Podstawy inżynierii oprogramowania

Iwona Dubielewicz

iwona.dubielewicz@pwr.edu.pl

Pok.201/16 bud.D-2, konsult. Środa 11-13

Katedra Inżynierii Oprogramowania, Wydział Informatyki i Zarządzania, 2017/2018

Politechnika Wrocławska

Zakres wykładu

Wytwarzanie oprogramowania – perspektywa inżynierska

- **Procesy tworzenia oprogramowania**: cykl życia oprogramowania; modele, metodyki
- **Wymagania**: analiza i modele specyfikacji
- **Konstrukcja systemu i oprogramowania**
- **Jakość oprogramowania**: testowanie, zarządzanie konfiguracją i zmianami

Literatura:

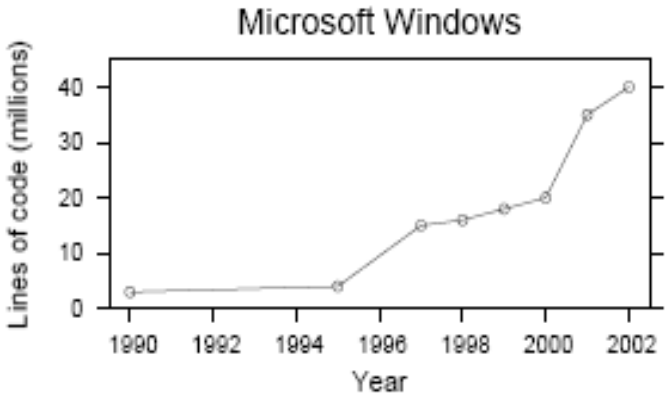
[1] Sacha K., Inżynieria oprogramowania, WNT 2009
[2]Patton J. Mapowanie historyjek użytkownika…,Helion 2016
[3] Fowler M., UML 2.0 w kropelce, Micom 2006
[4] Wrycza&al., UML 2.x, Ćwiczenia zaawansowane, Helion 2012
[4] Cocburn A., Jak pisać efektywne przypadki użycia, WNT, 2004.
[5] Adzic G. , Specyfikacja na przykładach, Helion, Gliwice 2014
[7] **OMG® Unified Modeling Language®** (OMG UML®)
http://www.omg.org/spec/UML/2.5.1

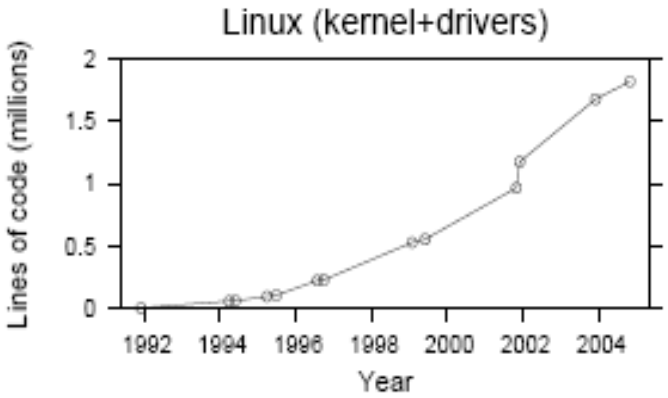
Wykład 1

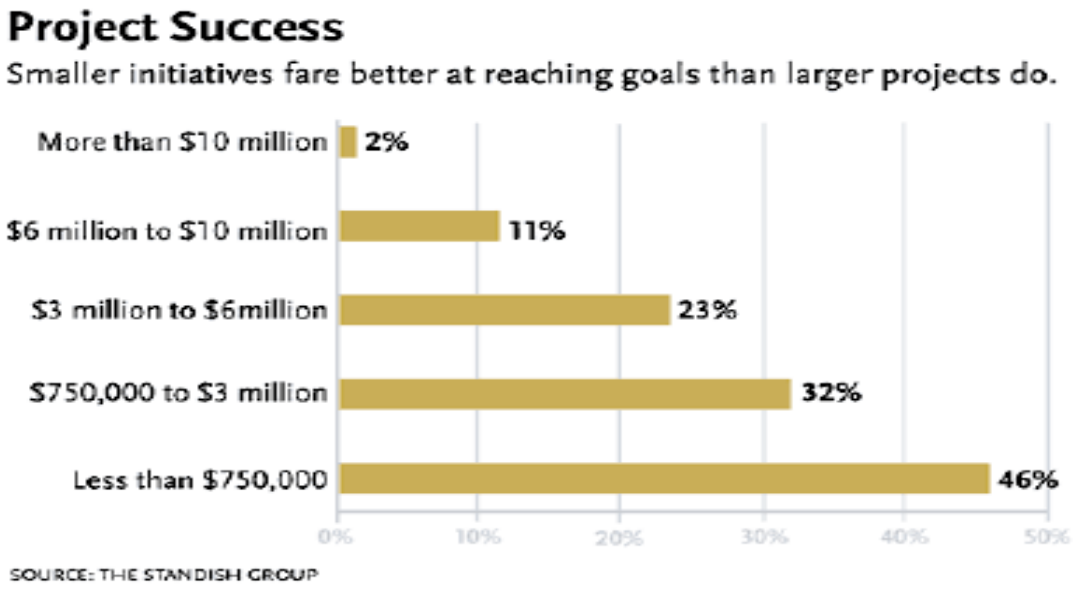
1. Wprowadzenie; dlaczego inżynieria?
2. Pojęcia podstawowe:
   - Modele cyklu życia oprogramowania
   - Metodyki wytwarzania oprogramowania
   - Artefakty (produkty) i języki ich specyfikacji

Kryzys wytwarzania oprogramowania

Wzrost rozmiaru oprogramowania







Historia sukcesów projektów informatycznych

- Badanie 280 000 projektów wytwórczych w 2000 roku

Tylko 28% projektów zostało pomyślnie zakończonych

- Badanie 9236 projektów wytwórczych przeprowadzone w 2004 roku

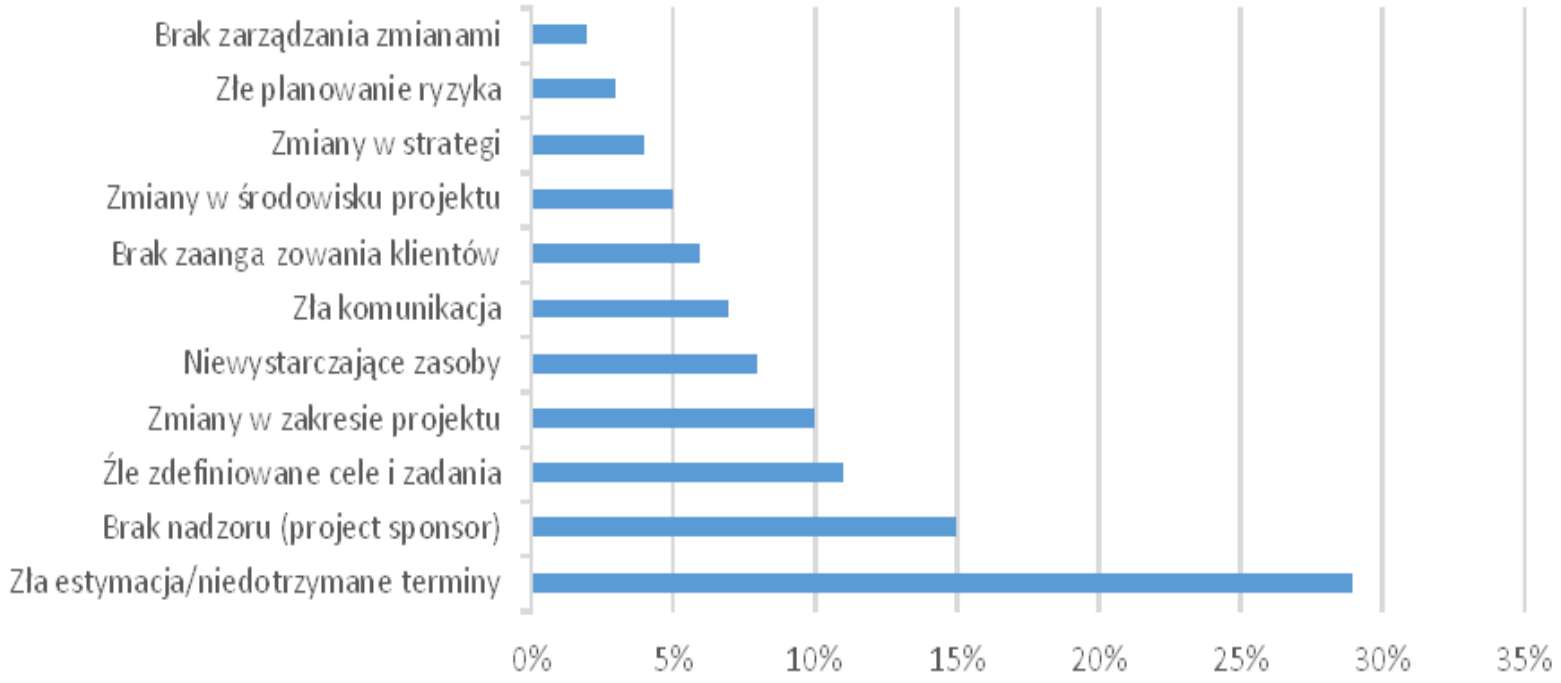
Tylko 29% projektów zostało zakończonych pomyślnie

- Badanie zrealizowane w 2006 roku

Tylko 35% projektów wytwórczych zostało pomyślnie zakończonych

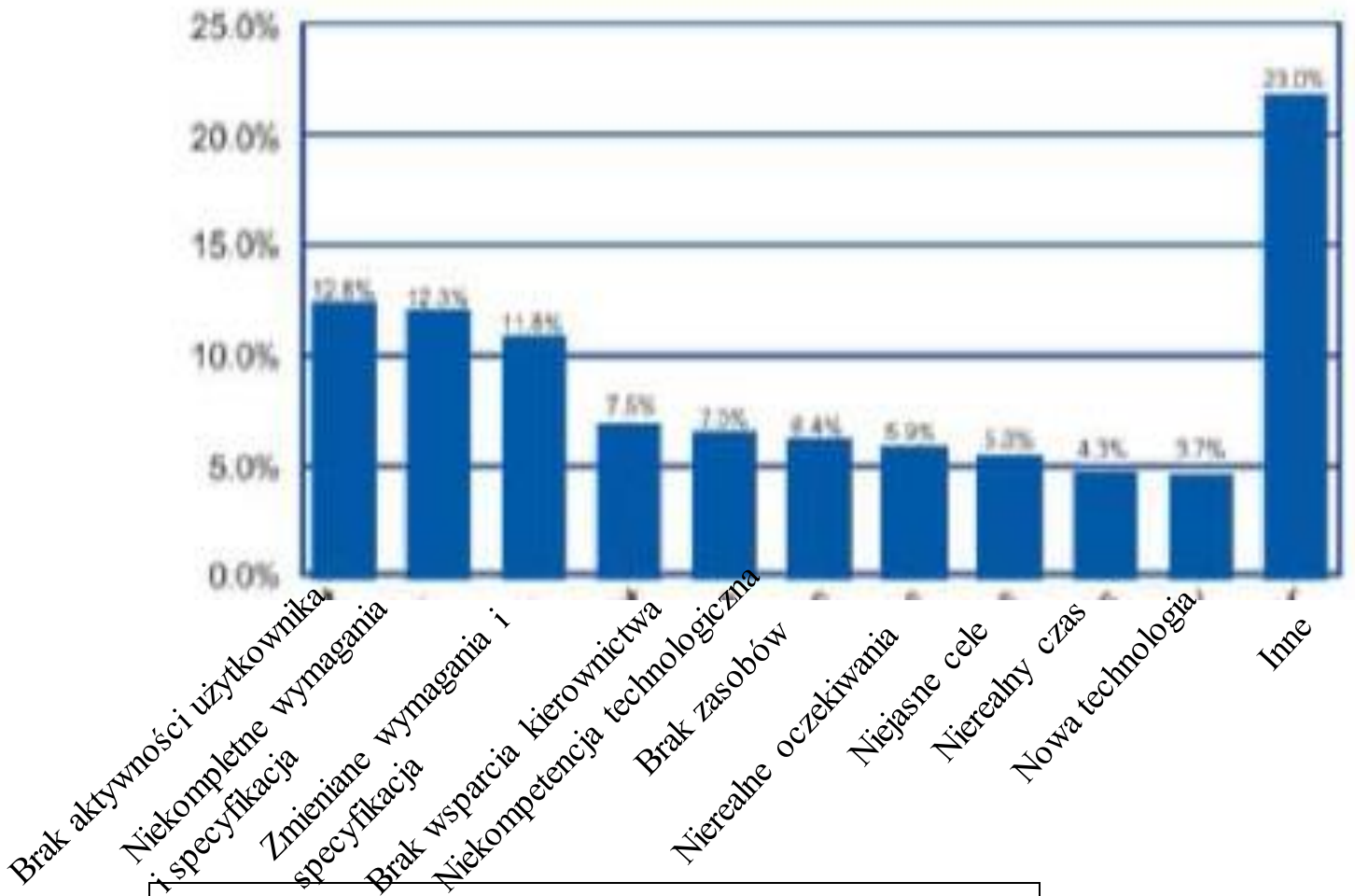
– (dzięki stosowaniu inżynierii oprogramowania?)

Przyczyny niepowodzeń



[Standish 2012]

Przyczyny niepowodzeń *cd.*



Projekty informatyczne – charakterystyka

- Oprogramowanie jako produkt (jakość, wydajność, użyteczność, pielęgnacja, ewolucja)
- Oddzielenie wymagań od projektowania i implementacji
- Zarządzanie projektem: personelem, czasem, budżetem
- Procesy wytwarzania
  Sekwencyjne
  Iteracyjne uszczegóławianie
  Przyrostowe (Agile)

Proces wytwarzania oprogramowania **fundamentalne założenia**

- Dobre procesy prowadzą do **dobrych produktów**
- Dobre procesy redukują **ryzyko**
- Dobre procesy polepszają **zrozumiałość**
- Dobre procesy umożliwiają **pracę zespołową**

INŻYNIERIA ***wiedza matematyczna i przyrodnicza*** *zdobyta podczas studiowania oraz wynikająca z doświadczeń i praktyki, która jest stosowana do opracowania takich* ***metod działania****, by wykorzystać* ***ekonomicznie*** *materiały i naturę,* ***na rzecz dobra człowieka***

OPROGRAMOWANIA* ***zestaw programów komputerowych, procedur i dokumentacji****;*

•pojęcie Inżynieria Oprogramowania (IO) pojawiło się 1968 r;

'podejście inżynierskie' to reakcja na tzw. *kryzys oprogramowania*

Inżynieria oprogramowania- charakterystyka

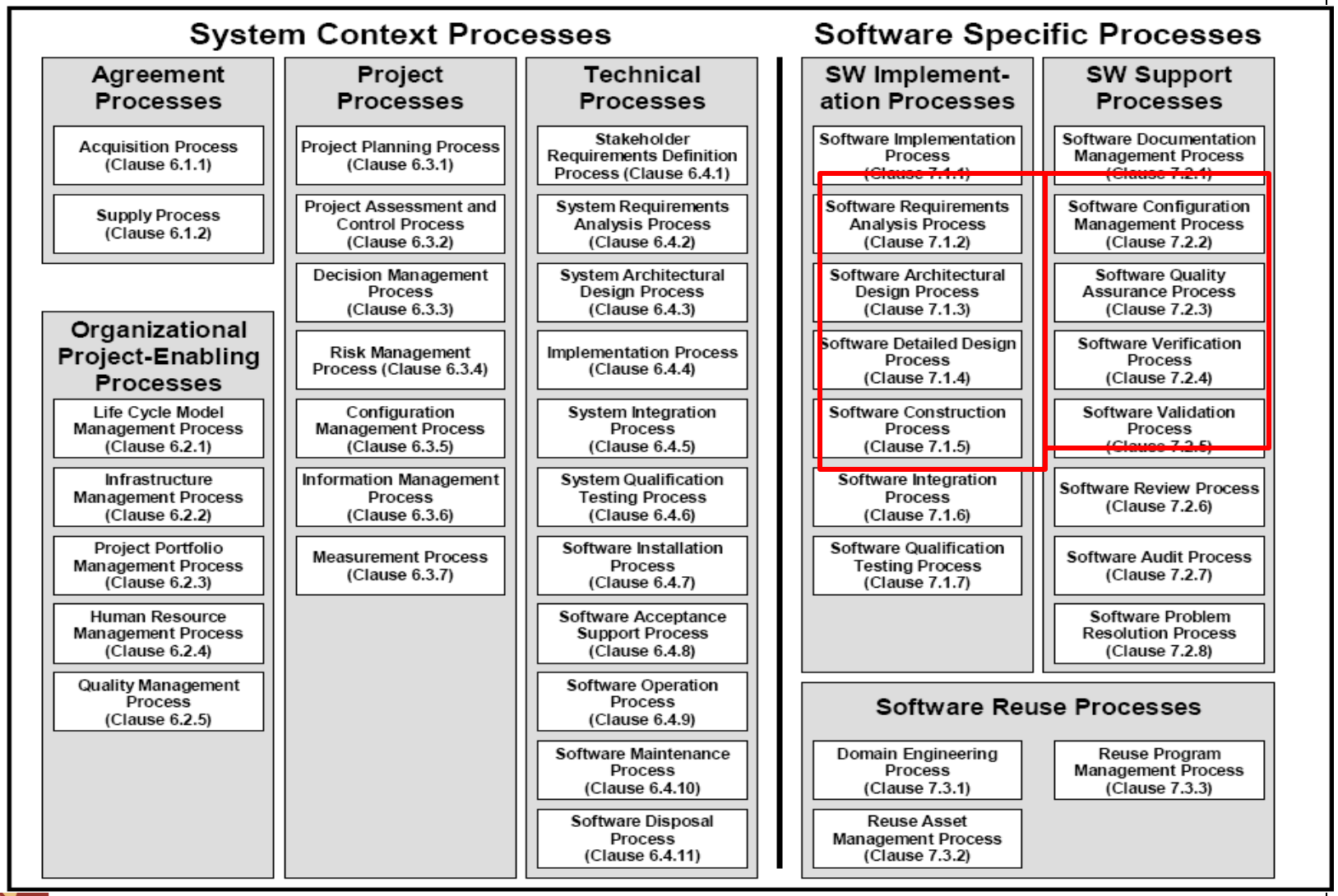
jest dyscypliną techniczną, której celem jest

- **produkcja bezawaryjnego oprogramowania**,
  - dostarczonego na czas,
  - w ramach ustalonego budżetu,

  które **spełnia potrzeby użytkownika**;

oprogramowanie wytworzone musi być łatwe do modyfikacji, gdy potrzeby użytkownika ulegną zmianie.

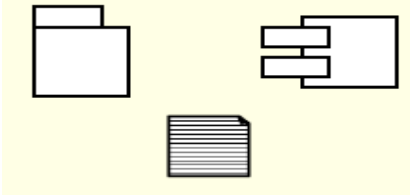
Zakres inżynierii oprogramowania jest bardzo szeroki.



[ISO12207:2007]

4 x P – elementy IO

Produkt (co)

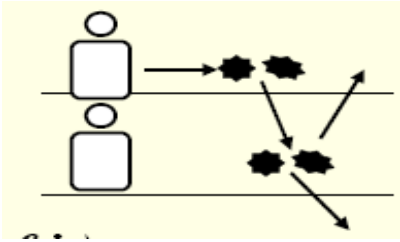


Proces

(jak)

Przedsięwzięcie

(realizacja)



Personel

(kto)

PRODUKT = OPROGRAMOWANIE

wykonuje **funkcje (usługi)** i ma **jakość (**realizuje potrzeby**)**

Cykl życia oprogramowania

obejmuje **okres życia oprogramowania:**

-od pojawienia się **koncepcji** (potrzeby) posiadania oprogramowania,

-poprzez **budowę**

-i **stosowanie**

- do momentu jego **wycofania** z użycia.

Model cyklu życia oprogramowania

to sposób **organizacji wybranych procesów** cyklu życia oprogramowania i wykonywanych (w ich ramach) czynności.

Zwykle rozważa się model dla procesów:

- wytwórczych
- zarządczych

Modele cyklu życia oprogramowania - klasyfikacja

kryterium: kompletność wymagań (ISO 12207):

- **kaskadowy**
- **przyrostowy**
- **ewolucyjny** (*syn*. **prototypowanie)**

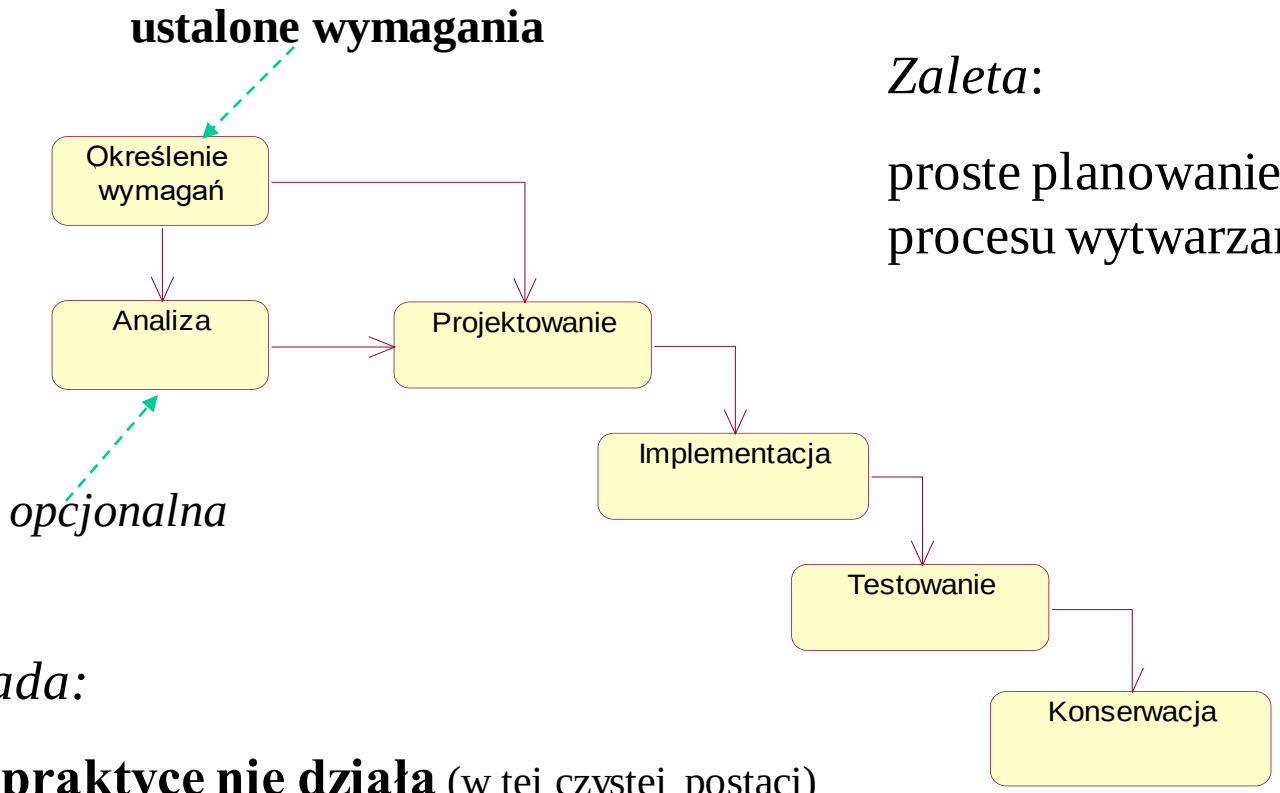
kryteria dodatkowe:

-iteracyjny *powtarzalność działań*

-spiralny *zarządzany ryzykiem*

Politechnika Wrocławska

Model kaskadowy (1970, Royce)



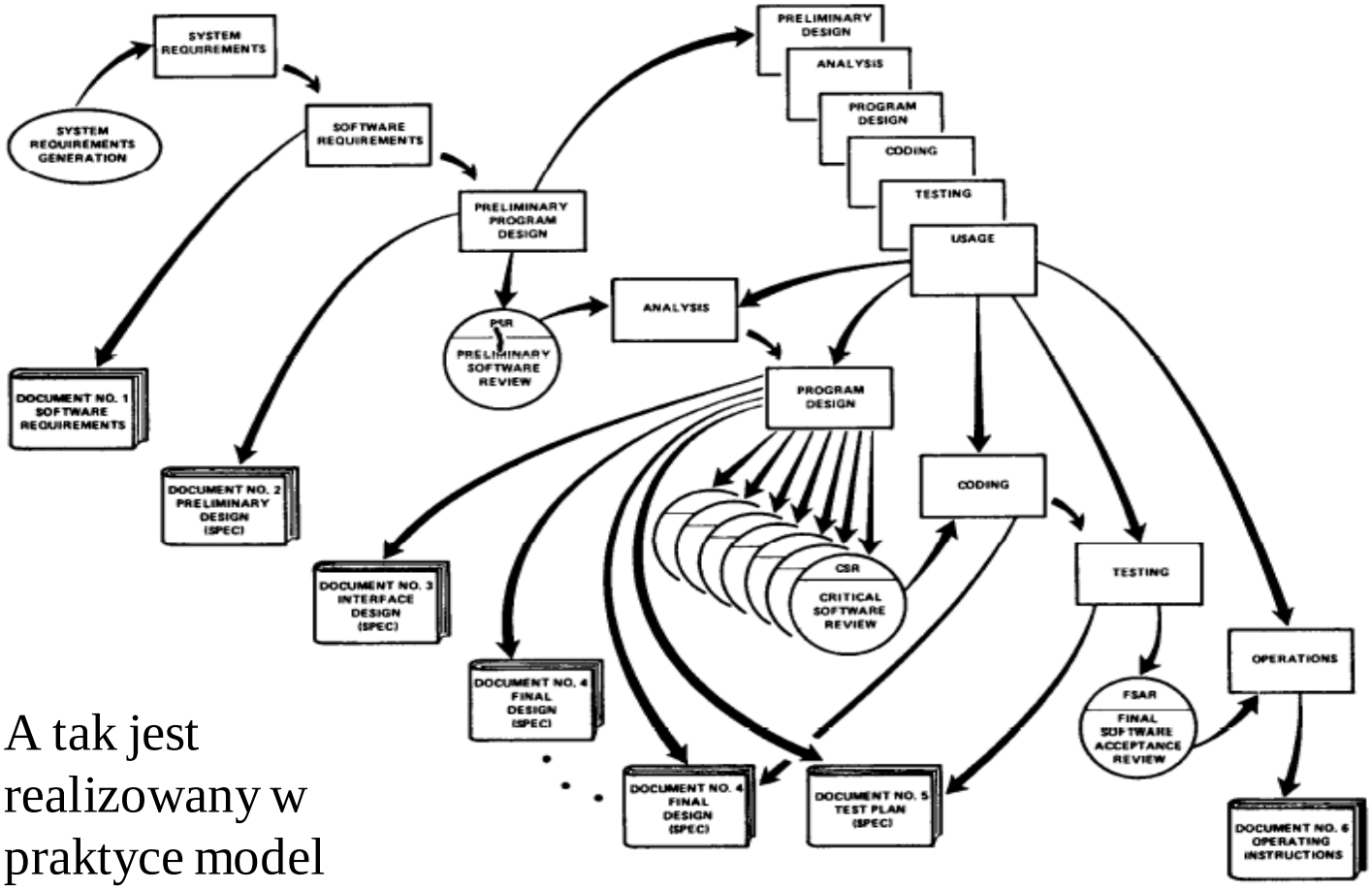
Zaleta:

proste planowanie procesu wytwarzania

Wada:

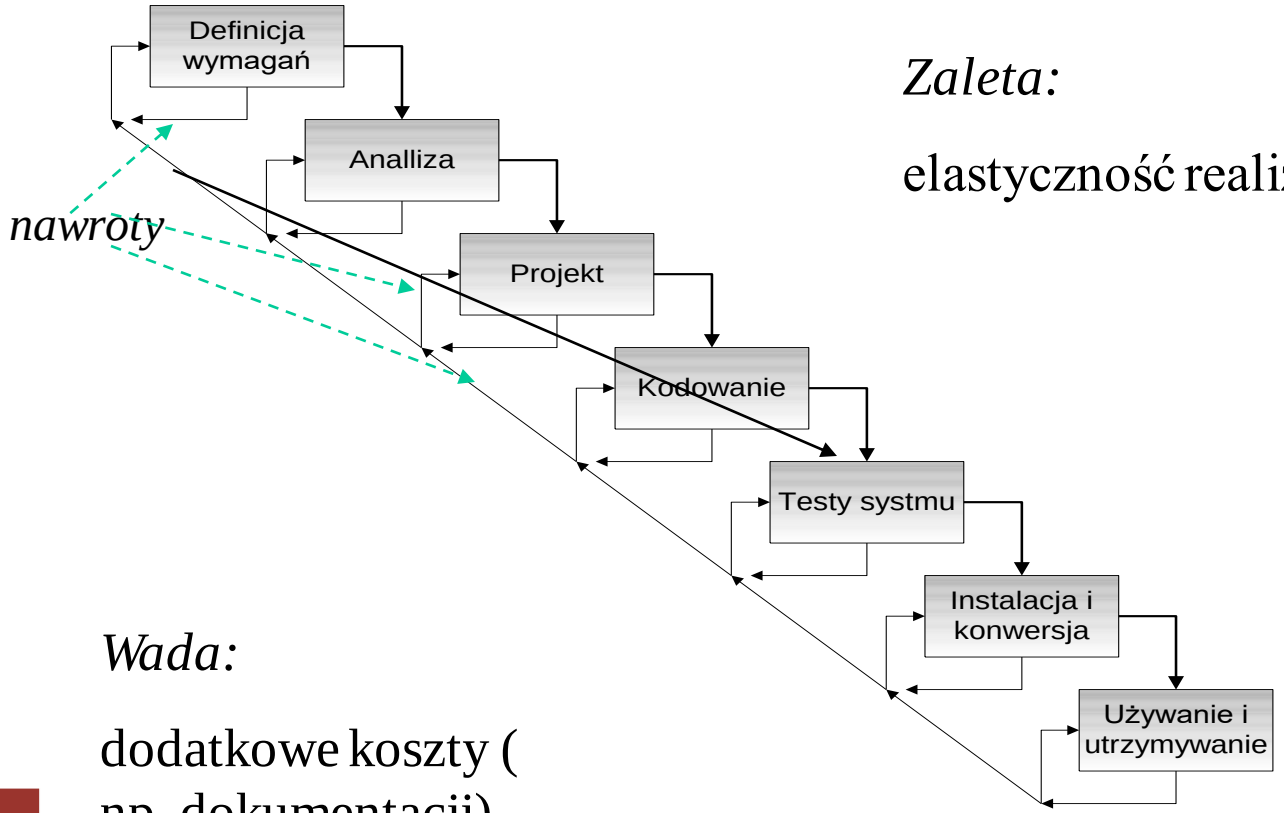
w praktyce nie działa (w tej czystej postaci)

Model kaskadowy + (*Royce 1973*)



A tak jest realizowany w praktyce model kaskadowy

Model kaskadowy +



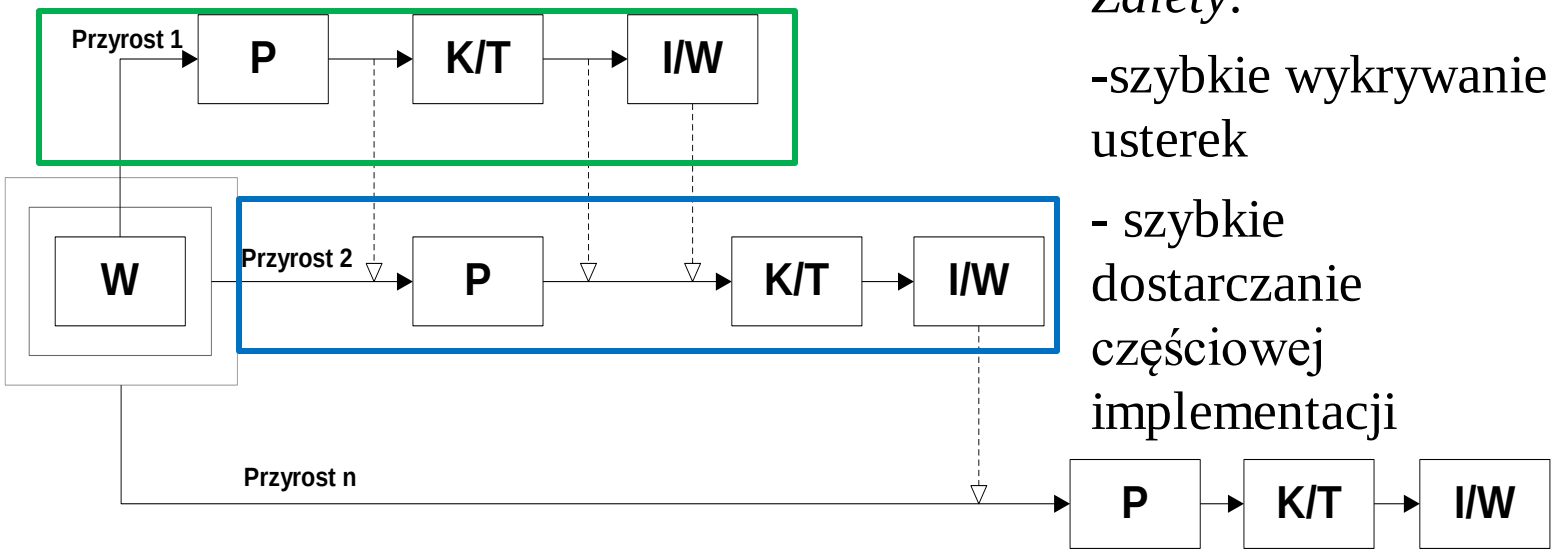
Zaleta:

elastyczność realizacji

Wada:

dodatkowe koszty (np. dokumentacji)

Model przyrostowy



Zalety:

-szybkie wykrywanie usterek

- szybkie dostarczanie częściowej implementacji

Możliwy przepływ informacji

W - wymagania

K/T - kodowanie i testy

P - projekt

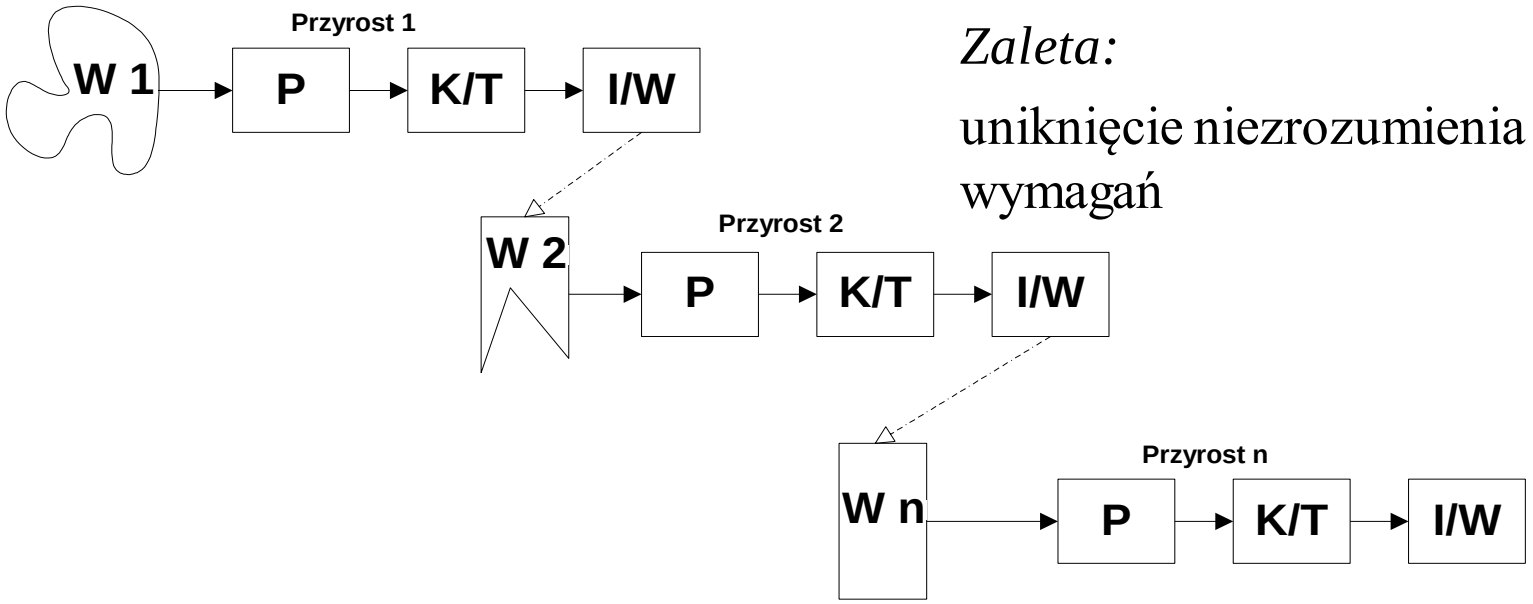
I/W - instalacja i wsparcie

Wady:

- trudniejsze planowanie
- dodatkowe koszty

wg ISO12207

Model ewolucyjny (prototypowania)



Zaleta:

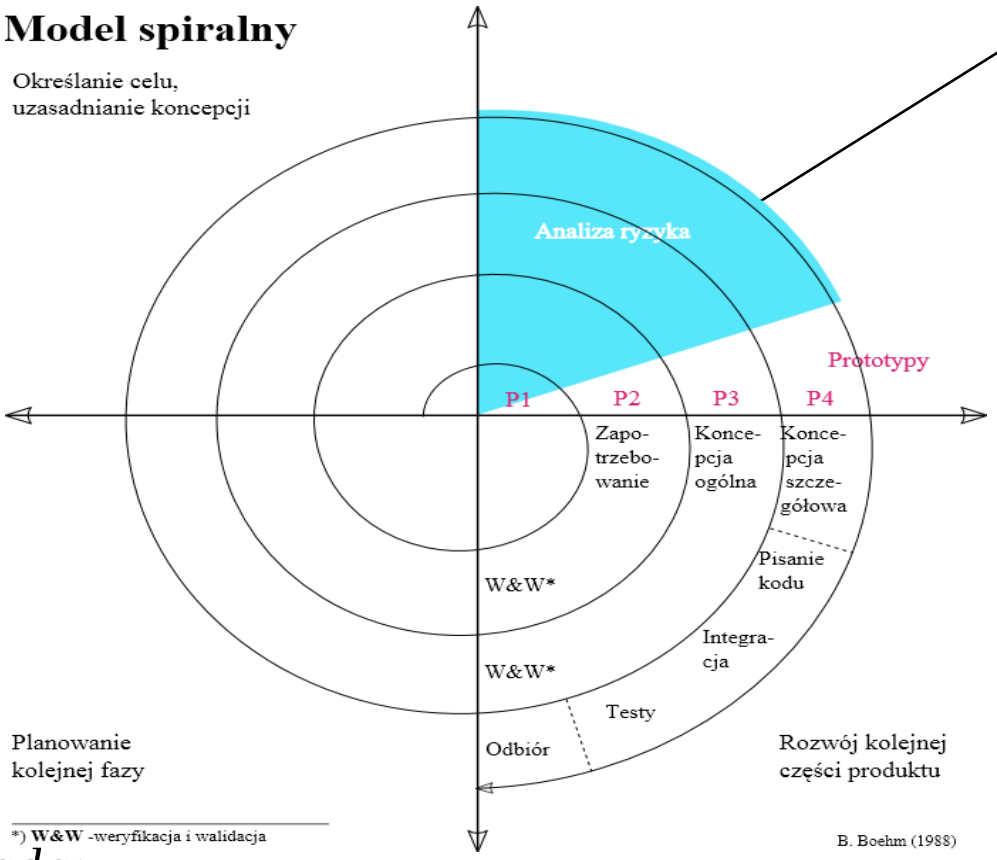
uniknięcie niezrozumienia wymagań

Wada:

koszty (budowanie wielu rozwiązań)

[ISO12207]

Model spiralny



Zaleta:

szybka identyfikacja ryzyka

Wada:

koszty!!

[Boehm,1988]

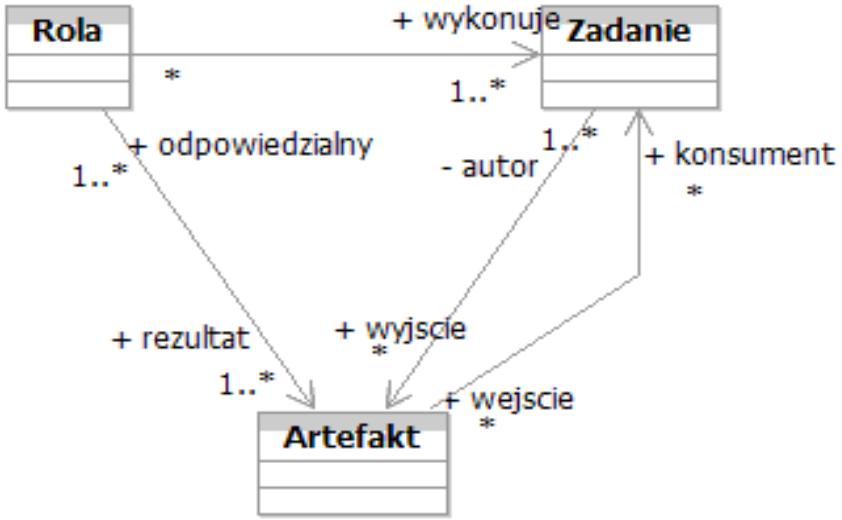
Metodyka wytwarzania oprogramowania

- to proces systematycznie wykonywany w celu wytworzenia oprogramowania (np. dobór pracowników, zadań, zasady współpracy, procedury postępowania);
- każda firma wytwórcza ma swoją własną, niepowtarzalną metodykę.

Metodyka wytwarzania oprogramowania

definicja wykorzystuje trzy podstawowe pojęcia:

- rola (ang. *role*),
- zadanie (ang. *task*),
- produkt pracy (ang. *work product*).



[SPEM 2008]

Atrybuty metodyki

Zakres metodyki– określa, które fazy wytwarzania oprogramowania są objęte metodyką, jakie role i aktywności metodyka definiuje.

Rozmiar metodyki– jest liczbą typów elementów kontrolowanych w metodyce (wytwarzany artefakt,, miara jakości, opis techniki, itd.).

Ceremoniał metodyki (stopień formalizacji) – określa wymagany stopień precyzji, nakładu pracy i stopnia formalizacji dokumentów (duży, średni, mały).

Waga metodyki – to „koncepcyjny iloczyn" rozmiaru i ceremoniału dla poszczególnych elementów kontrolnych.

Widzialność– łatwość oceny, czy projekt jest wykonywany zgodnie z metodyką.

Kategorie metodyk wytwarzania:

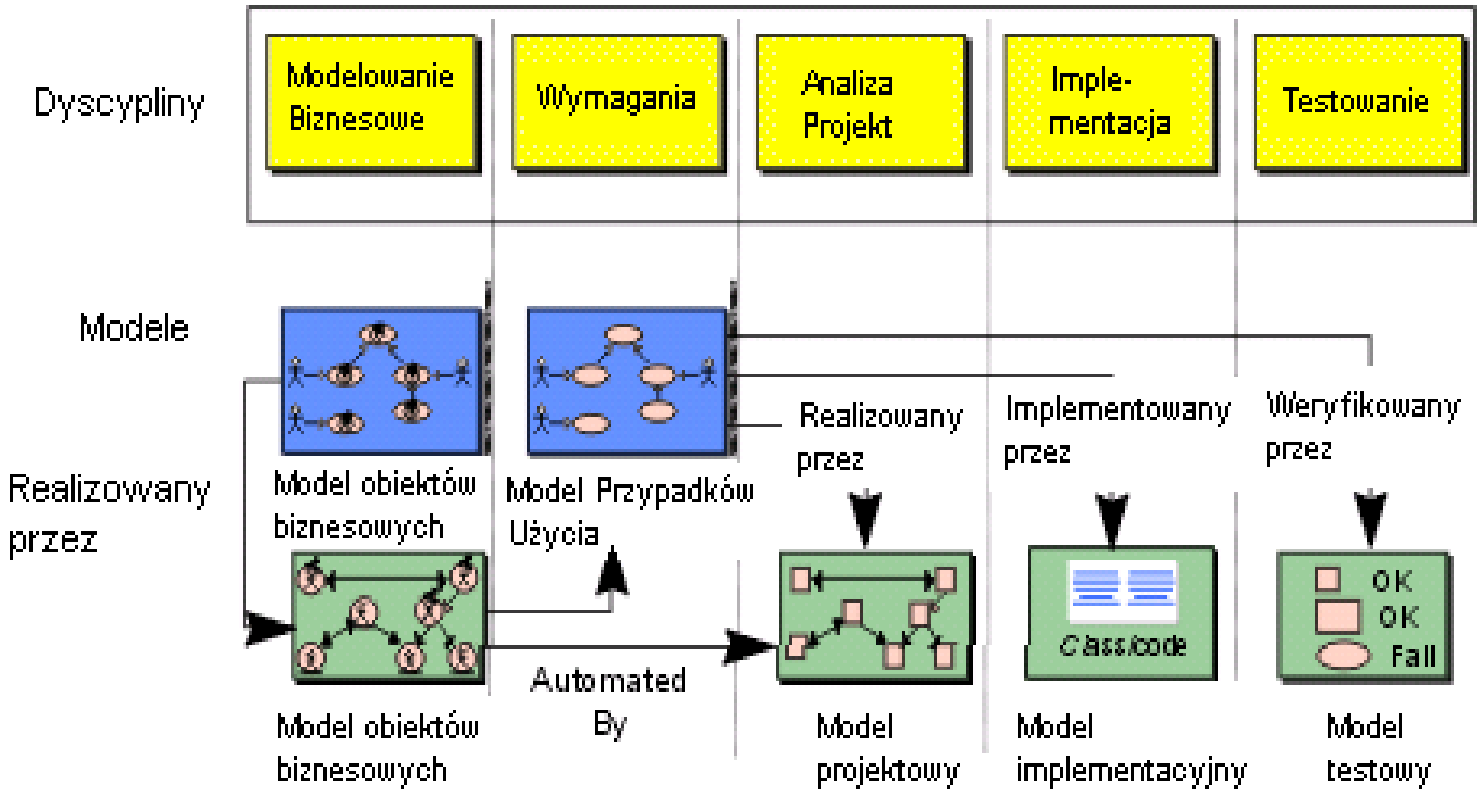
wg paradygmatu projektowania/programowania

- strukturalne
- **obiektowo-zorientowane**

wg złożoności

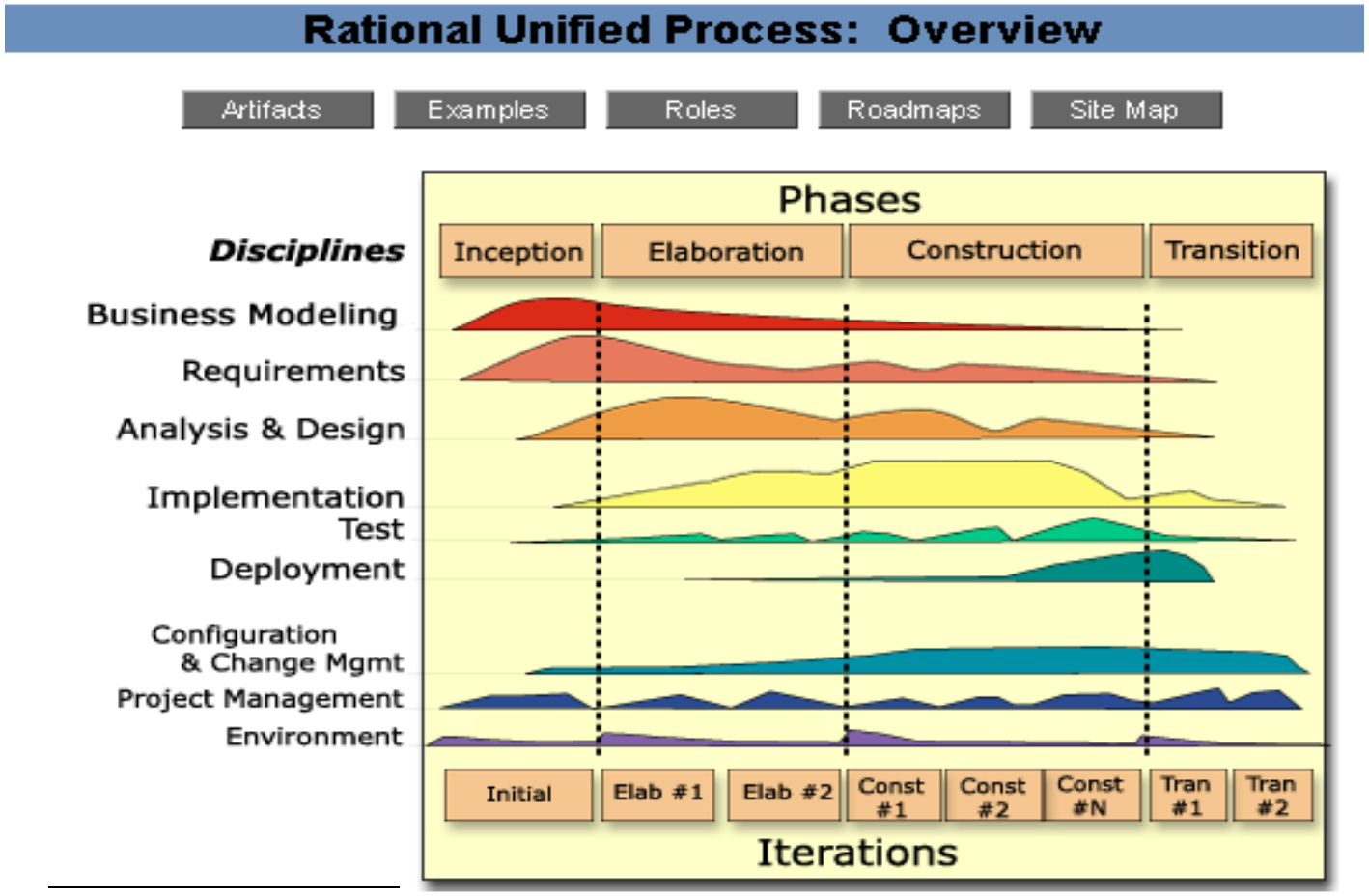
- tradycyjne/ciężkie: duży ceremoniał
- elastyczne/ zwinne: mały ceremoniał

Metodyki sterowane modelami



[RUP 2003]

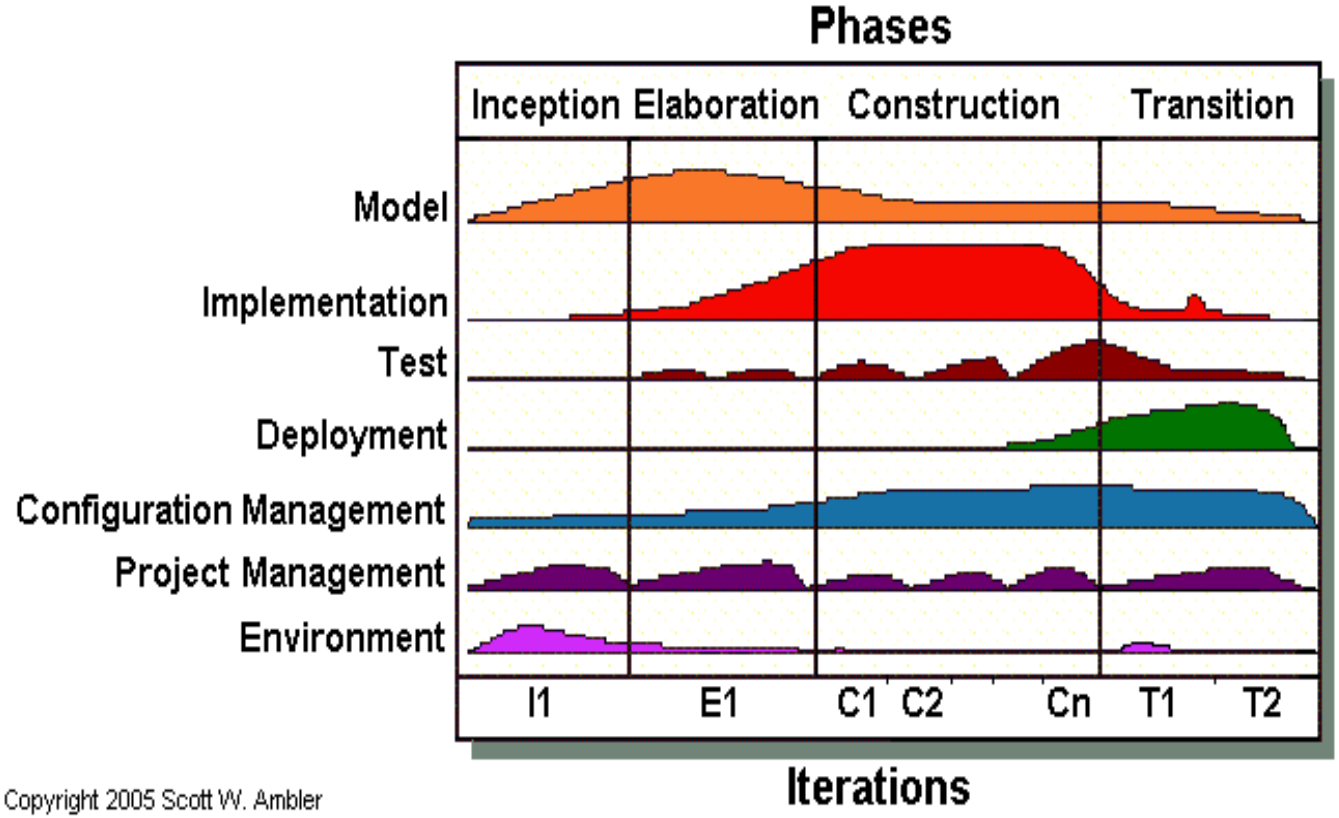
Metodyka RUP* – fazy i dyscypliny



*RUP- RationalUnified Process

[RUP,2003]

Metodyki AUP* v1.1 – fazy i dyscypliny

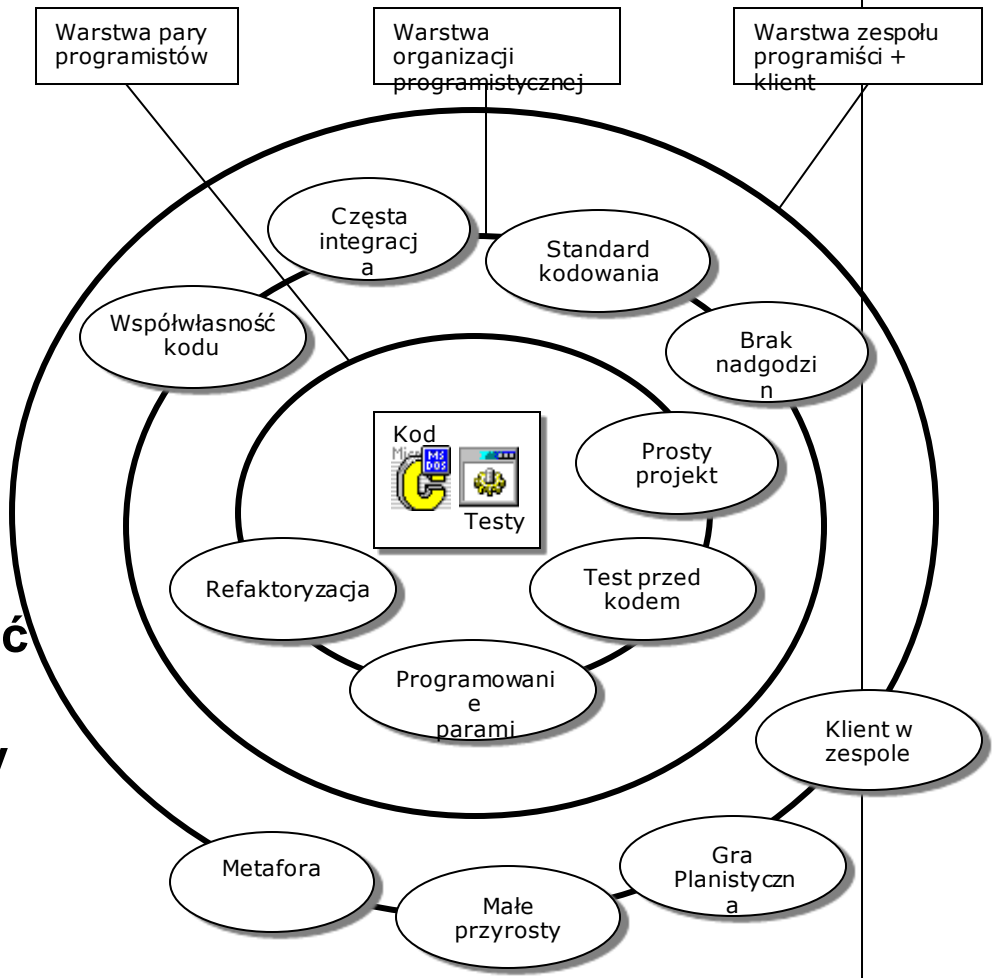


*AUP- Agile Unified Process; rodzina metodyk

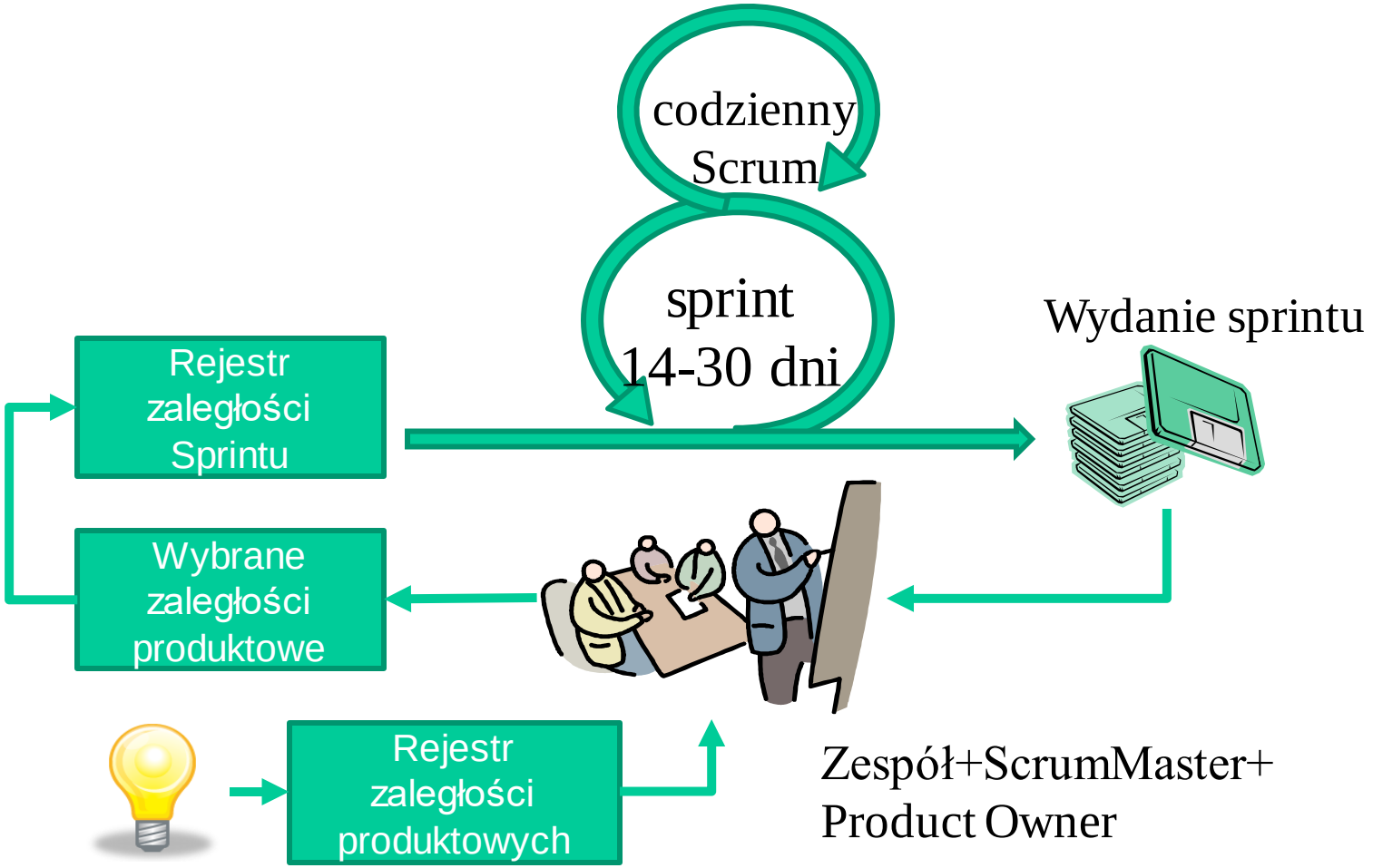
XP- przykład metodyki zwinnej

12 praktyk wg Kenta Becka:

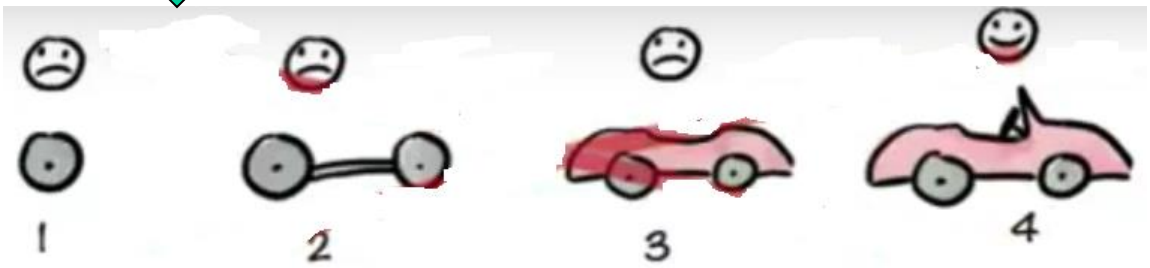
- **Planowanie**
- **Małe wydania**
- **Metafora systemu**
- **Prosty projekt**
- **Ciągłe testowanie**
- **Przerabianie**
- **Programowanie w parach**
- **Standard kodowania**
- **Wspólna odpowiedzialność**
- **Ciągłe łączenie**
- **40-godzinny tydzień pracy**
- **Ciągły kontakt z klientem**



SCRUM- przykład metodyki zwinnej



Agile vs tradycyjne podejście do wytwarzania



Model - charakterystyka

- Model jest *uproszczonym* opisem rzeczywistości
- Modele buduje się by lepiej zrozumieć budowany system
- Model dla systemów złożonych jest konieczny, bo nie jesteśmy w stanie zbudować całości poprawnie
- Modele mogą być formalne lub nieformalne.

Model

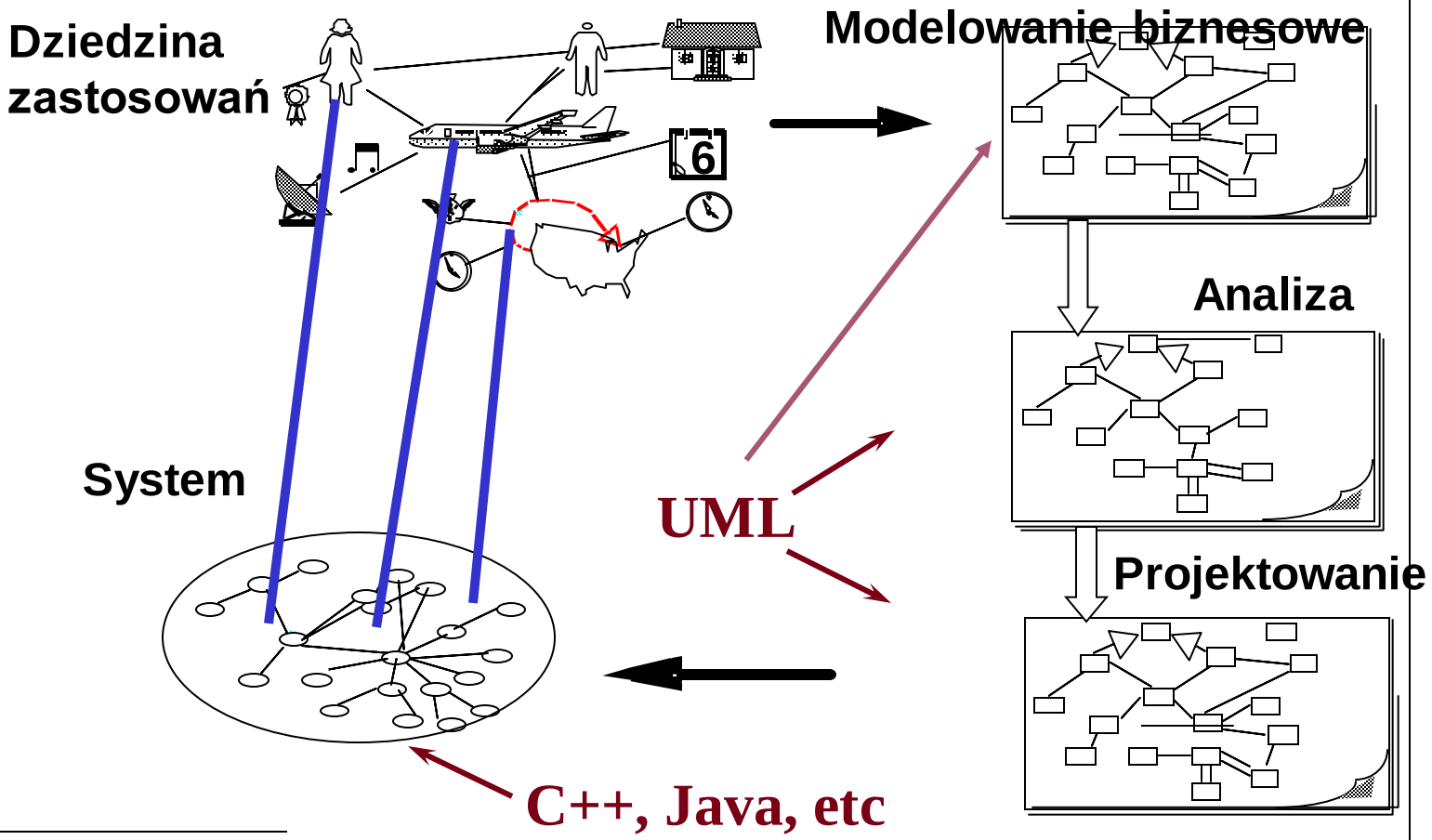
Po co model?

- wychwytuje i precyzyjnie wyraża wymagania
- pozwala zrozumieć i uchwycić decyzje projektowe
- organizuje różne elementy projektowe
- pozwala przebadać (ekonomicznie) wiele rozwiązań
- opisuje jasno systemy złożone

Poziomy modeli

- Przewodnik procesu myślowego - modele na wysokim szczeblu
- **Abstrakcyjna specyfikacja bazowej struktury**
- Pełna specyfikacja systemu końcowego
- Przykłady typowych systemów
- Całkowite lub częściowe opisy interakcji

Język UML* w obiektowym procesie wytwarzania oprogramowania



*UML - Unified Modeling Language

Język UML

Abstrakcyjny

Pozwala wyrazić najważniejsze zagadnienia z pominięciem zbędnych szczegółów

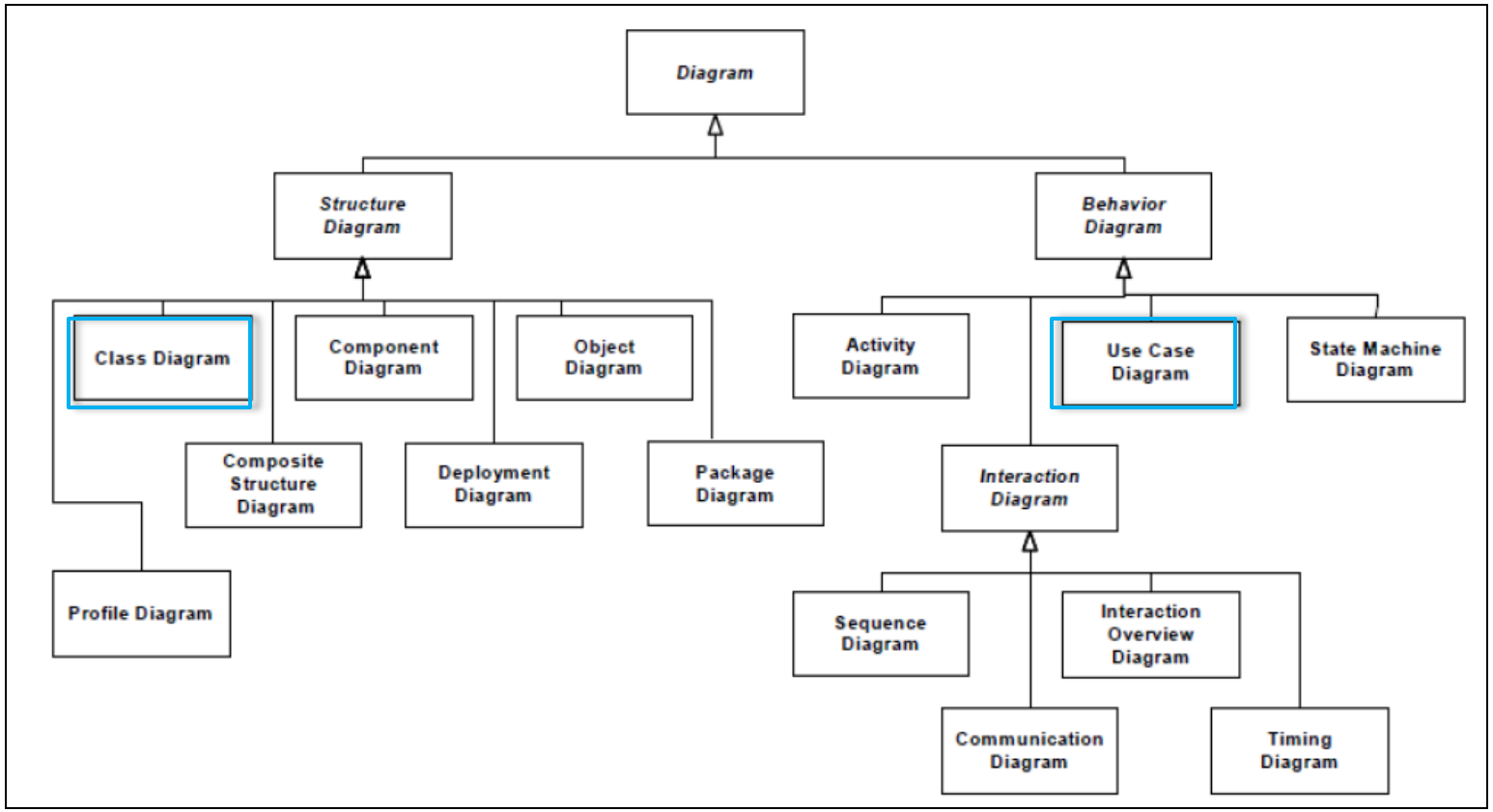
Precyzyjny

Pomaga znaleźć braki oraz niespójności

Graficzny

Pozwala zilustrować obiekty oraz związki zachodzące między nimi

Język UML - diagramy



Diagramy są graficzną reprezentacją fragmentów modelu

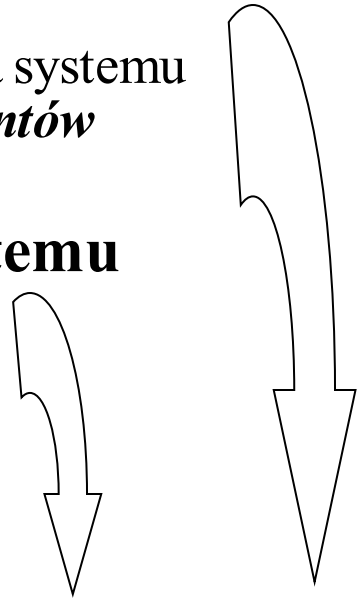
Język UML w budowie modeli oprogramowania

Modelowanie statyczne

- opisuje elementy i strukturę wnętrza systemu
- ***diagramy klas, obiektów, komponentów i rozmieszczenia***

Modelowanie funkcjonalności systemu

- opisuje co nowy system ma robić (lub co robi system istniejący)
- ***diagramy przypadków użycia***



Modelowanie dynamiczne

- opisuje zachowanie systemu w terminach interakcji elementów systemu

- diagramy aktywności, stanów, sekwencji i współdziałania

Podsumowanie

- Cykl życia oprogramowania obejmuje następujące etapy: [modelowanie biznesowe], **specyfikowanie wymagań, [**analizę] **projektowanie, implementację, testowanie i pielęgnację**.
- Wytworzenie oprogramowania wysokiej jakości wymaga doboru i stosowania metodyki wytwórczej
- Język UML jest językiem modelowania i specyfikacji; może być stosowany na wszystkich etapach wytwarzania oprogramowania