



O budowie systemów informatycznych



Wymagania w projektach informatycznych

- Projekty programistyczne można podzielić na trzy rodzaje:
 - kierowane przez użytkownika
 - konsultowane z użytkownikiem
 - niezależne od użytkownika



Projekty kierowane przez użytkownika

- W projekcie ***kierowanym przez użytkownika*** on sam ustala swe wymagania; budowniczy oprogramowania jest tylko kontrahentem użytkownika, a wymagania są sprecyzowane w kontrakcie.
 - W ten właśnie sposób przebiega budowa systemów informatycznych dla potrzeb rządu Stanów Zjednoczonych. Agencje rządowe tworzą obszerną listę wymagań, stanowiących podstawę do przetargu wśród dostawców oprogramowania.



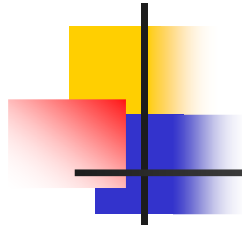
Projekty konsultowane z użytkownikiem

- W projekcie ***konsultowanym z użytkownikiem*** wymagania formułują twórcy oprogramowania (współpracujący jednocześnie z użytkownikiem).
 - W tego rodzaju projektach użytkownikowi przysługuje prawo zatwierdzenia wymagań, zazwyczaj w późniejszym stadium specyfikacji, a zwłaszcza specyfikacji zewnętrznych.



Projekty niezależne od użytkownika

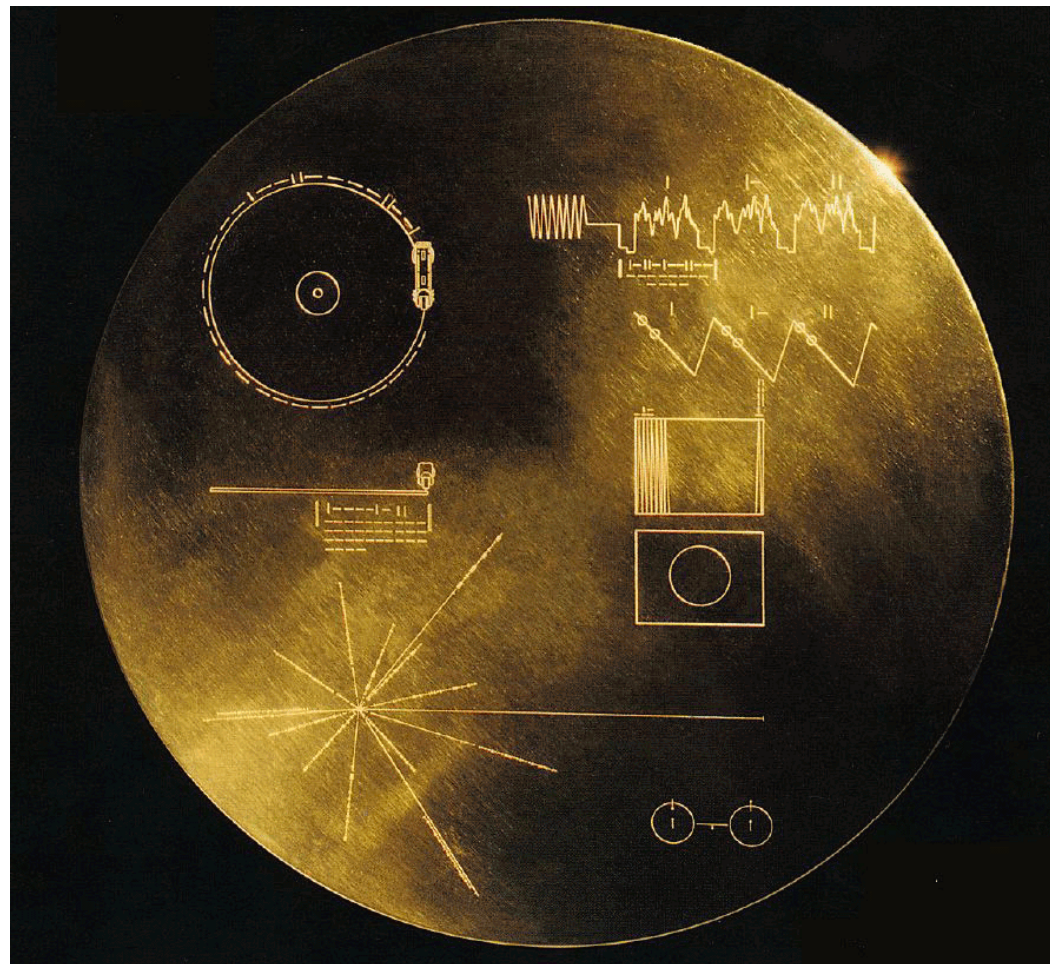
- W projekcie ***niezależnym od użytkownika*** cała odpowiedzialność za ustalenie wymagań spada na producenta programu.
 - Do tej grupy należy większość uniwersalnych systemów istniejących na rynku.



Czym jest modelowanie

- Model – uproszczenie rzeczywistości.
- Modele są budowane w celu lepszego zrozumienia systemu, który jest tworzony.
 - problem „sprzedania” modelu klientowi
- Modelowanie pomaga:
 - przedstawić budowany system,
 - specyfikować strukturę systemu i jego zachowanie,
 - dokumentować podjęte decyzje.

Ale z modelami nie jest tak
prosto...



Jak to z modelami bywa



How the customer explained it



How the Business Consultant described it



How the Analyst designed it

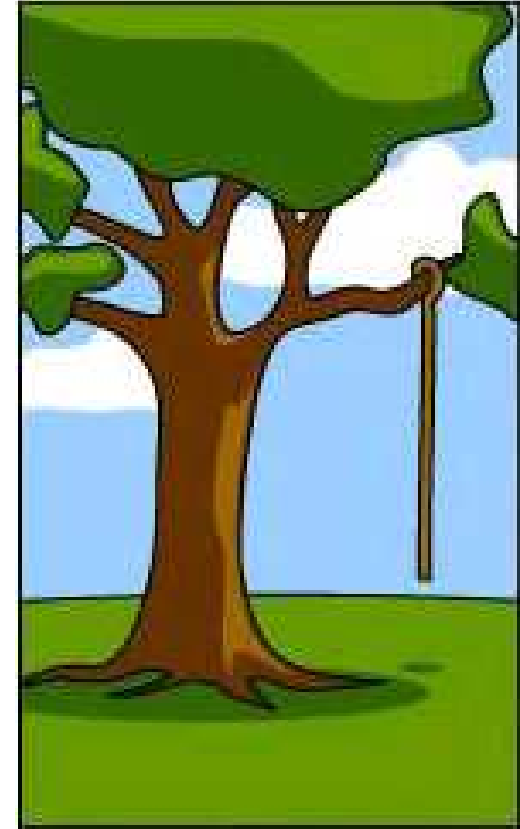
Jak to z modelami bywa



How the Project Leader understood it

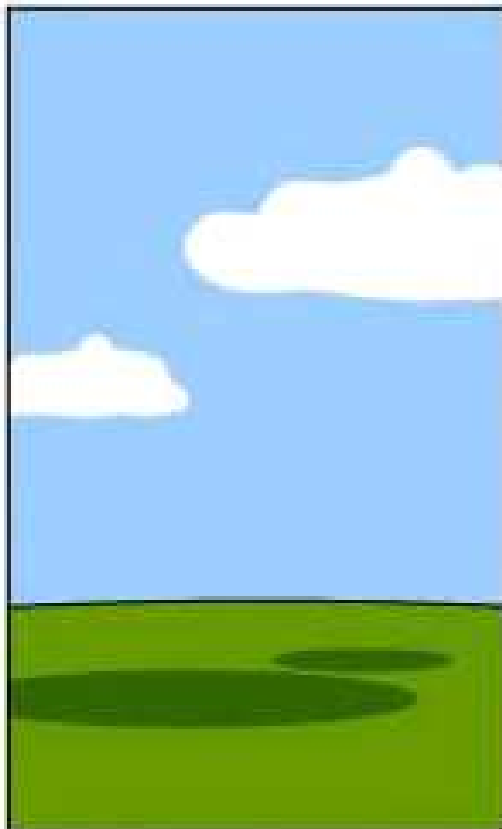


How the Programmer wrote it

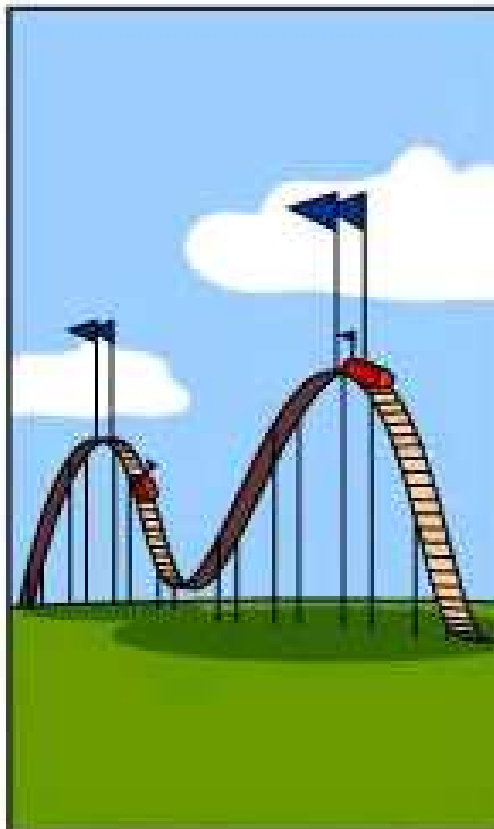


What operations installed

Jak to z modelami bywa



How the project was documented



How the customer was billed

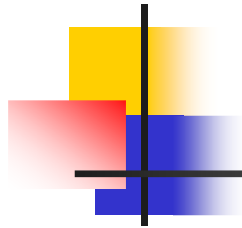


What the customer really needed



Jakie modele powstają?

- Skrócone (pewne byty są ukryte, żeby wszystko było prostsze – czasami aż za proste ☹).
- Niepełne (pewnych bytów może brakować).
- Niespójne (spójność modelu nie jest zapewniona).

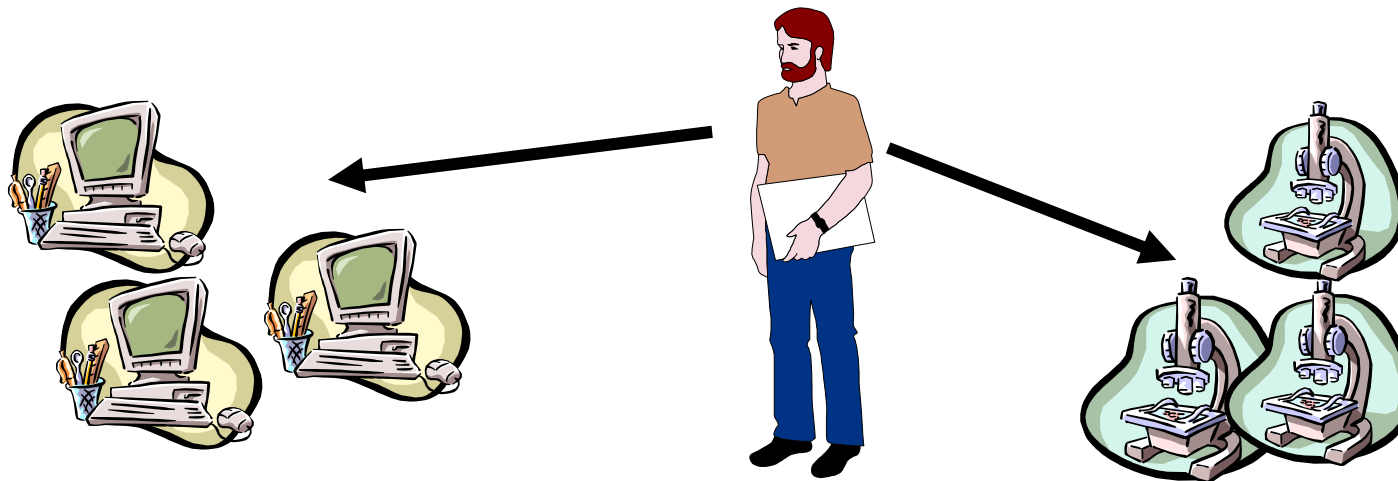


Czy jest abstrakcja?

- Abstrakcja polega na **eliminacji, ukryciu lub pominięciu mniej istotnych szczegółów rozważanego przedmiotu lub mniej istotnej informacji**; wyodrębnianiu cech wspólnych i niezmiennych dla pewnego zbioru bytów i wprowadzaniu pojęć lub symboli oznaczających takie cechy.
- Mechanizmy abstrakcji pozwalające na budowę abstrakcyjnych struktur i operowanie tymi strukturami bez wnikania w ich wewnętrzną budowę.

Co to jest obiektowość

- **Obiektowość** – jest to sposób myślenia oparty na kilku podstawowych zasadach;
 - Następuje zbliżenie paradygmatów: świata oprogramowania do świata rzeczywistego

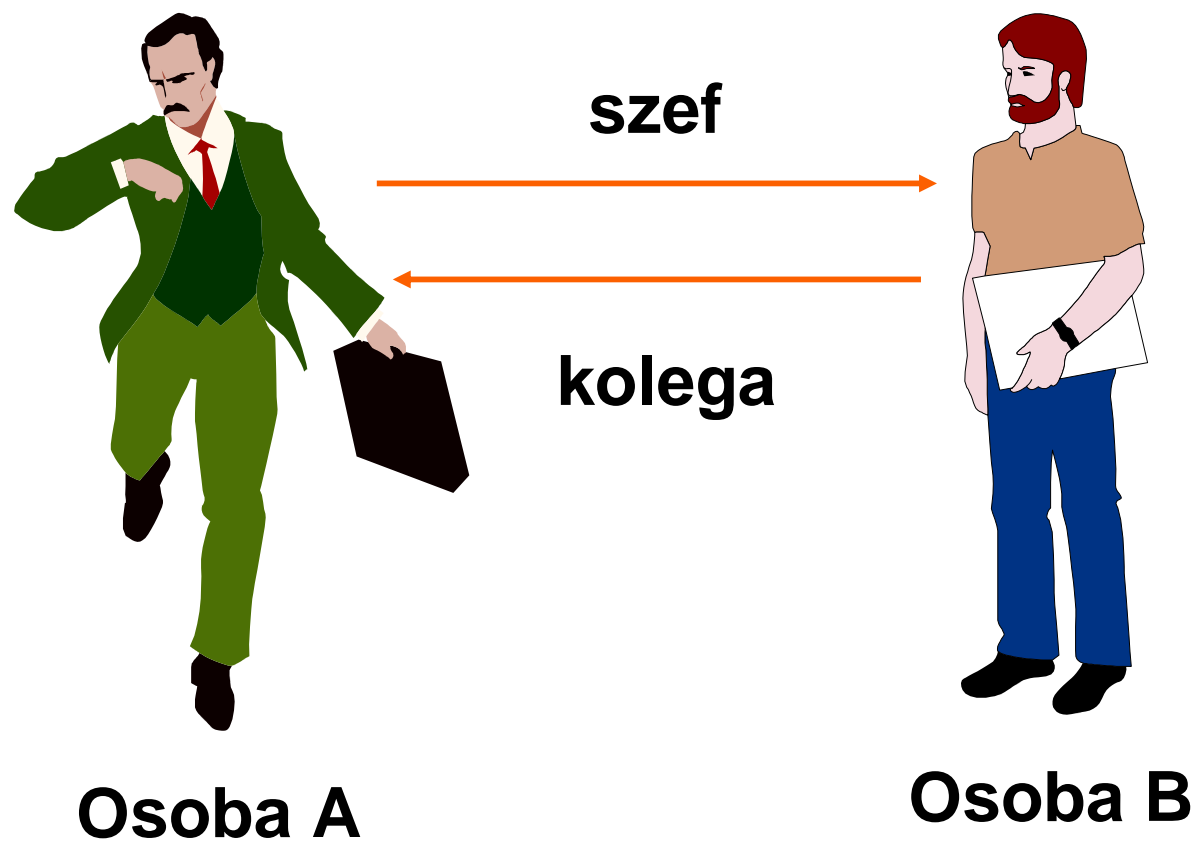




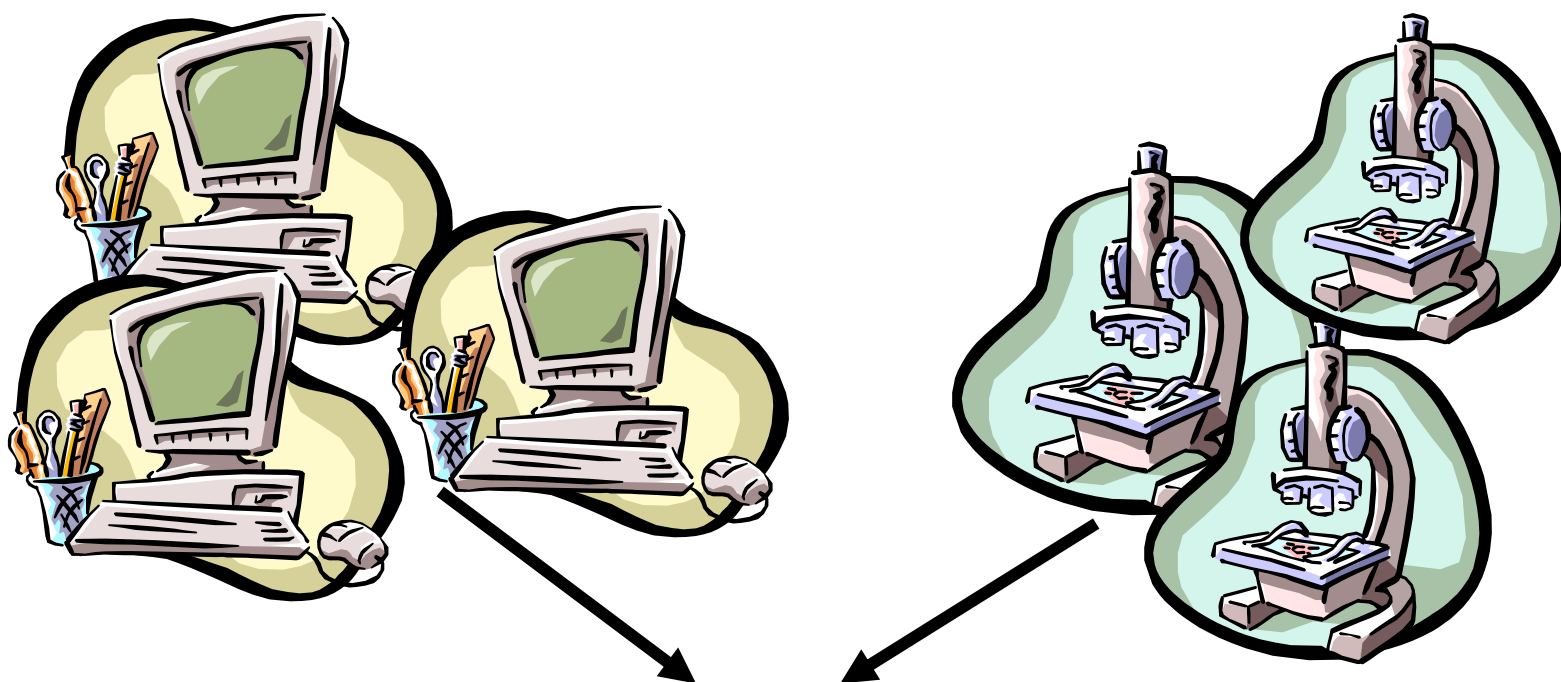
Obiekt i jego cechy

- **Obiekt** - struktura danych, występująca łącznie z operacjami dozwolonymi do wykonywania na niej, odpowiadająca bytowi wyróżnionemu podczas analizy.
 - Obiektem jest **rzecz** lub **pojęcie** występujące w świecie rzeczywistym.
 - Obiekt może być **złożony**, tj. może składać się z mniejszych obiektów.
 - Obiekt może być **powiązany** z innymi obiektami.
 - Obiekt posiada **stan**, który może zmieniać się w czasie.
 - Obiekty mogą **wysyłać komunikaty** między sobą.
 - Obiekt ma przypisane **zachowanie**, tj. zestaw operacji które wolno do niego stosować (implementacja operacji jest zwana metodą).
 - Obiekt ma przypisany **typ**, który określa dopuszczalną budowę obiektu oraz ustala operacje, które wolno wykonywać na obiekcie.

Przykłady obiektów



Przykłady obiektów



Czy mają wspólne właściwości?

Przykład klasy

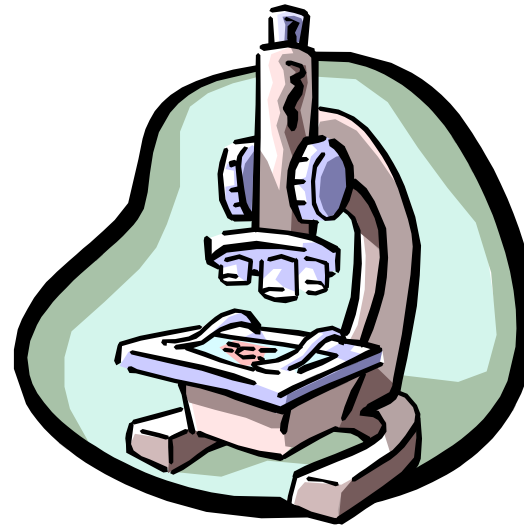
- Komputer:
 - Model
 - Waga
 - Zasilanie
 -
 - WłączUrządzenie()
 - WyłączUrządzenie()
 -



Przykład klasy

■ Mikroskop:

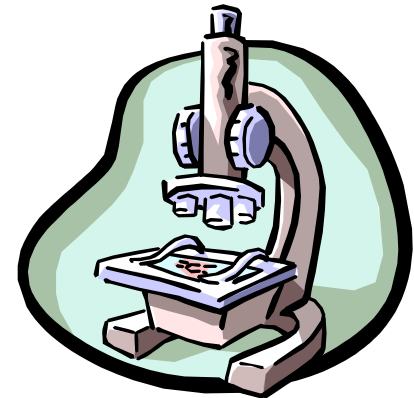
- Model
- Waga
- Zasilanie
-
- WłączUrządzenie()
- WyłączUrządzenie()
-



Przykład klasy

- Urządzenie:

- Model
- Waga
- Zasilanie
-
- WłączUrządzenie()
- WyłączUrządzenie()
-





Klasa, obiekt i ich cechy

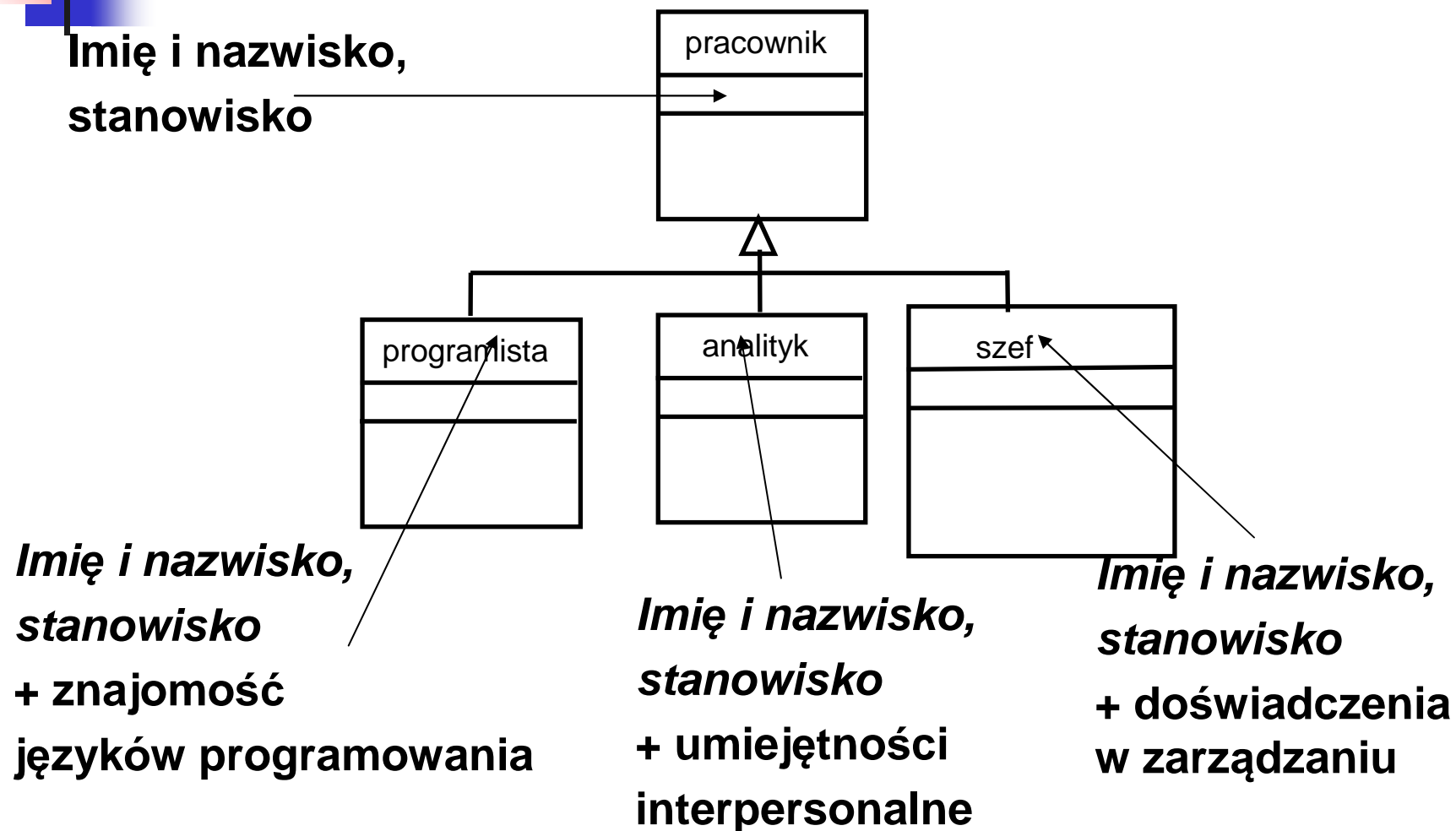
- **Klasa** - zgrupowanie obiektów o tych samych charakterystykach (atrybutach i operacjach - metodach).
 - Każdy obiekt należy do pewnej klasy obiektów.
 - Rodzaje klas:
 - Klasy abstrakcyjne – wzorce dla podklas
 - Klasy konkretne – wzorce dla tworzenia obiektów
 - Obiekt jest **egzemplarzem** klasy.

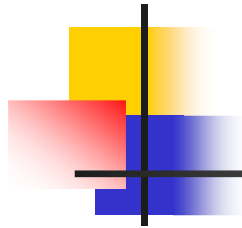


Obiekt i jego cechy

- **Dziedziczenie** (ang. inheritance) – związek pomiędzy klasami, w którym dana klasa przejmuje strukturę albo zachowanie zdefiniowane w innej klasie.
 - Wyróżnia się dziedziczenie proste i wielokrotne.
 - Dziedziczenie definiuje rodzaj hierarchii pomiędzy klasami, w której podklasa dziedziczy z jednej lub wielu klas nadrzędnych.
 - Podklasa zwykle powiększa definicje istniejących struktur i zachowań swojej klasy nadrzędnej.

Przykład dziedziczenia





Cele stworzenia UML

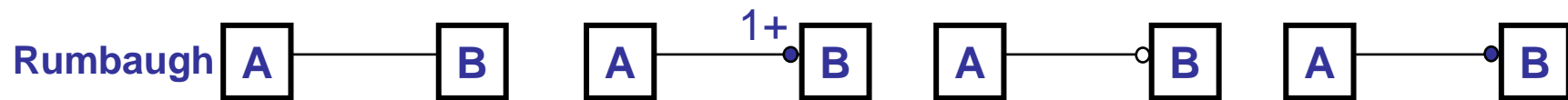
- Wizualny język modelowania do budowy i wymiany modeli.
 - Silny mechanizm rozbudowy.
- Niezależny od języków programowania i procesów tworzenia.
- Wspomaga wysoko-poziomowe rozwiązania.
- Zachęca (bardziej szczegółowo: to zadanie zostało już zrealizowane w praktyce) do wzrostu rynku narzędzi OO.



Początki UML (1a)

- Na przełomie lat 80-tych i 90-tych pojawiło się ponad 50 różnorodnych metodyk obiektowych m.in. Booch, OMT, OOSE/Objectory, Fusion, Coad/Yourdon.
- Poszczególni twórcy próbowali wykazać wyższość własnego rozwiązania nad pozostałymi, co niekorzystnie wpływało na przemysłowe próby zastosowania technologii obiektowych.
- Każda metodologia miała własną notację, otrzymywała wsparcie ze strony różnych narzędzi programistycznych, obejmowała różny zakres projektowania systemu informatycznego.

Początki UML (1b)



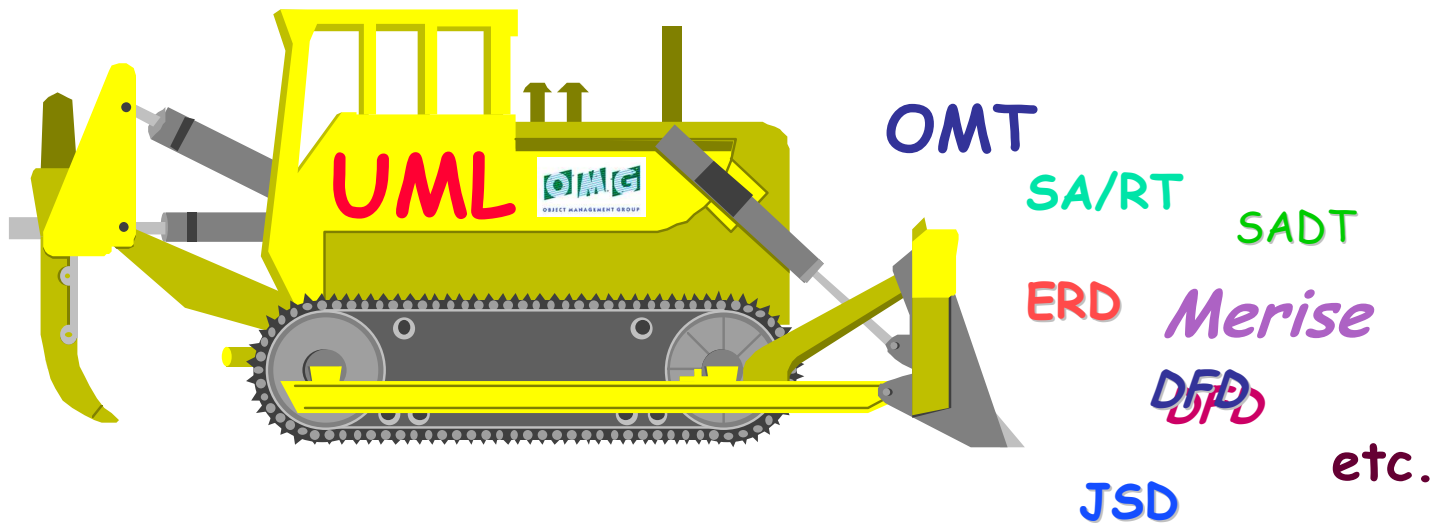
a A is always associated with a B.

a A is always associated with one or more B.

a A is associated with zero or one B.

a A is associated with zero or more B.

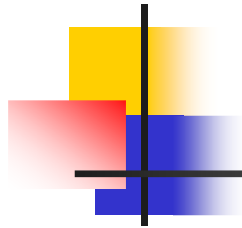
Początki UML (1c)





Stan obecny

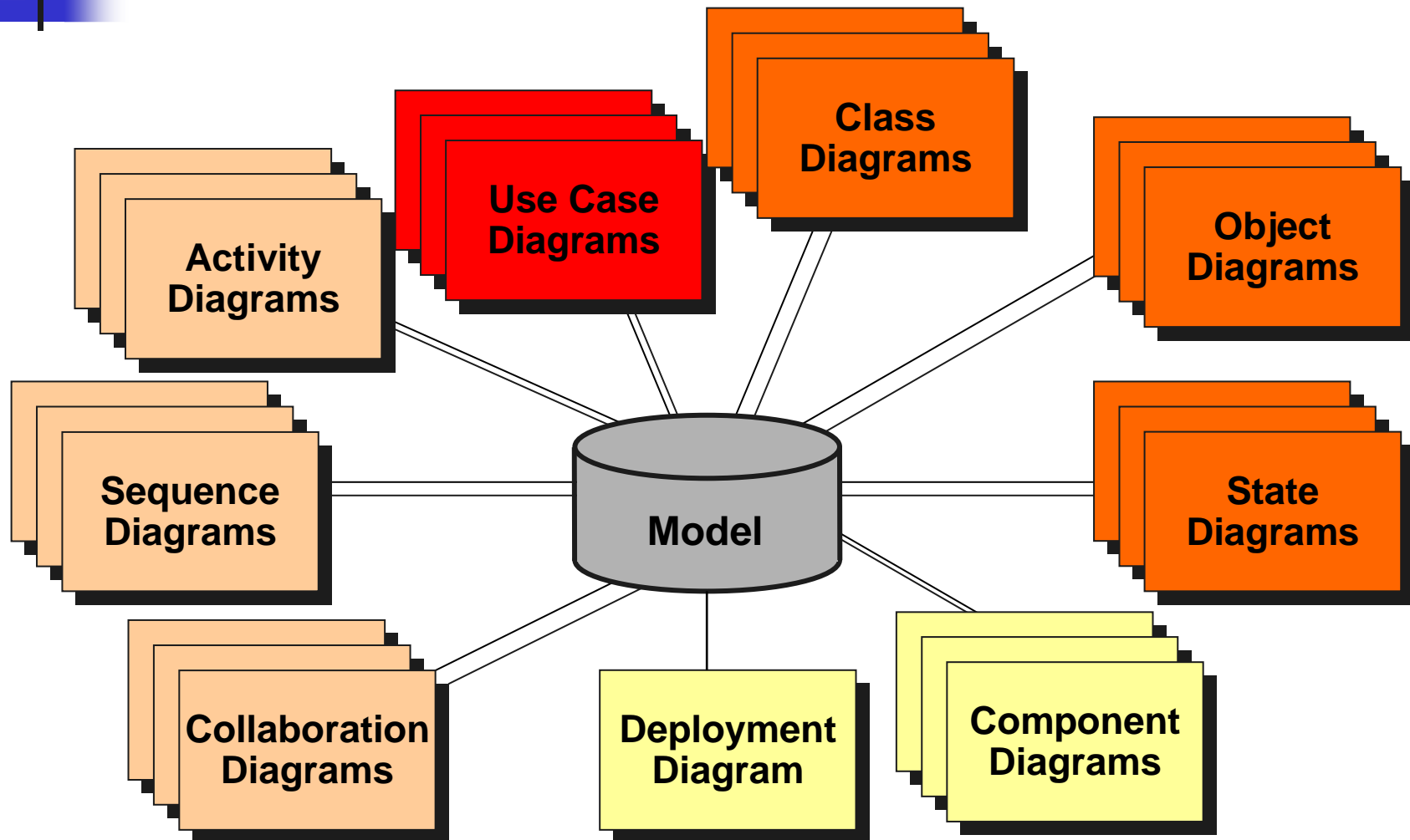
- Według firmy Gartner w 2006 r. więcej niż 10 mln używało UML.
- Według firmy Gartner w 2008 r. 70% firm informatycznych używało UML.
- Prace przy nowym teleskopie Webba będą prowadzone przy użyciu UML – ich zakończenie planuje się na rok 2013. Będą w nie zaangażowane: NASA, europejska i kanadyjska agencja kosmiczna oraz firmy zewnętrzne.



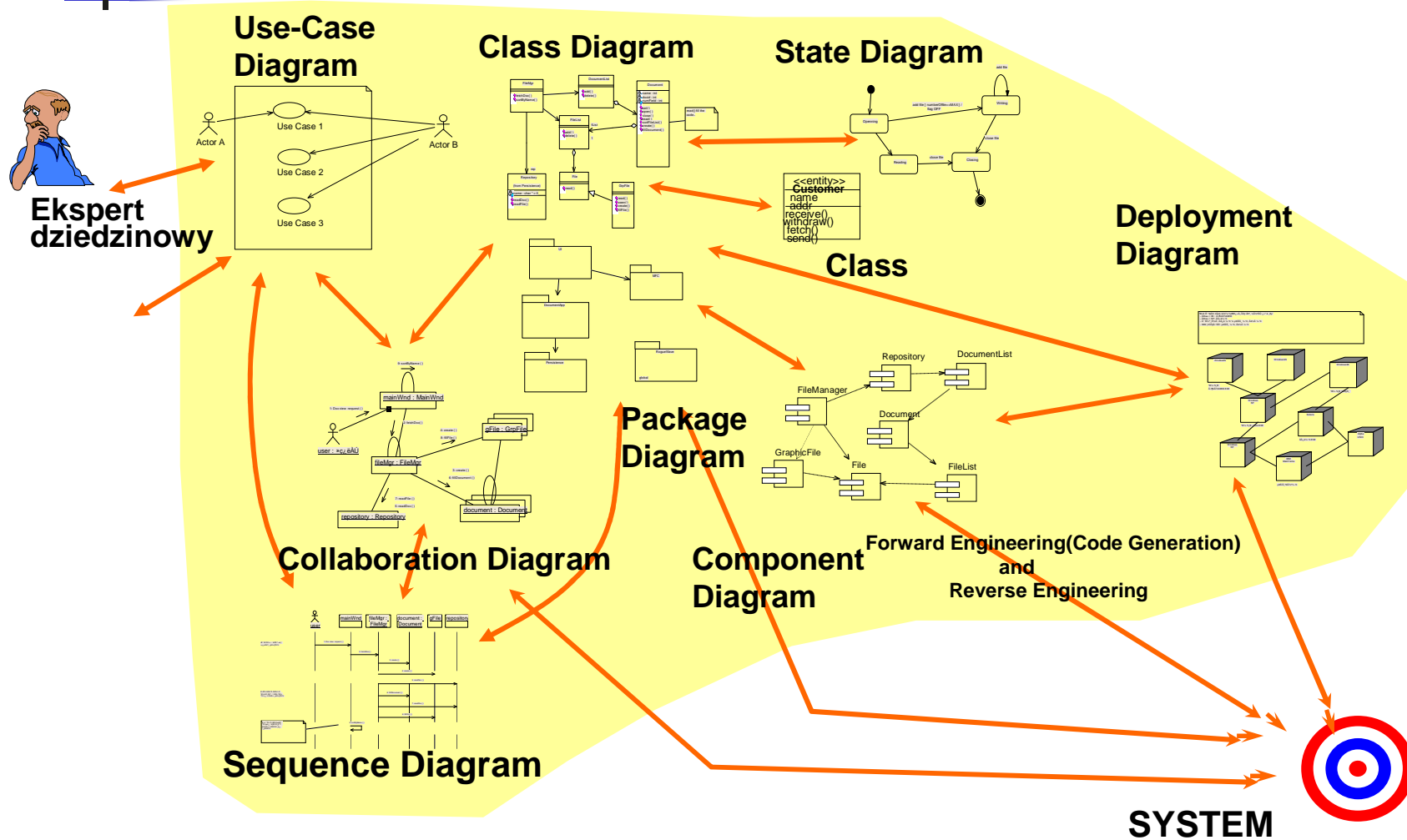
Charakterystyka UML

- UML jest językiem, tzn. zestawem pojęć, oznaczeń, diagramów.
- Notacja UML, która opiera się o podstawowe pojęcia obiektowości może być wykorzystana w dowolnej metodyce.
- Pojęcia UML, wynikające z doświadczenia jej twórców, mają w założeniu przykrywać większość istotnych aspektów modelowanych systemów.
- UML jest składową standardu OMG (CORBA).

UML wprowadza zestaw standardowych diagramów (wybrane)

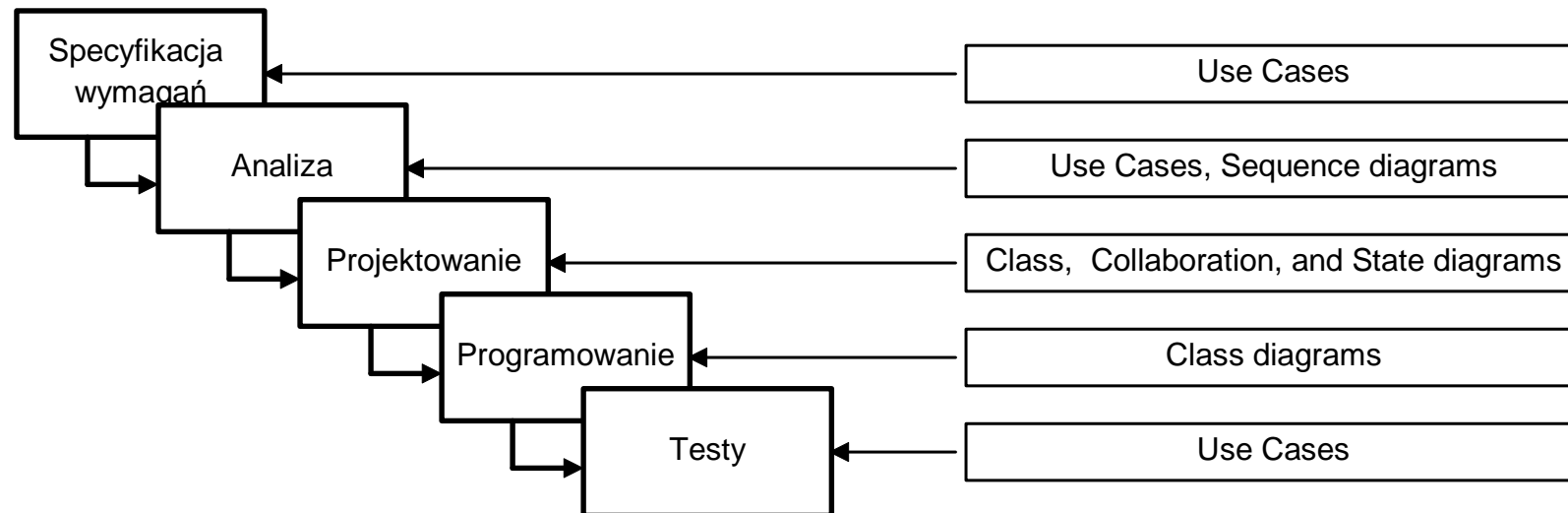


UML – jak to działa?



UML a etapy życia systemu informatycznego

UML może zostać zastosowany na różnym etapie projektowania systemu informatycznego, począwszy od specyfikacji wymagań aż po testy końcowe.



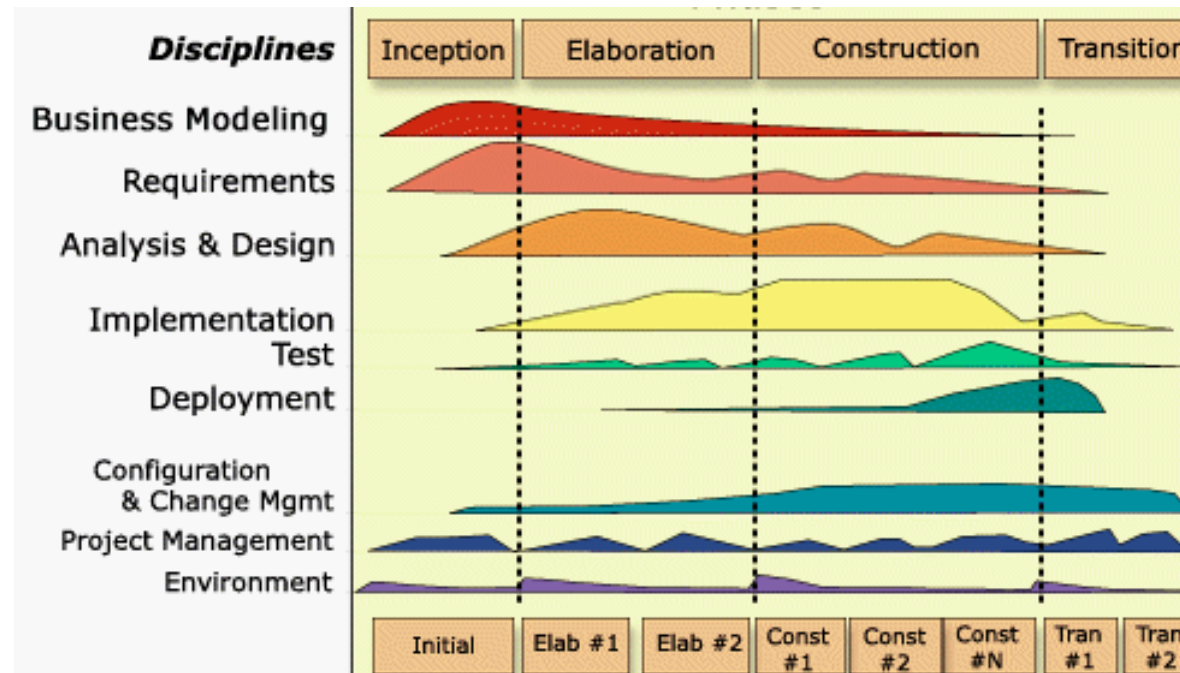


Czym jest RUP (Rational Unified Process)

- RUP jest procesem (wytwórczym) inżynierii oprogramowania, którego celem jest zapewnienie produkcji wysokiej jakości oprogramowania z przewidywalnym budżetem i harmonogramem, które spełni oczekiwania użytkowników końcowych. RUP reprezentuje dyscyplinarne podejście przypisujące zadania i obowiązki wewnątrz zespołu.
 - RUP jest konfigurowalnym procesem, ponieważ żaden pojedynczy proces nie jest odpowiedni do wytworzenia każdej aplikacji. W ten sposób istnieje możliwość dostosowania procesu do rozmaitych sytuacji.
- RUP jest produktem rozwijanym i utrzymywanym przez firmę IBM opartym na doświadczeniu w pracy z wieloma klientami firmy.
- RUP jest przewodnikiem jak skutecznie używać UML'a.

RUP - Rational Unified Process

Oś pionowa reprezentuje statyczne aspekty (zawartość) procesu; opisuje **proces** pod względem działań, artefaktów, pracowników (ang. workers) oraz przepływu zadań (ang. workflow).



Oś pozioma reprezentuje **czas** i przedstawia dynamiczne aspekty procesu używając pojęć: cykli, faz, iteracji oraz kamieni milowych (ang. milestones).
 Proces tworzenia oprogramowania podzielony jest na cykle, gdzie każdy cykl reprezentuje tworzenie nowej generacji (wydania) produktu.



Iteracje w RUP

- Każda faza RUP może zostać podzielona na iteracje.
- Iteracja jest kompletną „pętlą” tworzenia pewnej części finalnego produktu, który przyrasta inkrementalnie z jednej iteracji na drugą tworząc ostatecznie końcowy system.
- Korzyści podejścia iteracyjnego:
 - łatwiejsze zażegnanie elementów ryzyka,
 - większa zarządzalność zmianami,
 - możliwość nauki zespołu projektowego w trakcie projektu,
 - lepsza całkowita jakość,

Składniki RUP (1)

- Role i ich aktywności

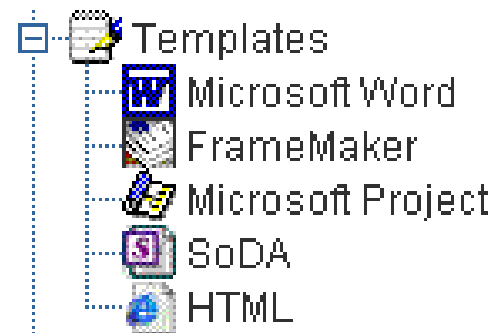


Składniki RUP (2)

■ Artefakty



■ Szablony





Składniki RUP (4)

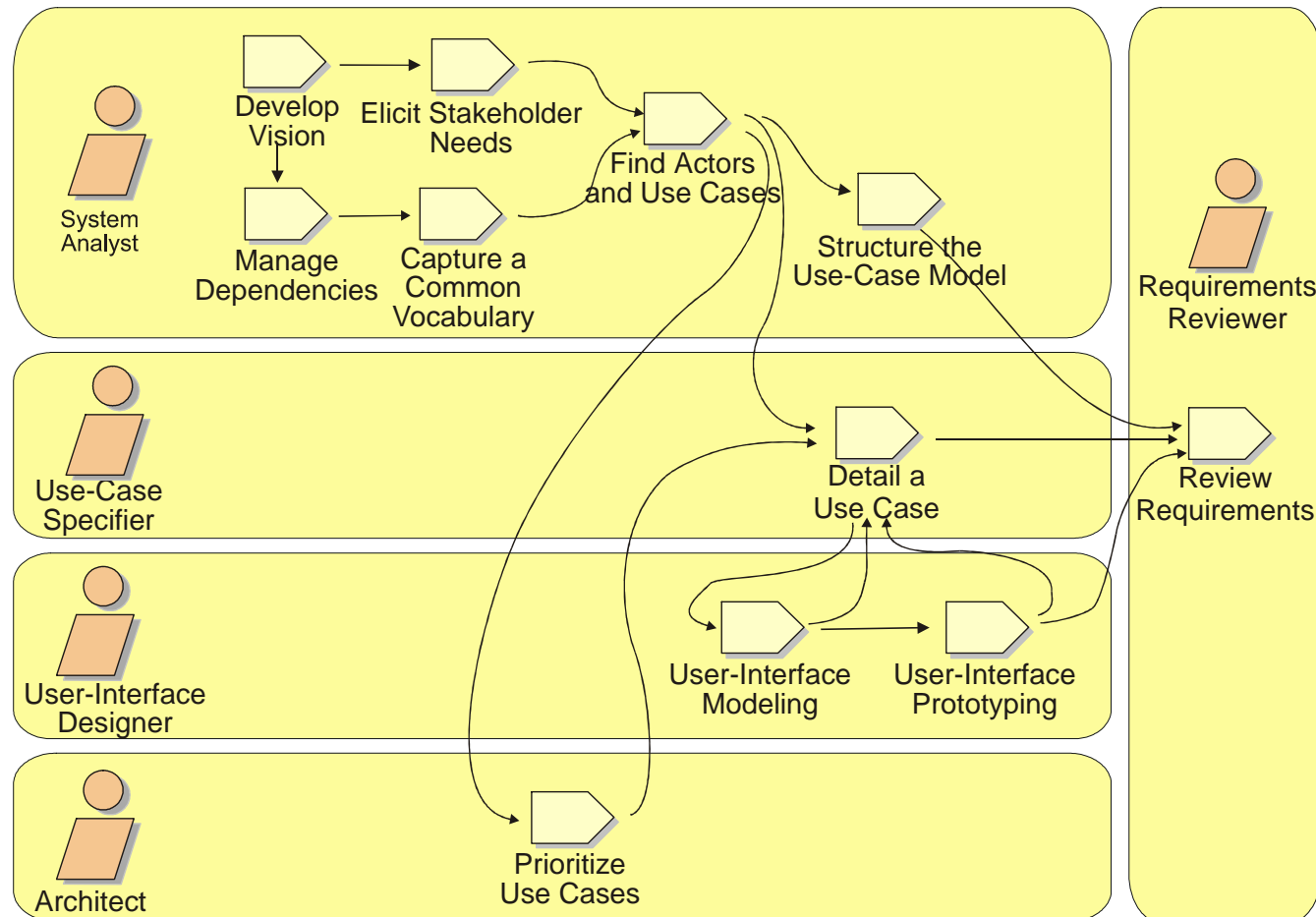
- Konceptcje (metody postępowania)



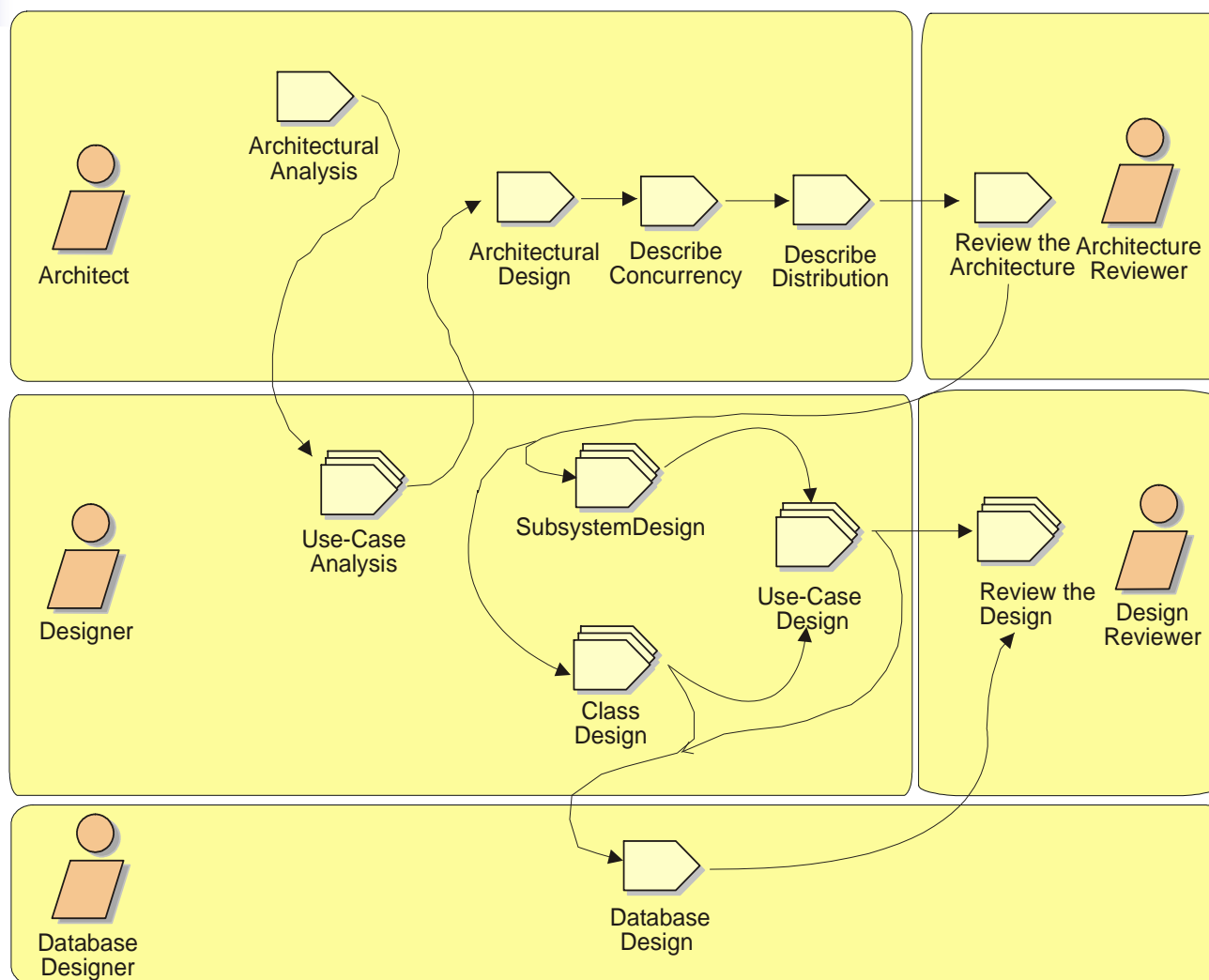
- Przepływy pracy („krok po kroku”)



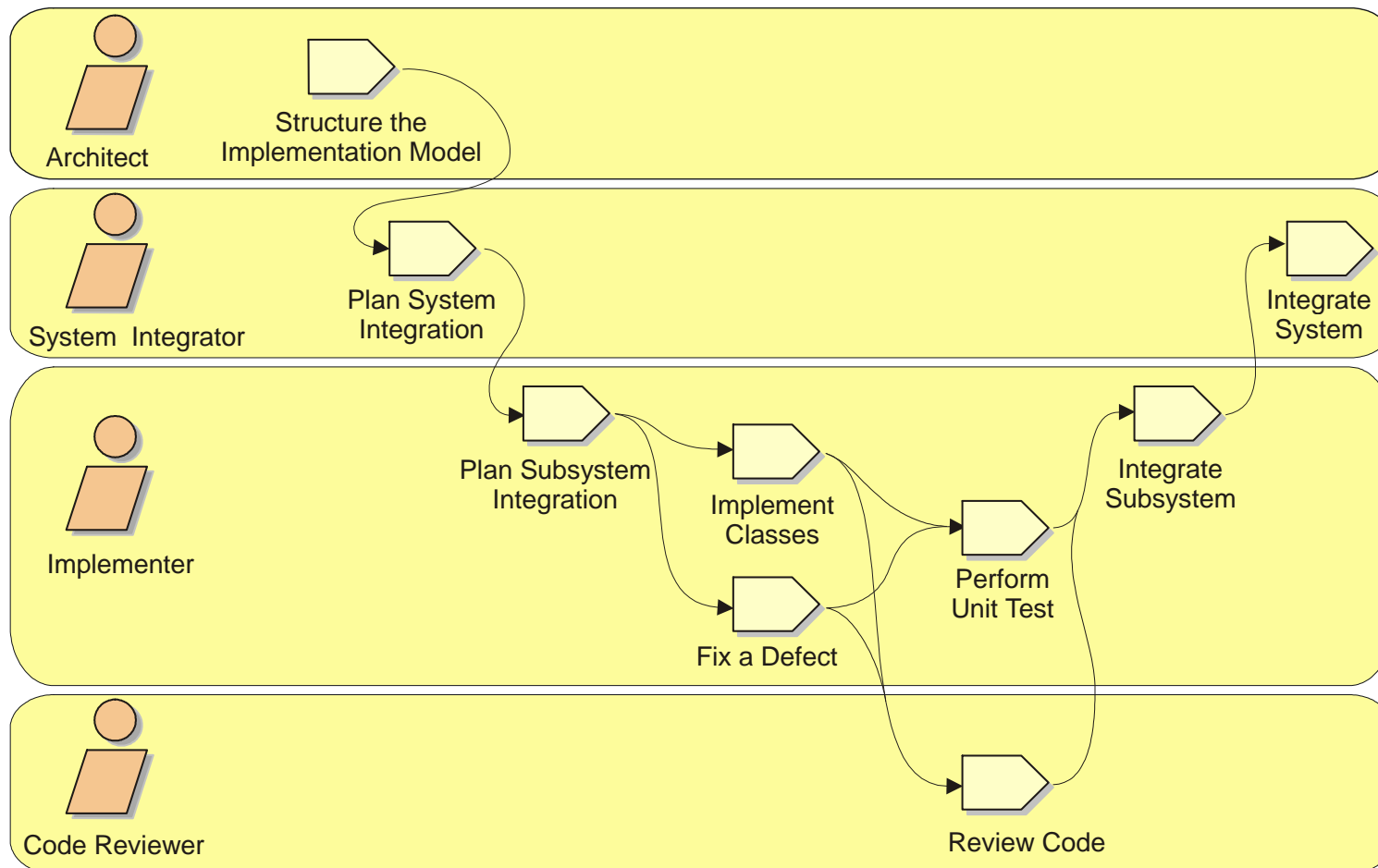
Przepływ pracy: inżynieria wymagań



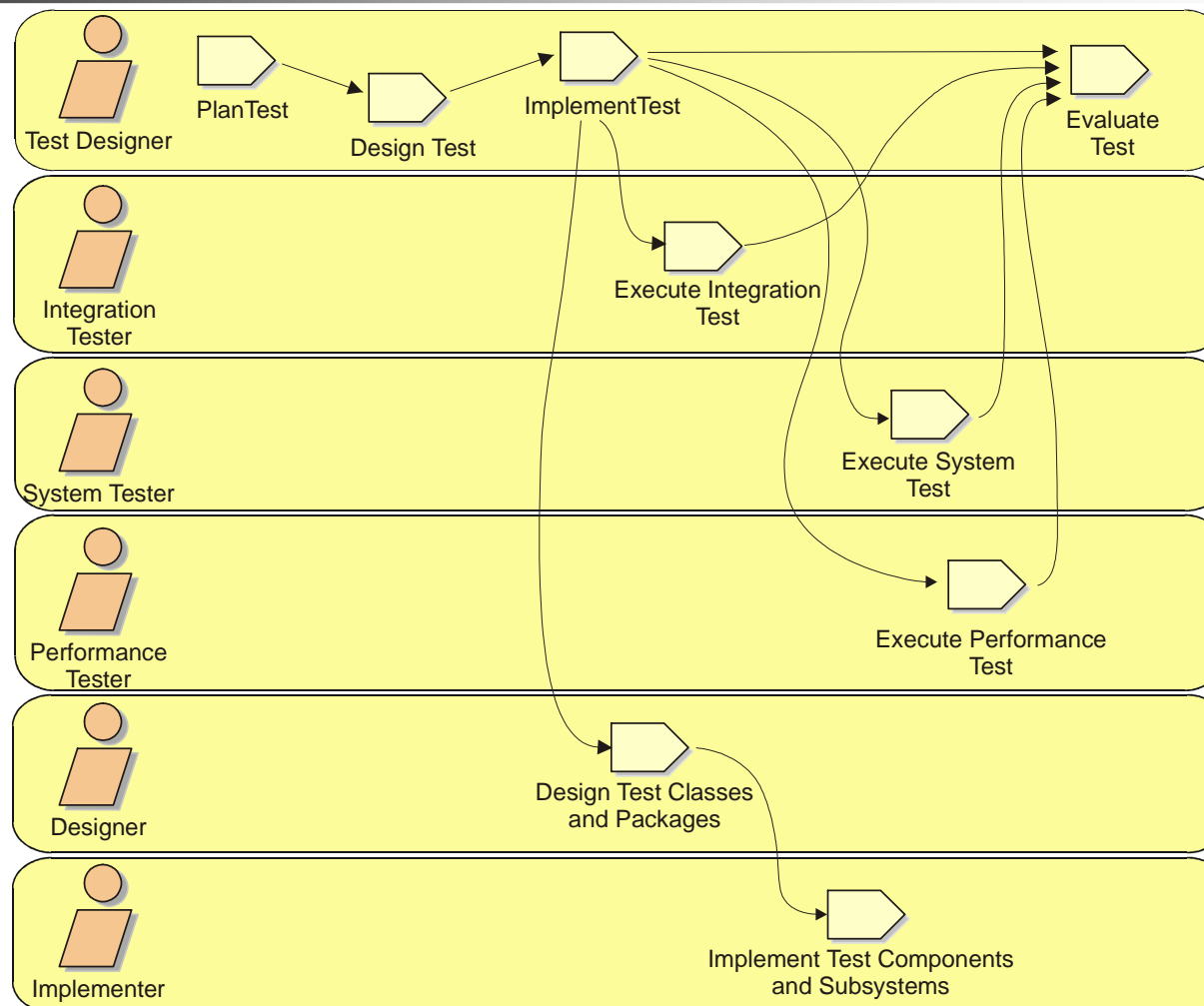
Przepływ pracy: analiza i projektowanie















Przepływ pracy: implementacja

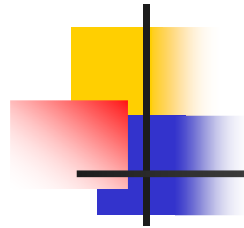


Przeływ pracy: testy



Składniki RUP (5)

- Produkty i
-  Rational Project Portal
-  Rational RequisitePro
-  Rational Rose
-  Rational Suite PerformanceStudio
-  Rational Purify
-  Rational PureCoverage
-  Rational Quantify
-  Rational SoDA
-  Rational Unified Process
-  Rational Robot
-  Rational TestManager
-  Rational TestFactory



Najlepsze praktyki w RUP

1. Iteracyjne tworzenie oprogramowania.
2. Zarządzanie wymaganiami.
3. Stosowanie architektur komponentowych.
4. Wizualne modelowanie oprogramowania.
5. Weryfikacja jakości oprogramowania.
6. Kontrola zmian w oprogramowaniu.



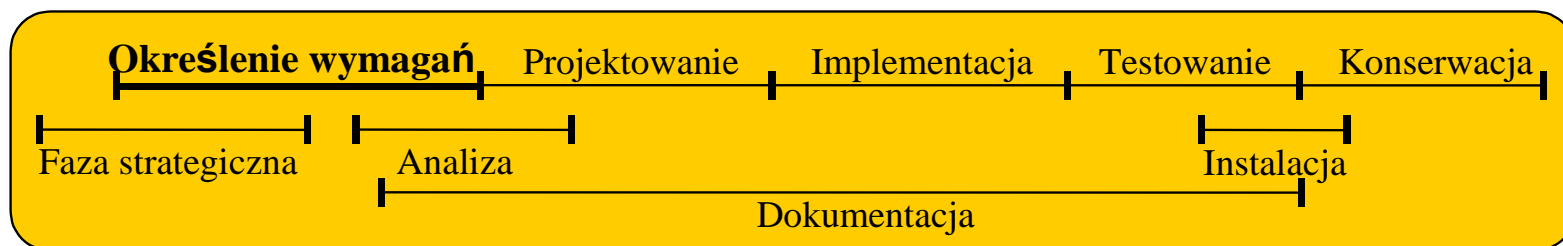
Podstawowe pojęcia - wymaganie

- Warunek lub zdolność systemu niezbędna użytkownikowi lub klientowi do rozwiązania problemu lub osiągnięcia zamierzonego celu;
- Warunek, który musi być spełniony, lub zdolność, którą musi posiadać system lub część systemu, aby spełnić umowę, normę, specyfikację lub inny formalnie przyjęty dokument;
- Udokumentowane wyrażenie zdolności lub warunków systemu określonych w punktach 1 i 2.

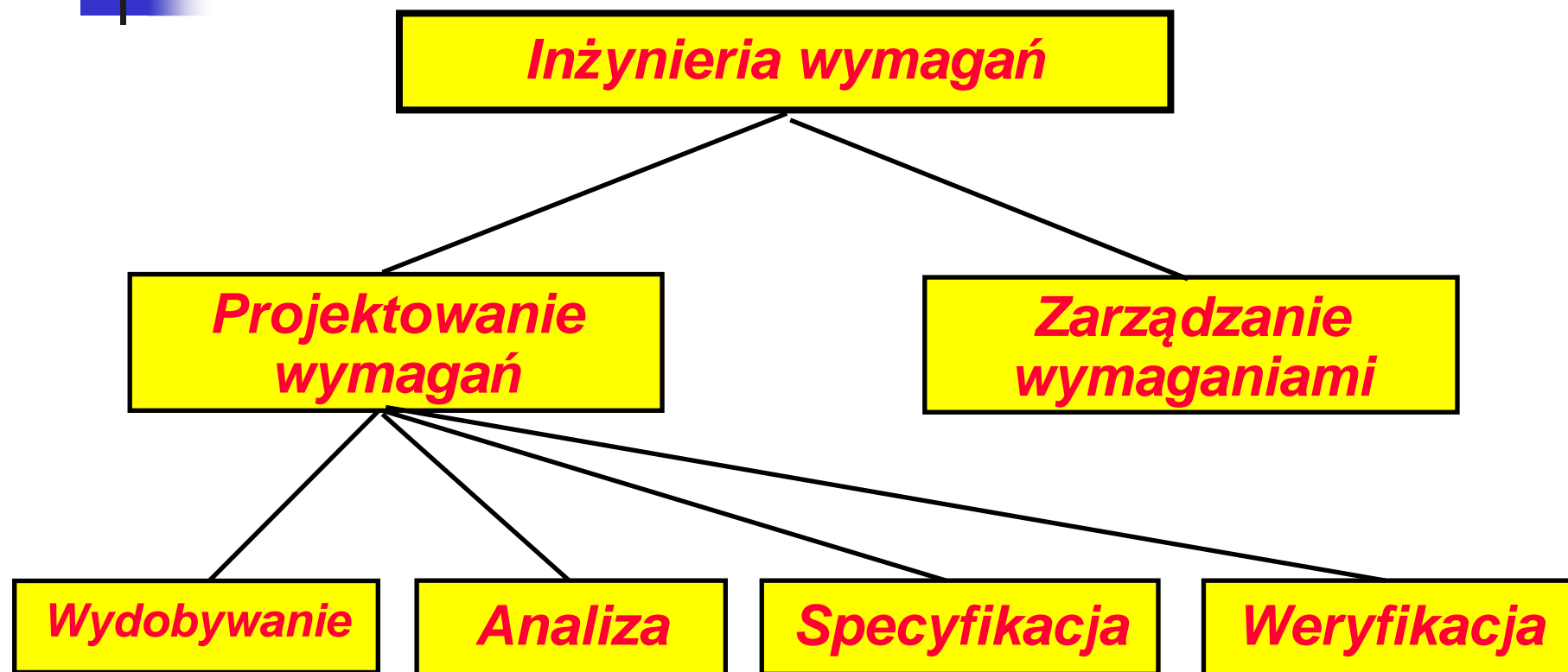
„IEEE Guide for Developing System
Requirements Specifications”

Miejsce inżynierii wymagań w inżynierii oprogramowania

- ♦ **Inżynieria wymagań** - jest dziedziną inżynierii oprogramowania zajmującą się **celami** systemów informatycznych, odnoszącymi się do świata rzeczywistego, ich funkcjami i dotyczącymi ich ograniczeniami.
- ♦ **Inżynierii wymagań** - jest zespołem działań obejmujących: (1) zbieranie informacji o problemie lub poznawanie problemu, który ma być rozwiązany, oraz (2) zdefiniowanie zachowań systemu, które mogą rozwiązać problem.



Inżynieria wymagań - klasyfikacja





Jak to z wymaganiami bywa (nowe spojrzenie)

Pewnego dnia odwiedziłem handlarza samochodów, aby zobaczyć interesujący mnie model samochodu. W pewnym momencie weszła elegancka blondynka i powiedziała, że chciałyby kupić przykrywkę nr 710 do jej samochodu.

Sprzedawca jak i wszyscy obecni w sklepie byli zaskoczeni.

- Przykrywkę nr 710 ? A do czego to jest? - spytał uprzejmie sprzedawca.

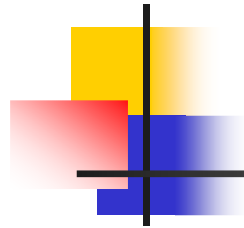
Kobieta powiedziała, że to było w jej samochodzie i że gdzieś to zgubiła.

Ponieważ nie mogła niczego więcej powiedzieć, sprzedawca poprosił, aby przynajmniej narysowała o co chodzi.

Kobieta narysowała kółko o średnicy ok 8 cm i w środku napisała 710.

Jak to z wymaganiami bywa (nowe spojrzenie)





Podstawowe pojęcia c.d.

- **Definicja wymagań** - ogólny opis w języku naturalnym. Opis taki jest rezultatem wstępnych rozmów z klientem.
- **Specyfikacja wymagań** - częściowo ustrukturalizowany zapis wykorzystujący zarówno język naturalny, jak i proste, częściowo przynajmniej sformalizowane notacje.
- **Specyfikacja oprogramowania** - formalny opis wymagań.



Klasyfikacja wymagań

- Wymagania są często klasyfikowane jako funkcjonalne lub нефункционалне.
 - **Wymagania funkcjonalne** opisują na kilka sposobów, co system powinien robić lub jakie powinien świadczyć usługi.
 - **Wymagania нефункционалне** można zdefiniować jako **ограничения** odnoszące się do usług lub czynności wykonywanych przez system.
 - **Ograniczenia** - „pseudo wymagania”
 - narzucone przez środowisko lub klienta

Wymagania niefunkcjonalne

(1)

Objętość: Ilu użytkowników będzie pracować jednocześnie? Ile terminali ma być podłączone do systemu? Ile danych będzie przechowywane?

Szybkość: Jak długo może trwać operacja lub sekwencja operacji? Liczba operacji na jednostkę czasu. Średni czas niezbędny dla jednej operacji.

Dokładność: Określenie stopnia precyzji pomiarów lub przetwarzania. Określenie wymaganej dokładności wyników. Zastąpienie wyników ilościowych jakościowymi lub odwrotnie.

Wymagania нефunkcjonalne

(2)



Interfejsy komunikacyjne: sieć, protokoły, wydajność sieci, poziom abstrakcji protokołów komunikacyjnych, itd.

Interfejsy sprzętowe: specyfikacja wszystkich elementów sprzętowych, które będą składały się na system, fizyczne ograniczenia (rozmiar, waga), wydajność (szybkość, RAM, dysk, inne pamięci), wymagania co do powierzchni lokalowych, wilgotności, temperatury i ciśnienia, itd.

Interfejsy oprogramowania: Określenie zgodności z innym oprogramowaniem, określenie systemów operacyjnych, języków programowania, kompilatorów, edytorów, systemów zarządzania bazą danych, itd.

Interakcja człowiek-maszyna: Wszystkie aspekty interfejsu użytkownika, rodzaj języka interakcji, rodzaj sprzętu (monitor, mysz, klawiatura), określenie formatów (układu raportów i ich zawartości), określenie komunikatów dla użytkowników (język, forma), pomocy, komunikatów o błędach, itd.

Wymagania niefunkcjonalne

(3)



Adaptowalność: Określenie w jaki sposób będzie organizowana reakcja na zmiany wymagań: dodanie nowej komendy, dodanie nowego okna interakcji, itd.

Bezpieczeństwo: założenia co do poufności, prywatności, integralności, odporności na hakerów, wirusy, wandalizm, sabotaż, itd.

Odporność na awarie: konsekwencje błędów w oprogramowaniu, przerwy w zasilaniu, kopie zabezpieczające, częstotliwości składowania, dziennika zmian, itd.

Standardy: Określenie dokumentów standardyzacyjnych, które mają zastosowanie do systemu: formaty plików, normy czcionek, polonizacja, standardy procesów i produktów, itd.

Zasoby: Określenie ograniczeń finansowych, ludzkich i materiałowych.

Skala czasowa: ograniczenia na czas wykonania systemu, czas szkolenia, wdrażania, itd.

Metody specyfikacji wymagań

(1)

- **Język naturalny** - najczęściej stosowany. Wady: niejednoznaczność powodująca różne rozumienie tego samego tekstu; utrudnia to wykrycie powiązanych wymagań i powoduje trudności w wykryciu sprzeczności.
- **Formalizm matematyczny** - stosuje się rzadko - dla specyficznych celów np. przy automatycznym generowaniu wymagań.
- **Język naturalny strukturalny** - język naturalny z ograniczonym słownictwem i składnią; tematy i zagadnienia wyspecyfikowane są w punktach i podpunktach.

Metody specyfikacji wymagań

(2)

- **Tablice, formularze** - wyspecyfikowanie wymagań w postaci (zwykle dwuwymiarowych) tablic, kojarzących różne aspekty (np. tablica ustalająca zależność pomiędzy typem użytkownika i rodzajem usługi).
- **Diagramy blokowe** - forma graficzna pokazująca cykl przetwarzania.
- **Diagramy kontekstowe** - ukazują system w postaci jednego bloku oraz jego powiązania z otoczeniem, wejściem i wyjściem.
- **Diagramy przypadków użycia** - poglądowy sposób przedstawienia aktorów i funkcji systemu.

Formularz wymagań funkcjonalnych

<i>Nazwa funkcji</i>	<i>Edycja dochodów pracownika</i>
Opis	Funkcja pozwala edytować łączne dochody podatnika uzyskane w danym roku.
Dane wejściowe	Informacje o dochodach pracowników uzyskane z różnych źródeł: kwoty przychodów, koszty uzyskania przychodów oraz zapłaconych zaliczek na poczet podatku dochodowego. Informacje o dokumentach opisujących dochody z poszczególnych źródeł.
Źródło danych wejściowych	Dokumenty oraz informacje dostarczone przez podatnika. Dane wpisywane przez pracownika firmy podatkowej.
Wynik	
Warunek wstępny	Kwota dochodu = kwota przychodu - kwota kosztów (zarówno dla konkretnych dochodów, jak i dla łącznych dochodów podatnika). Łączne kwoty przychodów, kosztów uzyskania dochodów oraz zapłaconych zaliczek są sumami tych kwot dla dochodów z poszczególnych źródeł.
Warunek końcowy	Jak wyżej.
Efekty uboczne	Uaktualnienie podstawy opodatkowania.
Powód	Funkcja pomaga przyspieszyć obsługę klientów oraz zmniejszyć ryzyko popełnienia błędów.