu stacjonarnego wraz z kodem miasta i kraju *

- Elementy złożone:

Zamówienie = Nr zamówienia + Zamawiający + Przedmiot + Cena + Data

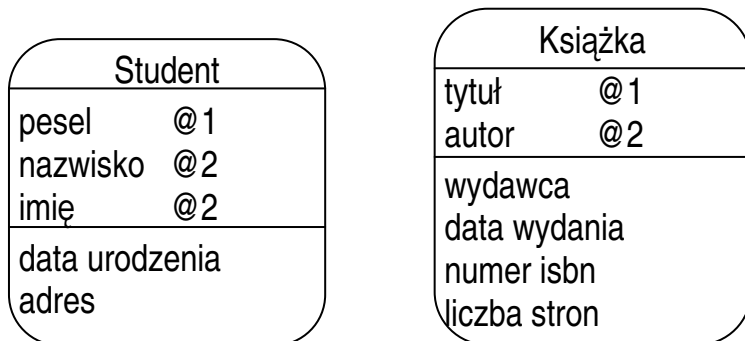
Student = nazwisko + imię + pesel + data urodzenia + adres + (nip)

Studenci = 1{ Student }3000

→ *Zbiór jest listą elementów tego samego typu (typ danych, encja)*

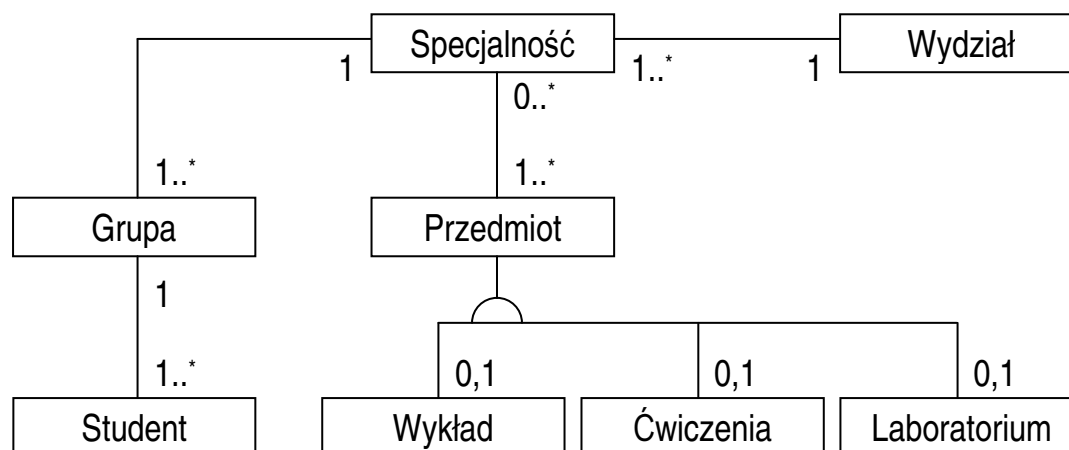
Reprezentacja graficzna — diagram danych (diagram encji)

- encje (prostokąty),



- relacje (romby lub same linie).

Przykład: ewidencja studentów



Wydział = nazwa wydziału + adres

Grupa studencka = numer

Student = nazwisko + imię + pesel + data urodzenia + adres + (nip)

Specjalność = nazwa + kierunek studiów + rodzaj

Przedmiot = nazwa + opis + sposób zaliczania

Wykład = liczba godzin + treść + literatura

.....

Implementacyjne konotacje ERD (*entity-relationship diagrams*):

encje → tabele,

atrybuty → kolumny tabeli,

relacje → powiązania przez klucze.

Diagram struktury

Tworzy model struktury programu (*structure chart*).

Elementy modelu:

- **Podprogram**
podstawowa jednostka programu (procedura, funkcja).
- **Przepływ sterowania**
wywołanie podprogramu przez inny podprogram.
- **Przepływ danych**
argumenty i wyniki wywołania.
- **Obszary danych**
dane lub zbiory wspólne dla kilku podprogramów.

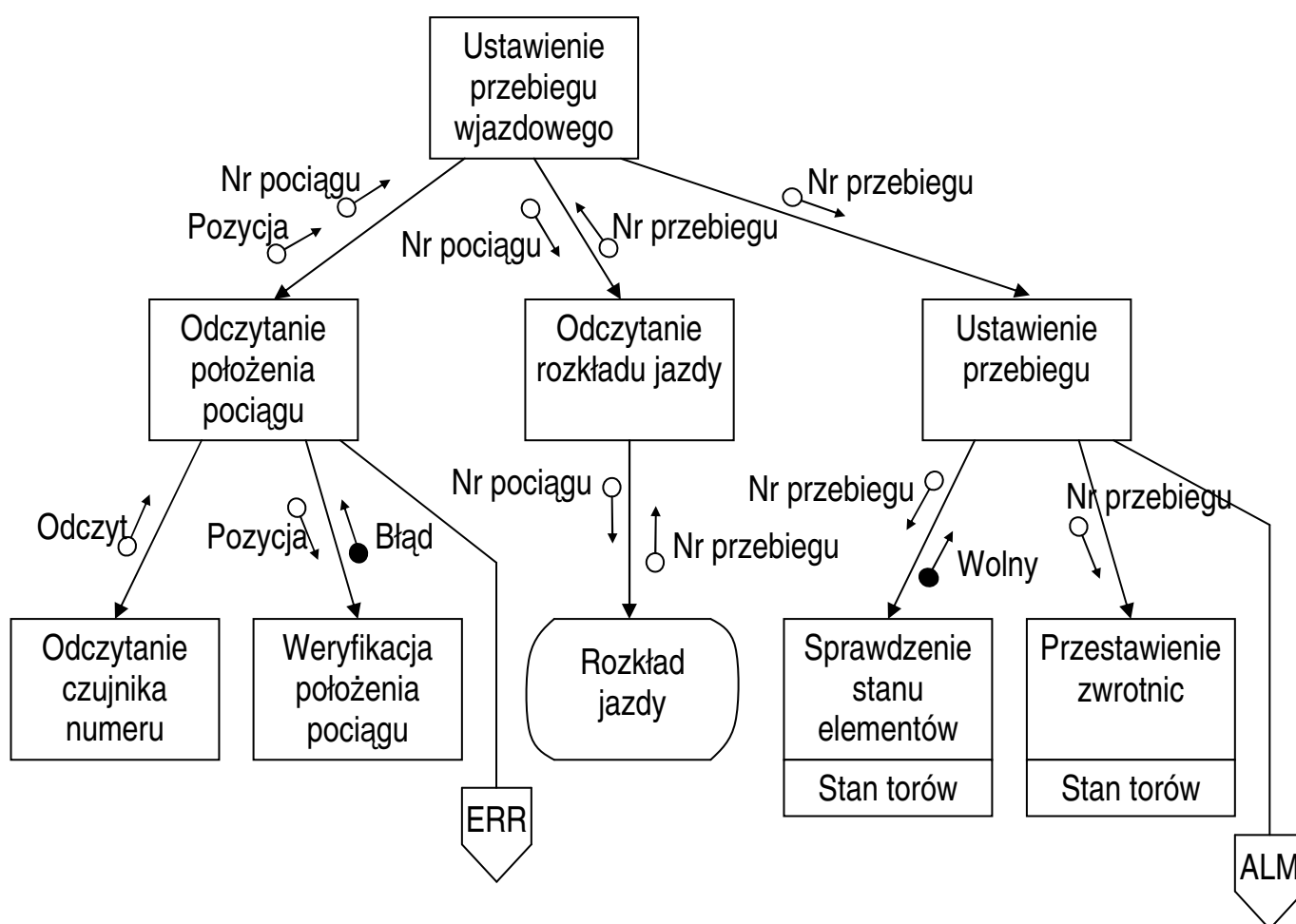
Celem diagramu jest pokazanie:

- dekompozycji programu na jednostki programowe (podprogramy),
- interfejsu każdej jednostki,
- przepływu sterowania między podprogramami.

Reprezentacja graficzna — diagram struktury

- podprogram (prostokąt),
- wywołanie (strzałka),
- argumenty (skierowane znaczniki),
- obszar danych (prostokąt zaokrąglony).

- Przykład: ustawienie przebiegu pociągu wjeżdżającego na stację:



Notacja diagramu bywa wzbogacana o emblematy:

- wywołania warunkowego (romb na początku strzałki),
- wywołania iteracyjnego (łuk wokół początku strzałki).

Diagram uzupełniają specyfikacje:

- algorytmów działania podprogramów
- przepływów i obszarów danych, np.:

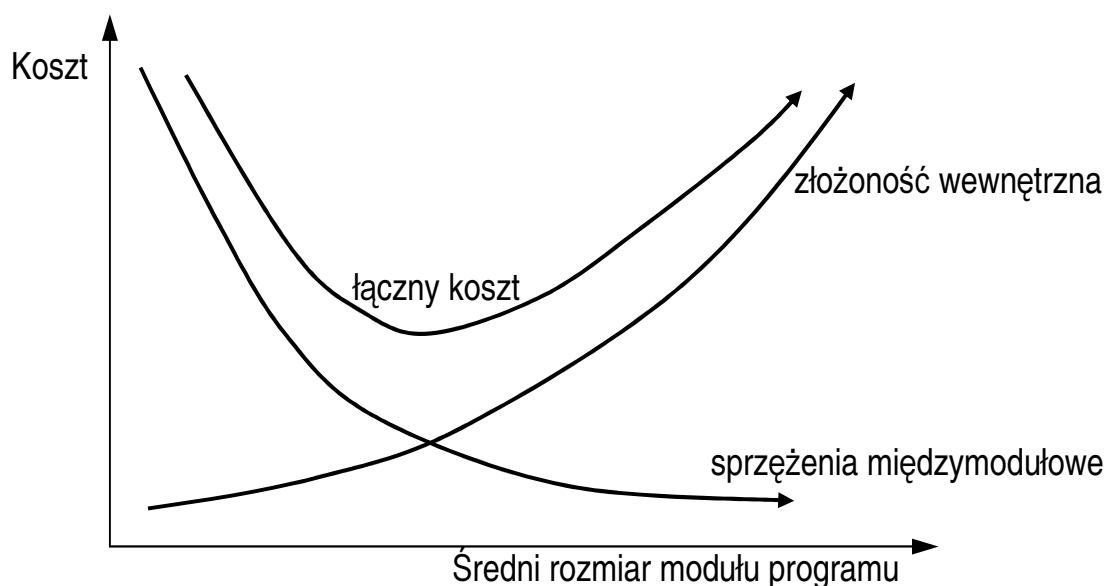
Pozycja = Nr pociągu + Nr toru

Odczyt = Nr pociągu + Nr czujnika + Czas +

Specyfikacje pochodzą z wcześniejszych modeli DFD i ERD.

Reguły określające dobrą strukturę programu:

- brak patologicznych połączeń — odwołań do prywatnych danych innego podprogramu (tylko interfejs proceduralny),
- wewnętrzna spójność podprogramów — brak argumentów sterujących (sugerują alternatywność działania jednostki),
- rozsądny rozmiar podprogramu (np. 1 strona), liczba wywoływanych podprogramów nie przekraczająca 7 ± 2 .

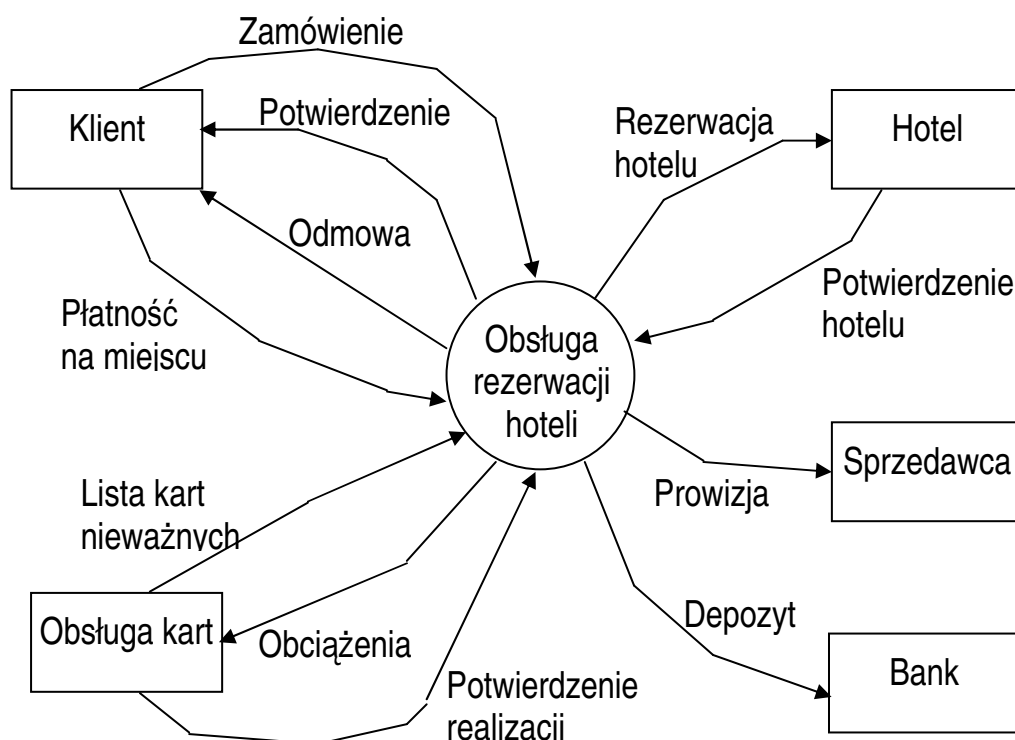


Analiza strukturalna

1. Specyfikacja wymagań: schemat kontekstu i listy funkcji.
2. Dokumentacja fizycznej struktury systemu (opcja).
3. Opracowanie logicznego modelu przetwarzania danych.
4. Budowa modelu danych.
5. Opracowanie interfejsu użytkownika i fizycznego modelu systemu.

Przykład: Rezerwacji miejsc hotelowych przez biuro obsługi kongresów i konferencji

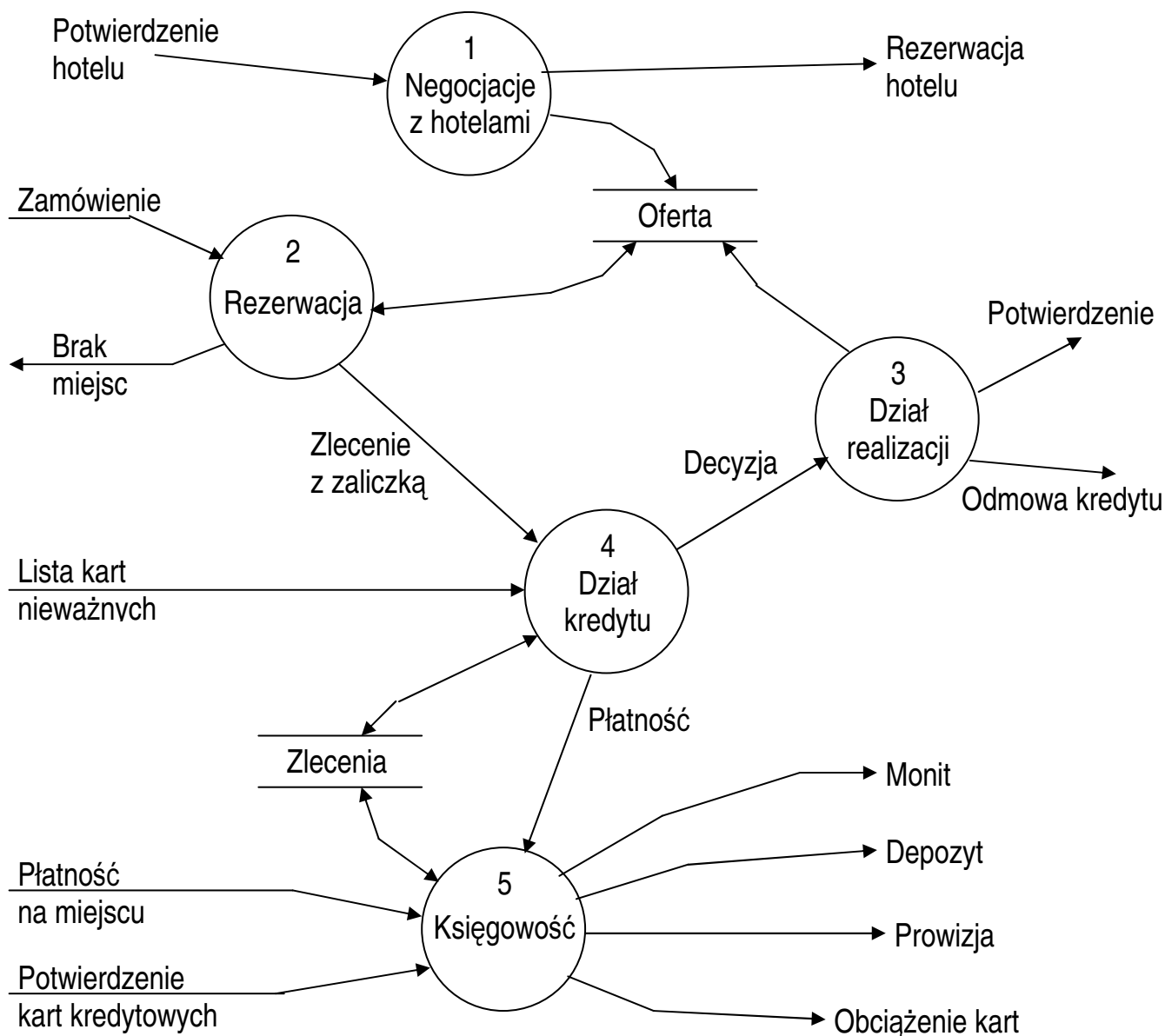
Schemat kontekstu



Zamówienie = Nazwisko klienta + Id konferencji + Liczba miejsc + Okres rezerwacji + Zakres ceny + (Zaliczka) + (Nazwa hotelu)

Schemat wyznacza granice systemu, ale bez fizycznego interfejsu.

Schemat całości



Zlecenie = Nr zlecenia + Nazwisko klienta + Okres rezerwacji + Nr pokoju + Cena + (Suma zaliczki + Typ) + Suma do zapłaty + Id sprzedawcy

Zaliczka = [Czek | Nr karty]

Brak miejsc = Nazwisko klienta + Okres rezerwacji + Zakres ceny + (Czek)

Decyzja = Zlecenie + [Akceptacja | Odmowa] + (Kopia paragonu)

Płatność = [Czek | Paragon]

Obciążenie kart = Formularz + { Paragon }

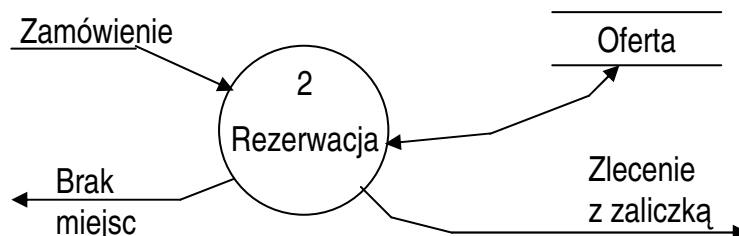
- Powstaje we współpracy z użytkownikiem
→ weryfikacja podczas przeglądu (*walkthrough*).
- Odwzorowuje dotychczasową strukturę organizacyjną firmy → bo z tym klient jest najlepiej obznajomiony.
- Konieczne zachowanie zgodności ze schematem kontekstu, tu:
 - kontekst: Odmowa ?
 - tutaj: Odmowa kredytu, Brak miejsc, Monit ?

Dekompozycja przepływu:

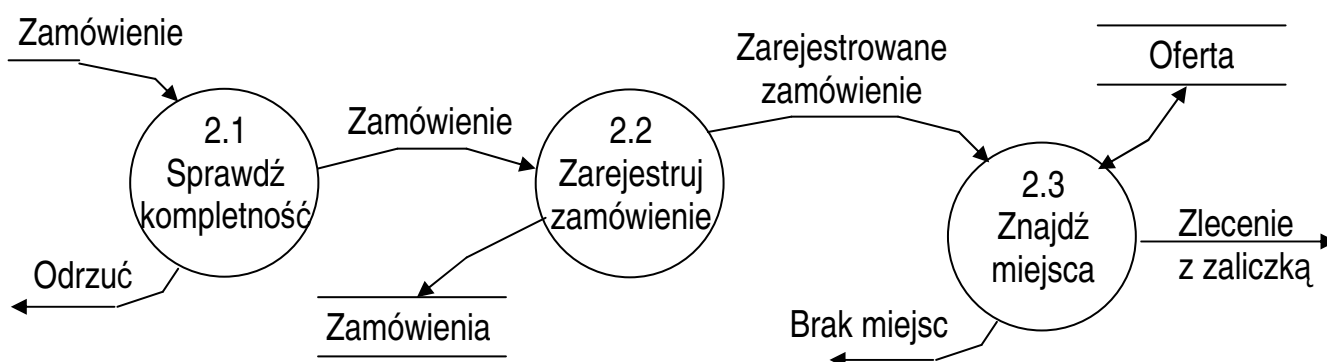
Odmowa = [Brak miejsc | Odmowa kredytu | Monit]

Hierarchiczna dekompozycja

Dział rezerwacji



Dekompozycja



- Zgodność ? → Odrzuć
- Słownik danych

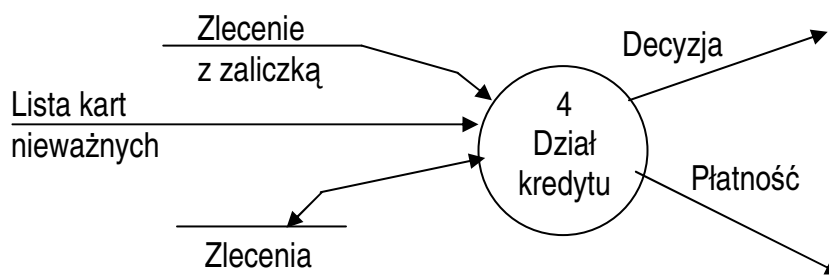
Zlecenie = Nr zlecenia + Nazwisko klienta + Okres rezerwacji + Nr pokoju + Cena + (Suma zaliczki + Typ) + Suma do zapłaty + Id sprzedawcy

Oferta = { Pokój }

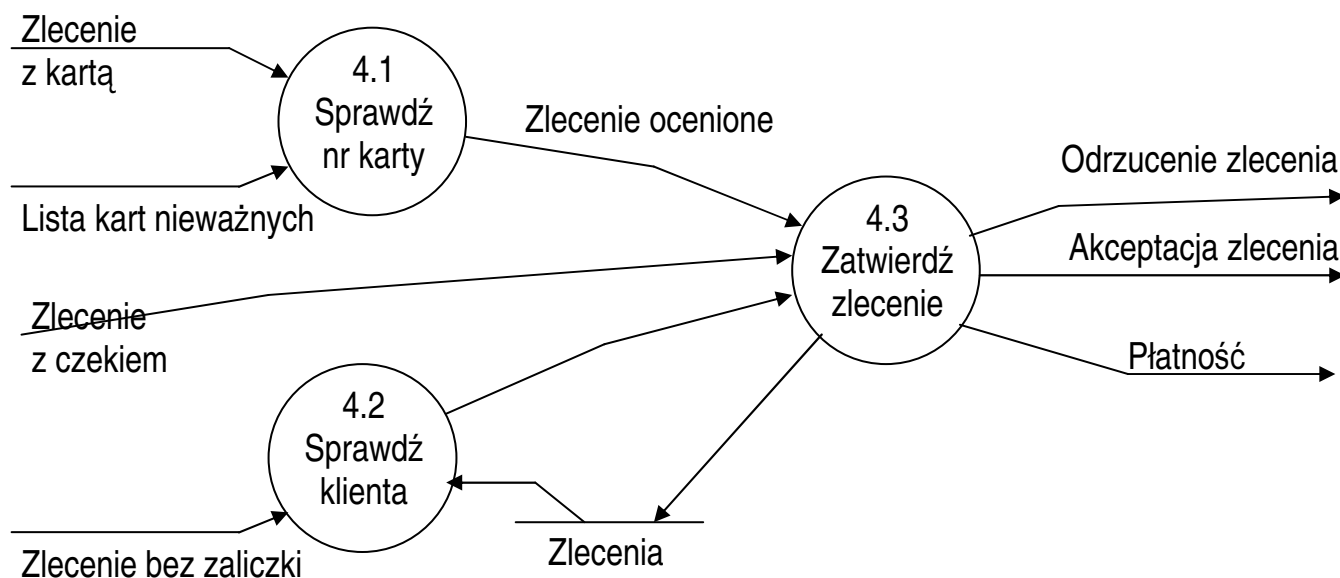
Pokój = Nazwa hotelu + Nr pokoju + Liczba miejsc + Cena + Okres ważności + ({ Okres rezerwacji + Status })

Status = [Przydzielony | Zarezerwowany]

Dział kredytu



Dekompozycja

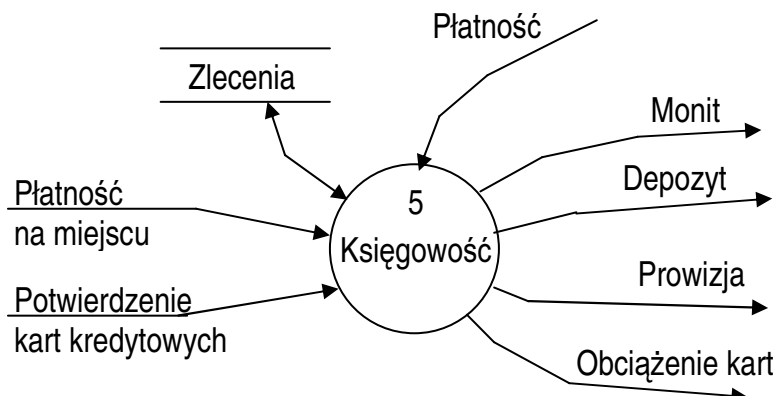


- Zgodność ?

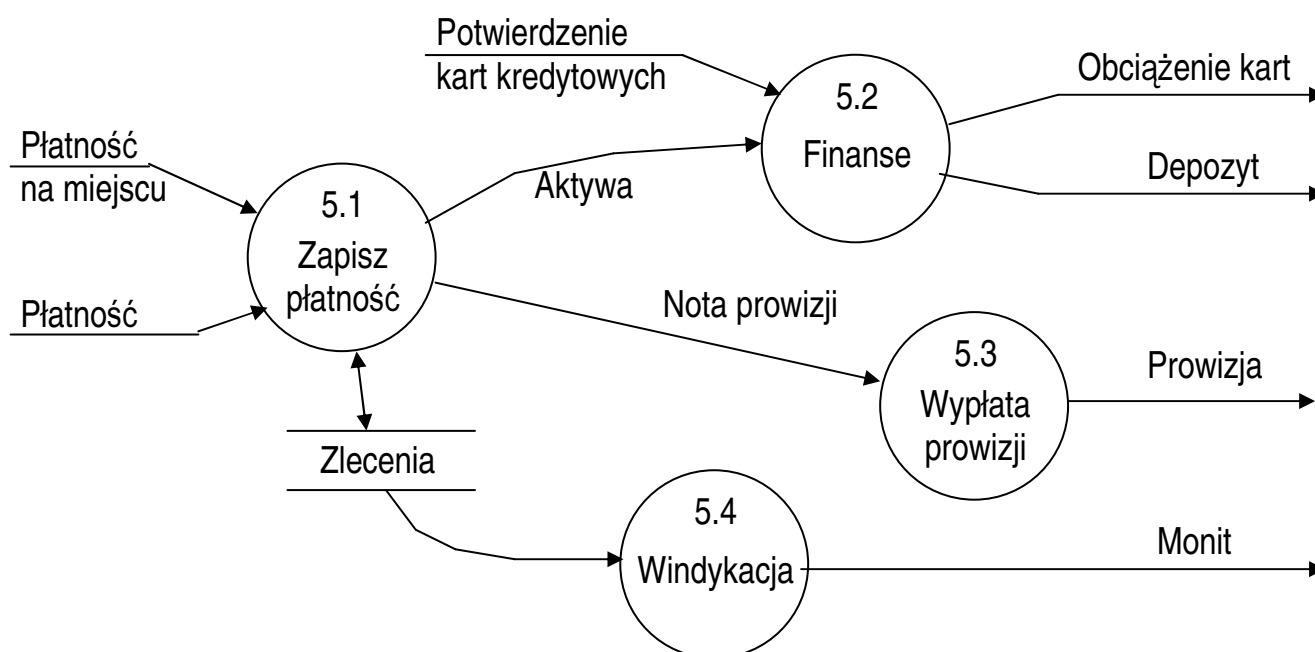
Zlecenie z zaliczką = [Zlecenie z kartą | Zlecenie z czekiem
| Zlecenie bez zaliczki]

Decyzja = [Akceptacja zlecenia | Odrzucenie zlecenia]

Dział księgowości



Dekompozycja



- Problem: przypisanie płatności do zlecenia

?

Płatność = [Czek | Paragon]

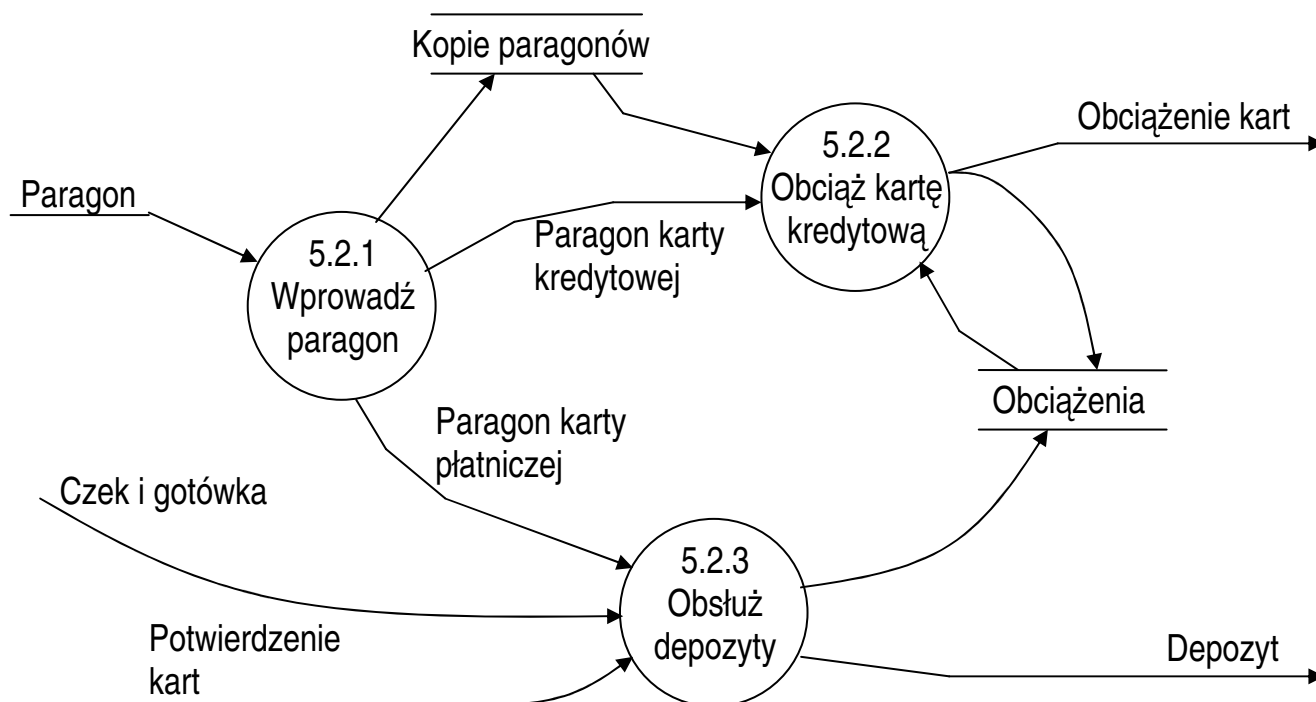
- Słownik danych

Płatność = [Czek | Paragon] + Nr zlecenia

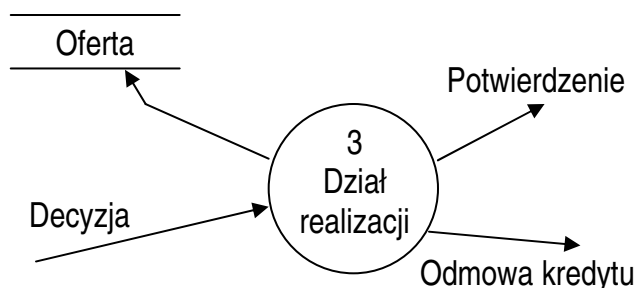
Nota prowizji = Id sprzedawcy + Nr zlecenia + Suma

Aktywa = [Czek | Paragon | Gotówka]

Proces Finanse



Dział realizacji



Minispecyfikacja

Proces 3: Dział realizacji.

Dla każdej otrzymanej decyzji wykonaj:

Znajdź Pokój w zbiorze Oferty hotelowej

Jeśli Akceptacja, to

Zmień status Pokoju z Przydzielony na

Zarezerwowany

Przygotuj pismo potwierdzenia

Jeśli Odmowa, to

Zmień status Pokoju z Przydzielony na Wolny

Przygotuj pismo odmowy kredytu

Wyjątek — Status Pokoju różny od Przydzielony

Wyjaśnij błąd w dziale sprzedaży

Słownik danych

Decyzja	= Zlecenie + [Akceptacja Odmowa] + (Kopia paragonu)
Potwierdzenie	= Nr zlecenia + Nazwisko klienta + Okres rezerwacji + Pokój + (Sposób zapłaty + Suma) + Suma do zapłaty + (Kopia paragonu)
Odmowa kredytu	= Nr zlecenia + Nazwisko klienta + Okres rezerwacji + Suma do zapłaty

Proces analizy

Opracowanie logicznego modelu przetwarzania danych

- zastąpienia procesów najwyższych poziomów ich rozwinięciami
→ płaski diagram obrazujący logikę przetwarzania danych,
- analiza diagramu i usprawnienie przebiegu przetwarzania.

Opracowanie diagramu danych:

- specyfikacja danych,
- uzupełnienie braków, np.:
 - zbiór danych adresowych hoteli i konferencji,
- likwidacja powtórzeń, np.:
 - dane klienta i okres rezerwacji w zbiorach Zamówienia i Zlecenia,
 - okres rezerwacji w zbiorach Oferta i Zlecenia.

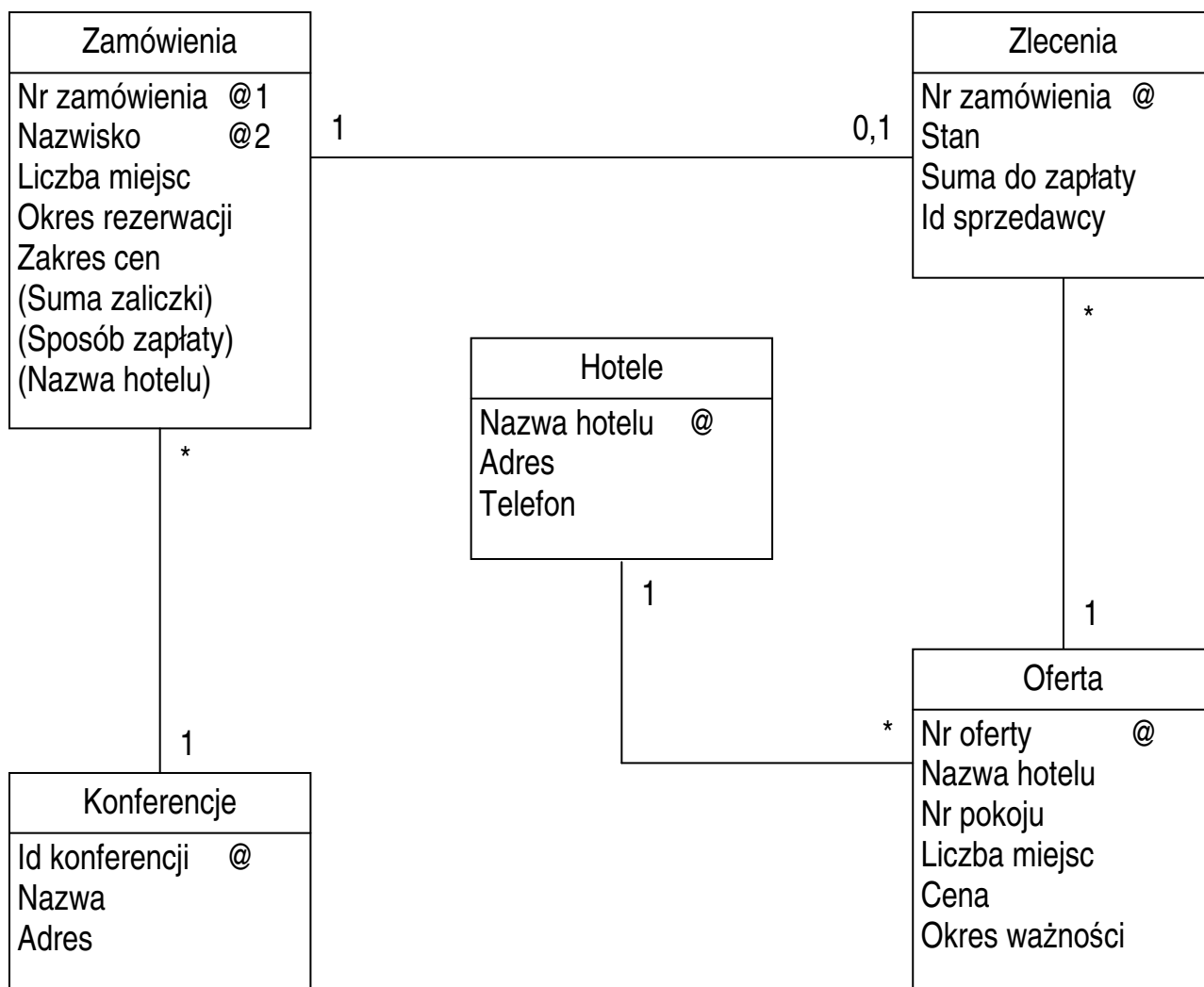
W rezultacie zmiany początkowych definicji danych:

Zamówienie = Nr zamówienia + Nazwisko klienta + Id konferencji
+ Liczba miejsc + Okres rezerwacji + Zakres ceny
+ (Suma zaliczki + Typ) + (Nazwa hotelu)

Zlecenie = Nr zamówienia + Pokój + Stan + Suma do zapłaty
+ Id sprzedawcy

Pokój = Nr oferty + Nazwa hotelu + Nr pokoju + Liczba miejsc + Cena
+ Okres ważności

- opracowanie diagramu encji
→ usunięte powtórzenia danych zastąpione asocjacjami



Opracowanie interfejsu użytkownika i fizycznego modelu systemu

- eliminacja dokumentów papierowych i czynności ręcznych
→ opracowanie interfejsu użytkownika,
- pogrupowanie transformacji i zbudowanie modelu hierarchicznego
→ modularność (*coupling, cohesion*).

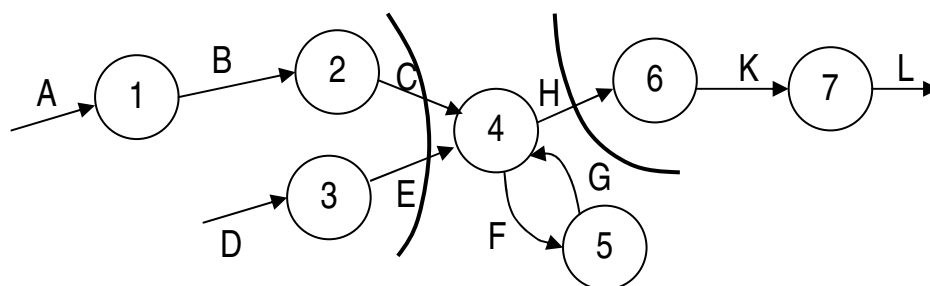
Projektowanie strukturalne

Przekształcenie hierarchicznego diagramu przepływu danych w schemat struktury programu:

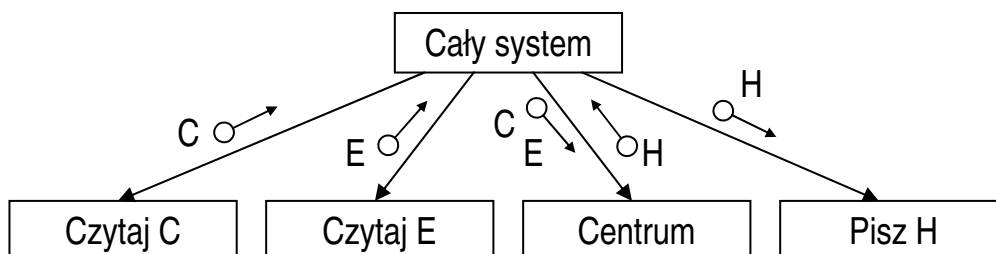
1. Określenie struktury programu.
2. Uzupełnienie i optymalizacja struktury.
3. Optymalizacja wydajności i pakietyzacja programu.

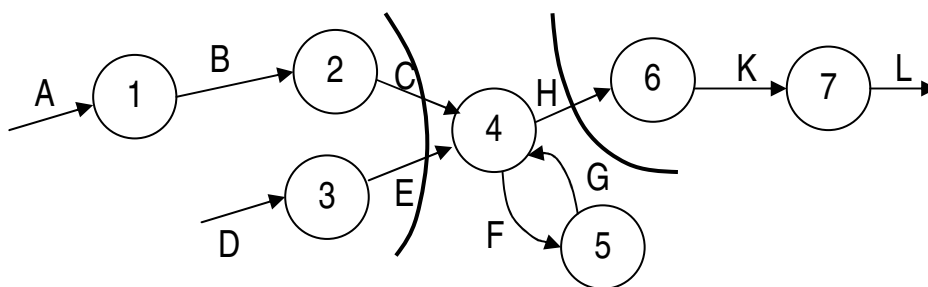
Struktura transformacyjna

Zawiera liniowe sekwencje przetwarzania wejść i wyjść, np.:



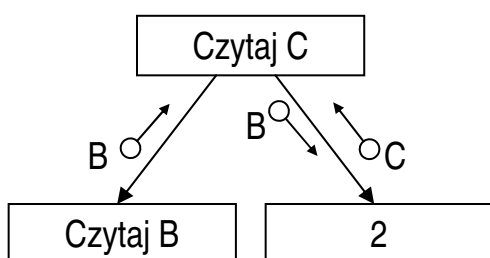
1. Określenie struktury programu głównego:



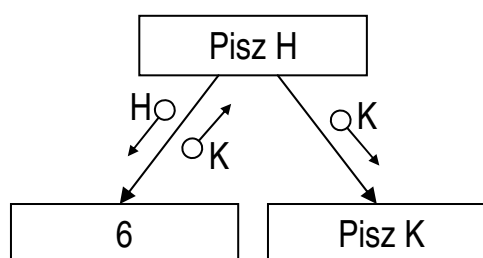


2. Przekształcanie kolejnych gałęzi wejściowych i wyjściowych:

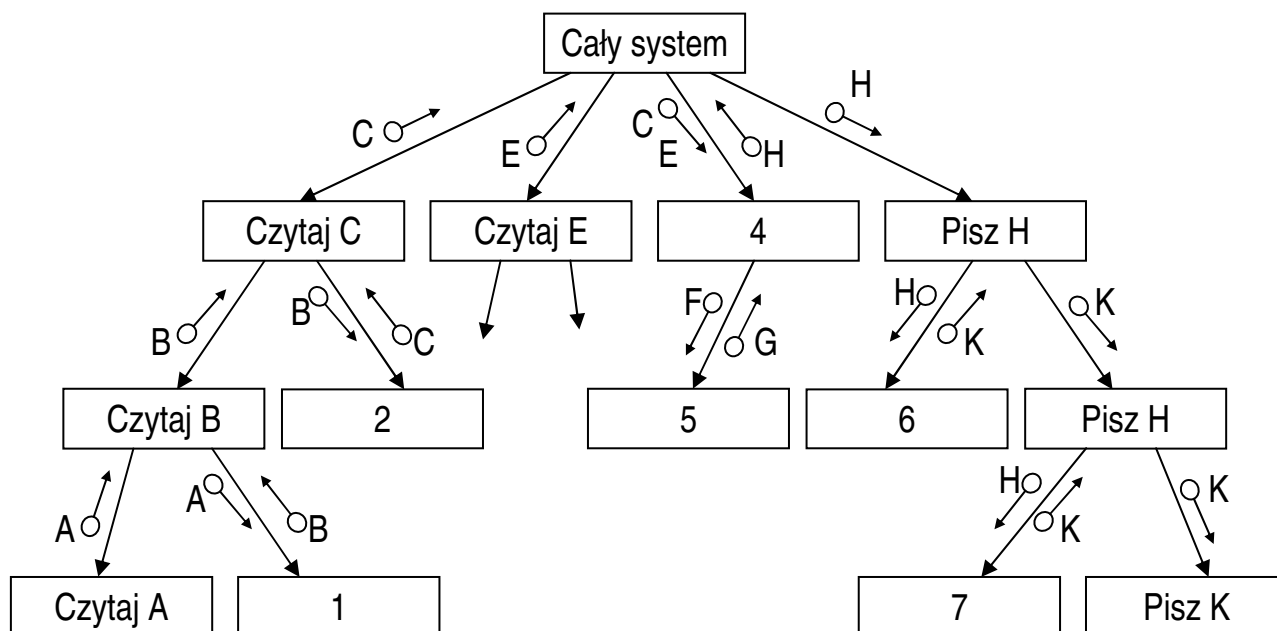
„Wejście”



„Wyjście”



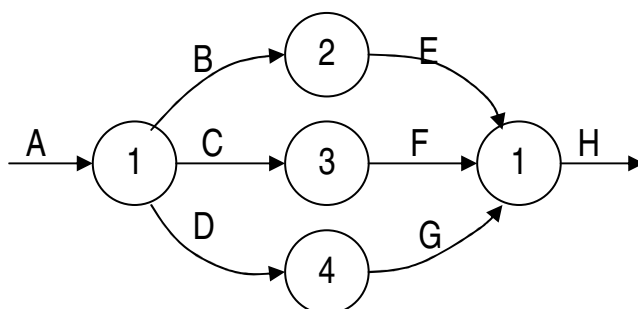
aż do uzyskania pełnego schematu:



→ Niższe poziomy DFD tworzą niższe warstwy podprogramów

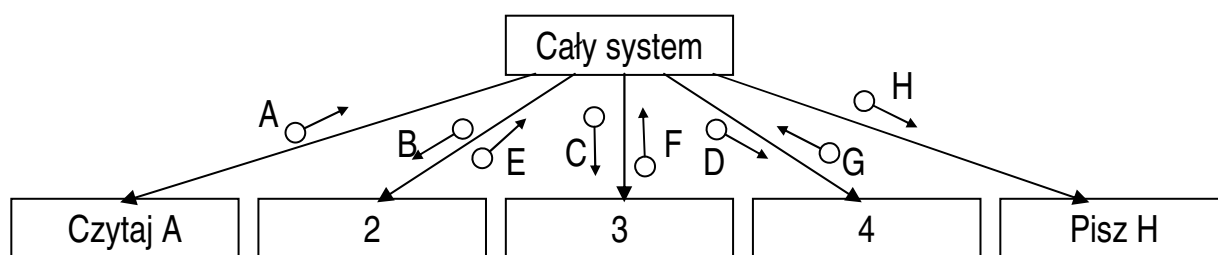
Struktura transakcyjna

Zawiera równoległe elementy przetwarzające, np.:



Budowa schematu struktury programu przebiega w kilku krokach:

1. Określenie struktury programu głównego



2. Rozwinięcie niższych warstw

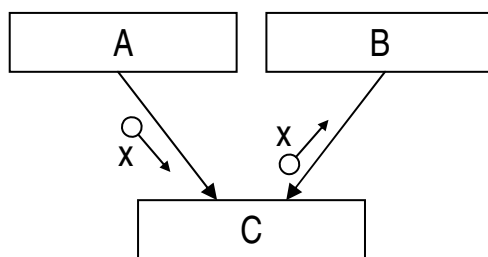
Heurystyczna praktyka — struktura 3-poziomowa:

- procesy transakcji (tu: 2, 3, 4),
- wywołują funkcje realizujące kroki transakcji,
- które korzystają z różnych funkcji usługowych.

Modyfikacja struktury

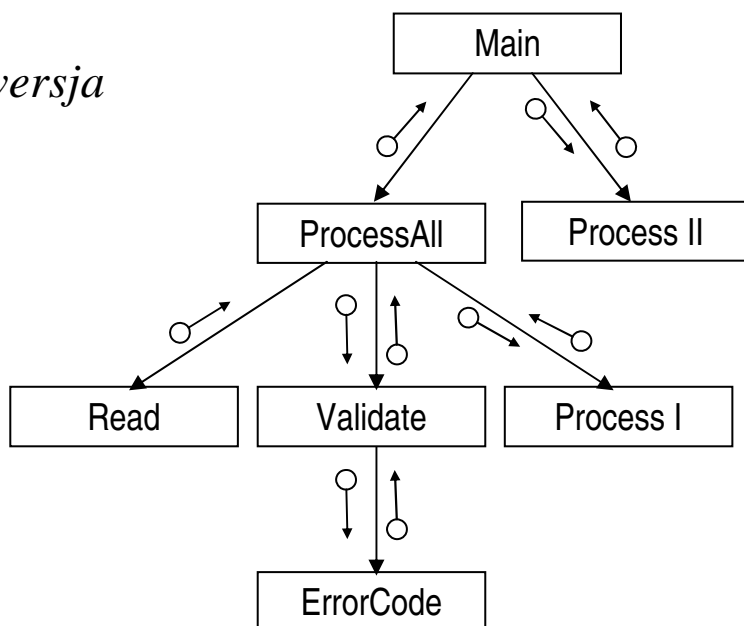
Cel: zwiększenie czytelności i modyfikowalności struktury programu.

- Uzupełnienie szczegółów obsługi błędów oczywistych.
- Optymalizacja struktury (poprawa spójności modułów).
- Eliminacja modułów zbędnych.
- Eliminacja modułów takich samych.
- Eliminacja patologicznych powiązań modułów — innych niż przez argumenty lub obszary danych, np.:
 - przez wskaźniki do wewnętrznych zmiennych statycznych
 - przez wspólne podprogramy niższego rzędu:



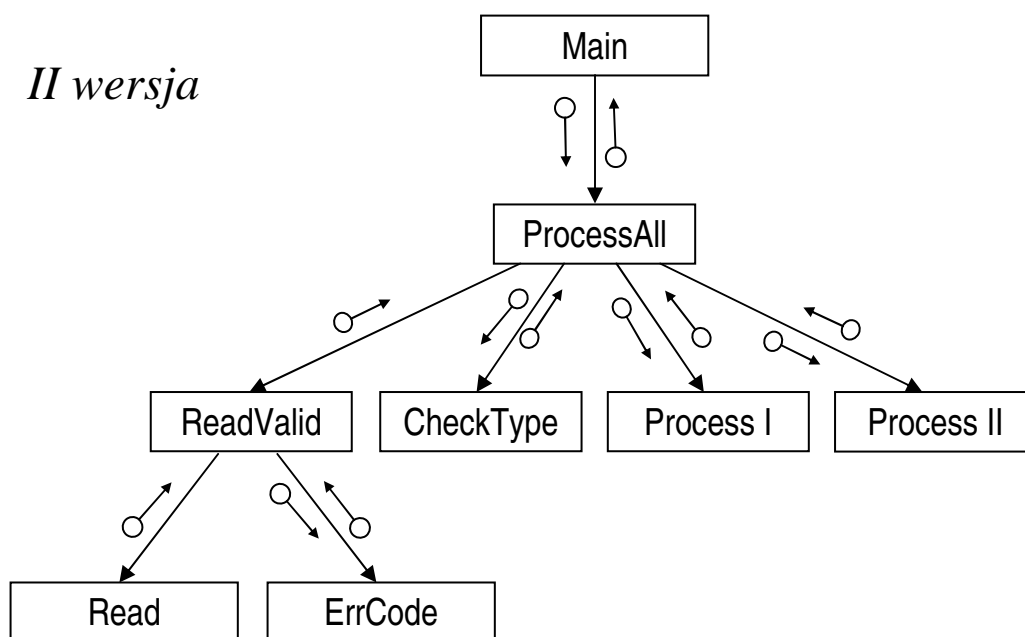
- Dopasowanie struktury sterowania i struktury programu:

I wersja



Validate:
 – *Ok / Err*
 – *Type (I, II)*

II wersja



- Optymalizacja wydajności wykonywana na samym końcu (po zbudowaniu poprawnie działającego systemu).