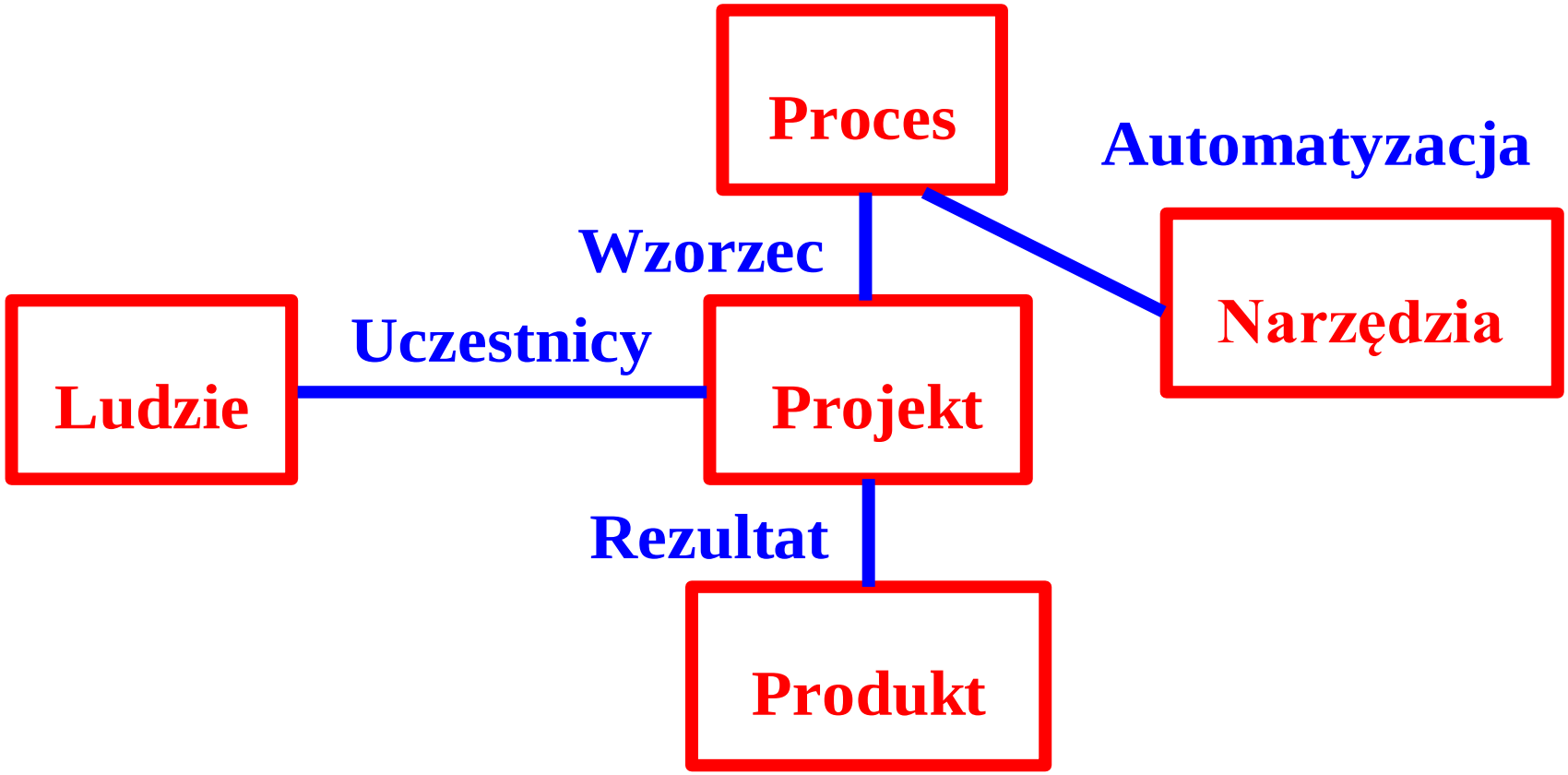
# Wprowadzenie, podstawowe pojęcia, projekt a produkt Wykład1

Zofia Kruczkiewicz

# Literatura


1. Roger S. Pressman, Praktyczne podejście do oprogramowania, WNT, 2004
2. Stephen H. Kan, Metryki i modele w inżynierii jakości oprogramowania, Mikom, 2006
3. Jacobson, Booch, Rumbaung, The Unified Software Development Process,Addison Wesley, 1999
4. Shalloway A.,Trott James R.,Projektowanie zorientowane obiektowo. Wzorce projektowe. Gliwice, Helion, 2005

# Elementy tworzenia oprogramowania – struktura [3]



# Elementy tworzenia oprogramowania - struktura

**Ludzie:** uczestnicy

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

**Proces (tworzenia oprogramowania):** wzorzec realizacji produktu
(wymagania użytkownika ⟶ oprogramowanie)

**Projekt (przedsięwzięcie):** organizacja tworzenia produktu

**Narzędzia:** sprzęt i oprogramowanie umożliwiające zautomatyzowanie procesu.

# Artefakt

**Artefakt** – reprezentuje różne **rodzaje informacji tworzonej, produkowanej, zmienianej lub używanej** podczas cyklu życia oprogramowania.

**Są to artefakty inżynierskie** związane:

- **z procesem wytwórczym**

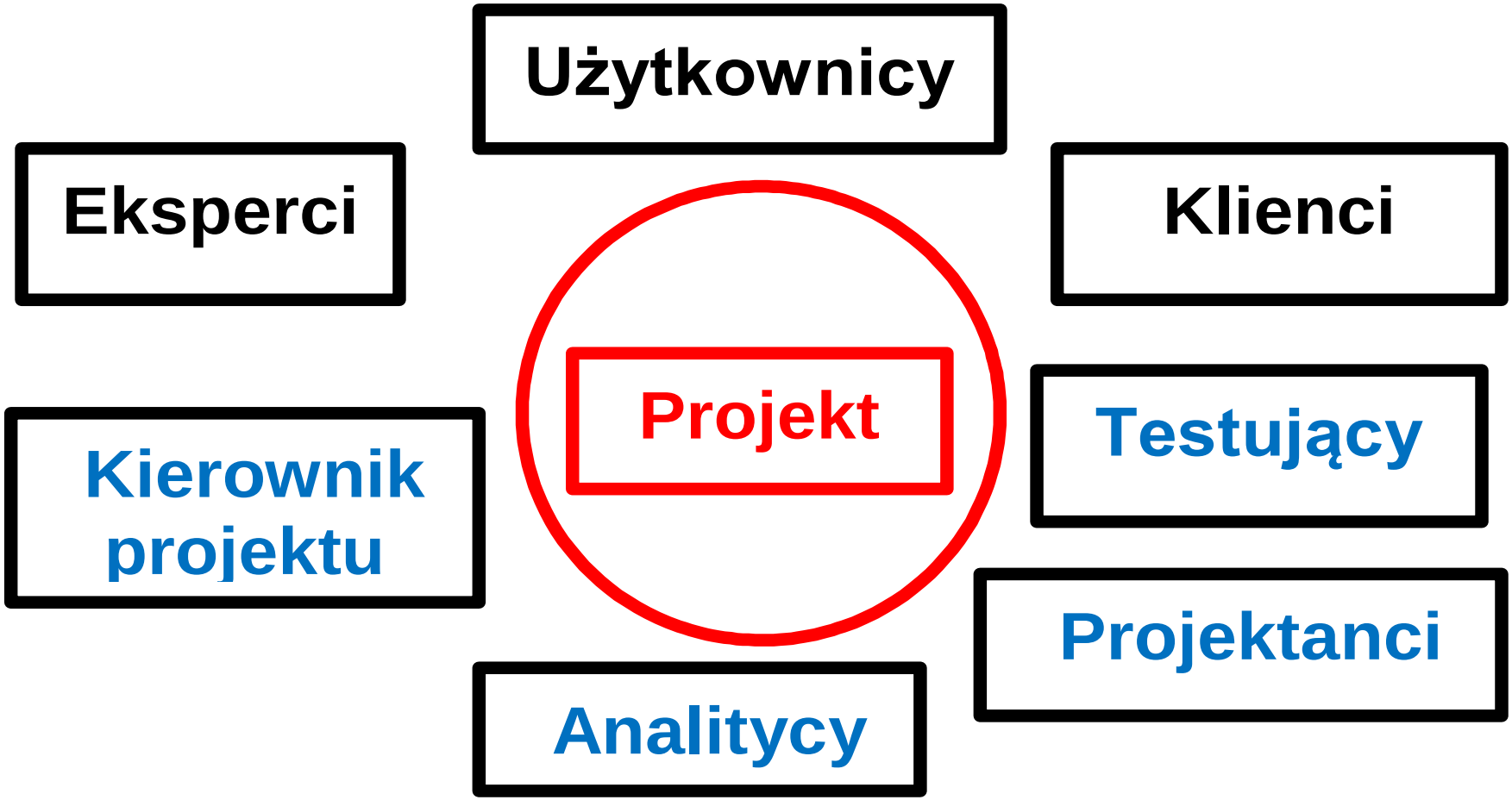
  (wymagania, analiza, projektowanie, programowanie, testy)

- **z kierowaniem projektu**

  (narzędzia programistyczne, kompilatory, komputery, programiści, architekci, testerzy, handlowcy, administratorzy)

# Ludzie [3]

# Ludzie


Użytkownicy
Eksperci
Klienci
Projekt
Kierownik projektu
Testujący
Projektanci
Analitycy

# Produkt [1]

# Produkt

- Co to jest?
- Kto się tym zajmuje?
- Dlaczego to jest ważne?
- Jak to się robi?
- Co jest produktem roboczym?

# Produkt

1. Modele, kod źródłowy, kod wynikowy, dokumentacja
2. Podsystemy
3. Diagramy: klas, interakcji, kooperacji, stanów
4. Wymagania, testy, produkcja , instalacja
5. **Wytworzone oprogramowanie**
6. Jest to system złożony z artefaktów związanych **z procesem wytwórczym** (wymagania, analiza, projektowanie, programowanie, testy)

# Zmieniająca się rola oprogramowania

Oprogramowanie jest produktem i mechanizmem jego dostarczania:

- Dlaczego trwa tak długo?
- Dlaczego tyle kosztuje?
- Dlaczego zawiera błędy?
- Nadal brak precyzyjnego pomiaru oprogramowania, analizowania procesu powstawania oprogramowania?

# Oprogramowanie – byt logiczny

Oprogramowanie to:

- rozkazy , których wykonanie pozwala wypełnić określone funkcje w oczekiwany sposób
- struktury danych, które umożliwiają programom manipulowanie informacjami
- oraz dokumenty, które opisują działanie i sposób użytkowania programów.

# Cechy charakterystyczne oprogramowania

- Oprogramowanie jest wytwarzane, ale nie jest fizycznie konstruowane (np. tak jak sprzęt)
- Oprogramowanie się nie zużywa, ale z czasem niszczeje
- Korzystanie z gotowych komponentów, ale większość jest tworzona od nowa

# Dziedziny zastosowań oprogramowania

- Oprogramowanie systemowe
- Systemy czasu rzeczywistego
- Systemy informacyjne dla przedsiębiorstw
- Oprogramowanie inżynierskie i naukowe
- Systemy wbudowane
- Oprogramowanie komputerów osobistych
- Oprogramowanie internetowe
- Sztuczna inteligencja

# Mity - ignorowanie zasad inżynierii oprogramowania

- **Mity kierownictwa**
  - Standardy i procedury zapewniają tworzenie dobrego oprogramowania
  - Dobre oprogramowanie narzędziowe działające na dobrym sprzęcie gwarantuje wykonanie dobrych programów
  - Jeśli prace się opóźniają, wystarczy przydzielić do zadania więcej programistów
  - Jeżeli oprogramowanie wykonuje inna firma (outsourcing), to można pozbyć się problemów

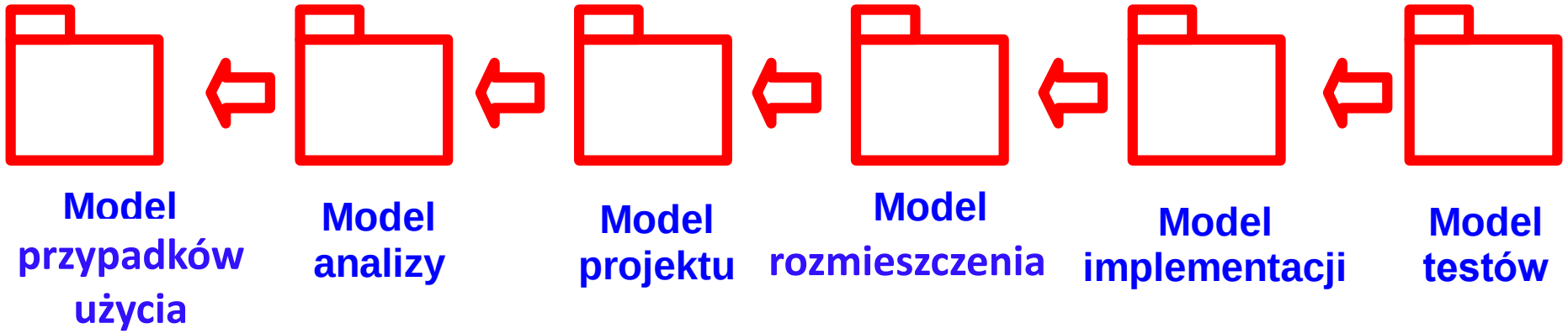
- **Mity klientów**
  - Ogólne określenie oczekiwań klienta wystarczy do rozpoczęcia prac, a szczegóły można dopracować później
  - Wymaganie wobec systemu wciąż się zmienia. Ale to nie problem, bo oprogramowanie jest elastyczne i łatwo je zmienić.

- **Mity informatyków**
  - Po napisaniu programu i uruchomieniu go praca jest wykonana
  - Dopóki program nie działa, nie da się ocenić jego jakości
  - Jedynym wynikiem pracy nad oprogramowaniem jest działający program komputerowy
  - Inżynieria oprogramowania zmusi nas do tworzenia przepastnych, zbędnych dokumentów i nieuchronnie spowolni pracę.

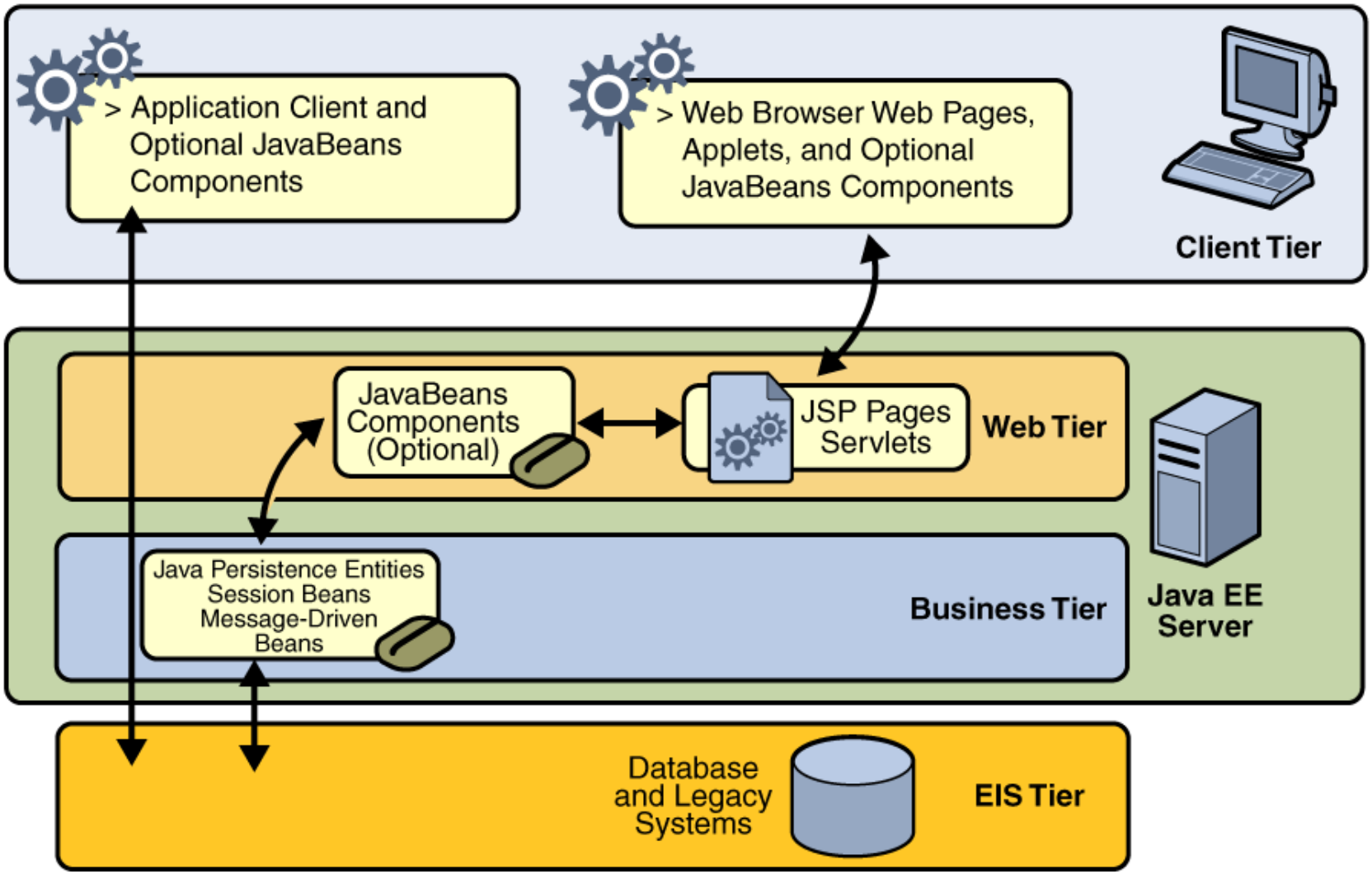
# Produkt – Modele [3]

**Modele:**

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



# **Produkt -** warstwy aplikacji (Java EE)



# Produkt

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

| | |
|---|---|
| **Warstwa klienta**<br>Klienci aplikacji, aplety, aplikacje i inne elementy z graficznym interfejsem użytkownika | Interakcja z użytkownikiem, urządzenia i prezentacja interfejsu użytkownika |
| **Warstwa prezentacji**<br>Strony JSP, serwlety i inne elementy interfejsu użytkownika | Logowanie, zarządzanie sesją, tworzenie zawartości, formatowania i dostarczanie |
| **Warstwa biznesowa**<br>Komponenty EJB i inne obiekty biznesowe | Logika biznesowa, transakcje, dane i usługi |
| **Warstwa integracji**<br>JMS, JDBC, konektory i połączenia z systemami zewnetrznymi | Adaptery zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania |
| **Warstwa zasobów**<br>Bazy danych, systemy zewnętrzne i pozostałe zasoby | Zasoby, dane i usługi zewnętrzne |

# **Produkt – diagramy UML** wspierające zunifikowany iteracyjno - przyrostowy proces tworzenia oprogramowania [3]

**Diagramy modelowania struktury**

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 produktu
- formalne, precyzyjne prezentowanie produktu
- tworzenie wzorca produktu
- możliwość testowania oprogramowania we wczesnym stadium jego tworzenia

# Narzędzia [1]

# Narzędzia

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

# Proces tworzenia oprogramowania - proces wytwórczy [1]

# Proces tworzenia oprogramowania

- Kto robi?
- Co robi?
- Kiedy robi?
- Jak robi?

# Proces tworzenia oprogramowania

- Technologie
- Narzędzia
- Ludzie
- Wzorce organizacyjne

# Proces tworzenia oprogramowania

1) **Proces tworzenia oprogramowania** jest definicją kompletnego zbioru aktywności potrzebnych do odwzorowania wymagań użytkownika w zbiór artefaktów, które reprezentują produkt (oprogramowanie)
2) Obejmuje czynniki:
   - Czynniki organizacyjne
   - Czynniki dziedzinowe
   - Czynniki **cyklu życia**
   - Czynniki techniczne

# Model procesu wytwarzania oprogramowania - czyli model cyklu życia oprogramowania [3], [4]

**Tworzenie** technicznego systemu informacyjnego jest powiązane z:

- budową oprogramowania: **co i jak wykonać? kiedy wykonać?**
- wdrażaniem oprogramowania

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

# Definicje inżynierii oprogramowania

- [Fritz Bauer]
  - Opracowanie sprawdzonych zasad inżynierii oraz ich zastosowanie w celu wytworzenia niedrogiego i niezawodnego oprogramowania, działającego efektywnie na rzeczywistych maszynach
- [IEEE 1993]
  - (1)Zastosowanie systematycznego, zdyscyplinowanego, poddającego się ocenie ilościowej, podejścia do wytwarzania, stosowania i pielęgnacji oprogramowania, czyli wykorzystanie technik tradycyjnej inżynierii w informatyce.
  - (2) Dziedzina wiedzy zajmująca się badaniem metod jak w p. (1)

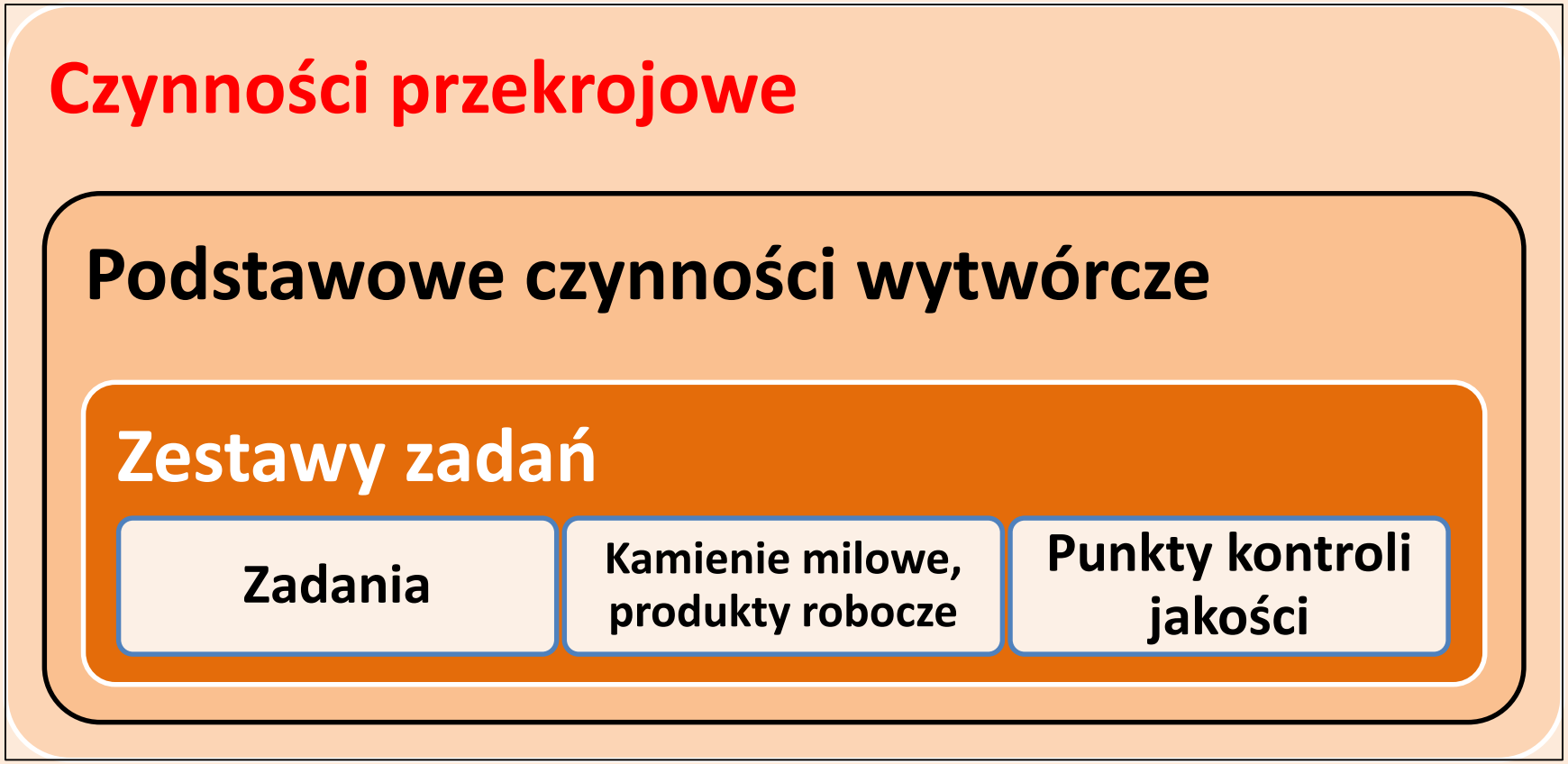
# Warstwowe podejście w inżynierii oprogramowania

# Ogólne spojrzenie na inżynierię oprogramowania

1) Jaki problem należy rozwiązać
2) Jakie cechy produktu umożliwiają rozwiązanie problemu
3) Jak ma wyglądać produkt (rozwiązanie problemu)
4) Jak skonstruować taki produkt
5) Jak wykrywać błędy w projekcie lub podczas konstrukcji produktu
6) Jak obsługiwać i pielęgnować gotowy produkt i jak uwzględniać uwagi, reklamacje i żądania jego użytkowników: **poprawianie, adaptowanie, rozszerzanie, zapobieganie**

# Proces wytwórczy

Uniwersalny schemat procesu wytwórczego



# Uzupełnienie faz procesu wytwórczego - zestaw czynności przekrojowych, wykonywanych przez cały czas działania projektu

1) Zarządzanie przedsięwzięciem programistycznym i śledzenie jego przebiegu
2) Formalne przeglądy techniczne
3) Zapewniane jakości
4) Zarządzanie konfiguracją
5) Przygotowanie i drukowanie dokumentacji
6) Zarządzanie elementami oprogramowania nadającymi się do wielokrotnego użycia
7) Pomiary
8) Panowanie nad ryzykiem

# **Poziomy dojrzałości procesów wytwórczych CMM** (capability maturity model) **ustalone przez SEI** (Software Engineering Institute )

- Poziom 1 - początkowy
- Poziom2 - powtarzalny
- Poziom 3 - zdefiniowany
- Poziom 4 - zarządzany
- Poziom 5 - optymalizowany

**Opis kluczowego elementu procesu** (KPA – key process areas) **każdego poziomu dojrzałości procesu, związanego z kluczowymi praktykami** (key practices) **kontrolowanymi za pomocą kluczowych wskaźników** (key indicators)

1) Cele
2) Zobowiązania podjęte do spełnienia celów
3) Możliwości konieczne do spełniania zobowiązań
4) Konkretne czynności
5) Metody monitorowania przebiegu prac
6) Metody sprawdzania

**5 poziomów dojrzałości procesów wytwórczych CMM** (capability maturity model) **ustalone przez SEI** (Software Engineering Institute)**. Każdy poziom dojrzałości procesu jest powiązany z listą kluczowych elementów procesu (razem 18 KPA)**

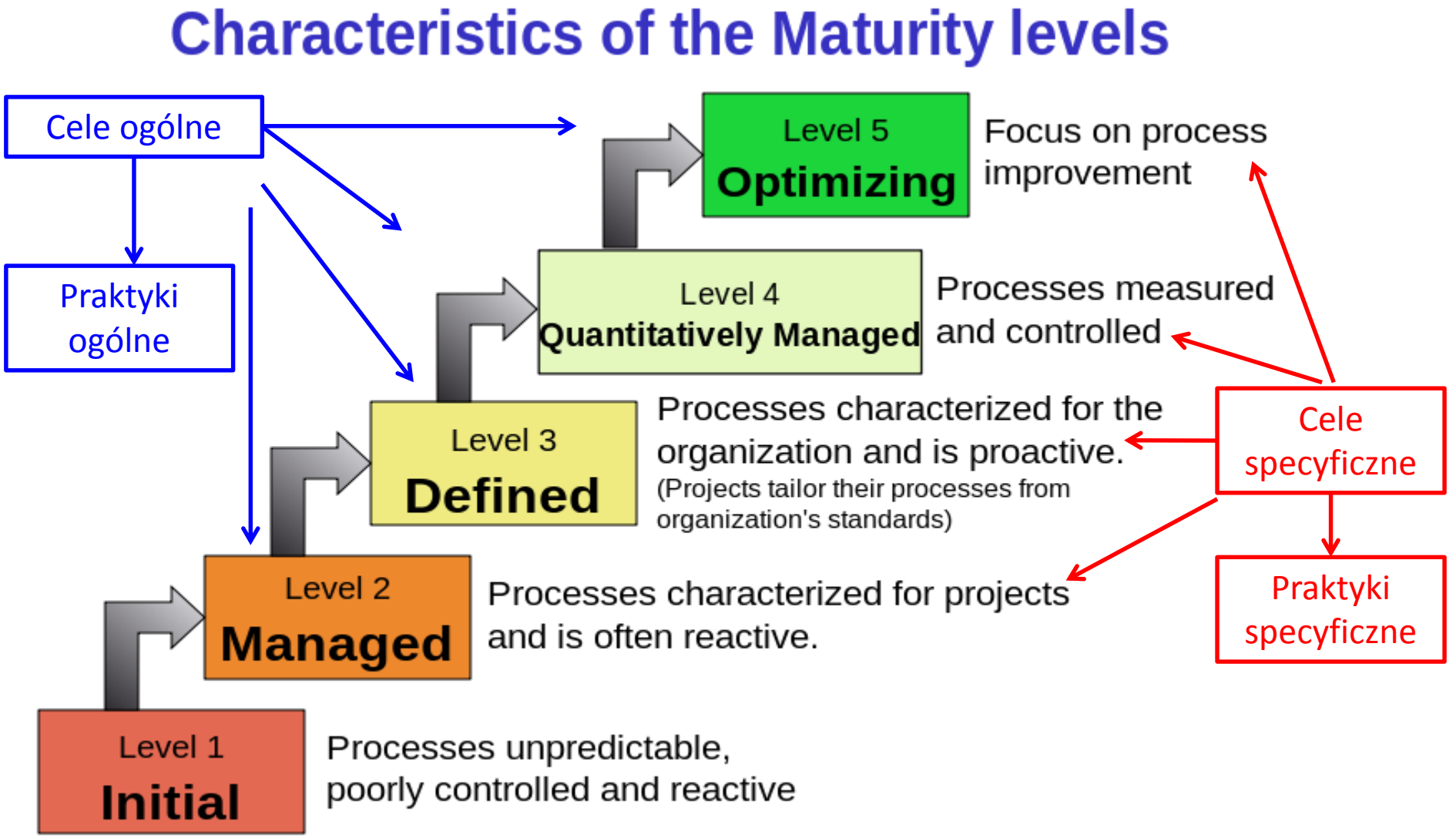
1) Poziom 1- początkowy: chaotyczny
2) Poziom2 – powtarzalny:
    1. Zarządzanie konfiguracją
    2. Zapewnianie jakości
    3. Zarządzanie współpracą z podwykonawcami
    4. Śledzenie i nadzór nad przedsięwzięciem programistycznym
    5. Planowanie przedsięwzięcia programistycznego
    6. Zarządzanie wymaganiami

3) Poziom 3- zdefiniowany (po osiągnięciu poziomu 2)
   7. Wewnętrzne recenzje i przeglądy
   8. Zarządzanie współpracą między zespołami
   9. Zastosowanie metod inżynierii oprogramowania podczas tworzenie produktu
   10. Zintegrowanie zarządzanie oprogramowaniem
   11. Program szkoleń
   12. Określenie schematu procesu wytwórczego
   13. Duże znaczenie procesu wytwórczego w działalności firm

4) Poziom 4 – zarządzany (po osiągnięciu poziomu 3)
   14. Zarządzanie jakością oprogramowania
   15. Zarządzanie procesami określone ilościowo

5) Poziom 5 – optymalizowany (po osiągnięciu poziomu 4)
   16. Zarządzanie zmianami procesu
   17. Zarządzanie zmianami technologicznymi
   18. Zapobieganie błędom

# 4.2. Poziomy dojrzałości modelu CMMI



# Modele procesów wytwórczych

1. Sekwencyjny model liniowy
2. Model oparty na prototypowaniu
3. Model szybkiej rozbudowy aplikacji
4. Modele ewolucyjne
   - Model przyrostowy
   - Model spiralny
   - Model spiralny WINWIN
   - Model równoległy
5. Model oparty na metodach formalnych
6. Model oparty na komponentach
7. Techniki czwartej generacji

# Technologia procesu wytwórczego

- Narzędzia do
  - analizy procesu,
  - organizowania zadań,
  - kontrolowania i monitorowania postępu prac,
  - zarządzanie techniczną jakością oprogramowania,

# Produkt a proces wytwórczy
# (dopasowanie procesu do produktu)

1) Wadliwy proces może spowodować powstanie produktu o niskiej jakości
2) Dualna natura procesu i produktu – wspierająca wielożywalność produktu
3) Czerpanie satysfakcji z faktu tworzenia i jego rezultatu
4) Ścisły związek między procesem i produktem zachęca twórczych ludzi do pracy

# Projekt – wstęp do zarządzania projektami programistycznymi [1], [2]

# **Projekt** = przedsięwzięcie

**Główny, organizacyjny element** powiązany z:

- Ludzie, Produkt, Proces

**Pojęcia**:

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

# Zarządzanie projektem - wstęp

1) Co to jest – ogólny plan pracy nad oprogramowaniem
2) Kto się tym zajmuje – Informatycy, kierownicy i klienci
3) Dlaczego jest to ważne – bo wprowadza stabilność, przewidywalność i uporządkowanie
4) Jak to się robi – (Ludzie, Produkt, Proces, Projekt) np. postać projektu zależy od rodzaju wytwarzanego oprogramowania (produktu)
5) Co jest produktem roboczym – programy, dokumenty i dane
6) Jak to zrobić dobrze – można ocenić dojrzałość projektu. Jego jakość potwierdza wysoka jakość stworzonego produktu, jego zdolność do długotrwałego utrzymania się na rynku i terminowość procesu

# Zarządzanie projektem: planowanie, organizowanie, monitorowanie i kontrolowanie

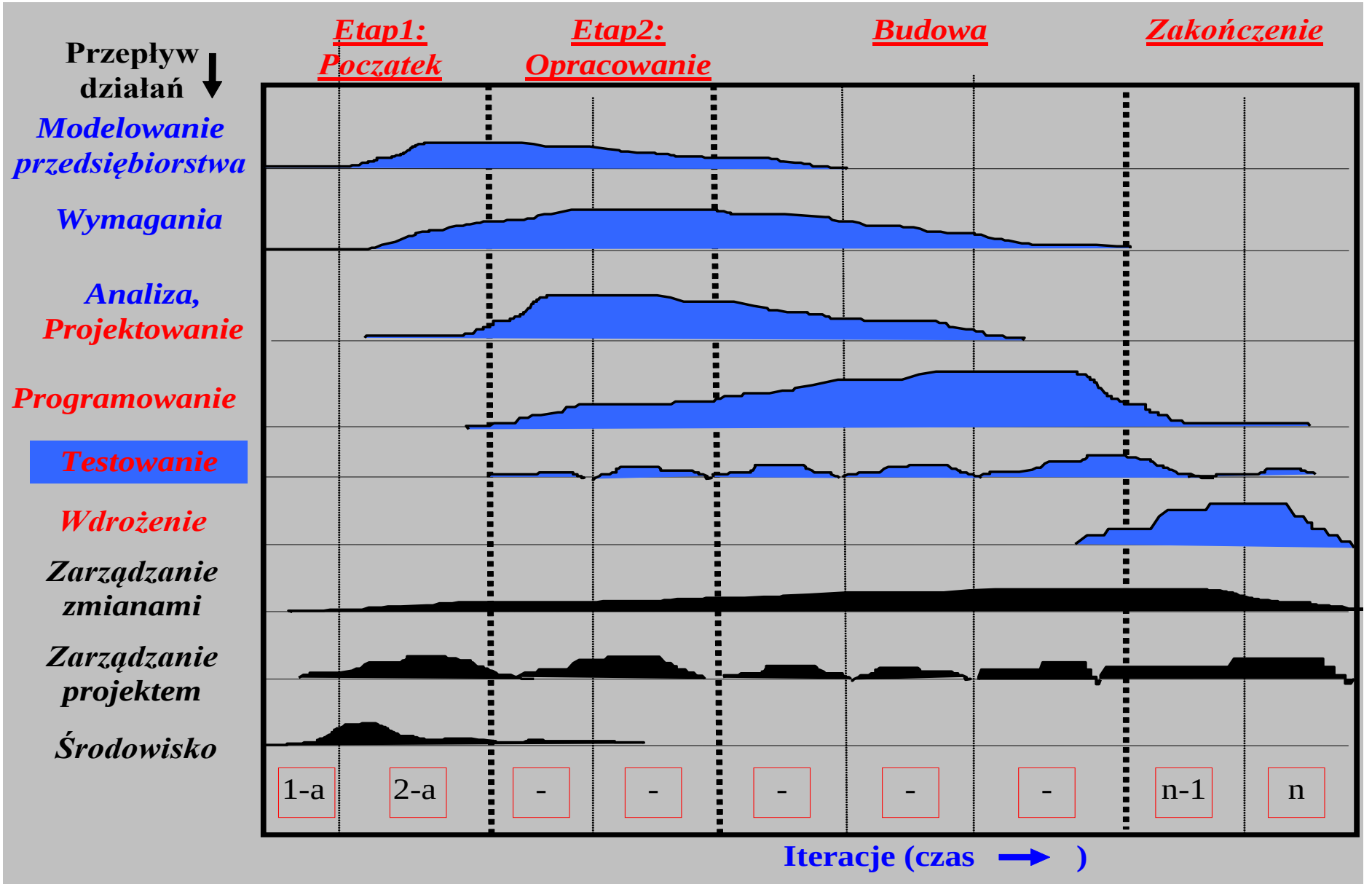
1) Jak zarządzać pracownikami, procesami i problemami podczas tworzenia oprogramowania?
2) Czym są miary i jak ich używać do zarządzania projektami i procesami wytwórczymi?
3) Jak sporządzać wiarygodne prognozy kosztów pracochłonności i czasu trwania prac?
4) Jakimi metodami można formalnie ocenić zagrożenia, które mogą mieć wpływ na powodzenie projektu?

4) Jak wybrać zadania, które trzeba zrealizować, aby stworzyć nowy produkt?
5) Jak sporządzić harmonogram?
6) Jak zdefiniować jakość oprogramowania, aby można było ją kontrolować?
7) Na czym polega zapewnianie jakości?
8) Dlaczego formalne przeglądy techniczne są tak ważne?
9) Jak zarządzać zmianami w czasie procesu wytwórczego i po oddaniu produktu do użytku?

# Dodatek (Modelowanie i analiza systemów informatycznych)

- **Przykład procesu wytwórczego [3]**

# Zunifikowany iteracyjno- przyrostowy proces tworzenia oprogramowania – kiedy?



# Przepływy czynności

- 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 zmianami – panowanie nad zmianami i dbanie o spójność elementów systemu
- Zarządzanie projektem - opisane różnych strategii prowadzenia procesu iteracyjnego
- Określenie środowiska – opisanie struktury niezbędnej do opracowania systemu

# Co i jak wykonać?

## Perspektywy projektowania obiektowych systemów informacyjnych [4]

- **koncepcji** (model analizy)

  ( co obiekty 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)*

- **używania obiektów** (implementacja)

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

  *opiera się na hermetyzacji i polimorfiźmie obiektu B )*

# Perspektywy rozumienia obiektów – identyfikacji obiektów [4]

- **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 [4]

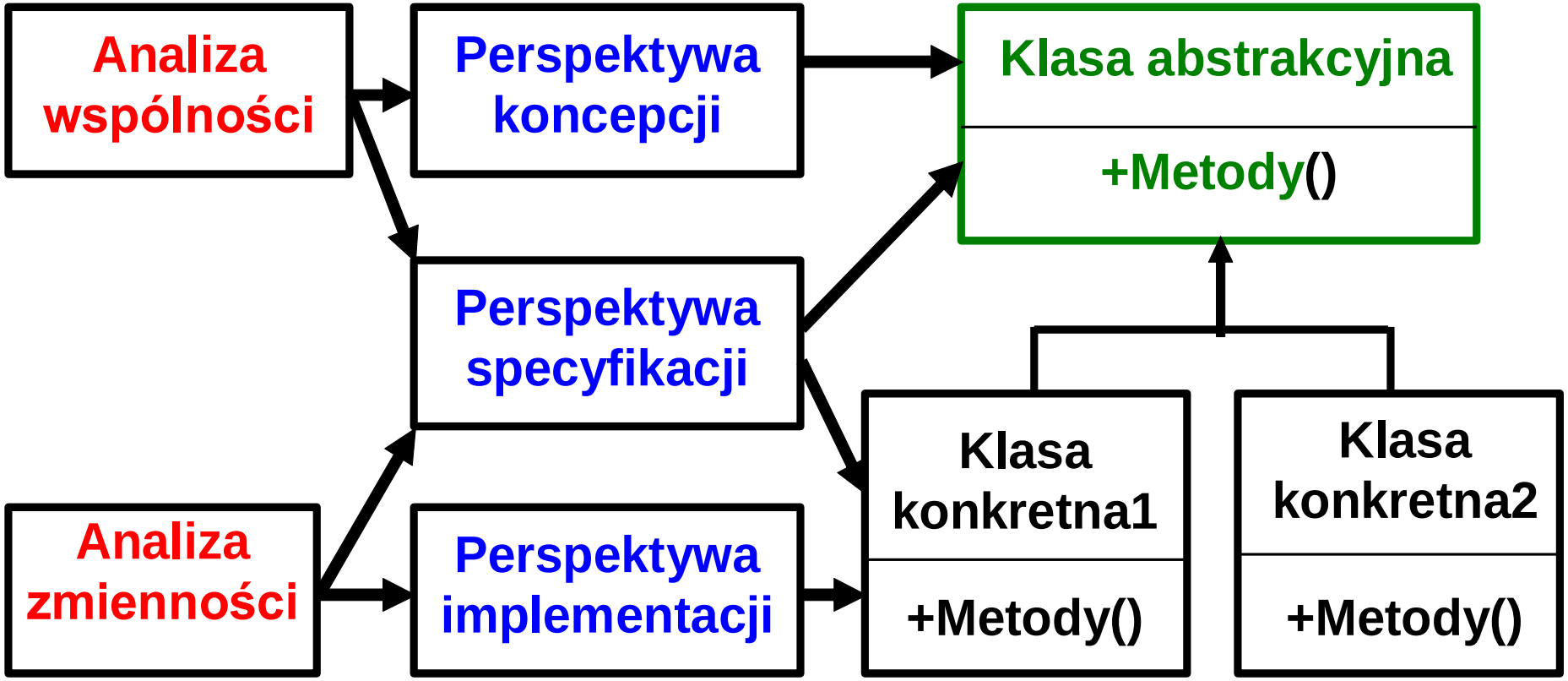
- **Perspektywa tworzenia i zarządzania obiektami**

  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 [4]




Związek między perspektywą specyfikacji, koncepcji i implementacji

# Zależności między analizą, projektowaniem i implementacją [4]

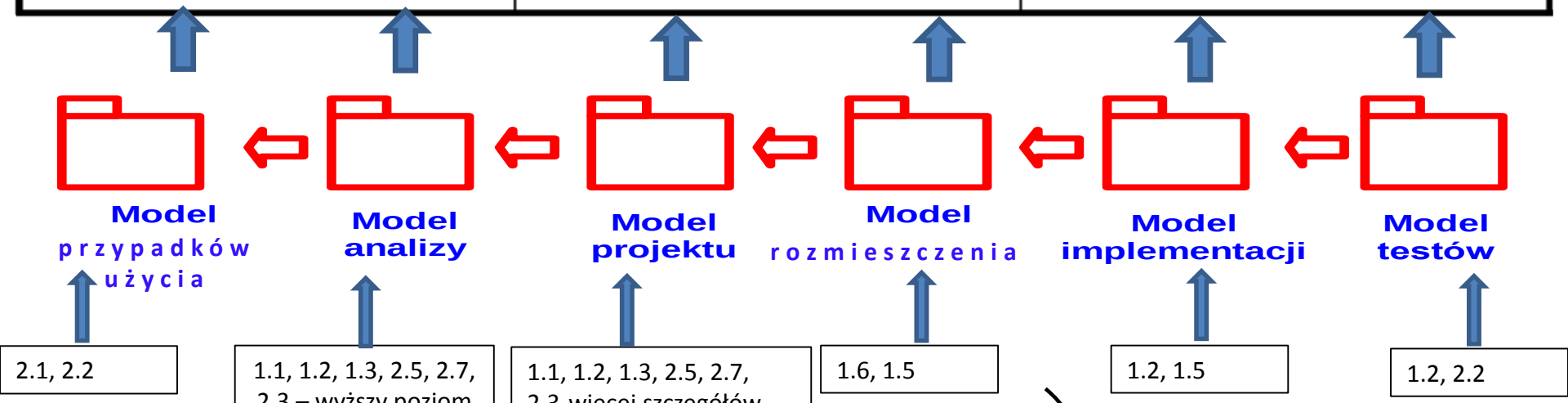
**Związek między perspektywą koncepcji i specyfikacji**

- 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**)

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




Numery diagramów UML: slajdy 21 i 22