

Wprowadzenie do projektowania

Dr inż. Ilona Bluemke

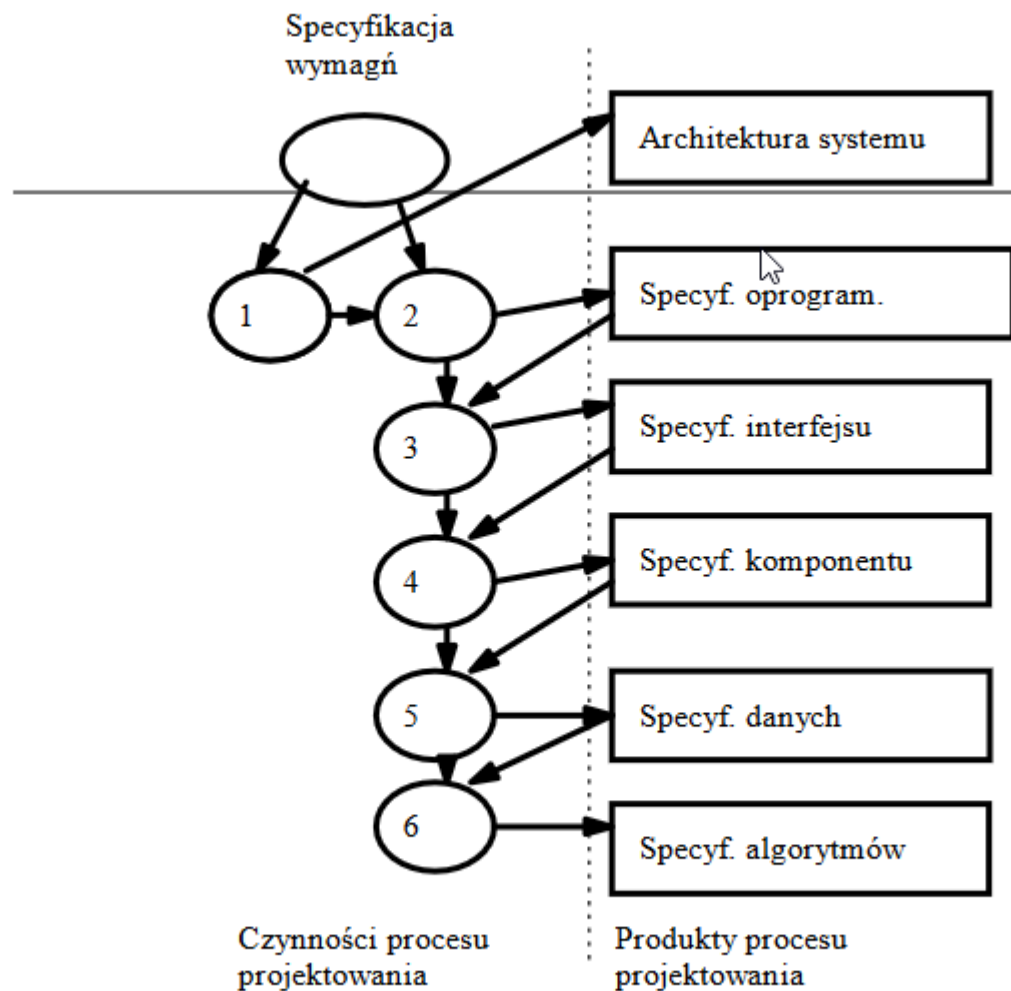


Projektowanie oprogramowania

Jest to proces tworzenia, nauczyć się go można poprzez doświadczenie, studiowanie istniejących systemów.

Kroki:

- Studiowanie i zrozumienie problemu, badanie problemu z różnych punktów widzenia.
- Szukanie wielu rozwiązań i ich ocena (wady, zalety).
- Opis każdej abstrakcji użytej w rozwiązaniu.





Czynności procesu projektowania

1. Projekt architektury - wyodrębnienie podsystemów i ich relacji, dokumentacja
2. Specyfikacja abstrakcji - dla każdego podsystemu specyfikacja dostarczanych usług i ograniczeń pracy
3. Projekt interfejsu dla każdego podsystemu, dokumentacja
4. Projekt komponentu - usługi przyporządkowane poszczególnym komponentom, projekt interfejsu komponentu
5. Projekt struktur danych używanych w systemie
6. Projekt algorytmów dostarczających usługi.



Metody projektowania

Rady, notacje prowadzące do powstania projektu sensownego

- **podejście funkcjonalne, strukturalne**

System modelowany jako transformacje danych. Zaczyna się od wysokiego poziomu, stopniowo ulepsza się w projekt bardziej szczegółowy.

- **podejścia obiektowe**

System widziany jako zbiór obiektów, powiązanych relacjami. Obiekty komunikują się przekazując komunikaty. Mają atrybuty, zbiory operacji



Jakość projektu


Nie ma obiektywnej metody stwierdzającej jaki projekt jest dobry.

"**dobry**" projekt to projekt:

- spełniający specyfikację,
- pozwala na produkcję efektywnego kodu,
- posiada "pożądane" własności np. daje się łatwo pielęgnować.

System daje się **łatwo pielęgnować** gdy:

- jest **spójny**,
- ma **mało powiązań**,
- jest **łatwo adaptowalny**.



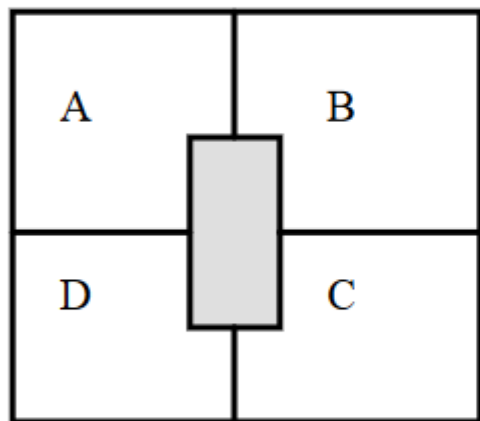
Spójność (cohesion)

Komponent implementuje logiczną funkcję. Jego części biorą udział w tej implementacji.

Poziomy spójności Constantine & Yourdon(1979):

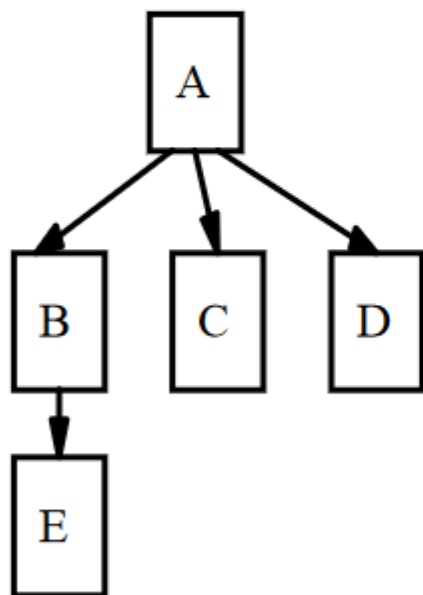
- **przypadkowa**,
- **logiczna** (elementy wykonujące podobne funkcje są zgrupowane w jednostce),
- **czasowa** (elementy aktywowane w tym samym czasie są zgrupowane w jednostce),
- **proceduralna** (elementy w jednostce tworzą sekwencję sterującą),
- **komunikacyjna** (elementy pracują na tych samych danych wejściowych lub produkują dane wyjściowe),
- **sekwencyjna** (wyjście jednego elementu jest wejściem następnego),
- **funkcyjna** (każda część jest konieczna do wykonania funkcji jednostki).

Silne powiązania



Systemy silnie powiązane jednostki są zależne jedna od drugiej np. pracują na wspólnych danych lub wymieniają informacje sterujące

Niski stopień zależności



Reprezentacji
informacji
zamknięta w
jednostce.

Jeśli nie jest to
możliwe to dostęp
do danych
wspólnych poprzez
odrębną jednostkę.



Adaptowalność

Łatwość rozumienia (understandability)

Wykonując modyfikacje powinniśmy rozumieć działanie jednostki systemu.

Dla łatwości rozumienia istotne są:

- złożoność jednostki (ale także nazwy ..)
- jakość dokumentacji.

Adaptowalność możliwa jeśli projekt jest:

- dobrze udokumentowany,
- spójny,
- o niewielkim stopniu zależności między elementami.



modelowanie

Pomaga zrozumieć funkcjonowanie,
ułatwia komunikację z użytkownikiem.

Różne modele prezentują system z
różnych perspektyw:

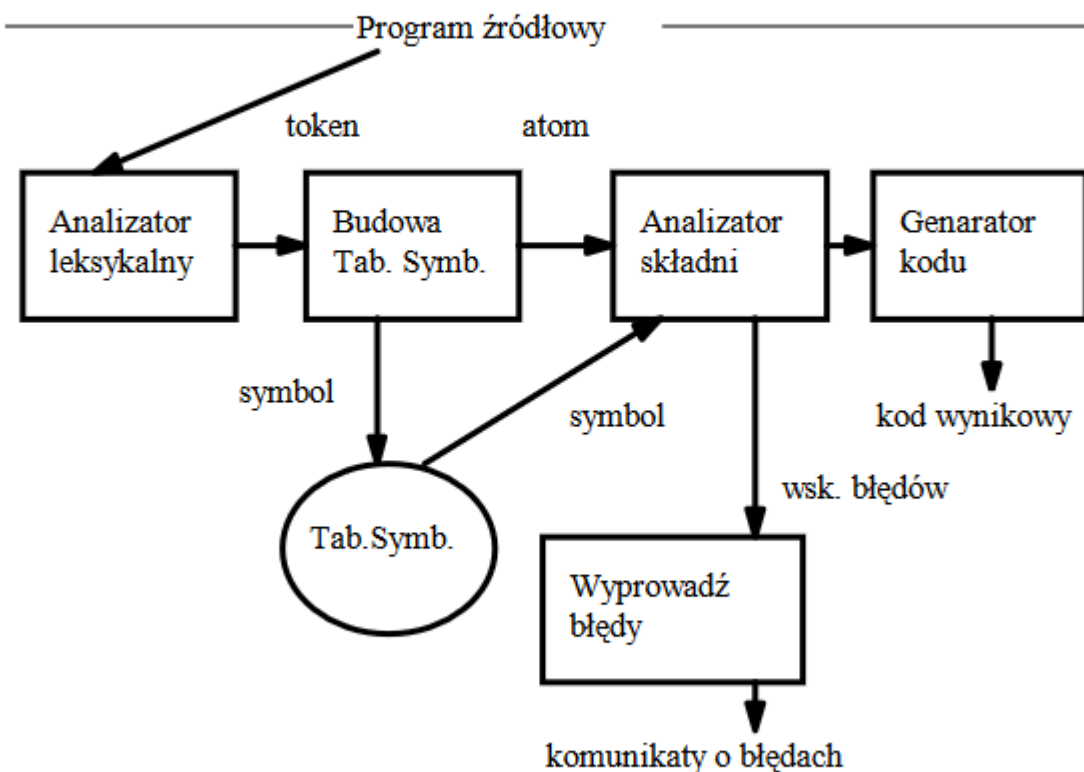
- Zewnętrzna – pokazuje kontekst systemu, otoczenie,
- Behavioralna – pokazuje zachowanie systemu,
- Strukturalna – architektura systemu, danych.



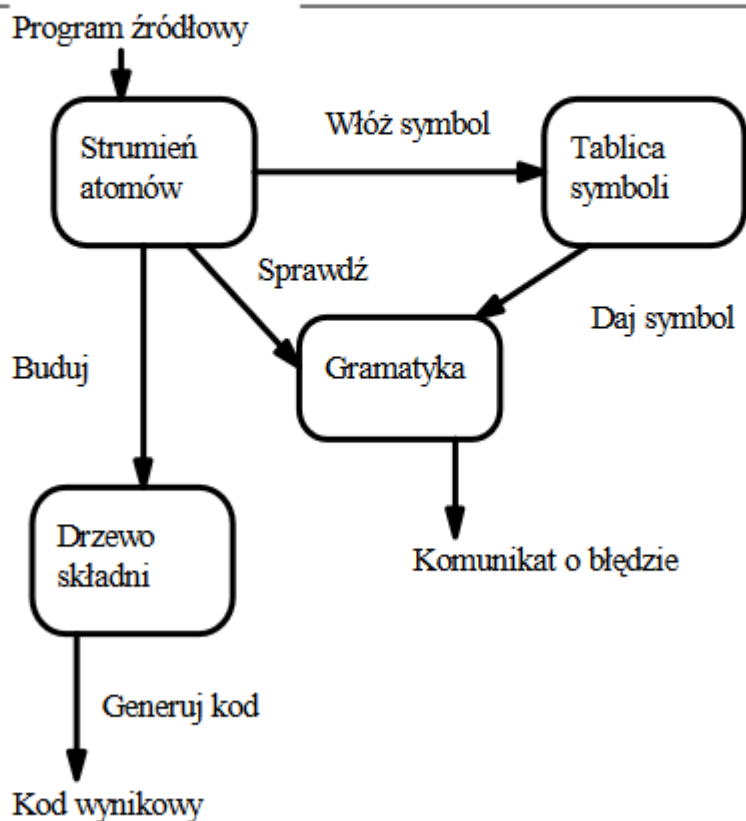
Typy modeli

- Przetwarzania, procesów
- Kompozytowe – pokazują budowę encji
- Architektoniczne - pokazują pod-systemy
- Klasyfikacyjne - pokazują wspólne cechy encji
- Zdarzeniowe - pokazują reakcję systemu na zdarzenia

Kompilator – ujęcie strukturalne



Kompilator – ujęcie obiektowe





Modele architektury

Dokumentują projekt architektoniczny

- Statyczne modele strukturalne pokazują główne komponenty systemu.
- Model dynamiczny pokazuje strukturę procesów systemowych.
- Model interfejsów definiuje interfejsy podsystemów.
- Model relacji (zależności) np. przepływu danych pokazuje zależności podsystemów
- Model instalacyjny pokazuje, jak podsystemy są rozmieszczone



Organizacja systemu

Podstawowe style organizacji systemu:

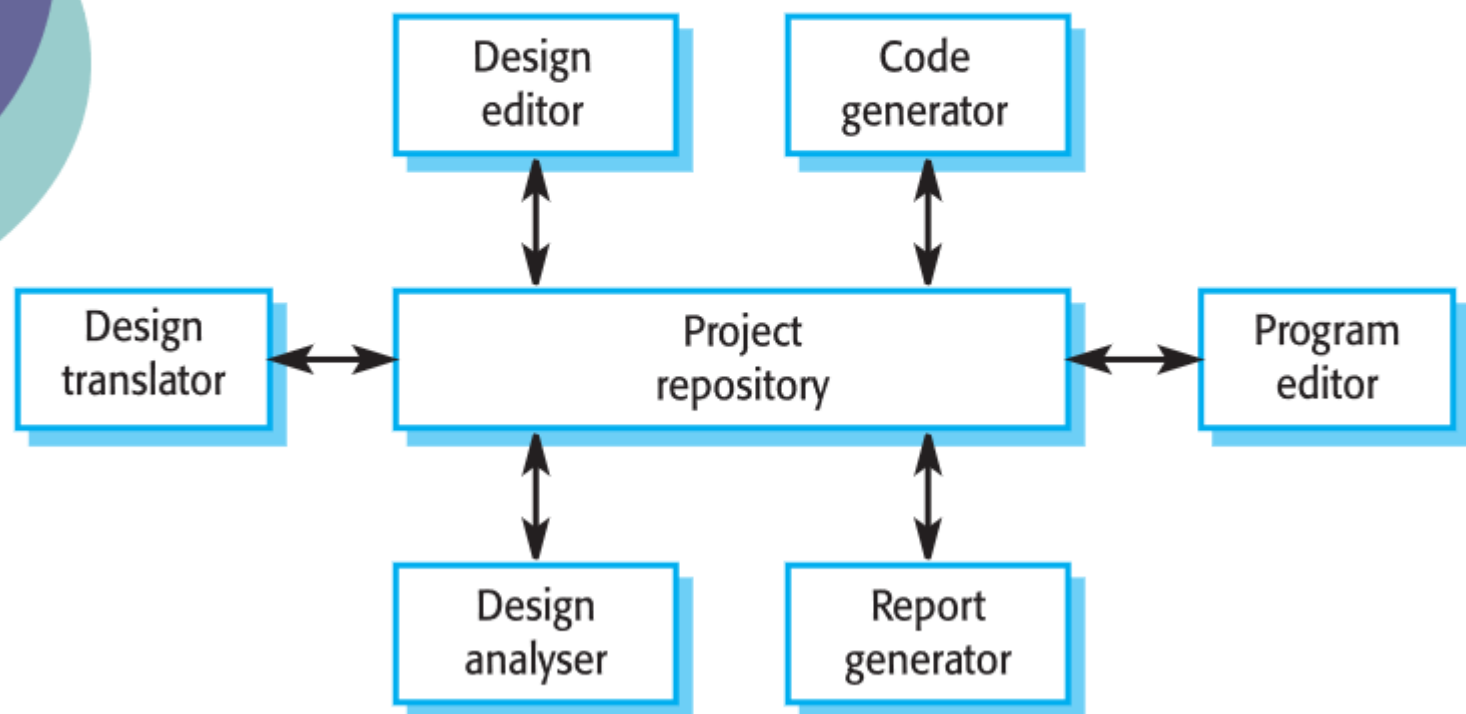
- Dzielone repozytorium
- Dzielone serwisy – styl serwerowy
- Warstwowa



Model z repozytorium

- Podsystemy muszą wymieniać dane.
Metody:
 - Dane przechowywane są w centralnej bazie danych – repozytorium i dostęp do nich mają wszystkie podsystemy.
 - Każdy podsystem ma własną bazę danych i przekazuje je explicite do innych podsystemów.
- Przy dużych ilościach danych model z repozytorium jest częściej używany.

Przykład architektury z repozytorium - CASE





Wady i zalety modelu z repozytorium

○ Zalety

- Efektywna metoda dzielenia dużych ilości danych
- Podsystemy nie muszą wiedzieć jak dane są produkowane. Centralne mechanizmy zarządzania np. backup, bezpieczeństwo, etc.
- Dzielony model is publikowany jako schemat repozytorium.

○ Wady

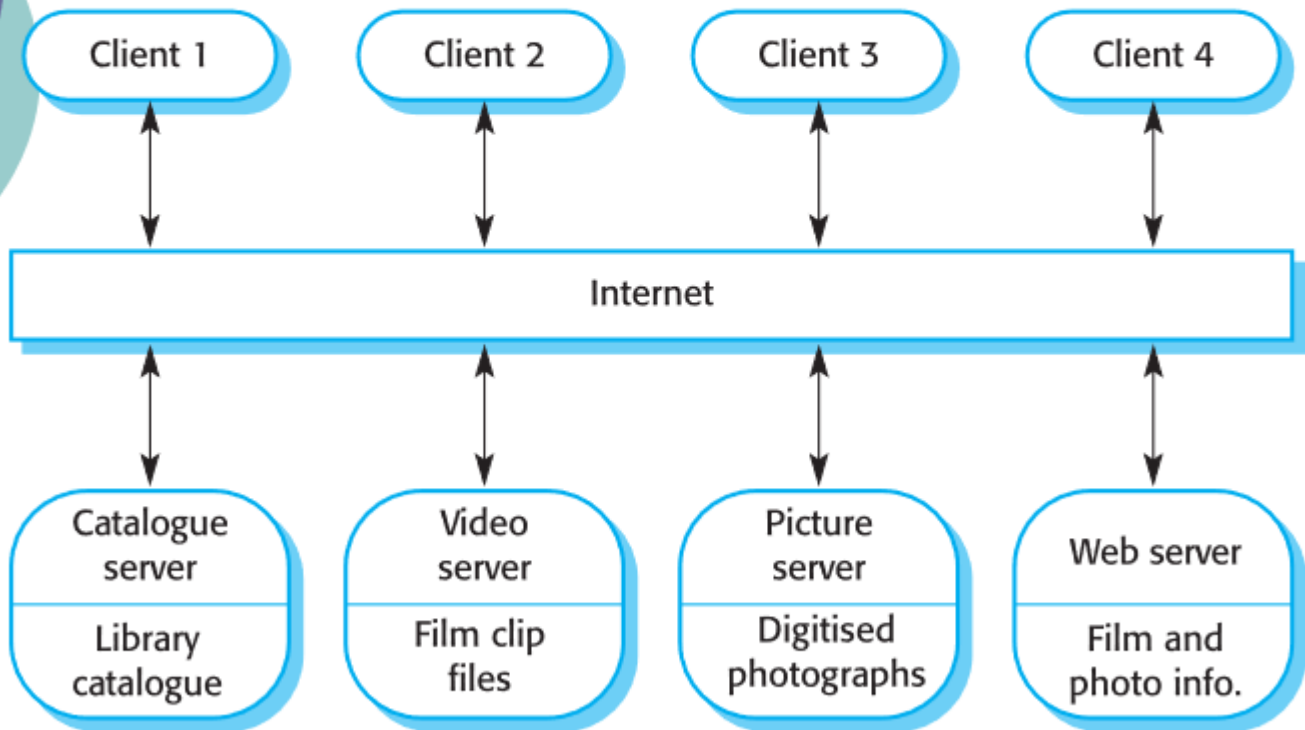
- Podsystemy muszą zgodzić się na model danych w repozytorium. Konieczny kompromis;
- Ewolucja danych jest trudna i kosztowna;
- Brak reguł zarządzania;
- Trudno jest efektywnie dystrybuować.



Model klient - serwer

- Model rozproszony, pokazujący jak dane i przetwarzanie są rozproszone na różnych komponentach.
- Zbiór samodzielnych serwerów dostarcza specyficznych usług np. Drukowanie, zarządzania danymi itd..
- Zbiór klientów wywołujących usługi.
- Sieć, pozwalająca klientom na dostęp do serwerów.

Biblioteka filmów: klient - serwer






Wady i zalety modelu klient-serwer

○ Zalety

- Prosta dystrybucja danych;
- Efektywne wykorzystanie systemów sieciowych.
- Łatwe dodawanie nowego serwera lub upgrade istniejącego.

○ Wady

- Brak modelu danych dzielonych więc podsystemy mają różną organizację danych. Wymiana danych może być nieefektywna;
- Redundancyjne zarządzanie w każdym serwerze;
- Brak centralnego rejestru nazw i usług – może być trudne stwierdzenie jakie serwery i usługi są dostępne.



Abstrakcyjne maszyny – model warstwowy

- Organizacja systemu jako zbioru warstw (maszyn abstrakcyjnych). Każda warstwa dostarcza zbioru usług.
- Wspiera przyrostowy rozwój podsystemów w różnych warstwach. Zmiana interfejsu warstwy wpływa tylko na warstwy przyległe.
- Często jest trudno strukturalizować system w ten sposób.



Architektura warstwowa

Configuration management system layer

Object management system layer

Database system layer

Operating system layer



Style sterowania

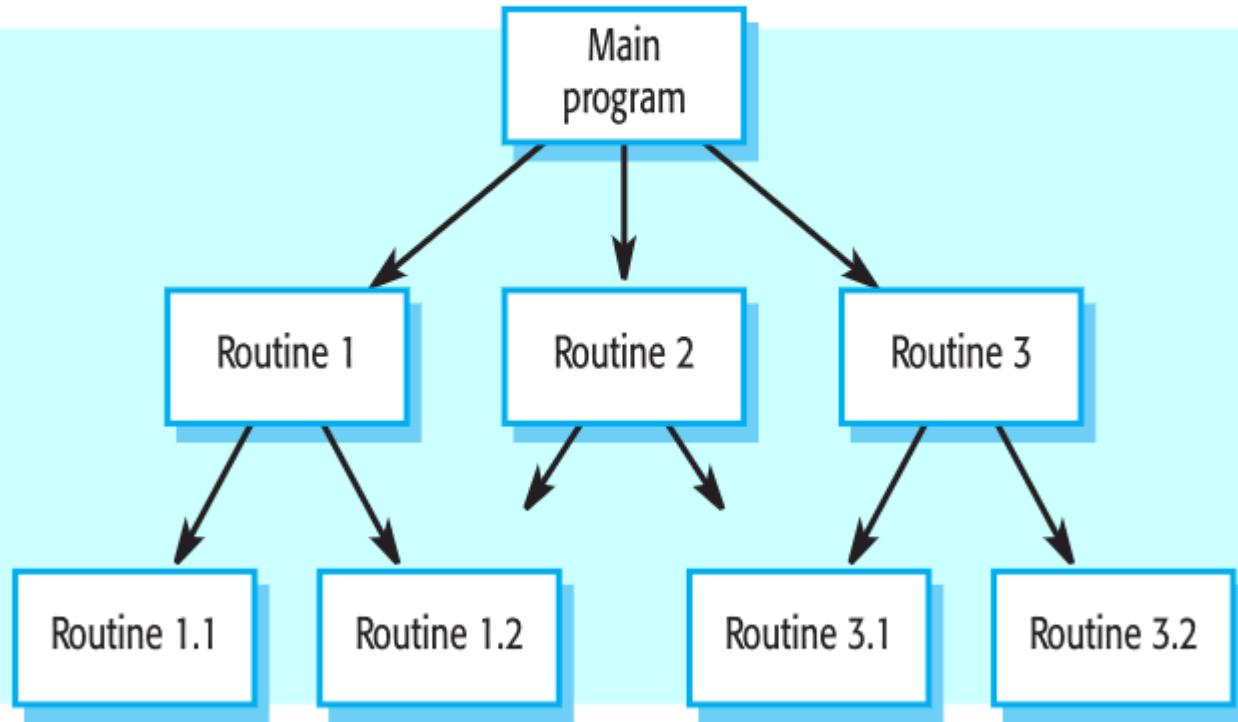
- Związane z przepływem sterowania pomiędzy podsystemami. Różne od modelu dekompozycji.
- Scentralizowane sterowanie
 - Jeden podsystem jest odpowiedzialny za sterowanie, uruchamia, zatrzymuje inne podsystemy.
- Sterowanie zdarzeniowe
 - Każdy podsystem odpowiada na zewnętrznie generowane zdarzenia z innych podsystemów lub z otoczenia.



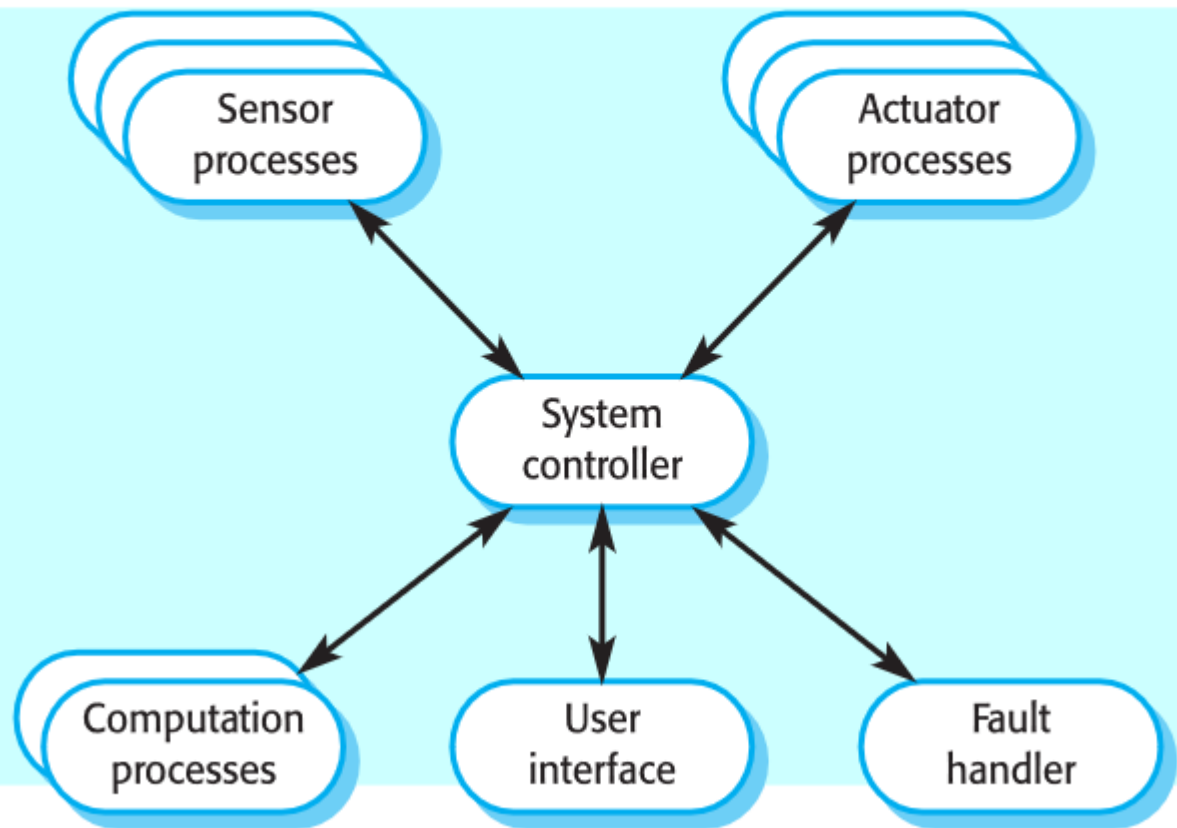
Zcentralizowane sterowanie

- Podsystem sterowania jest odpowiedzialny za wykonanie innych podsystemów.
- Model **Call-return**
 - Top-down model podprogramów, sterowanie rozpoczyna się w wierzchołku hierarchii podprogramów i przesuwa się w dół. Nadaje się do systemów sekwencyjnych.
- Model **Manager**
 - Nadaje się do systemów równoległych. Jeden z komponentów systemu steruje zatrzymaniem, uruchamianiem i koordynacją innych procesów systemowych.

Model call-return



Sterowanie w systemach czasu rzeczywistego



Systemy zdarzeniowe

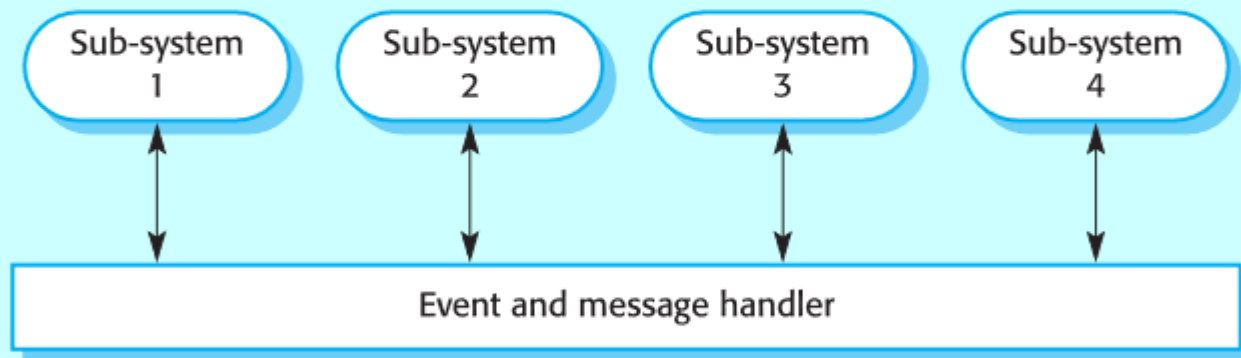
- Sterowane zewnętrznie generowanymi zdarzeniami, pojawienie się zdarzenia nie jest kontrolowane przez system obsługujący zdarzenie.
- Modele systemów zdarzeniowych
 - Model rozgłaszania (broadcast). Zdarzenie jest rozgłaszane do wszystkich podsystemów. Dowolny podsystem, który jest w stanie, może je obsłużyć;
 - Model sterowany przerwaniem. Używany w systemach czasu rzeczywistego, przerwanie są wykrywane przez interrupt handler i przekazane do innych komponentów do obsługi.
- Inne – systemy produkcyjne



Model rozgłaszania

- Efektywne w integracji podsystemów na różnych komputerach w sieci.
- Podsystemy rejestrują zainteresowanie pewnymi zdarzeniami. Jeżeli pojawia się zdarzenie, sterowanie jest przekazywane do podsystemu obsługującego je.
- Reguły sterownia nie są wbudowane w event i message handler. Podsystemy decydują, które zdarzenia je interesują.
- Podsystemy nie wiedzą, czy i kiedy pojawi się zdarzenie do obsługi.

Rozgłaszanie



Systemy sterowane przerwaniami

- Używane w systemach czasu rzeczywistego gdzie potrzebna jest szybka reakcja na zdarzenie.
- Znane są typy przerwań, dla każdego typu jest zdefiniowany program obsługi.
- Każdy typ jest związany z określonym miejscem w pamięci, sprzętowy przełącznik przełącza do programu obsługi.
- Pozwala na szybką odpowiedź ale trudne do oprogramowania i walidacji.

System sterowany przerwaniem

