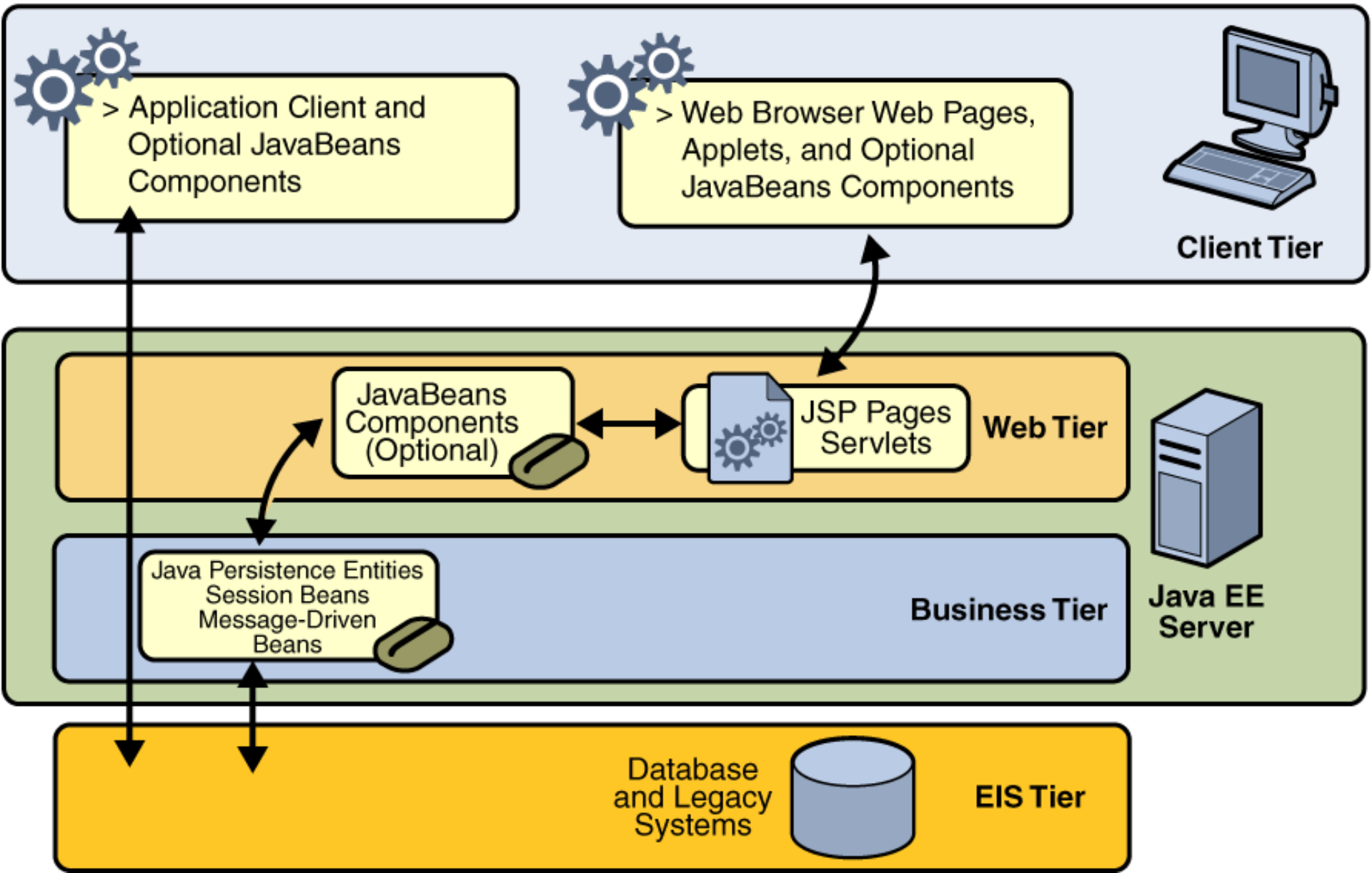
Modelowanie i analiza systemów informatycznych

1. **Warstwowa budowa systemów informatycznych**
2. **Model procesu wytwarzania oprogramowania - model cyklu życia oprogramowania**
3. **Wstęp do modelowania systemów informatycznych**
4. **Środowisko tworzenia oprogramowania**
5. **Podstawy UML**

Modelowanie i analiza systemów informatycznych

1. **Warstwowa budowa systemów informatycznych**

Warstwy aplikacji (Java EE)*



Pięciowarstwowy model logicznego rozdzielania zadań (wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

Warstwa klienta
Klienci aplikacji, aplety, aplikacje i inne elementy z graficznym interfejsem użytkownika

Interakcja z użytkownikiem, urządzenia i prezentacja interfejsu użytkownika

Warstwa prezentacji
Strony JSP, serwlety i inne elementy interfejsu użytkownika

Logowanie, zarządzanie sesją, tworzenie zawartości, formatowania i dostarczanie

Warstwa biznesowa
Komponenty EJB i inne obiekty biznesowe

Logika biznesowa, transakcje, dane i usługi

Warstwa integracji
JMS, JDBC, konektory i połączenia z systemami zewnetrznymi

Adaptery zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania

Warstwa zasobów
Bazy danych, systemy zewnętrzne i pozostałe zasoby

Zasoby, dane i usługi zewnętrzne

Modelowanie i analiza systemów informatycznych

1. **Warstwowa budowa systemów informatycznych**
2. **Model procesu wytwarzania oprogramowania - model cyklu życia oprogramowania**

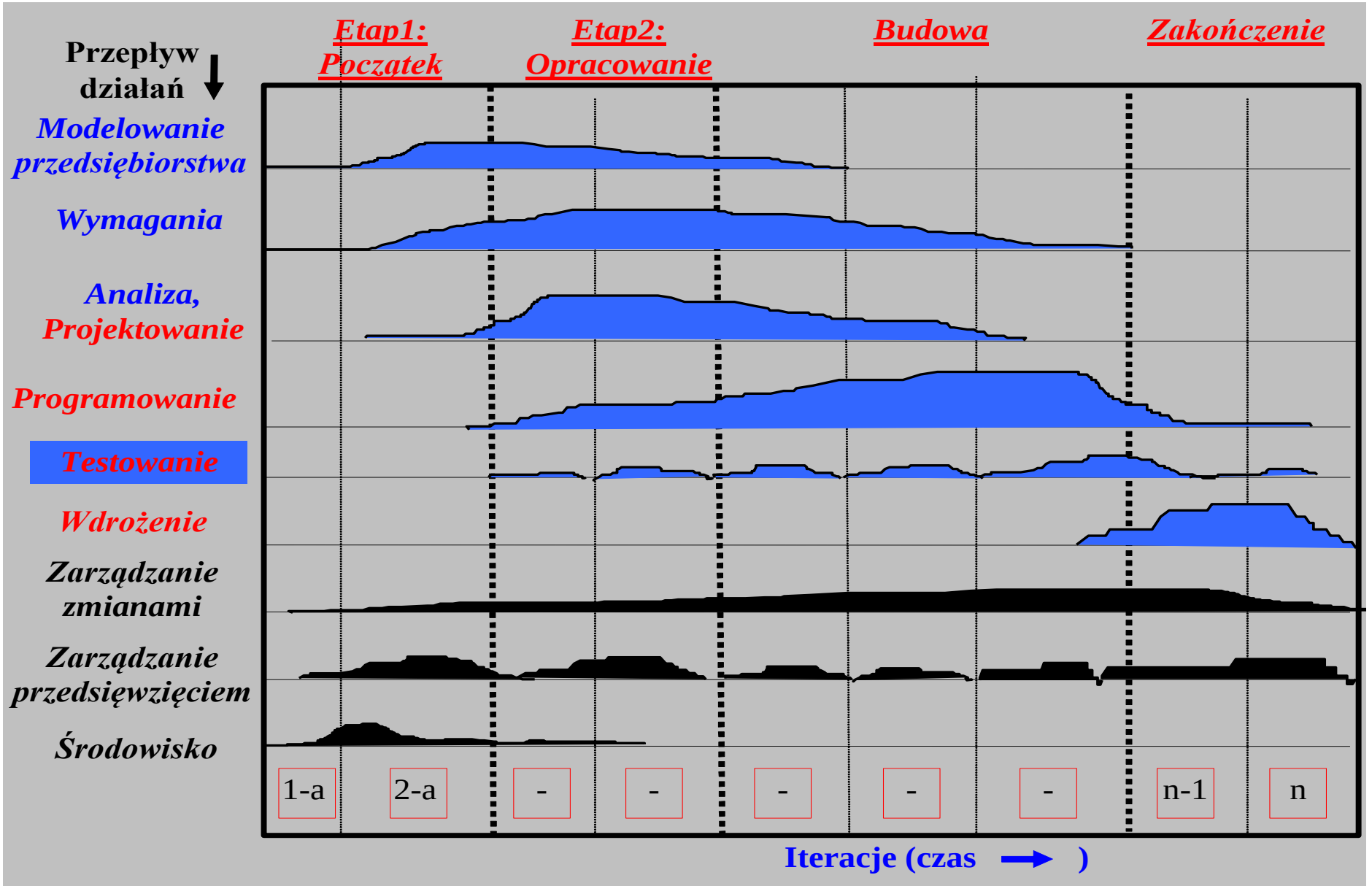
Model procesu wytwarzania oprogramowania - czyli model cyklu życia oprogramowania*

Tworzenie technicznego systemu informacyjnego jest powiązane z:

- budową oprogramowania: co i jak wykonać?
- zarządzaniem procesem tworzenia oprogramowania: kiedy wykonać?
- wdrażaniem oprogramowania

Modelowanie struktury i dynamiki systemu	**Implementacja systemu,**	**struktury i dynamiki generowanie kodu**
Perspektywa koncepcji *co należy wykonać?*	Perspektywa specyfikacji *jak należy używać?*	Perspektywa implementacji *jak należy wykonać?*
• model problemu np. przedsiębiorstwa • wymagania • analiza (model konceptualny) • testy modelu	• projektowanie (model projektowy: architektura sprzętu i oprogramowania; dostęp użytkownika; przechowywanie danych) • testy projektu	• programowanie (specyfikacja programu : deklaracje, definicje; dodatkowe struktury danych: struktury „pojemnikowe”, pliki, bazy danych) • testy oprogramowania • wdrażanie • testy wdrażania

Zunifikowany iteracyjno- przyrostowy proces tworzenia oprogramowania – kiedy?



Przepływy czynności (wg G.Booch, J. Rumbaugh, I.Jacobson)

- **Modelowanie przedsiębiorstwa – opis dynamiki i struktury przedsiębiorstwa**
- **Wymagania – zapisanie wymagań metodą opartą na przypadkach użycia**
- **Analiza i projektowanie – zapisanie różnych perspektyw architektonicznych**
- **Implementacja – tworzenie oprogramowania, testowanie modułów, scalanie systemu**
- **Testowanie – opisanie danych testowych, procedur i metryk poprawności**
- **Wdrożenie – ustalenie konfiguracji gotowego systemu**
- **Zarządzanie konfiguracjami – panowanie nad zmianami i dbanie o spójność elementów systemu**
- **Zarządzanie przedsięwzięciem - opisane różnych strategii prowadzenia procesu iteracyjnego**
- **Określenie środowiska – opisanie struktury niezbędnej do opracowania systemu**

Modelowanie i analiza systemów informatycznych

1. Warstwowa budowa systemów informatycznych
2. Model procesu wytwarzania oprogramowania - model cyklu życia oprogramowania
3. Wstęp do modelowania systemów informatycznych

Co i jak wykonać?

Perspektywy projektowania obiektowych systemów informacyjnych

(wg Alan Shalloway, James R.Trott)

- **koncepcji** (model analizy)

  (co obiekty powinny powinny robić?)

- **specyfikacji interfejsów** (model projektowy)

  (jak używać obiektów?)

- **implementacji** (implementacja)

  (w jaki sposób zaimplementować interfejs ?)

- **tworzenia i zarządzania obiektami** (implementacja)

  (*obiekt A w roli fabryki obiektów tworzy obiekt B i/lub zarządza obiektem B*)

- **używania obiektów** (implementacja)

  (*obiekt A tylko używa obiektu B – nie może go jednocześnie tworzyć*)

Perspektywy rozumienia obiektów – identyfikacji obiektów

- **Perspektywa koncepcji** (modelu konceptualnego)

  - obiekt jest zbiorem różnego rodzaju odpowiedzialności

- **Perspektywa specyfikacji** (modelu projektowego)

  - obiekt jest zbiorem metod (zachowań), które mogą być wywoływane przez metody tego obiektu lub innych obiektów

- **Perspektywa implementacji** (kodu źródłowego)

  - obiekt składa się z kodu metod i danych oraz interakcji między nimi

Perspektywy skalowania systemu – tworzenia, zarządzania i używania obiektów

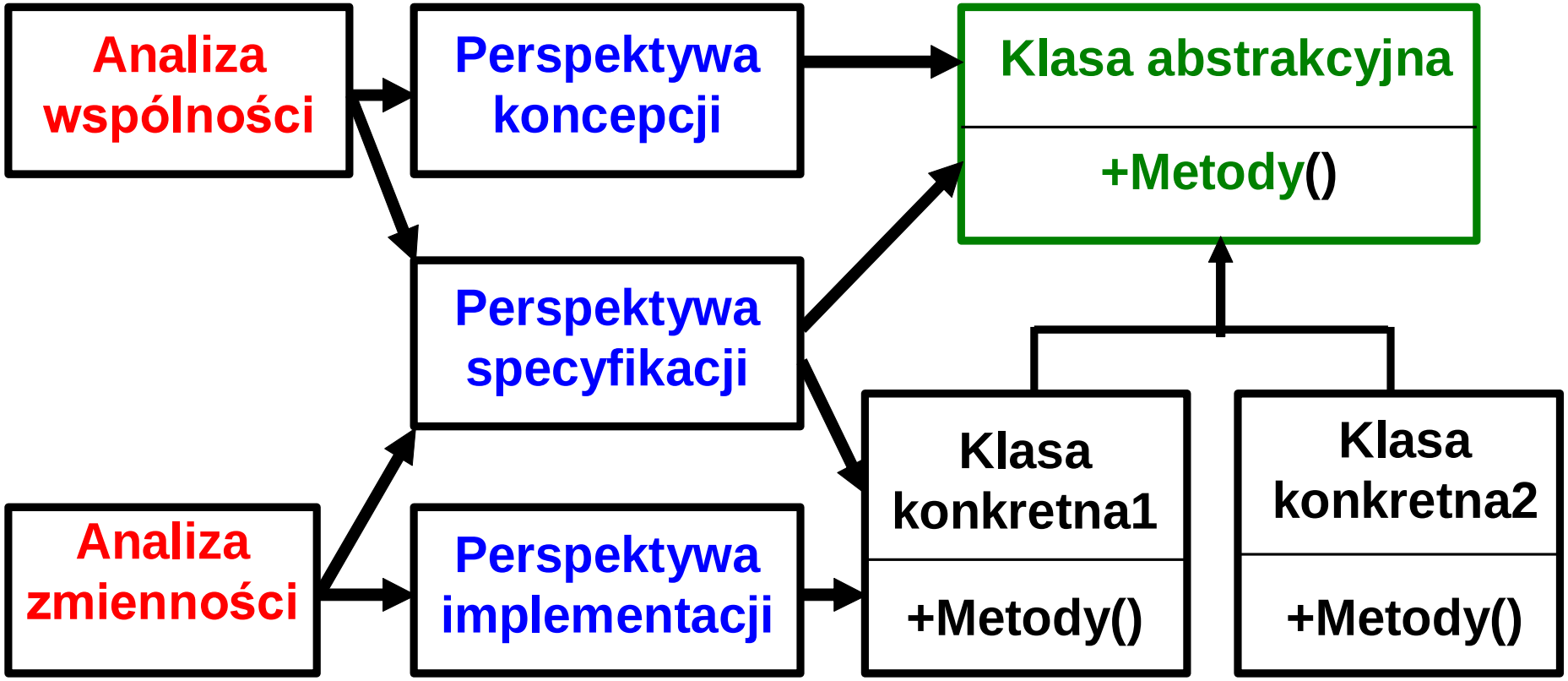
- **Perspektywa tworzenia i zarządzania obiektami**

  Zmiany w implementacji obiektów dotyczą obiektów czyli fabryk obiektów (tworzących te obiekty i zarządzających tworzeniem tych obiektów)

- **Używanie obiektów**

  Zmiana implementacji obiektów nie zmienia implementacji obiektów, które używają zmieniane obiekty

Metoda identyfikacji obiektów i klas



Związek między perspektywą specyfikacji, koncepcji i implementacji

Zależności między analizą, projektowaniem i implementacją

Związek między perspektywą specyfikacji i koncepcji

- Perspektywa specyfikacji określa interfejs potrzebny do obsługi wszystkich przypadków danego problemu (czyli **część wspólną** określoną przez perspektywę koncepcji)

Związek pomiędzy perspektywą specyfikacji i implementacji

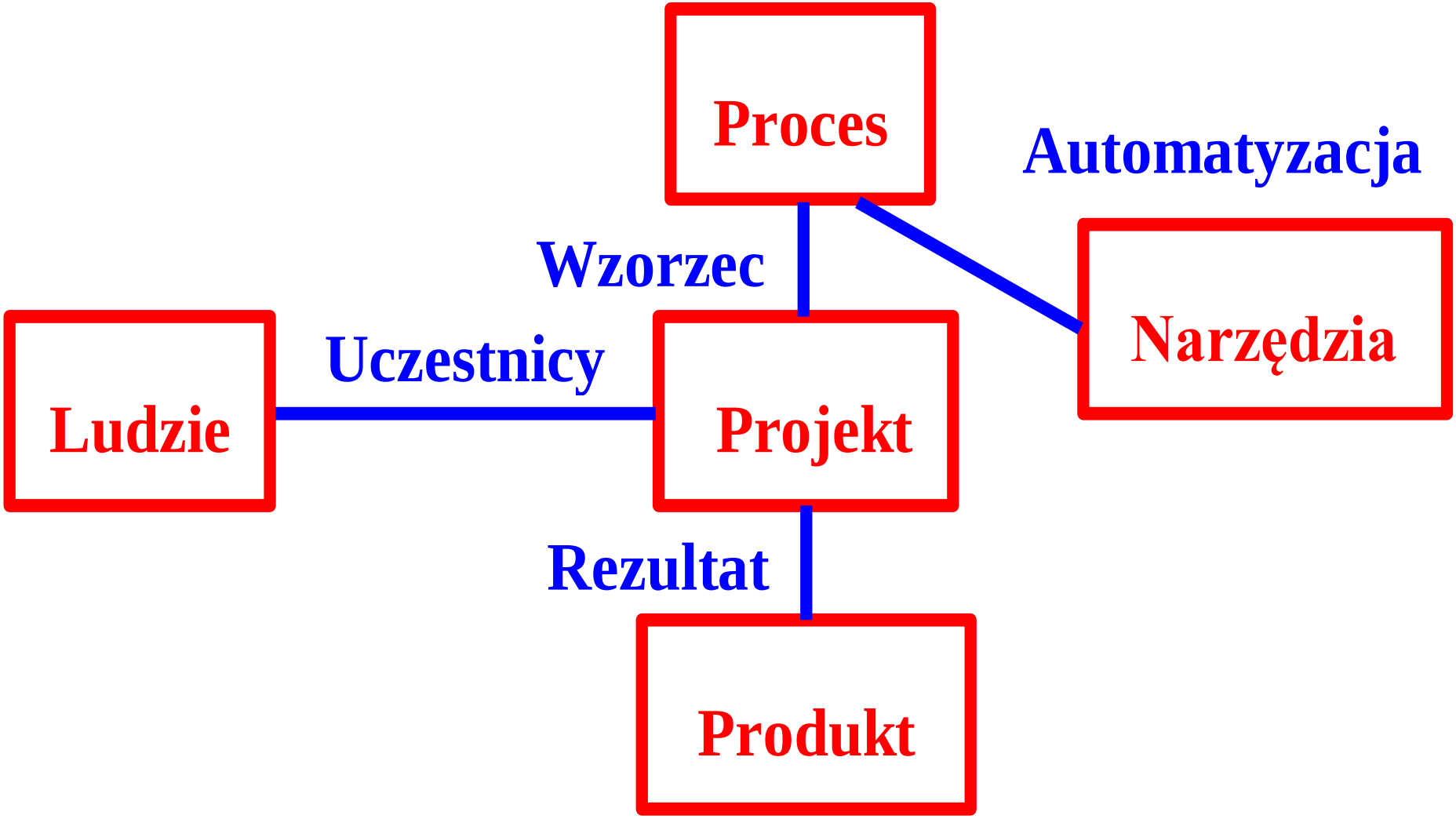
- Biorąc pod uwagę określoną specyfikację ustala się, w jaki sposób należy zaimplementować poszczególne przypadki (czyli **część zmienną**)

Podsumowanie zasad obiektowości

- Obiekty definiuje się przez pryzmat ich odpowiedzialności
- Hermetyzacja oznacza dowolny rodzaj ukrywania: danych, implementacji, klas (za pomocą klas abstrakcyjnych, lub interfejsu), projektu, instancji
- Wykorzystanie analizy wspólności i zmienności w celu utworzenia abstrakcji reprezentujących zmienność danych i zachowań
- Wykorzystanie dziedziczenia jako sposobu przedstawienia zmienności
- Dążenie do niskiego stopnia powiązań
- Dążenie do dużego stopnia spójności
- Oddzielenie kodu używającego obiektów od kodu, który je tworzy
- Zasada pojedynczej reguły - implementacja tylko raz pewnej operacji określonej jedną regułą
- Stosowanie nazw jasno opisujących przeznaczenie obiektów

Modelowanie i analiza systemów informatycznych

1. Warstwowa budowa systemów informatycznych
2. Model procesu wytwarzania oprogramowania - model cyklu życia oprogramowania
3. Wstęp do modelowania systemów informatycznych
4. Środowisko tworzenia oprogramowania


Proces
Automatyzacja
Wzorzec
Narzędzia
Uczestnicy
Ludzie
Projekt
Rezultat
Produkt

W tworzenia oprogramowania istotne są następujące elementy:

Ludzie: eksperci dziedziny biznesowej, analitycy, projektanci, testujący, użytkownicy oprogramowania, konserwatorzy oprogramowania

Projekt: główny organizacyjny element, określający tworzenie oprogramowania: *projekt tworzy produkt*

Produkt: elementy tworzone podczas cyklu życia oprogramowania: modele, kod źródłowy, kod wynikowy, dokumentacja

Proces: wszystkie czynności podejmowane w celu przekształcenia wymagań użytkownika w oprogramowanie; jest to wzorzec realizacji projektu

Narzędzia: oprogramowanie umożliwiające zautomatyzowanie procesu.

Produkt

1. Podsystemy
2. Diagramy: klas, interakcji, kooperacji, stanów
3. Wymagania, testy, produkcja , instalacja
4. System złożony z artefaktów reprezentujących narzędzia programistyczne, kompilatory, komputery, programistów, architektów, testujących, handlowców, administratorów
5. Artefakty to różne rodzaje informacji tworzonej, produkowanej, zmienianej lub używanych podczas cyklu życia oprogramowania. Są to artefakty inżynierskie związane z tworzeniem oprogramowania (wymagania, analiza, projekt, programowanie, testy) i artefakty procesu zarządzania projektem

Projekt

Podstawowe pojęcia związane z projektem:

1. Wykonalność projektu
2. Zarządzanie ryzykiem
3. Struktura grup projektowych
4. Szeregowanie zadań projektowych
5. Zrozumiałość projektu
6. Sensowność działań w projekcie

Cechy projektu:

1. Sekwencja zmian w projekcie
2. Seria iteracji
3. Wzorzec organizacyjny

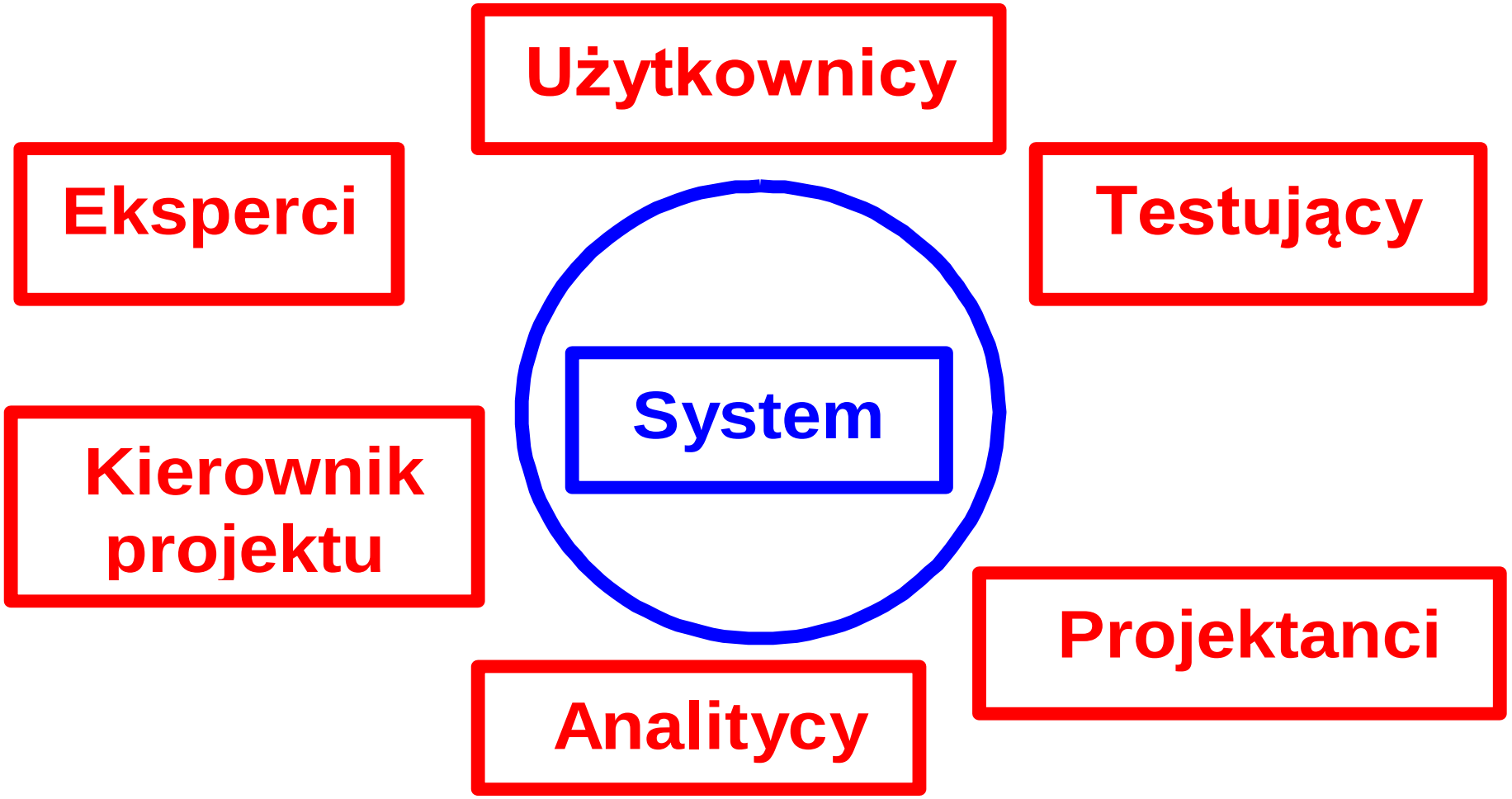
Proces

1. Proces tworzenia oprogramowania jest definicją kompletnego zbioru aktywności potrzebnych do odwzorowania wymagań użytkownika w zbiór artefaktów, które reprezentują oprogramowanie
2. Czynniki organizacyjne
3. Czynniki dziedzinowe
4. Czynniki cyklu życia
5. Czynniki techniczne

Narzędzia

- Automatyzacja procesu
- Standaryzacja procesu i produktu
- Wspomaganie całego cyklu życia oprogramowania: wymagania, wizualne modelowanie i projektowanie, programowanie, testowanie

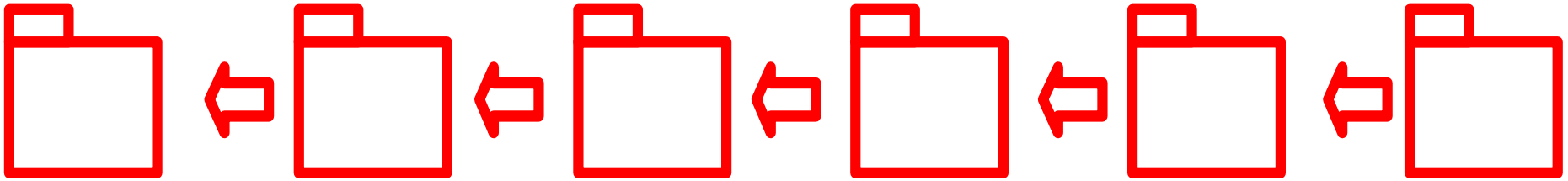
Ludzie


Użytkownicy
Eksperci
Testujący
System
Kierownik projektu
Projektanci
Analitycy

Modele

Modele:

- Abstrakcja systemu
- Przedstawianie różnych perspektyw systemu
- Związki między modelami



Modelowanie i analiza systemów informatycznych

1. Warstwowa budowa systemów informatycznych
2. Model procesu wytwarzania oprogramowania - model cyklu życia oprogramowania
3. Wstęp do modelowania systemów informatycznych
4. Środowisko tworzenia oprogramowania
5. Podstawy UML

UML – język wspierający zunifikowany iteracyjno - przyrostowy proces tworzenia oprogramowania

Diagramy UML modelowania strukturalnego

1.1. Diagramy pakietów

1.2. Diagramy klas

1.3. Diagramy obiektów

1.4. Diagramy mieszane

1.5. Diagramy komponentów

1.6. Diagramy wdrożenia

Diagramy UML modelowania zachowania

2.1. Diagramy przypadków użycia

2.2. Diagramy aktywności

2.3. Diagramy stanów

2.4. Diagramy komunikacji

2.5. Diagramy sekwencji

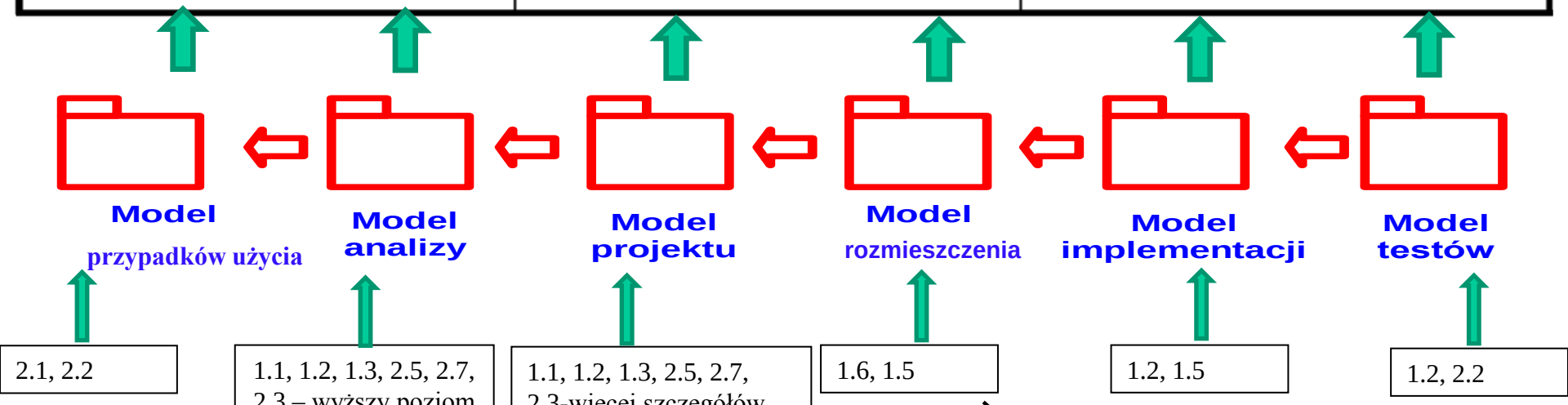
2.6. Diagramy czasu

2.7. Diagramy interakcji

Rola diagramów UML 2

- praca zespołowa
- pokonanie złożoności projektu
- formalne, precyzyjne prezentowanie projektu
- tworzenie wzorca projektu
- możliwość testowania oprogramowania we wczesnym stadium jego tworzenia

Modelowanie struktury i dynamiki systemu	Implementacja systemu,	struktury i dynamiki generowanie kodu
Perspektywa koncepcji *co należy wykonać?*	Perspektywa specyfikacji *jak należy używać?*	Perspektywa implementacji *jak należy wykonać?*
• model problemu np. przedsiębiorstwa • wymagania • analiza (model konceptualny: diagram przypadków użycia, diagram klas, diagramy sekwencji,) • testy modelu	• projektowanie (model projektowy: architektura sprzętu i oprogramowania; dostęp użytkownika; przechowywanie danych) • testy projektu	• programowanie, wdrażanie (specyfikacja programu : deklaracje, definicje; dodatkowe struktury danych: struktury „pojemnikowe”, pliki, bazy danych) • testy oprogramowania • wdrażanie • testy wdrażania



Numery diagramów UML: slajdy 26 i 27