Wykład 7

1. Faza analizy wytwarzania oprogramowania

- Warstwy i mechanizmy architektoniczne
- Klasy analizy; Realizacja przypadków użycia

3. Faza projektu wytwarzania oprogramowania

- Projektowanie architektury oprogramowania
- Wzorce projektowe
- Projektowanie bazy danych

Dyscypliny wytwórcze dla oprogramowania

dyscypliny

modelowanie biznesowe

specyfikacja wymagań

analiza i projektowanie

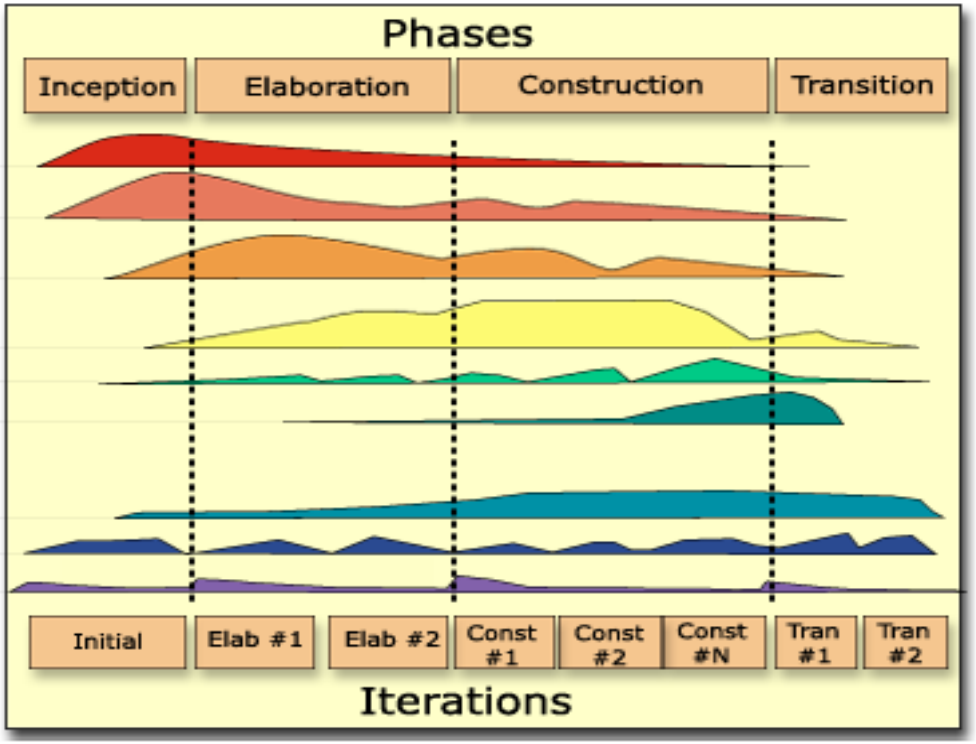
implementacja

testowanie

zarządzanie konfiguracją

zarządzanie projektem

środowisko



ANALIZA a Specyfikacja wymagań

Model przypadków użycia	**Model analizy**
Wyrażony w języku klienta	Wyrażony w języku wytwórcy systemu
Zewnętrzny widok systemu	Wewnętrzny widok systemu
Strukturalizowany przez przypadki użycia; nakłada strukturę na widok zewnętrzny systemu	Strukturalizowany przez stereotypowe klasy i pakiety; nakłada strukturę na widok wewnętrzny systemu
Początkowo używany jako kontrakt pomiędzy wytwórcą a klientem	Początkowo używany przez wytwórcę do zrozumienia kształtu, zakresu systemu
Może być nadmiarowy, niespójny itp.	Nie powinien być nadmiarowy, niespójny itp.
Opisuje funkcjonalność systemu	Wyjaśnia, jak system realizuje daną funkcjonalność; służy jako pierwsze przybliżenie projektu
Definiuje przypadki użycia, które są później analizowane w ramach modelu analizy	Definiuje realizację przypadków użycia

Analiza architektoniczna

Cele:

- Definicja architektury kandydującej systemu w oparciu o doświadczenia w wytwarzaniu podobnych systemów
- Dostarczenie wejść dla procesów planowania

Kroki:

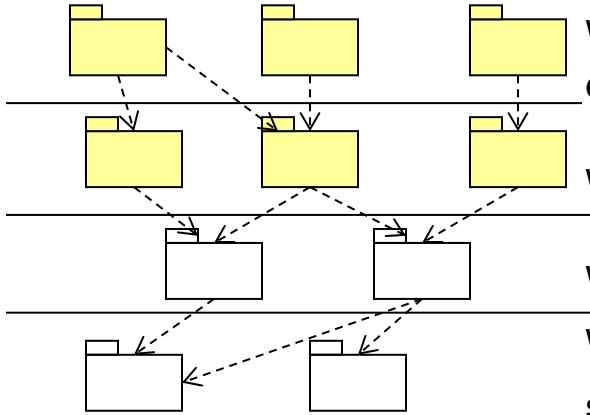
1. Definicja organizacji podsystemów (wysokiego poziomu) – zwykle wykorzystuje się model warstwowy
2. Wytworzenie modelu rozmieszczenia wysokiego poziomu (RUP)
3. Identyfikacja kluczowych abstrakcji
4. Identyfikacja mechanizmów analizy

Architektura SI - definicje

podstawowa organizacja systemu wraz z jego komponentami, wzajemnymi powiązaniami, środowiskiem pracy i regułami ustanawiającymi sposób jej budowy i rozwoju. [*ISO/IEC 42010:2007*]

Architekturę oprogramowania danego systemu można traktować jako **kolekcję komponentów obliczeniowych** – lub prościej komponentów – wraz z **opisem interakcji pomiędzy tymi komponentami**

Architektura logiczna SI - charakterystyka



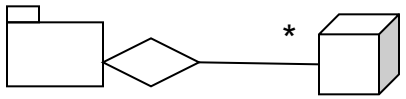
Warstwa specyficzna dla aplikacji

Warstwa ogólna

Warstwa pośrednia

Warstwa oprogramowania systemowego

Istniejące oprogramowanie/narzędzia:
- systemów operacyjnych
- systemów baz danych
- oprogramowanie komunikacyjne
- technologie rozpraszania obiektów

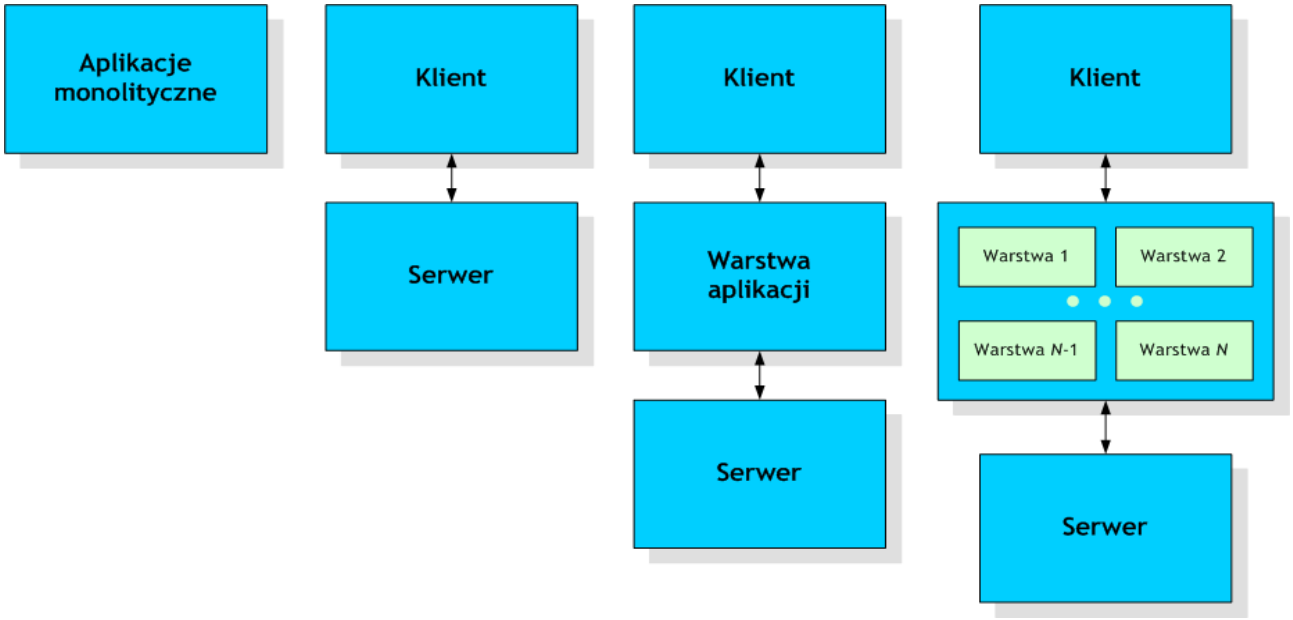


Model rozmieszczenia

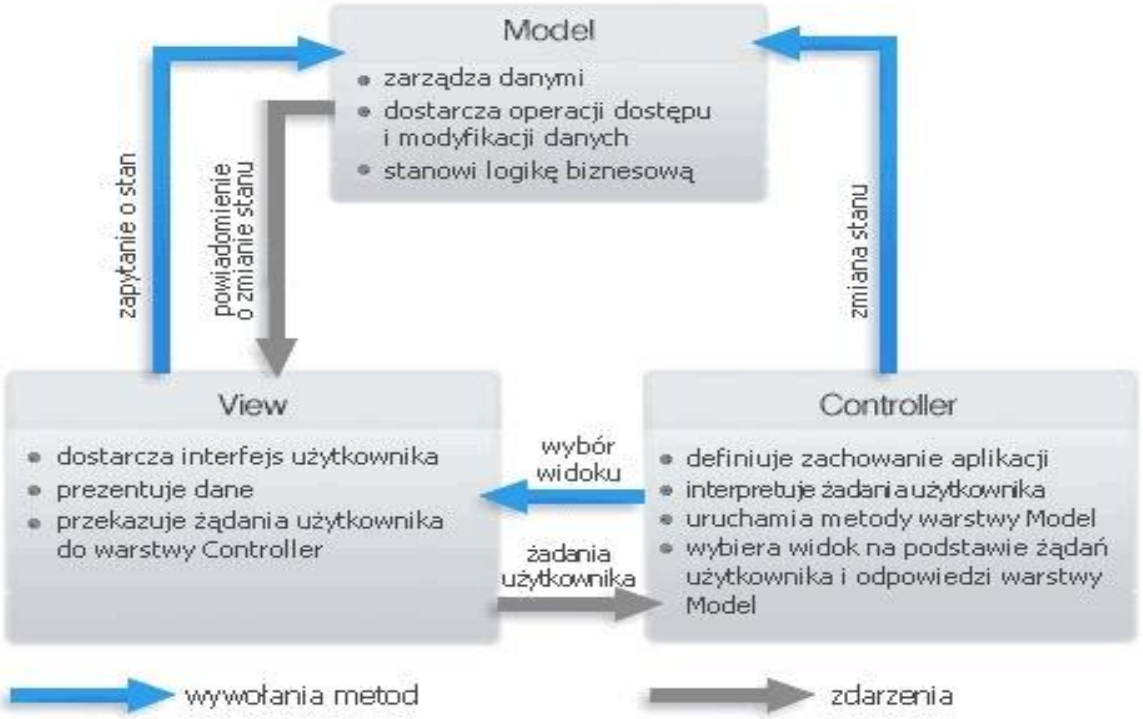
Węzeł

Wizja architektury fizycznej:
- Aplikacja jednostanowiskowa
- Aplikacja wielowarstwowa (tiers)
- Architektura heterogeniczna
- Architektura agentowa

Architektura systemu informatycznego - kategorie

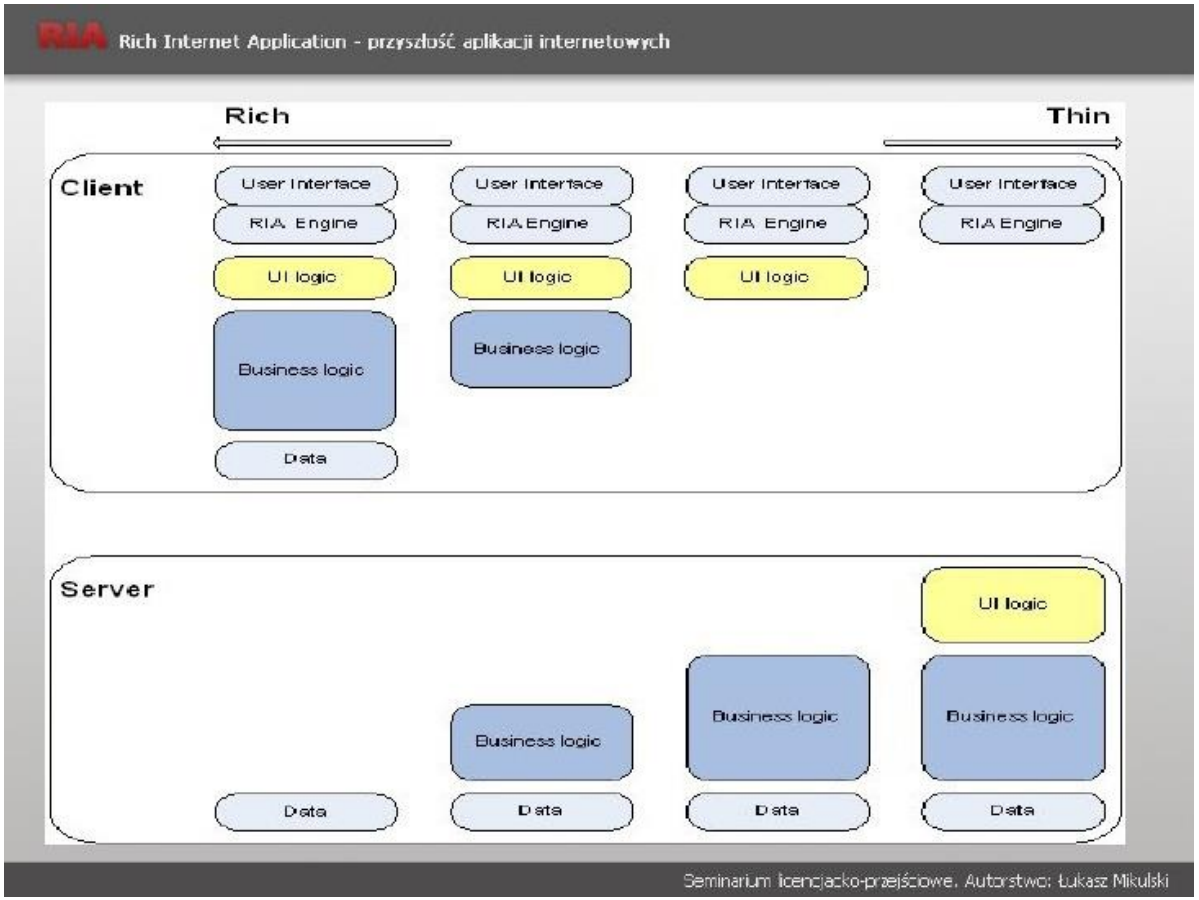


Architektura SI – wzorzec Model-View-Controller (MVC)



organizacja struktury aplikacji - podział na 3 części wzajemnie zależne; Wykorzystywany w innych wzorcach

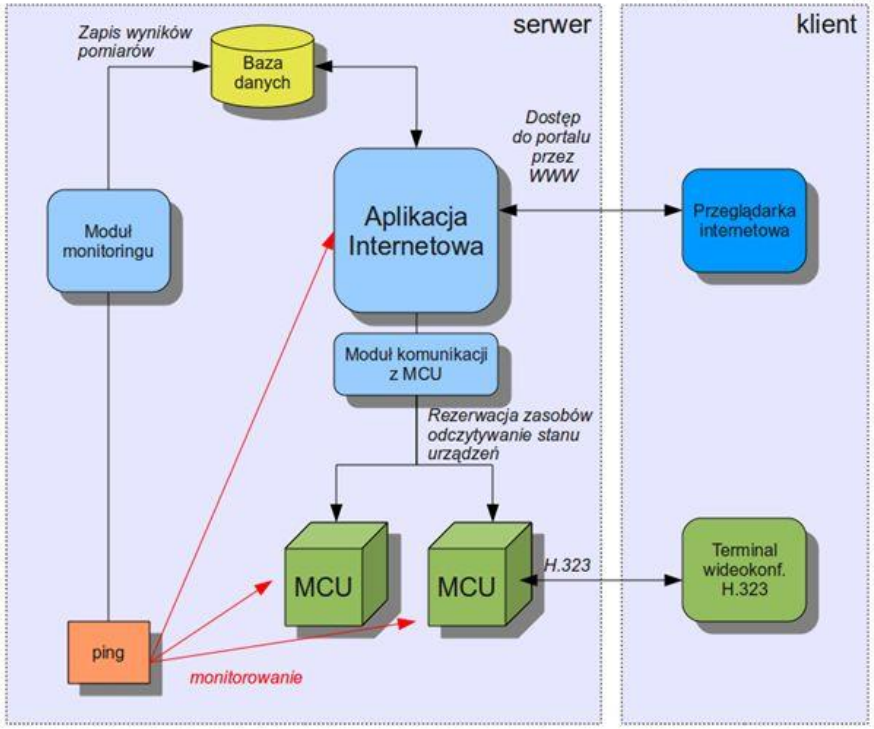
Schematy architektur logicznych dla aplikacji internetowych (wzorzec klient –serwer)



Przykład architektury klient-serwer

www. platon.pionier.net.pl PLATFORMA OBSŁUGI NAUKI PLATON

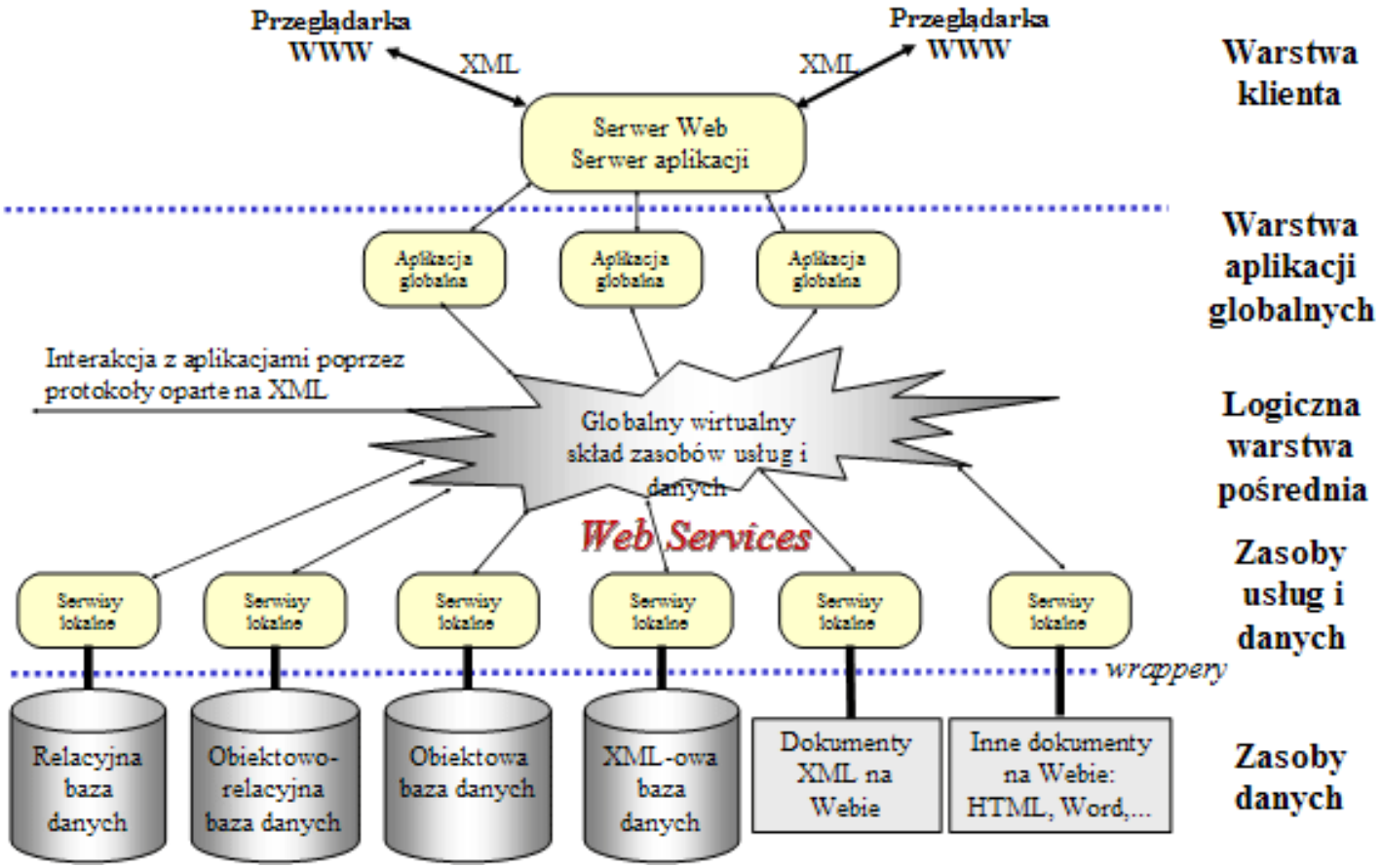
Portal – architektura aplikacji



INNOWACYJNA GOSPODARKA
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ ROZWOJU REGIONALNEGO

Przykład rozbudowanej architektury klient-serwer

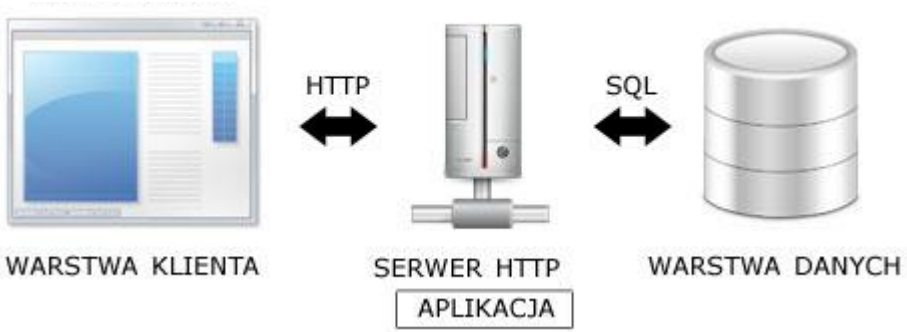


Architektury logiczne - przykłady

Architektura aplikacji internetowych

- **Architektura trójwarstwowa** (*3-tier*)

Warstwa prezentacji	• Interfejs użytkownika • Prezentacja wyników
Warstwa aplikacji	• Logika biznesowa • Przetwarzanie danych • Kontrola przepływu informacji
Warstwa danych	• Bazy danych • Składowanie i pobieranie danych



Analiza architektoniczna - mechanizmy

• **Mechanizm analizy reprezentuje wzorzec**, który stanowi rozwiązanie powszechnego problemu.

• **Mechanizmy analizy są używane** do reprezentacji umiejscowienia złożonej technologii w warstwach pośrednich (middleware) oraz oprogramowania systemowego.

• **Zwykle są niezwiązane** z dziedziną problemu.

Przykładowe mechanizmy dotyczą trwałości danych, komunikacji pomiędzy procesami, obsługi błędów, powiadomień, zarządzania transakcjami, bezpieczeństwa itp.

Np. charakterystyk dotyczących opisu trwałości danych:
o Wielkość (Granularity)
o Liczbę (Volume)
o Okres przechowywania (Duration)
o Mechanizm odtwarzania (Retrieval mechanism)
o Częstość modyfikacji (Update frequency)
o Pewność (Reliability)

Analiza architektoniczna – przykładowe mechanizmy

Mechanizm architektoniczny	**Opis**
Obsługa błędów (ang. *error management*)	Wykonuje operacje niezbędne do powrotu systemu do normalnego stanu po wystąpieniu przewidywalnej sytuacji, która zmienia normalny przepływ wykonania programu.
Zarządzanie zdarzeniami (ang. *event management*)	Pozwala oddzielnym częściom aplikacji na wzajemną interakcję poprzez wymianę asynchronicznych wiadomości.
Licencjonowanie (ang. *licensing*)	Pozwala nabyć licencję, a także śledzi i monitoruje sposób jej wykorzystywania.
Poczta (ang. *mail*)	Wspomaga wymianę wiadomości e-mail pomiędzy aplikacjami i zewnętrznymi systemami.
Zarządzanie wiadomościami (ang. *messaging*)	Pozwala procesom na wymianę wiadomości podczas wykonywania programu.
Pomoc online (ang. *online assistance*)	Wspomaga użytkownika w formie pomocy online, powiadomień pojawiających się po najechaniu na element myszką, pomocy kontekstowej itd.
Trwałość danych (ang. *persistence*)	Umożliwia aplikacji na zapis danych na trwałych nośnikach.
Drukowanie (ang. *printing*)	Umożliwia aplikacji wyświetlanie informacji na wielu rodzajach urządzeń, takich jak drukarki i plottery.
Zarządzanie zasobami (ang. *resorce management*)	Zarządza zasobami systemowymi takimi jak czas procesora oraz pamięć.
Bezpieczeństwo (ang. *security*)	Uwierzytelnia i autoryzuje dostęp do funkcjonalności systemu.
Zarządzanie transakcjami (ang. *transcation management*)	Zarządza przetwarzaniem informacji w niepodzielnych, pojedynczych operacjach, zwanych transakcjami. Każda transakcja musi być atomowa (ang. *atomic*), spójna (ang. *consistent*), izolowana (ang. *isolated*) oraz trwała (ang. *durable*) – ACID.
Przepływ zadań (ang. *workflow*)	Służy do definicji zadań i określa porządek i warunki ich wykonania.

ANALIZA przypadków użycia

Cele:

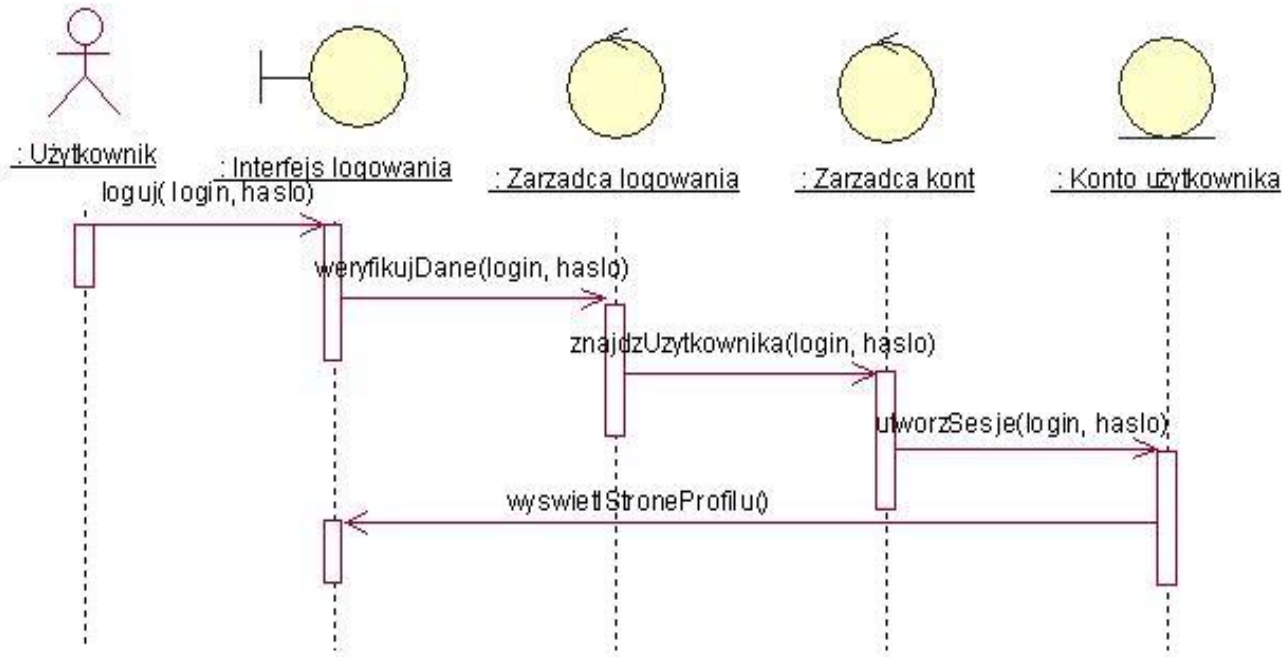
- Identyfikacja klas analizy, które będą odpowiedzialne za realizację przypadku użycia
- Odkrycie wymagań dodatkowych związanych z przypadkiem użycia

Kroki:

1. Identyfikacja klas analizy
2. Opisanie interakcji obiektów analizy
3. Odkrywanie wymagań dodatkowych

ANALIZA – realizacja przypadku użycia

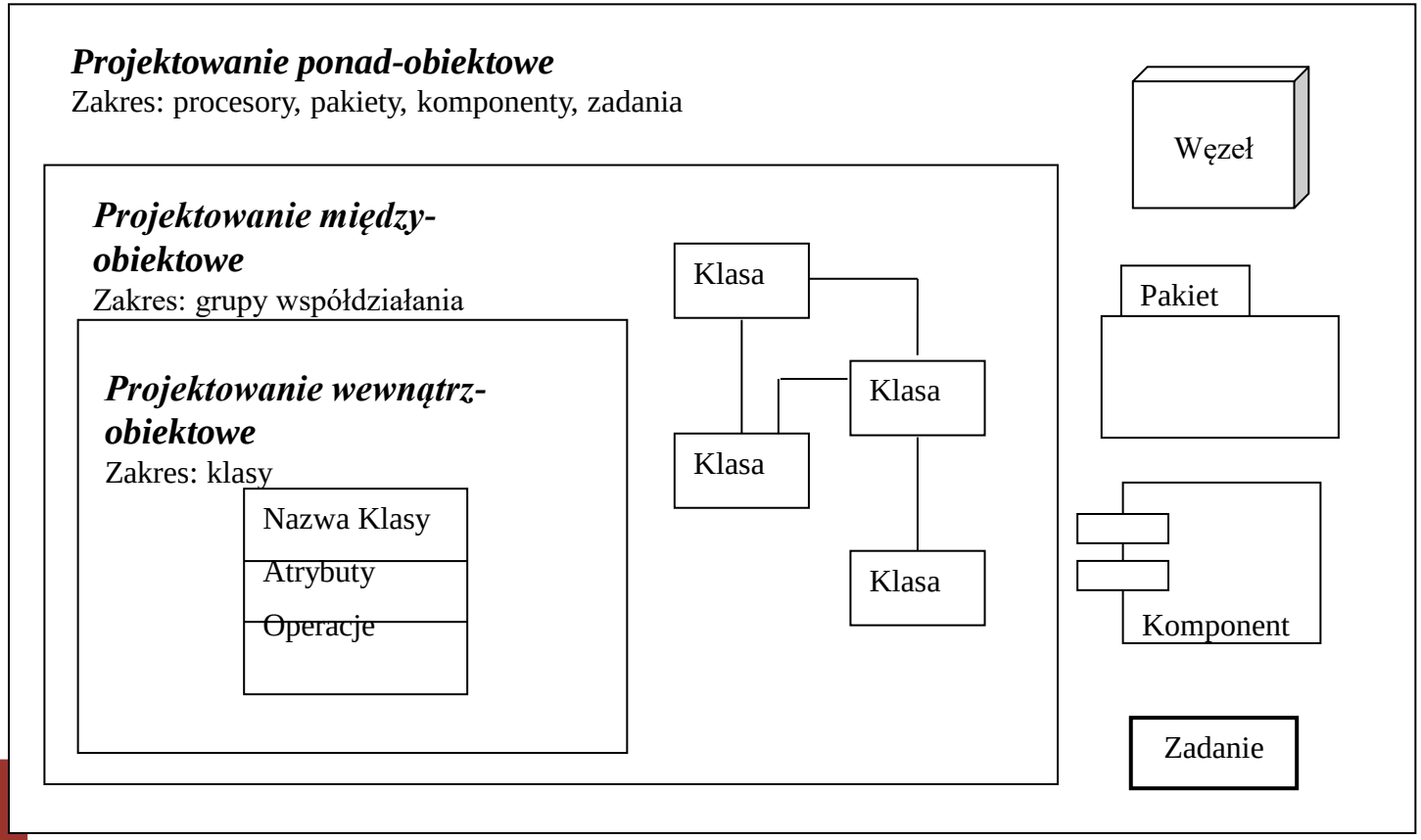
(zachowanie –diagram sekwencji)



PROJEKT a Analiza

Model analizy	**Model projektowy**
Model konceptualny; abstrakcja systemu, pomijająca szczegóły implementacyjne	Model fizyczny; szczegółowy plan implementacji (blueprint of the implementation)
Generyczny; rozumiany jako specyfikacja wielu projektów	Specyficzny dla danej implementacji
Dopuszcza użycie trzech predefiniowanych stereotypów klas	Dopuszcza użycie dowolnej liczby stereotypów klas (liczba jest ograniczona jedynie językiem implementacji)
Architektura zdekomponowana na niewiele poziomów	Architektura zdekomponowana na wiele poziomów
Nie musi być pielęgnowany w dalszych fazach	Powinien być pielęgnowany w dalszych fazach
Definiuje bazową strukturę do zarysowania systemu	Zakreśla system przy maksymalnym zachowaniu architektury wypracowanym w modelu analizy

Projektowanie - etapy



PROJEKT - *Projektowanie architektury*

Celem projektowania architektury jest podanie modeli:

projektowego i rozmieszczenia, oraz ich architektury.

Można to uzyskać dzięki identyfikacji:

Architektury fizycznej

- Węzłów i ich sieciowej konfiguracji,

Architektury logicznej

- Podsystemów i ich interfejsów,
- Architektonicznie znaczących klas projektowych np. klas aktywnych,
- Generycznego mechanizmu projektowego, który obsłuży wspólne wymagania np. specjalne wymagania na trwałość, rozproszenie, wydajność i inne, określone w fazie analizy klas i przypadków użycia z modelu analizy(-> wymagania niefunkcjonalne!!)

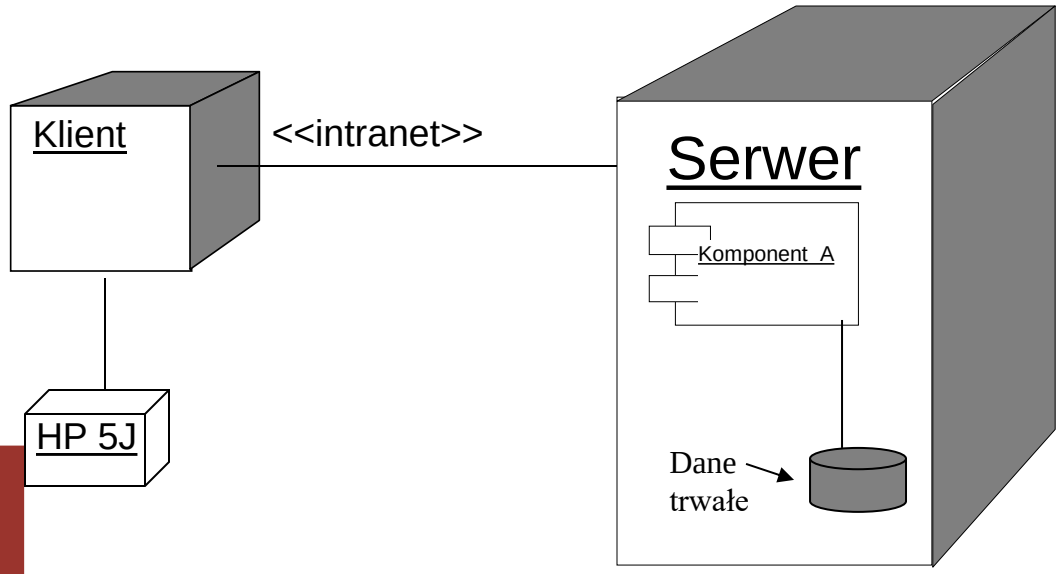
Projekt architektury - poziom logiczny

Wykorzystuje się diagram rozmieszczenia

Przedstawia konfigurację węzłów (przetwarzających) oraz umieszczonych w nich komponentów; odzwierciedla statyczny aspekt perspektywy instalacyjnej

Elementy

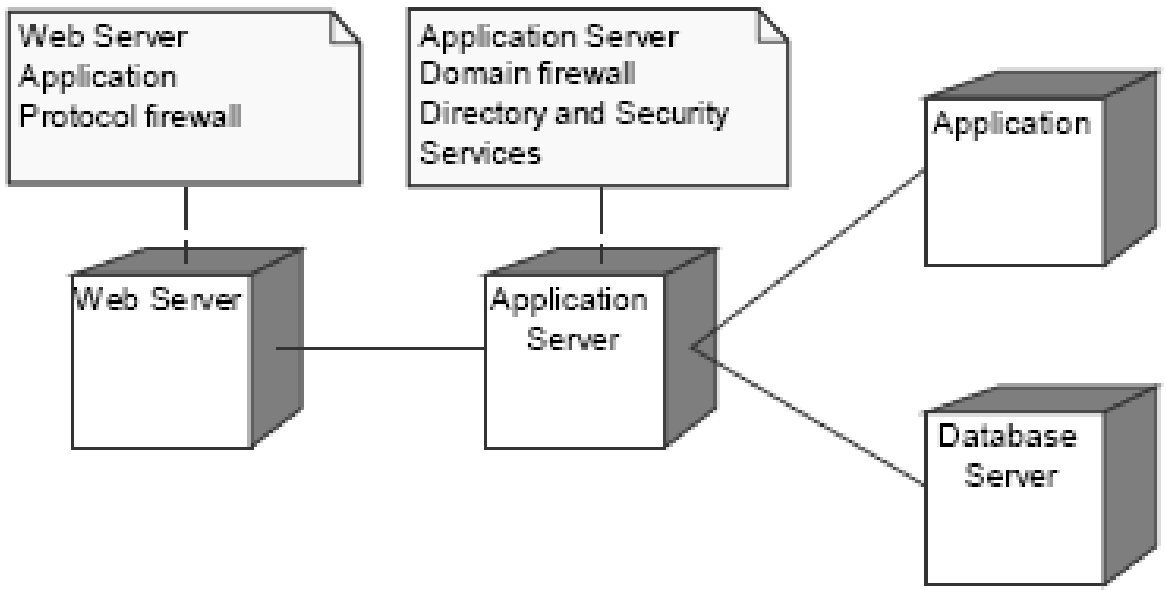
węzły; komponenty; połączenia pomiędzy węzłami



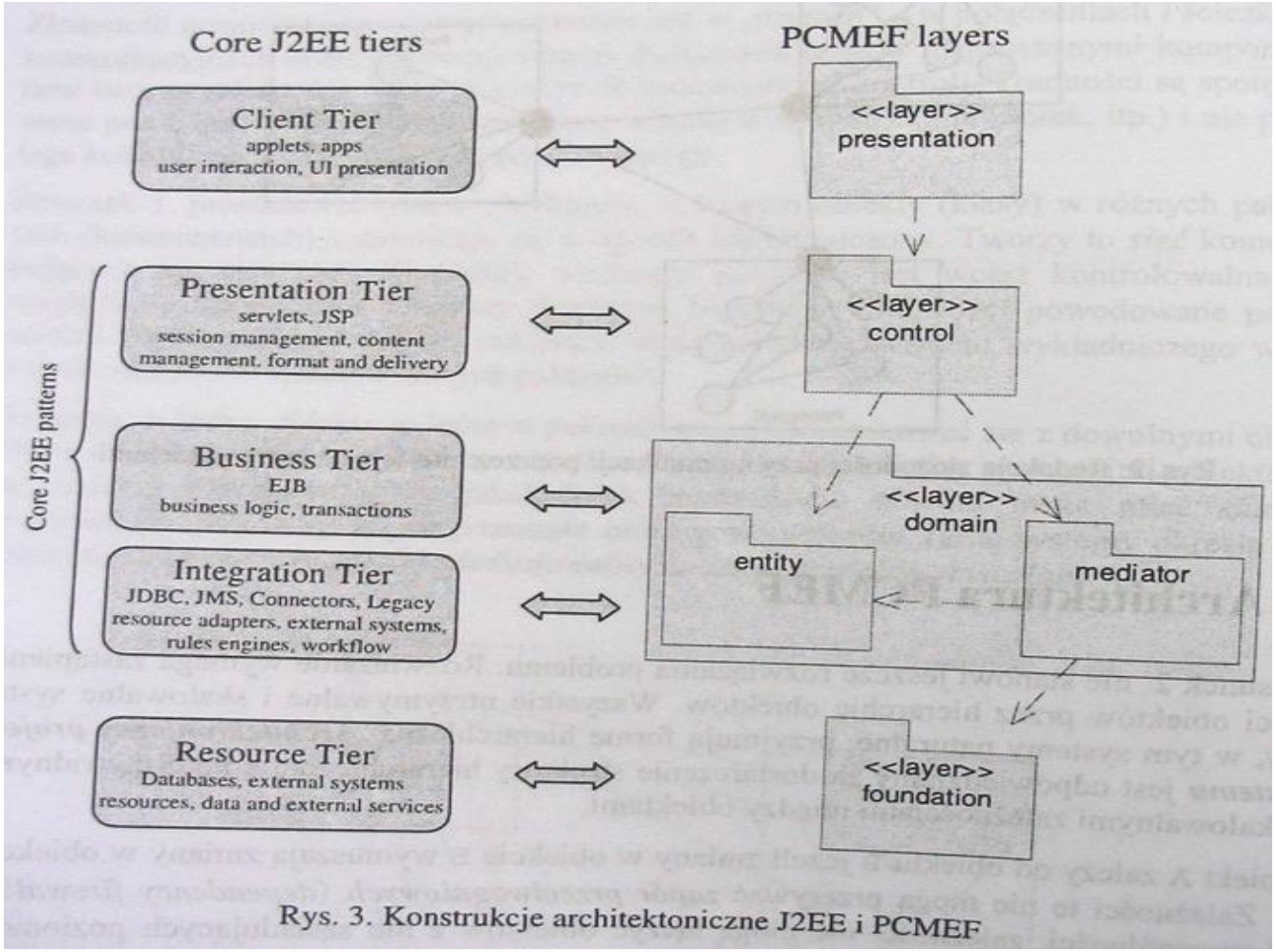
Projekt architektury - poziom fizyczny

ARCHITEKTURA FIZYCZNA – ITERACJA 2

- Wskazanie usług i aplikacji, alokowanych do węzłów
- Wstępna charakterystyka połączeń, np. nazwa protokołu, przepustowość

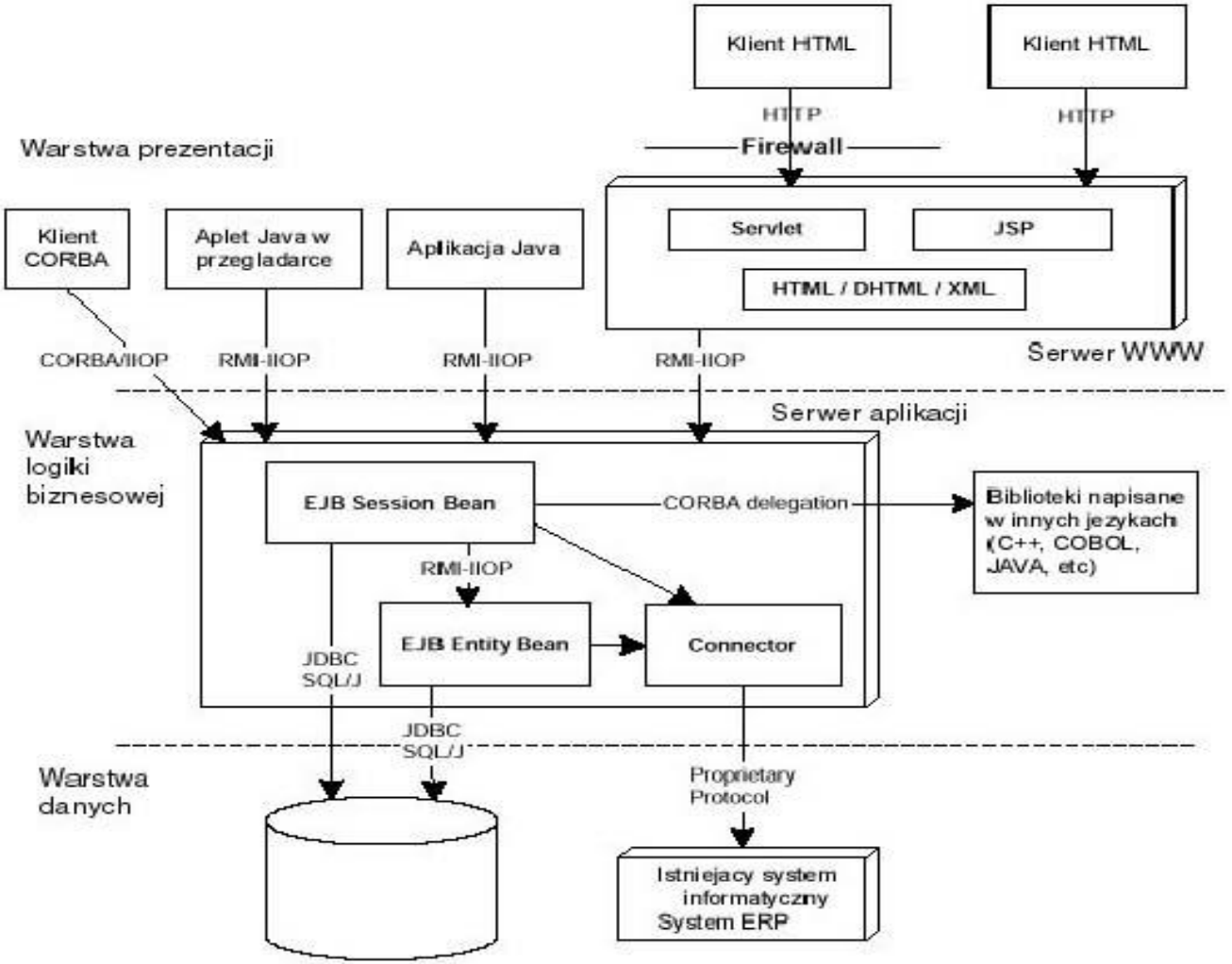


Architektura analizy vs. projektowa



Rys. 3. Konstrukcje architektoniczne J2EE i PCMEF

Przykład architektury projektowej



PROJEKT - *Projektowanie przypadku użycia*

Celem projektowania PU jest :

- **identyfikacja klas/podsystemów, które uczestniczą w PU**
- **rozdzielenie zachowania między klasy/podsystemy**
- **zdefiniowanie wymagań dotyczących operacji klas/podsystemów**
- **rozpoznanie wymagań implementacyjnych dla PU**

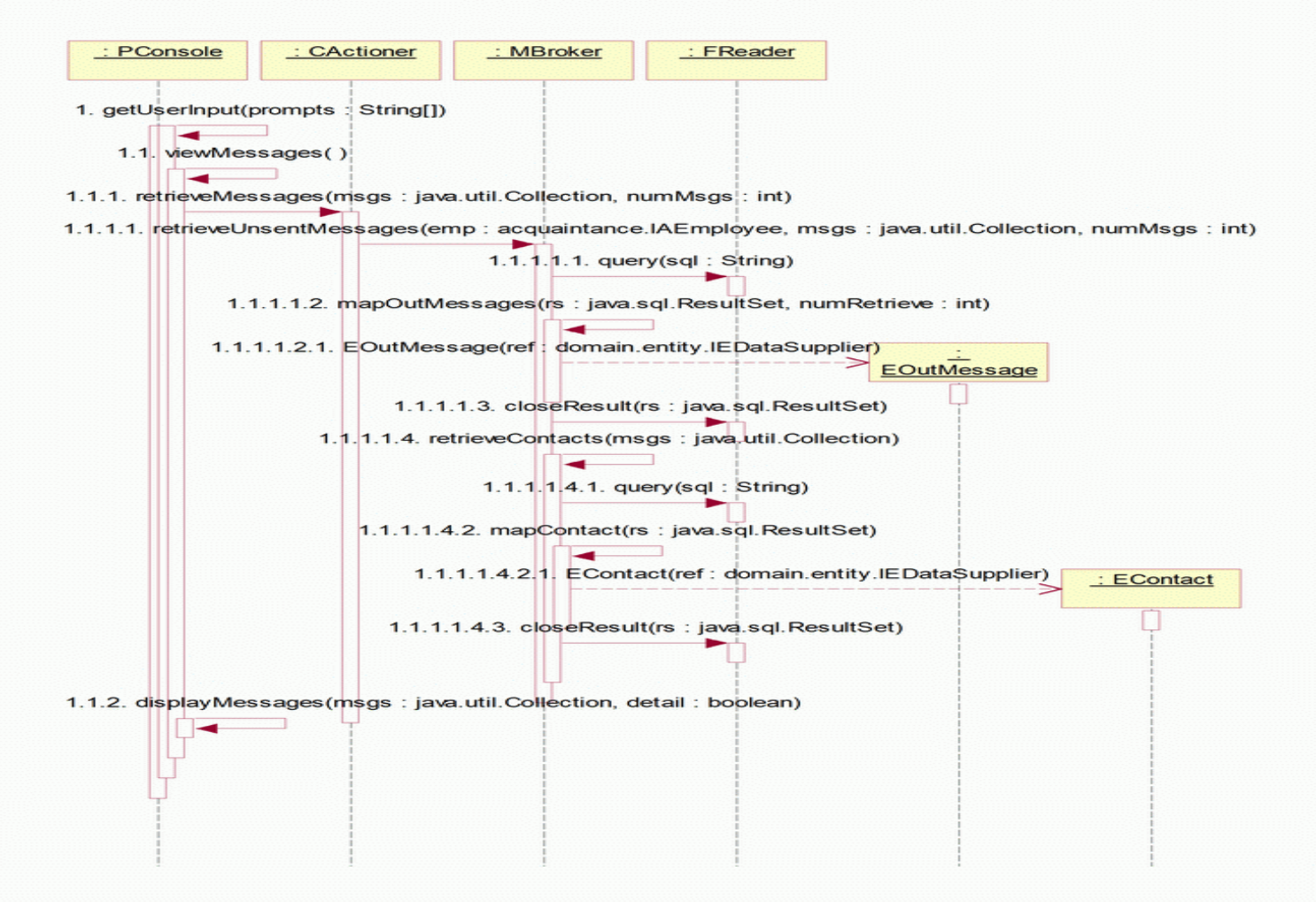
Etapy :

- **identyfikacja klas projektowych** (z analizy uzupełnione o klasy dla wymagań niefunkcjonalnych);

- **opis interakcji obiektów (**diagram sekwencji poszerzony o nowe klasy; nazwa wiadomości staje się nazwą operacji);

- **rozważenie ścieżek alternatywnych**: timeouty, błędy we, błędy systemów spadkowych.

PROJEKT - *Projektowanie przypadku użycia (przykład)*



PROJEKT - wzorce projektowe

Cel: **tworzenie projektu wielokrotnego użytku**

Wzorzec projektowy - opisanie istoty rozwiązania problemu (spotykanego często w otoczeniu) w sposób, który umożliwia **wielokrotne wykorzystywanie** tego rozwiązania - niekoniecznie w ten sam sposób.
Na Wzorzec projektowy składają się cztery elementy:

- nazwa wzorca,
- problem,
- sposób rozwiązania,
- konsekwencje stosowania wzorca.

Wzorce projektowe dotyczą pewnego poziomu abstrakcji problemu tzn. nie są wzorcami budowy list, drzew itp.; nie są też wzorcami budowy całych aplikacji/podsystemów (np. bankowego, sterowania procesem itp.)

Wzorce projektowe - definicje

Wzorzec projektowy identyfikuje i specyfikuje abstrakcję ‘wyższą’ niż klasa, instancja czy nawet komponent (Gamma, 1993)

Wzorce projektowe są opisem komunikujących się obiektów i klas dostosowanych do rozwiązania problemu projektowego w konkretnym kontekście (Gamma 1995)

Wzorce projektowe stanowią zbiór reguł określających jak osiągnąć konkretne cele w dziedzinie wytwarzania oprogramowania (Pree, 1994)

Wzorzec stanowi rozwiązanie powtarzających się problemów projektowych (Bushmann, 1996)

Wzorzec ~ przepis na upieczenie ciasta

Wzorce projektowe - klasyfikacja

- **Kreujące** – przejmują na siebie tworzenie nowych instancji obiektów, ukrywając rodzaj obiektów (singleton, proxy, fabryka abstrakcyjna)
- **Strukturalne** – wspomagają organizowanie obiektów w duże struktury, np. UI czy operacje na danych (fasada, adapter, most, dekorator, kompozycja)
- **Behawioralne** – wspomagają definiowanie komunikacji i przepływu zdarzeń pomiędzy obiektami (łańcuch odpowiedzialności, mediator, iterator)

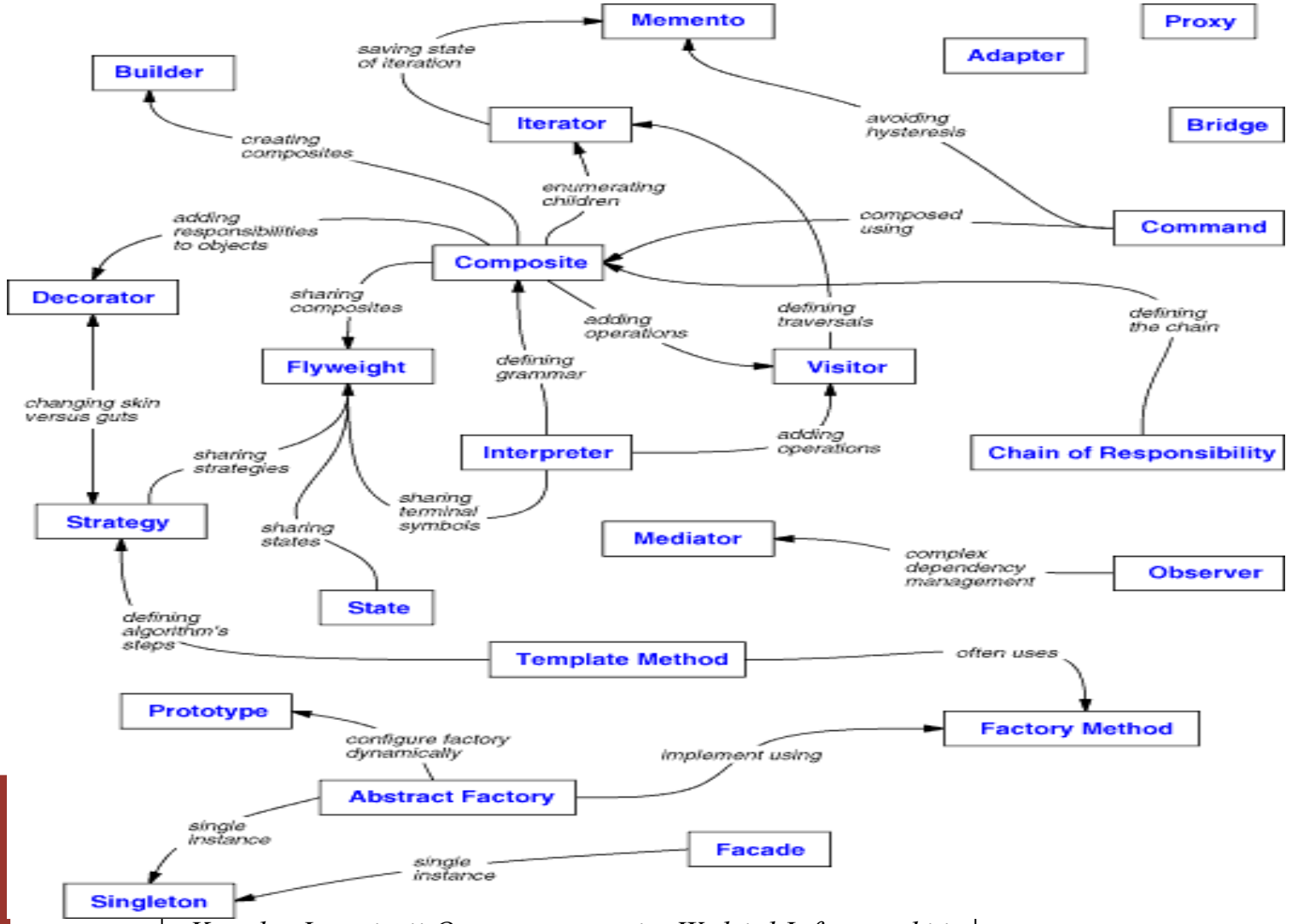
Wzorce projektowe - przykłady

CEL	NAZWA	PROBLEM rozwiązywany
Kreowanie		
	Abstrakcyjna fabryka	kreacja rodziny obiektów-produktów
	Budowniczy	kreacja złożonego obiektu
	Metoda fabryczna	podklasa obiektów, które są instantowane
	Prototyp	klasa obiektów,które są instantowane
	Singleton	pojedyncza instancja klasy
Struktura		
	Adapter	interfejs do obiektu
	Most	implementacja obiektu
	Kompozycja	struktura i kompozycja obiektu
	Dekorator	odpowiedzialność obiektu bez podklas
	Fasada	interfejs do podsystemu
	Ziarnistość obiektu	koszty pamiętania obiektu
	Pośrednik	dostęp do obiektu, jego położenie

Wzorce projektowe - przykłady

CEL Zachowanie	NAZWA	PROBLEM rozwiązywany
	Łańcuch odpowiedzialności	obiekt, który spełnia żądania
	Rozkaz	kiedy i jak żądanie jest spełnione
	Interpreter	gramatyka i interpretacja języka
	Iterator	jak elementy agregatu są dostępne
	Mediator	jak i które obiekty współdziałają
	Memento	jaka prywatna informacja jest przechowywana poza obiektem i kiedy
	Obserwator	uaktualnianie stanupewnej liczby obiektów
	Stan	stany obiektu
	Strategia	algorytm
	Wizytator	operacje, które mogą być stosowane do obiektów bez zmiany ich klasy

Wzorce projektowe - zależności



WZORZEC SINGLETON

Wzorzec Singleton – wzorzec projektowy, który daje pewność że istnieje tylko jedna instancja danej klasy.

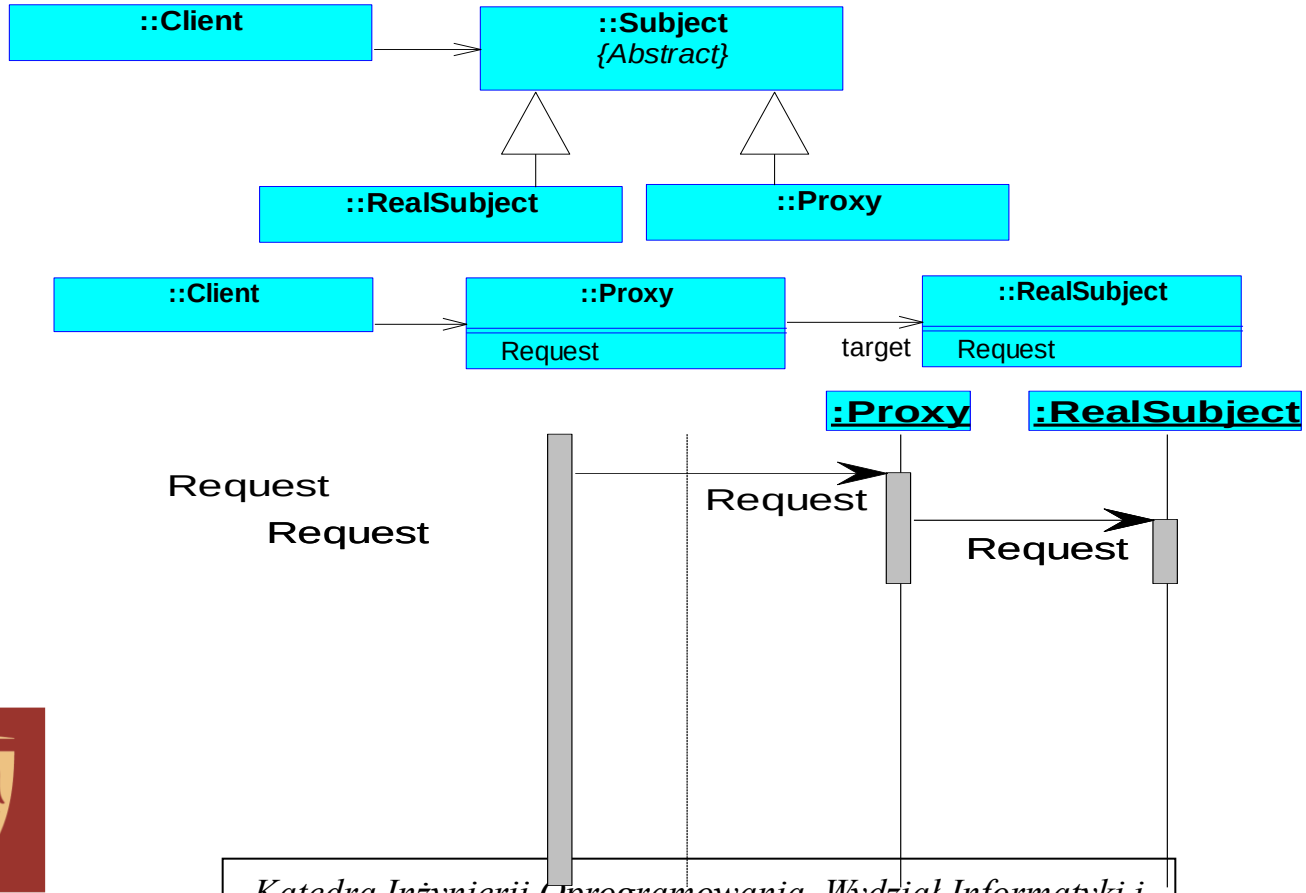
```
final class Singleton{
        private static Singleton ref = new Singleton();

        private Singleton(){}

        static public Singleton getReference(){
                return ref;
        }
   // publiczne metody dostępu do obiektu klasy Singleton
 }
```

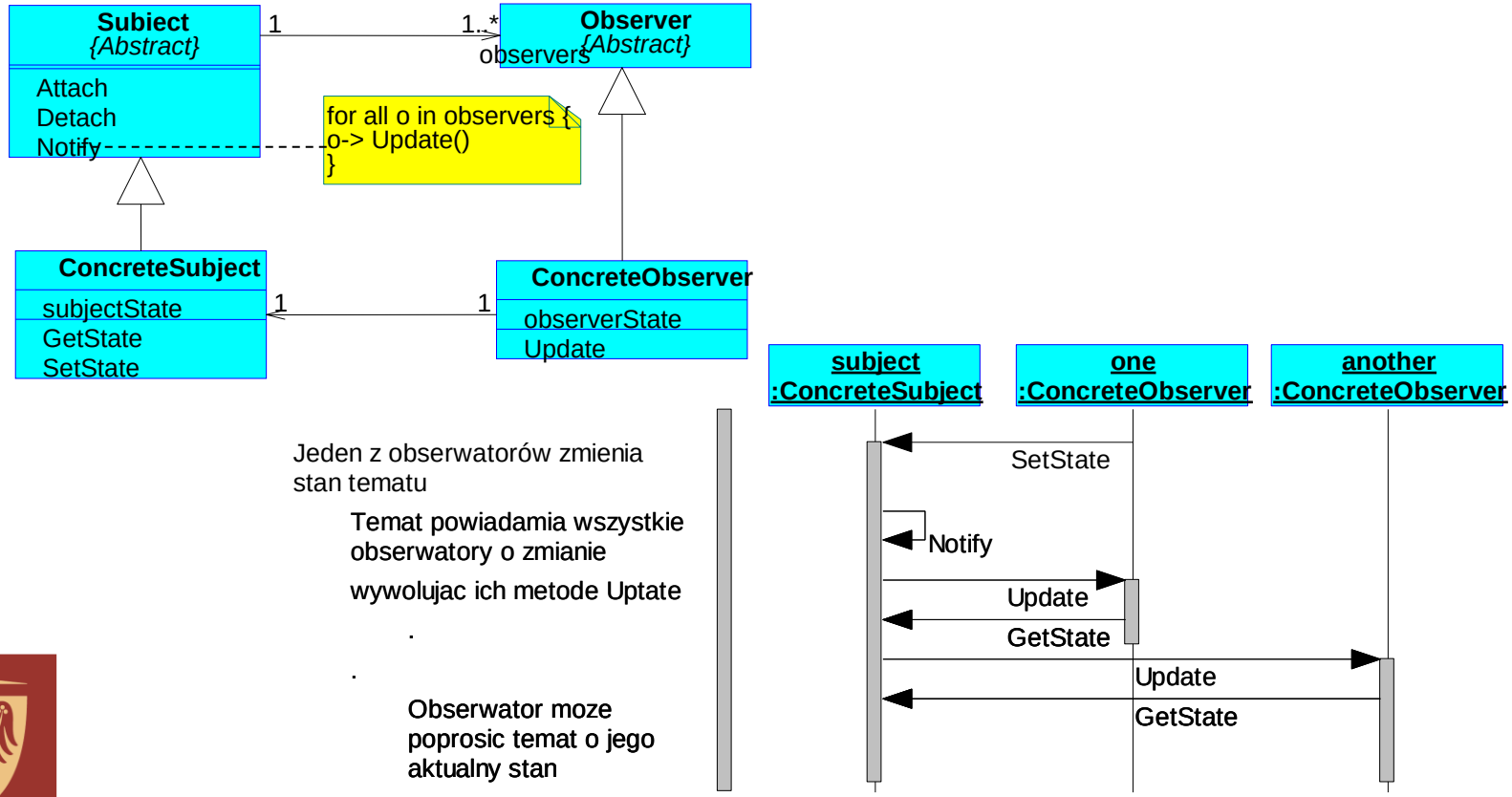
WZORZEC PROXY

Wzorzec Proxy – umożliwia dostęp do obiektu klasy rzeczywistej, poprzez obiekt klasy pośredniczącej (obiekt proxy).



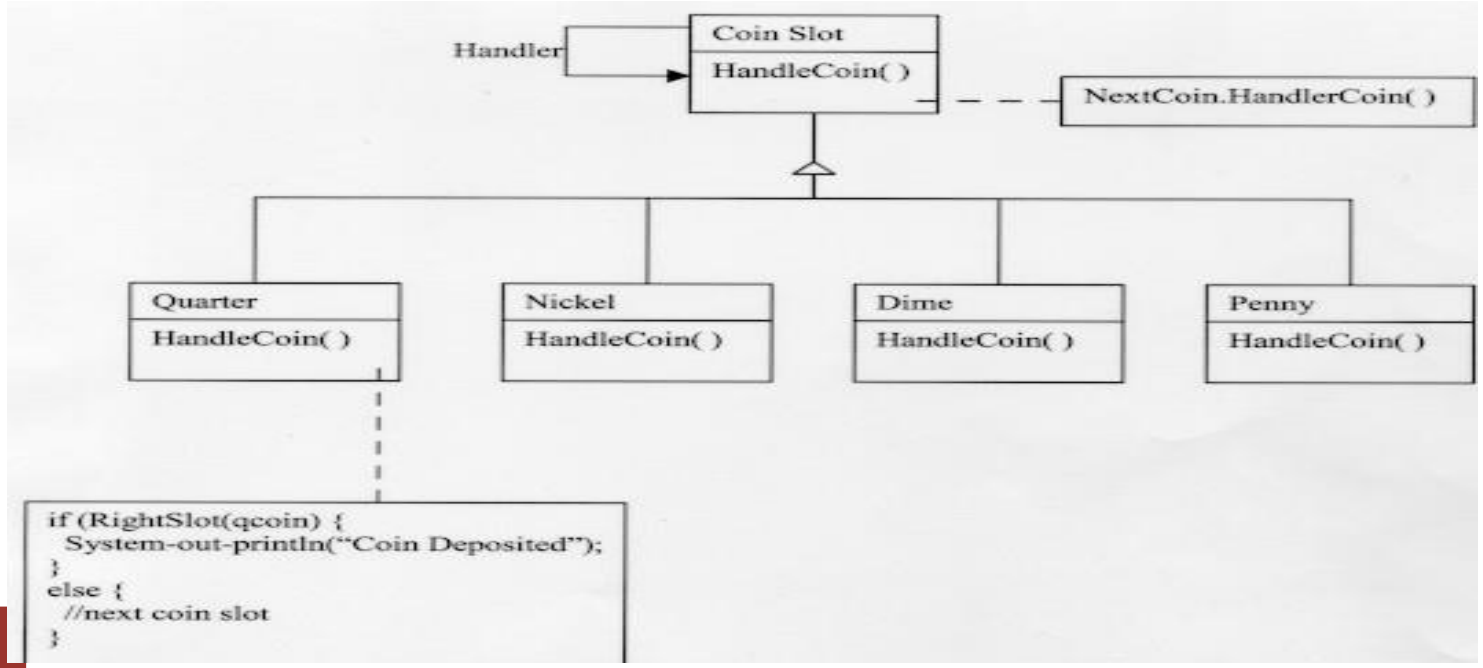
WZORZEC OBSERWATOR

Wzorzec obserwator (observer) definiuje zależność pomiędzy obiektami w ten sposób, że gdy jeden z nich zmienia stan, reszta jest o tym powiada-miana i uaktualniana automatycznie.

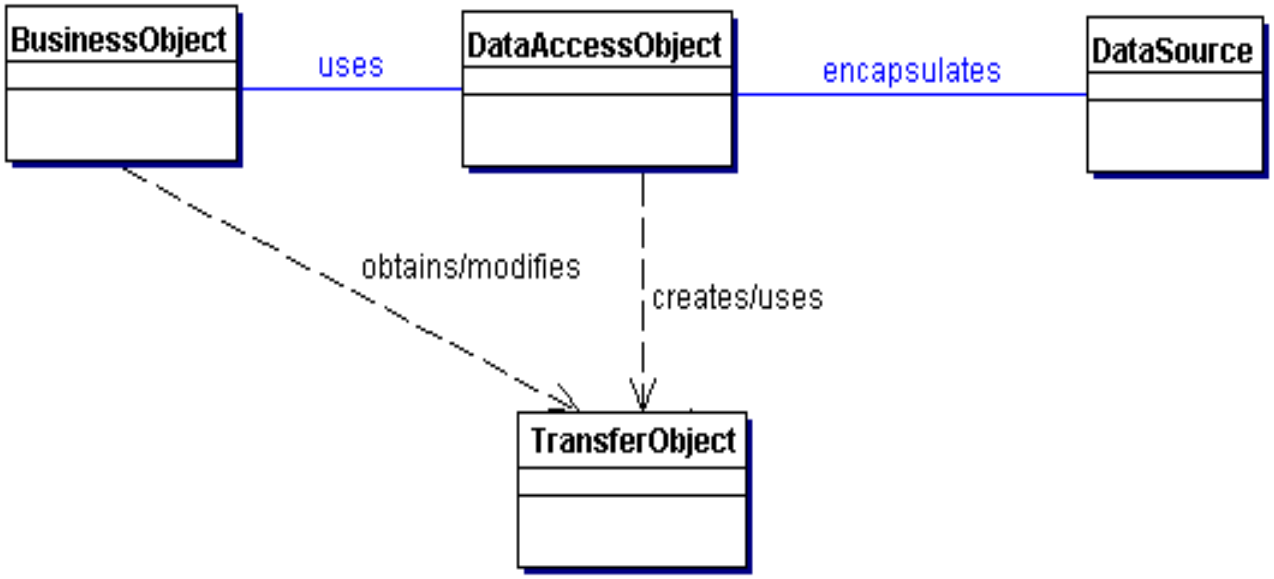


WZORZEC
ŁAŃCUCH ODPOWIEDZIALNOŚCI

Wzorzec Łańcuch odpowiedzialności (Chain of Responsibility) umożliwia różnym obiektom, połączonym w listę, na podjęcie próby obsłużenia pewnego żądania, podczas gdy żaden z nich nic nie wie o możliwościach innych.

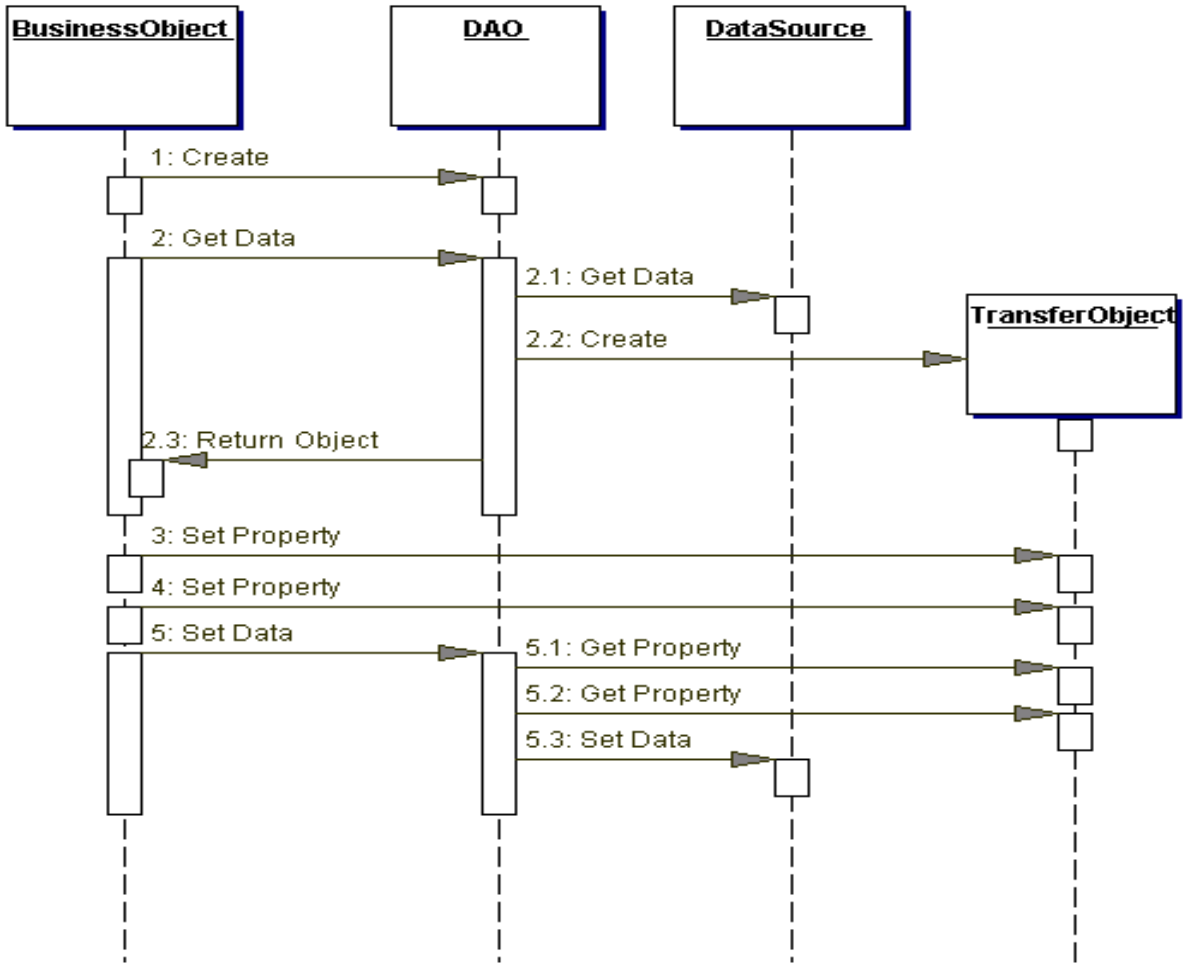


Wzorce projektowe – Data Access Object (DAO)



- jednolity interfejs do komunikacji między aplikacją a źródłem danych (np. bazą danych czy plikiem; często łączony z innymi wzorcami projektowymi.
-często stosowany w modelu Model-View-Controller do oddzielenia dostępu do danych od logiki biznesowej i warstwy prezentacji.

Wzorce projektowe – DAO zachowanie



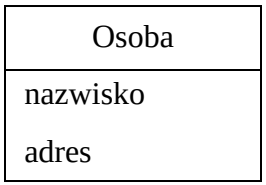
Wzorce projektowe - podsumowanie

- Wzorce projektowe ułatwiają proces modelowania aplikacji obiektowych
- Pozwalają na wykorzystanie gotowych przepisów na rozwiązanie trudnych problemów
- Kluczem do ich stosowania jest znajomość przynajmniej specyfiki problemów z jakimi poszczególne wzorce sobie radzą
- Proces przechodzenia z modelu analizy do modelu projektowego powinien następować z uwzględnieniem użytecznych wzorców projektowych

PROJEKTowanie danych -

odwzorowanie klas

Model obiektowy



Model tabeli

Nazwa atrybutu	Czy pusty?	Zakres
ID-osoby	N	ID
nazwisko	N	nazwisko
adres	T	adres

Kod SQL

```
CREATE TABLE Osoba
            (id-osoby    ID          not null,
            nazwisko    char(30)    not null,
            adres       char(30)    not null,
            PRIMARY KEY (ID-osoba));

CREATE SECONDARY INDEX (Osoba-indeks-nazwisko) ON  Osoba (nazwisko);
```

PROJEKTowanie danych - *odwzorowanie asocjacji*

Asocjacja (n do m) jest odwzorowywana na tabelę

Asocjacja (1 do n) jest odwzorowywana:

- na tabelę lub
- jest reprezentowana przez klucz obcy w tabeli reprezentującej klasę z licznością n

Asocjacja (1 do 1) oraz (1 do n) może być odwzorowana na jedną klasę (jeżeli asocjacje nie tworzą cyklu) reprezentującą zarówno oba końce (klasy) asocjacji, jak i asocjację

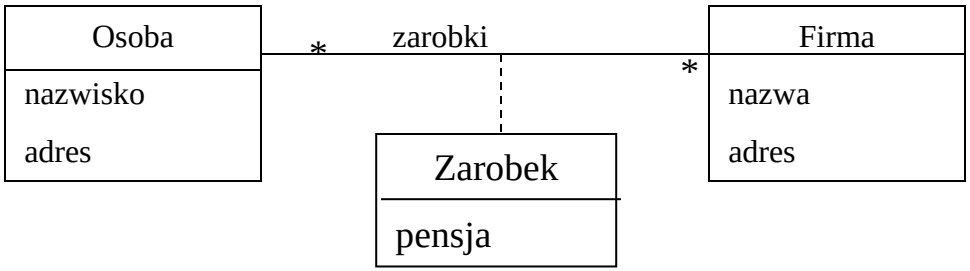
Asocjacja n-arna jest odwzorowywana na tabelę

Asocjacja kwalifikowana jest odwzorowywana na tabelę z przynajmniej trzema atrybutami: 2 klucze główne dla każdego końca asocjacji oraz kwalifikator

Agregacja jest odwzorowywana tak, jak asocjacja

PROJEKTowanie danych - *Odwzorowanie asocjacji*

Model obiektowy



Model tabeli

Zarobek

Nazwa atrybutu	Czy pusty?	Zakres
ID-firmy	N	ID
ID-osoby	N	ID
pensja	T	waluta

Kod SQL

```
CREATE TABLE Zarobki
        (id-firmy      ID          not null,
         id-osoby      ID          not null,
        pensja         char(30)    not null,
        PRIMARY KEY (ID-firma, ID-osoba),
        FOREIGN Key (ID-firma REFERENCES Firma),
        FOREIGN Key (ID-osoba REFERENCES Osoba));
```

PROJEKTowanie danych - *Odwzorowanie generalizacji*

Zasady odwzorowania dla generalizacji prostej:

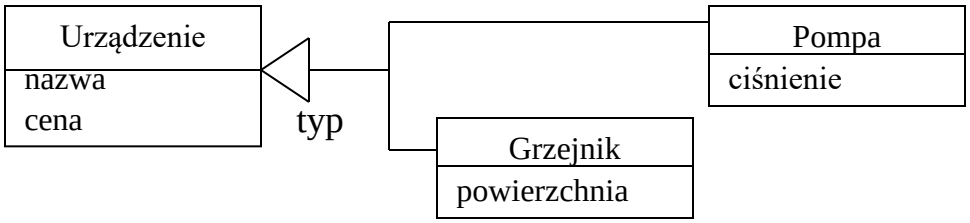
- każda klasa (nadklasa oraz podklasy) są odwzorowywane na tabele lub
- nie ma tabeli dla nadklasy; atrybuty nadklasy są replikowane w każdej tabeli podklasy lub
- nie ma tabeli dla podklasy; wszystkie atrybuty podklas są przeniesione do tabeli nadklasy

Zasady odwzorowania dla generalizacji wielokrotnej:

- dla klas rozłącznych - każda klasa (nadklasy oraz podklasy) są odwzorowywane na tabele lub
- dla klas nierozłącznych - każda klasa (nadklasy oraz podklasy) są odwzorowywane na tabele oraz generalizacja jest odwzorowywana na tabelę

PROJEKTowanie danych - *Odwzorowanie generalizacji*

Model obiektowy



Modele tabel

I dzielony atrybut; łatwe rozszerzanie, wolny dostęp, integralność!?

Sprzęt

ID-sprzętu	N
nazwa	N
cena	T
typ	N

Pompa

ID-sprzętu	N
ciśnienie	T

Grzejnik

ID-sprzętu	N
powierzchnia	T

II W celu przyspieszenia dostępu --> eliminacja nadklasy

Pompa

ID-sprzętu	N
nazwa	N
cena	T
ciśnienie	T

Grzejnik

ID-sprzętu	N
nazwa	N
cena	T
powierzchnia	T

PROJEKT - PODSUMOWANIE

Model projektowy obejmuje:

- architekturę logiczną tzn. podsystemy projektowe, podsystemy usługowe (*service*) wraz z ich interfejsami, zawartością oraz związkami zachodzącymi pomiędzy nimi; widok architektoniczny uwzględnia mechanizmy architektoniczne, w tym wzorce projektowe
- architekturę fizyczną opisującą węzły i i ich sieciową konfigurację (wyrażoną modelem rozmieszczenia)
- realizację projektowych przypadków użycia, wykorzystującą klasy projektowe z uwzględnieniem klas aktywnych; realizacja obejmuje wątki alternatywne wynikające z ograniczonych zasobów oraz błędów interfejsu