FAZA PROJEKTOWANIA

Technologie Obiektowe

Cele projektowania

- **Faza określenia wymagań:**

  Co system ma robić?

- **Faza analizy:**

  JAK system ma działać?

- **Faza projektowania:**

  JAK system ma być zaimplementowany?

Wynikiem fazy analizy jest model projektowy

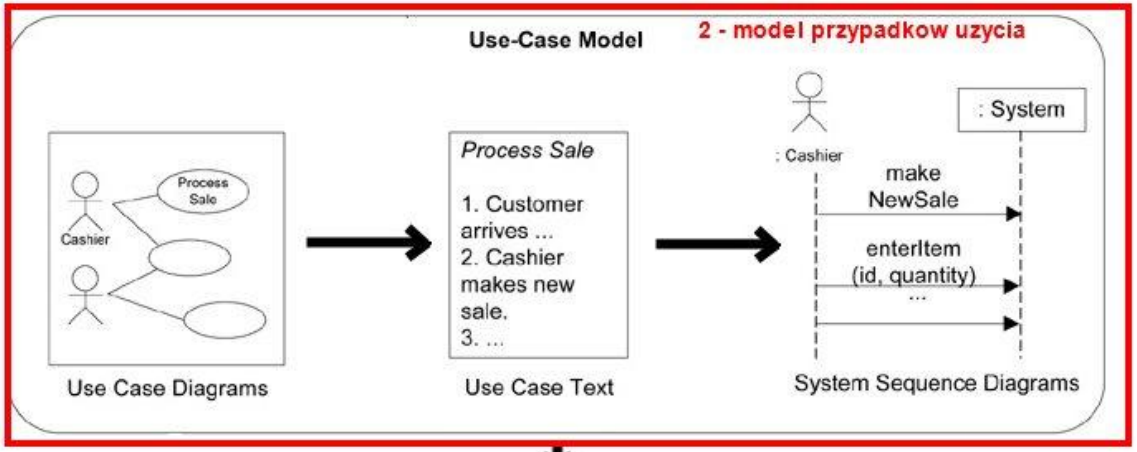
W skrócie

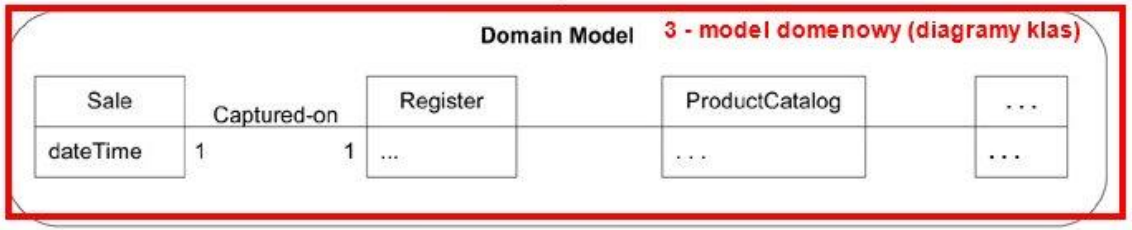
- trzeba wiedzieć **jak wygląda proces,** bez tego nie ma co się brać za dalsze projektowanie (np. przypadki użycia)
- **model przypadków użycia**
  - diagramy przypadków użycia
  - przypadki użycia w formie tekstowej (maksymalnie po stronie na przypadek)
  - uproszczony diagram sekwencji
- na podstawie modelu przypadków użycia można zbudować **model domenowy -** uproszczony diagram klas (tylko atrybuty i relacje)
- **model projektowy –** mając wykonane dwa powyższe modele można przystąpić do budowy modelu projektu ("Design Model"). W jego ramach tworzy się jednocześnie diagram klas (DCD) oraz diagram sekwencji.

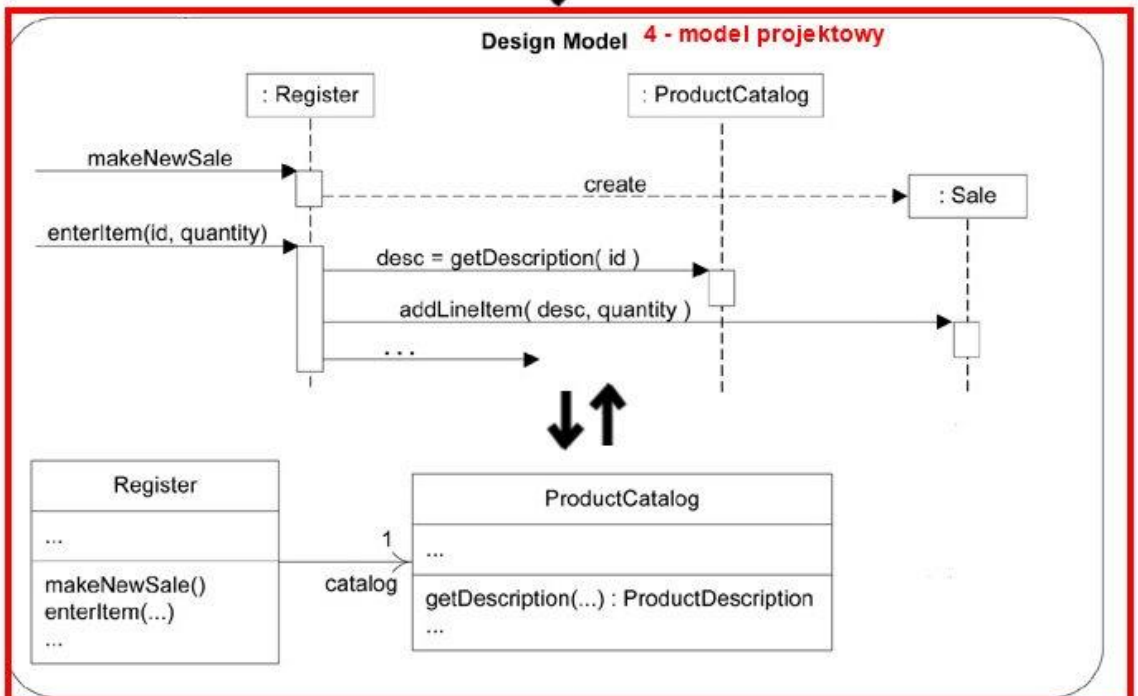
Unified Process

Unified Process to przebieg procesu/metodyki wytwarzania oprogramowania.
Jest to podstawowa metodyka iteracyjna, jej pochodne to: RUP, Agile UP, XP, Scrum, itp

Schemat z książki "Applying UML and Patterns"


Use-Case Model
2 - model przypadkow uzycia
Process Sale
Cashier
Use Case Diagrams
Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...
Use Case Text
: Cashier
: System
make NewSale
enterItem (id, quantity)
...
System Sequence Diagrams


Domain Model
3 - model domenowy (diagramy klas)
Sale
dateTime
Captured-on
1
1
Register
...
ProductCatalog
. . .
. . .
. . .


Design Model
4 - model projektowy
: Register
: ProductCatalog
: Sale
makeNewSale
create
enterItem(id, quantity)
desc = getDescription(id)
addLineItem(desc, quantity)
. . .
Register
...
makeNewSale()
enterItem(...)
...
1
catalog
ProductCatalog
...
getDescription(...) : ProductDescription
...

Unified Process jest konfigurowalny, w razie konieczności można dodać, czy usunąć pewne elementy (np. w prostych przypadkach tworzenie modelu domenowego może być zbędne, natomiast w skomplikowanych problemach pomocny może być dodatkowy model analityczny).

Czynności w fazie projektowania

- **Uszczegóławianie modelu analitycznego**
  - Dodatkowe elementy notacji (metody (+) publiczna, (#) zabezpieczona, (–) prywatna)
- **Uszczegóławianie metod**
  - Podanie nagłówków wraz z parametrami, określenie rodzaju wiązania (statyczne, późne), zastąpienie wysyłania metod przez wyrażenie wyliczające ich wartość
- **Określenie implementacji związków (asocjacji)** np. wprowadzenie dodatkowych (prywatnych) atrybutów (referencyjnych)
- **Reguły transformacji schematów obiektowych w relacyjne**

Dodatkowe składowe modelu projektowego

1. Interfejs użytkownika
2. Zarządzanie danymi trwałymi
3. Zarządzanie pamięcią operacyjna
4. Zarządzanie zadaniami

   RAD — Szybkie tworzenie aplikacji (Rapids Application Development)

Borland Delphi RAD Pack, IBM Visual Age (4 Cobol, Java, C++, Smalltalk), MS Access Developer's Toolkit, MS Visual Studio, Power Buiokder Desktop, itp.

Projektowanie interfejsu użytkownika

Gotowe narzędzia (Zapp Factory, Visual Basic) zawierające gotowe biblioteki do interaktywnego i wizualnego projektowania: dialogów, okien, menu, map bitowych, ikon, pasków narzędziowych, itp., w oparciu o gotowe elementy.

- Schematy akcji na zdarzenia (np. wybór opcji z menu)
- Sposoby interakcji systemu z użytkownikiem
  - polecenie linii komend (szybkie dla sprawnych użytkowników)
  - mysz (wygodne dla początkujących)
  - głos (śpiew przyszłości)

UWAGA!: Należy zawsze pamietac o skrótach klawiszowych

Zasady projektowanie interfejsu użytkownika

- Umieszczać etykietę nad/obok pól edycyjnych
  - *Typowe pola/klawisze (OK, Anuluj) zawsze w tym samym miejscu (np. od dołu, lub od prawej)*
- Zachowywać sens i spójność tłumaczeń
- Sensowność doboru okien dialogowych
- Skróty klawiaturowe
- Wymaganie potwierdzenia poleceń
- Prosta obsługa błędów

Zasady projektowanie interfejsu użytkownika

- Odwoływanie akcji
  - *Zachowanie kontroli nad systemem przez użytkownika*
- System nie może robić rzeczy, które nie wynikają z poleceń
- Unikać skomplikowanych zależności kontekstowych
- Grupować powiązane logicznie operacje
- Reguła Millera: $7(\pm 2)$ *(zapamiętywanie informacji)*

Reguła Millera 7+/-2

Człowiek może się jednocześnie skupić na **5-9** elementach.

Ta reguła powinna być uwzględniona przy projektowaniu interfejsu użytkownika.
Dotyczy to liczby opcji menu, podmenu, pól w dialogu, itd.

Techniki i diagramy strukturalne

- **Moduł**

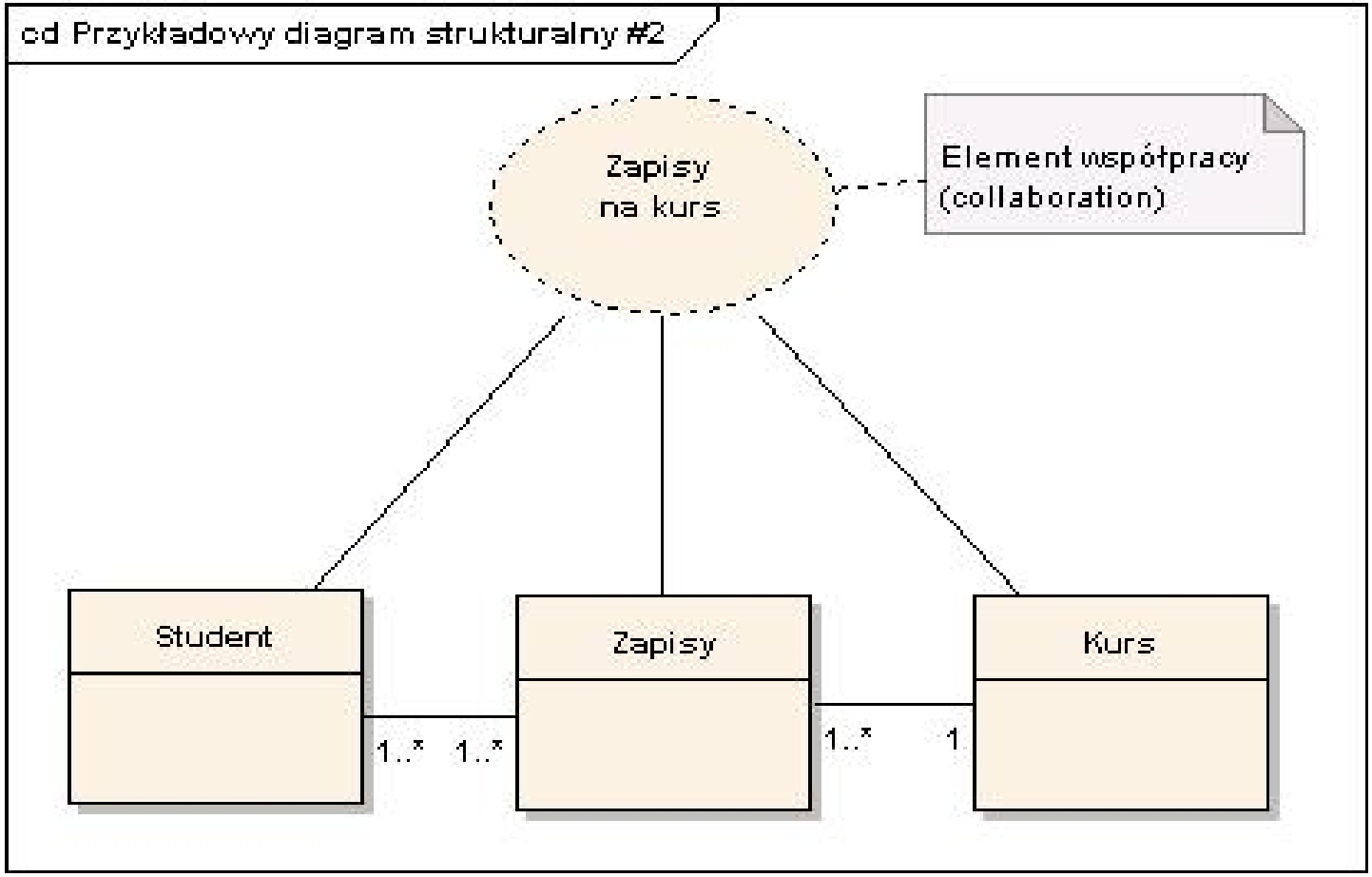
  Aktywna składowa programu (procedura, funkcja lub ich grupa)
- **Wywołanie**

  Akcja jednego modułu inicjowana przez inny
- **Moduł biblioteczny, dane**
- **Relacja w bazie danych, plik, zmienne programu**
- **Flagi przepływu danych**

  od (np. wyniki) i do (np. parametry) procedury wywołującej

Diagram strukturalny (przykład)

Diagram strukturalny = uszczegółowienie DFD



Projektowanie składowej zarządzania danymi

- **Każdy większy system musi w sposób trwały przechowywać dane**

Sposoby trwałego przechowywania:

- w pliku (lokalnie u klienta)
- w bazie danych

- **Metody przechowywania:**
  - scalone (w jednym pliku/bazie)
  - rozproszone (w kilku plikach/bazach)
- **SZDB -** System/Składowa Zarządzania Bazą Danych

System **Z**arządzania **B**azą **D**anych

oprogramowanie bądź system informatyczny służący do zarządzania bazą danych. System zarządzania bazą danych może być również **serwerem bazy danych** (**SBD**) lub też może udostępniać bazę danych lokalnie – na określonym komputerze.

- DB2
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Microsoft Access
- Kexi

Zalety i wady baz danych

- Trwałość
- Struktura bazy wymaga strukturalizacji danych
- Wsparcie dla szybkiego analizowania
- Koszt założenia i utrzymywania

Cechy dobrej bazy danych

- ✓ Wysoka efektywność i stabilność
- ✓ Bezpieczeństwo i prywatność danych, spójność i integralność przetwarzania
- ✓ Automatyczne sprawdzanie warunków integralności danych
- ✓ Wielodostęp, przetwarzanie transakcji
- ✓ Rozszerzalność (zarówno dodawanie danych jak i dodawanie ich rodzajów)
- ✓ Możliwość geograficznego rozproszenia danych
- ✓ Możliwość kaskadowego usuwania powiązanych danych

Integralność: poprawność danych w sensie ich organizacji i budowy.

Spójność: zgodność danych z rzeczywistością lub z oczekiwaniami użytkownika.

Cechy dobrej bazy danych

- Stabilność
- Bezpieczeństwo
- Wielodostęp
- Skalowalność
- Możliwość rozproszenia danych
- Obsługa zapytać wysokiego poziomu (SQL, OQL)

Kryteria doboru SZBD

- Wydajność
- Skalowalność
- Funkcjonalność
- Zgodność ze standardami
- Łatwość uzycia
- Niezawodność
- Wspomaganie
- Środowisko
- Cena

Wady relacyjnych baz danych

- **RBD — 90% rynku**
- **Niezgodność OO metod analizy oraz technologii relacyjnej**
  - Konieczność tłumaczenia
  - Problem reprezentacji pól zmiennej długości jako krótki
- **„Implementacyjność" modelu relacyjnego**
- **Brak zgodność SQL z C++**
- **Brak możliwości zagnieżdżania typów**

Optymalizacja projektu (1)

1. **Bezpośrednia implementacja projektu może prowadzić do systemu o zbyt niskiej efektywności.**
   - Wykonanie pewnych funkcji jest zbyt wolne
   - Struktury danych mogą wymagać zbyt dużej pamięci operacyjnej i masowej
2. **Optymalizacja może być dokonana:**
   - Na poziomie projektu
   - Na poziomie implementacji
   - **Sposoby stosowane na etapie implementacji:**
     - Stosowanie zmiennych statycznych zamiast dynamicznych (lokalnych).
     - Umieszczanie zagnieżdżonego kodu zamiast wywoływania procedur.
     - Dobór typów o minimalnej, niezbędnej wartości.

Optymalizacja projektu (2)

Co może przynieść zasadnicze zyski optymalizacyjne?

- **Zmiana algorytmu przetwarzania**. Np. zmiana algorytmu sortującego poprzez wprowadzenie pośredniego pliku zawierającego tylko klucze i wskaźniki do sortowanych obiektów może przynieść nawet 100-krotny zysk.
- **Wyłowienie "wąskich gardeł"** w przetwarzaniu i optymalizacja tych wąskich gardeł poprzez starannie rozpracowane procedury. Znane jest twierdzenie, że 20% kodu jest wykonywane przez 80% czasu.
- **Zaprogramowanie "wąskich gardeł" w języku niższego poziomu,** np. w C dla programów w 4GL.
- **Denormalizacja relacyjnej bazy danych,** łączenie dwóch lub więcej tablic w jedną.
- **Stosowanie indeksów, tablic wskaźników i innych struktur pomocniczych**.
- **Analiza mechanizmów buforowania danych** w pamięci operacyjnej i ewentualna zmiana tego mechanizmu (np. zmniejszenie liczby poziomów)

Ograniczenia środowiska implementacji

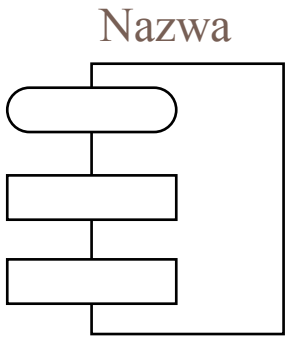
- Brak wielokrotnego dziedziczenia
- Brak dziedziczenia
- Brak metod wirtualnych (przesłaniania)
- Brak złożonych atrybutów
- Brak obsługi typów multimedialnych

Określenie struktury systemu

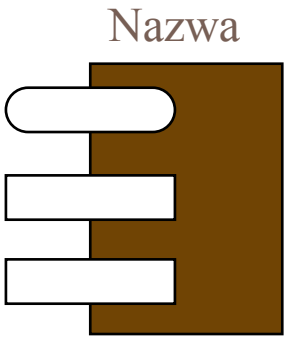
- Określenie struktury kodu źródłowego
- podział na pliki, zależność miedzy nimi, podział składowych miedzy pliki
- Podział systemu na aplikacje
- Rozmieszczenie danych i aplikacji na stacjach roboczych i serwerach

Oznaczenia (Booch)

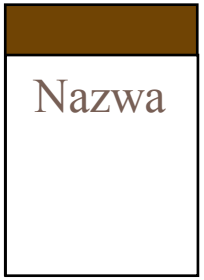
Deklaracja:



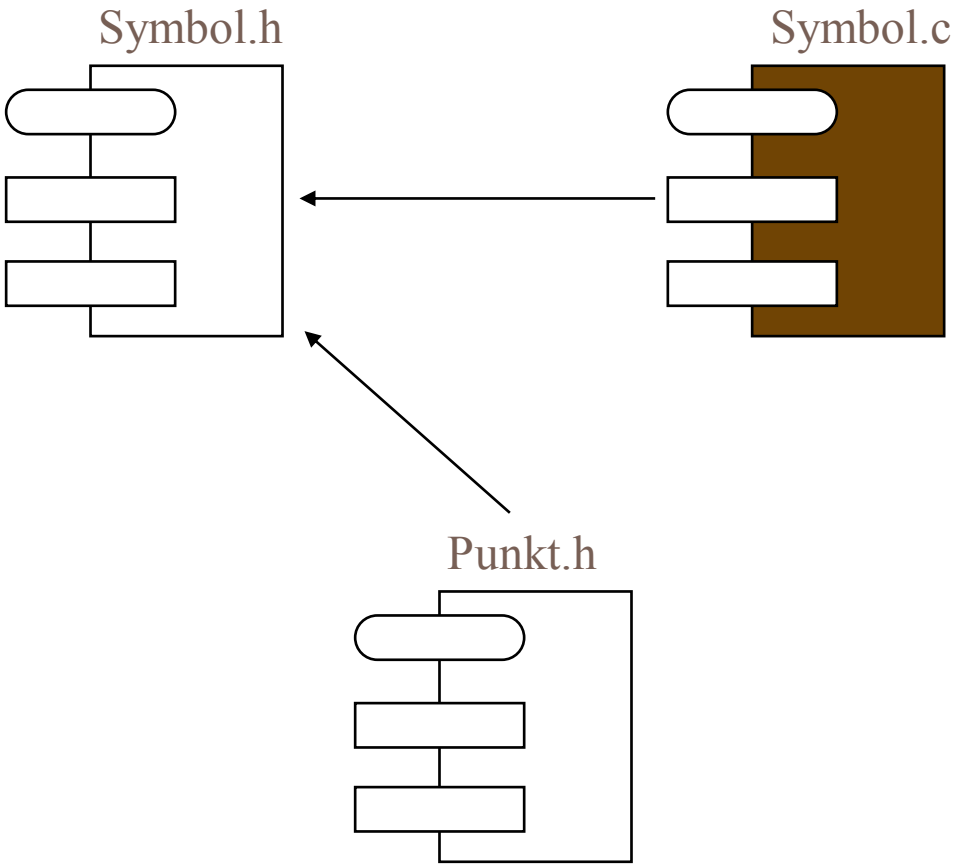
Definicja:



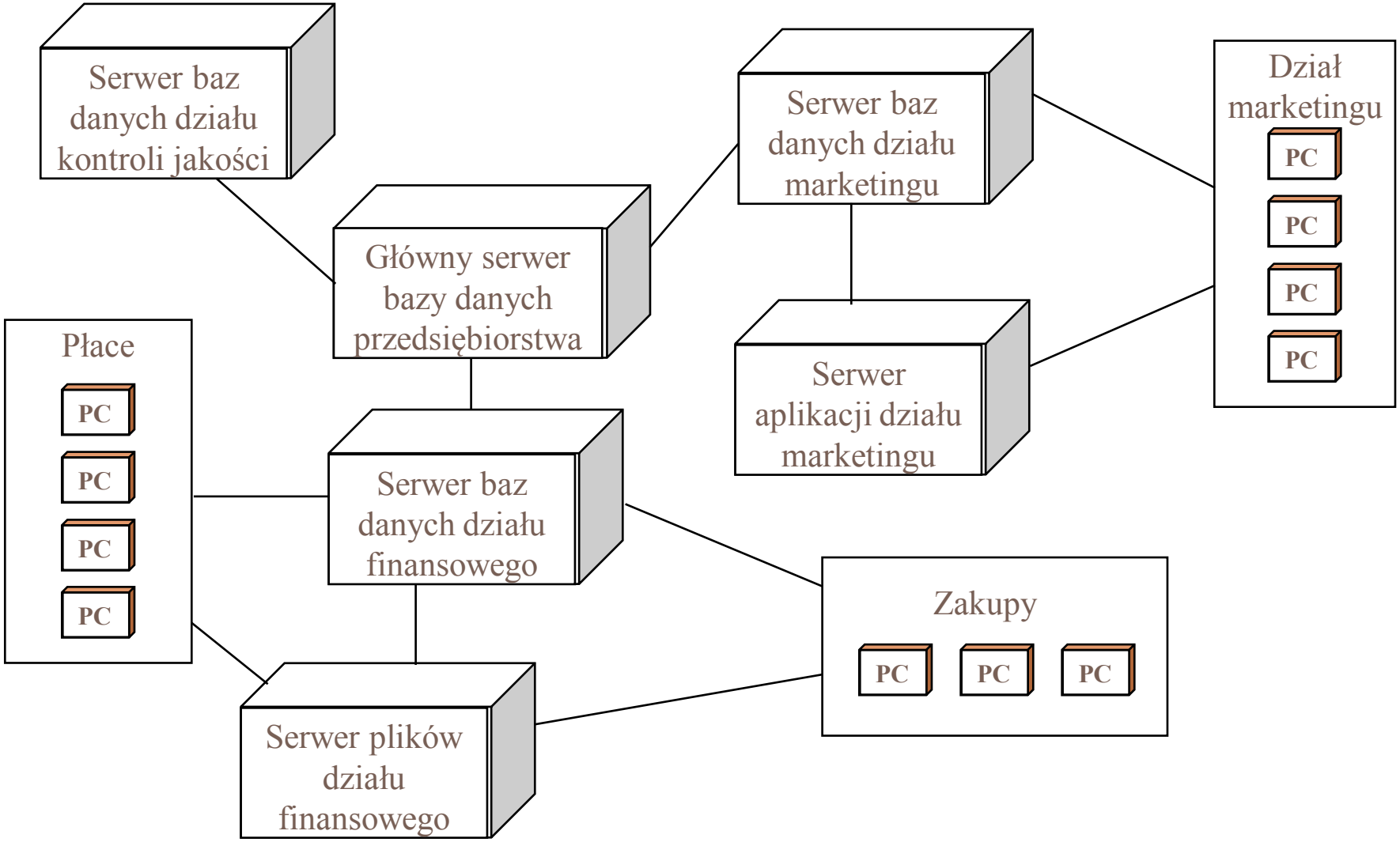
Moduł główny:



Przykład zależności kompilacji dla C++



Graficzny opis sprzętowej konfiguracji systemu



Poprawność i jakość projektu

Każdy projekt należy sprawdzać pod względem poprawności i jakości.

Poprawność oznacza, że opis projektu jest zgodny z zasadami posługiwania się notacjami. Nie gwarantuje, że projekt jest zgodny z wymaganiami użytkownika.

- **Kompletność = zdefiniowane są:**
  - wszystkie klasy
  - wszystkie pola (atrybuty)
  - wszystkie metody
- **Niesprzeczność**
- **Spójność =** semantyczna zgodność wszystkich informacji zawartych na poszczególnych diagramach i w specyfikacji.
- **Zgodność z regułami notacji**

Poprawność diagramów klas

- Acykliczność związków generalizacji-specjalizacji
- Opcjonalność cyklicznych związków agregacji
- Opis parametrów WE/WY i działania metod w specyfikacji sygnatur

Poprawność diagramów stanów

- Brak stanów nieosiągalnych z początkowego
- Brak stanów zakleszczonych (oprócz stanów końcowych)
- Jednoznaczność interpretacji warunków przejścia pomiędzy stanami

Jakość projektu

Metody projektowe i stosowane notacje są w dużym stopniu nieformalne, zaś ich użycie silnie zależy od rodzaju przedsięwzięcia programistycznego.

Jest więc dość trudno ocenić jakość projektu w sensie jego adekwatności do procesu konstruowania oprogramowania i stopnia późniejszej satysfakcji użytkowników: stopień spełnienia wymagań, niezawodność, efektywność, łatwość konserwacji i ergonomiczność.

Pod terminem **jakość** rozumie się bardziej szczegółowe kryteria:

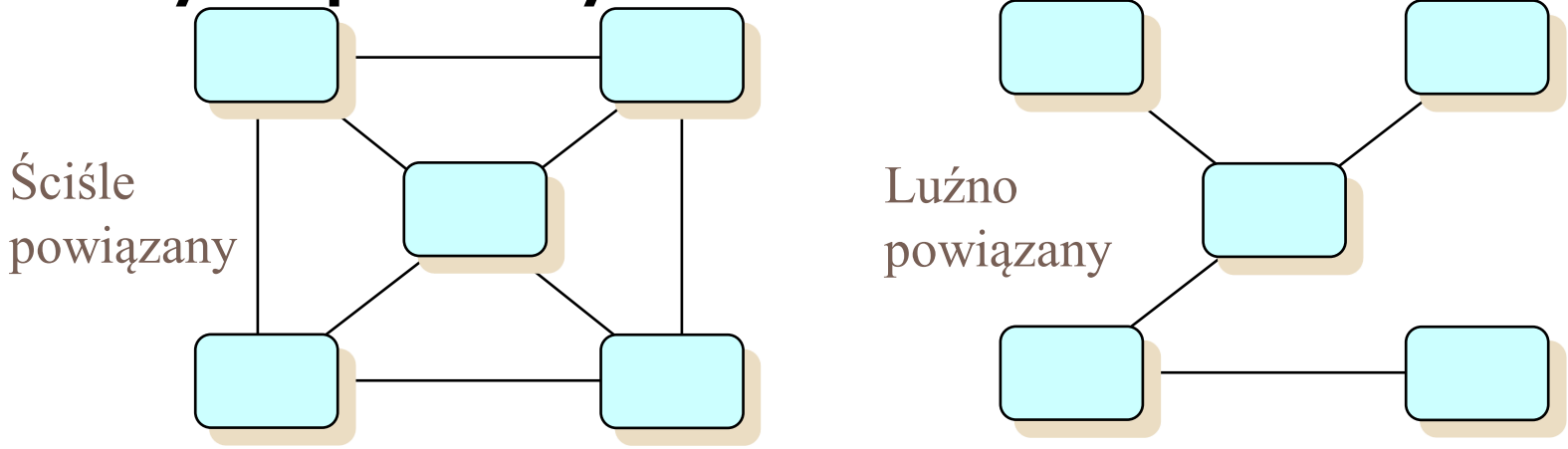
- spójność
- stopień powiązania składowych
- przejrzystość

Jakość projektu — spójność

Rodzaje spójności	
Logiczna	części wydzielone ze względu na funkcje
Czasowa	czynności wykonywane w tym samym czasie (start, koniec pracy)
Sekwencyjna	składowe wykonywane jedna po drugiej
Komunikacyjna	składowe wykorzystują te same dane WE, wspólnie produkują dane WY
Funkcyjna	składowe realizują jedną funkcją
Przypadkowa	Podział na moduły (części) wynika wyłącznie z tego, że całość jest za duża (utrudnia wydruk, edycję)

Jakość projektu - stopień powiązań składowych

- **Ścisły — niepolecany**
- **Luźny — polecany**



Powiązania pomiędzy składowymi:

- Korzystanie przez procesy/moduły z tych samych danych
- Przepływy danych pomiędzy procesami/modułami
- Związki pomiędzy klasami
- Przepływy komunikatów
- Dziedziczenie

Jakość projektu — przejrzystość

- Odzwierciedlenie rzeczywistości
- Spójność i powiązanie składowych
- Nazewnictwo
- Czytelna i pełna specyfikacja
- Stopień abstrakcji odpowiedni dla danego poziomu

Wymagania niefunkcjonalne w fazie projektowania

- Odnośnie wydajności
- Odnośnie interfejsu *(protokoły, formaty plików)*
- Operacyjne *(aspekty ergonomiczne, języki, pomoce)*
- Odnośnie zasobów *(ilość procesorów, pojemność dysków)*
- Odnośnie weryfikacji *(sposoby przeprowadzenia)*
- Odnośnie akceptacji i testowania
- Odnośnie dokumentacji

Wymagania niefunkcjonalne w fazie projektowania

- **Odnośnie bezpieczeństwa**
- **Odnośnie przenaszalności**
- **Odnośnie jakości**
  - *wybór metod projektowania*
  - *decyzje dotyczące ponownego użycia*
  - *wybór narzędzi*
  - *wybór metod oceny projektu przez ciała zewnętrzne*
- **Odnośnie niezawodności**
- **Odnośnie pielęgnacji**
- **Odnośnie odporność na awarie**

Kluczowe czynniki sukcesu fazy projektowania

- **Wysoka jakość modelu projektowego**
- **Dobra znajomość środowiska implementacji**
- **Zachowanie przyjętych standardów,** np. konsekwentne stosowanie notacji i formularzy.
- **Sprawdzenie poprawności projektu** w ramach zespołu projektowego
- **Optymalizacja projektu** we właściwym zakresie. Powinna być ograniczona do istotnych, krytycznych miejsc
- **Poddanie projektu ocenie przez niezależne ciało** oceniające jego jakość pod względem formalnym i merytorycznym.

Rezultaty fazy projektowania

- **Poprawiony dokument opisujący wymagania**
- **Poprawiony model analityczny**
- **Uszczegółowiona specyfikacja projektu**
- **Dokument opisujący projekt**:
  - *Diagramy: klas, interakcji obiektów, stanów, modułów, konfiguracji*
  - *zestawień zawierających:*
    - *Definicje: klas, atrybutów, danych złożonych i elementarnych, metod*
- **Zasoby interfejsu użytkownika** *(menu, dialogi)*
- **Projekt bazy danych**
- **Projekt fizycznej struktury systemu**
- **Poprawiony plan testów**
- **Harmonogram fazy implementacji**

Dokument Detali Projektu — DDP

<u>Informacje organizacyjne:</u>
Streszczenie (max 200 słów),
spis treści,
Status dokumentu (autor, firmy, daty,..),
zmiany względem poprzedniej wersji

Cześć 1 — Opis ogólny

1. Wprowadzenie

- 1.1 Cel
- 1.2 Zakres
- 1.3 Definicje, akronimy i skróty
- 1.4 Referencje i odsyłacze do innych dokumentów
- 1.5 Krótkie omówienie

Dokument Detali Projektu — DDP (Cd)

2. Standardy projektu, konwencje, procedury

2.1 Standardy projektowe

2.2 Standardy dokumentacyjne

2.3 konwencje nazewnicze

2.4 Standardy programistyczne

2.5 Narzędzia

Dokument wymagań na oprogramowanie (Cd)

Cześć 2 — Specyfikacja komponentów

Identyfikacja komponentu

1 Typ

2 Cel

3 Funkcja

4 Komponenty podporządkowane

5 Zależności

6 Interfejsy

7 Zasoby

8 Odsyłacze

9 Przetwarzanie

10 Dane

Dodatek A. Wydruki kodu źródłowego

Dodatek B. Macierz zależności miedzy wymaganiami, a komponentami

Przesłanki oceny jakości DDP

- **DDP** — pełna wiedza o kształcie projektu
- Modyfikowalność dokumentu
- Ewolucja dokumentu — rygorystyczna kontrola
- Odpowiedzialność za dokument — osobista, jednoosobowa
- Medium — jednolita elektroniczna wersja wzorcowa
- Sprawdzenie poprawności i spójności modelu

KONIEC