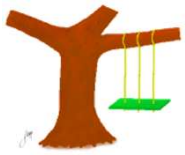
Inżynieria Oprogramowania

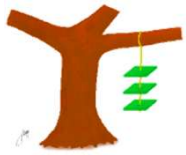
Podsumowanie

Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski
www.mimuw.edu.pl/~dabrowski

Pół żartem, pół serio

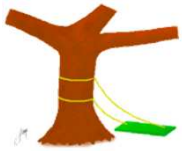


To, co klient zamówił

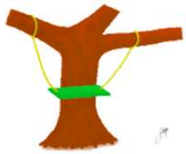


To, co analityk zrozumiał

To, co opisywał projekt



To, co wykonali programiści

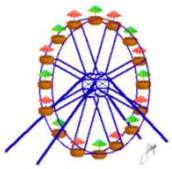


Wykorzystano materiały: Patryk Sielski, Politechnika Warszawska

Inżynieria Oprogramowania, MIMUW

Pół żartem, pół serio

Projekt po wdrożeniu



To, co za co klient zapłacił

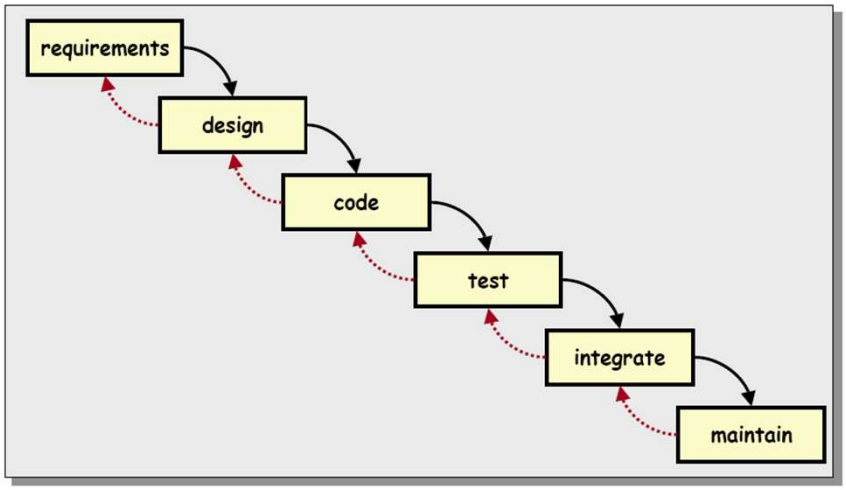
A to, czego klient potrzebował

Praktyczne zastosowanie projektu

3

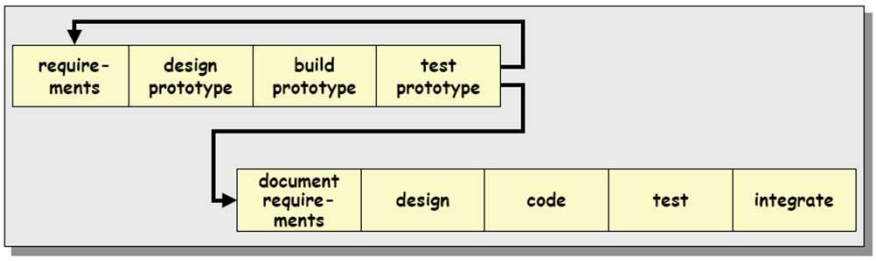
Inżynieria Oprogramowania, MIMUW

Model wodospadowy



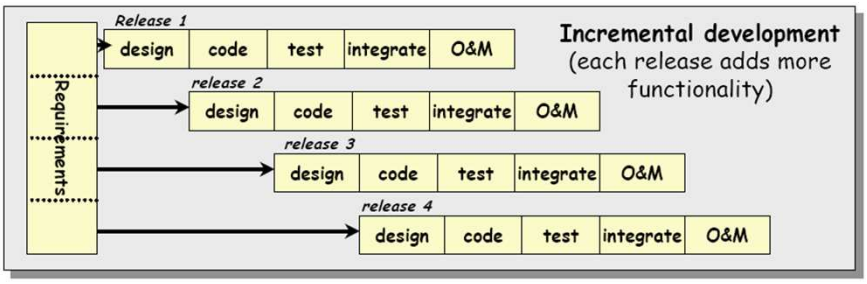
4

Model prototypowany



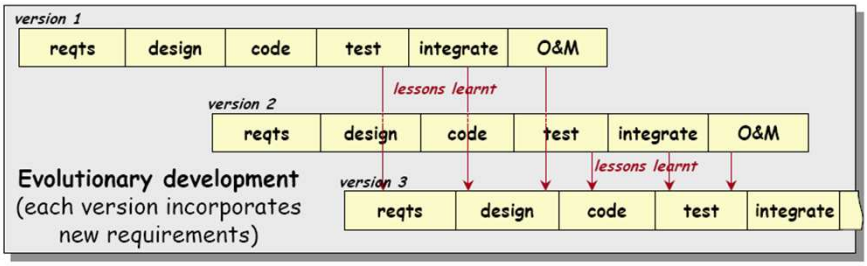
5

Model przyrostowy



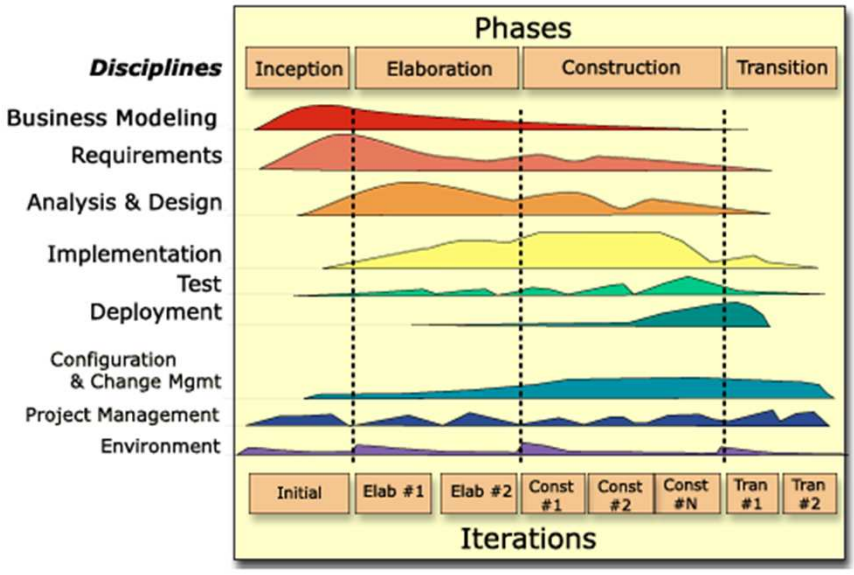
6

Model ewolucyjny



7

Model ujednolicony (Unified Process)



8

Esencja procesu wytwórczego

- Wiele modeli – jedna idea:
  - opisać problem
  - opisać rozwiązanie
  - zweryfikować
    - czy rozwiązanie rozwiązuje postawiony problem
  - zwalidować
    - czy rozwiązaliśmy właściwy problem

9

Praktyki wytwórcze

	A	B	C	D	E
5	[REQM] Zarządzanie wymaganiami	[PP] Planowanie projektu	[PMC] Monitorowanie i kontrola projektu	[MA] Pomiary i analizy	[PPQA] Zapewnianie jakości procesu i produktu
4	[CM] Zarządzanie konfiguracją	[SAM] Zarządzanie dostawcami	[RD] Wydobywanie wymagań	[TS] Projektowanie i budowanie rozwiązań	[PI] Integracja produktów
3	[VER] Weryfikacja	[WAL] Walidacja	[OPF] Poprawa procesów organizacji	[OPD] Definiowanie procesów organizacji	[OT] Szkolenia w organizacji
2	[IPM] Zintegrowane zarządzanie projektami	[RM] Zarządzanie ryzykiem	[DAR] Wnioskowanie i analiza decyzji	[OEI] Integracja środowisk	[IT] Integracja zespołów
1	[ISM] Zintegrowane zarządzanie dostawcami	[OPP] Wydajność procesów w organizacji	[QPM] Ilościowe zarządzanie projektem	[OID] Wdrażanie innowacji w organizacji	[CAR] Analizy przyczyn źródłowych

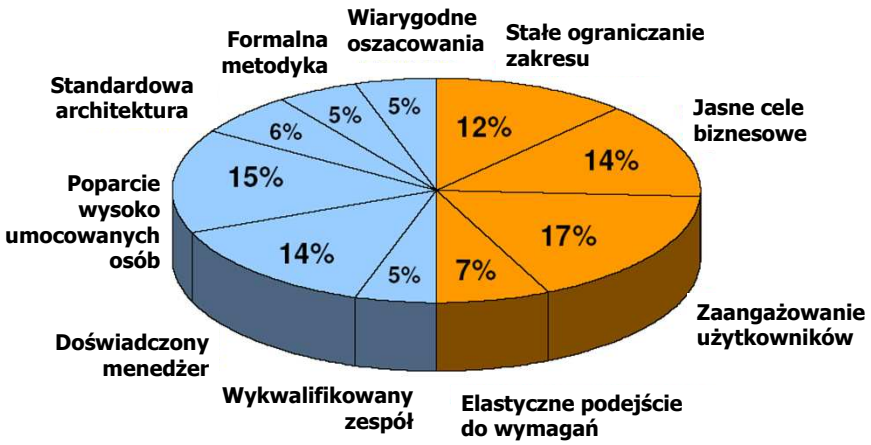
10

Praktyki wytwórcze – Wymagania

	A	B	C	D	E
5	[REQM] Zarządzanie wymaganiami	[PP] Planowanie projektu	[PMC] Monitorowanie i kontrola projektu	[MA] Pomiary i analizy	[PPQA] Zapewnianie jakości procesu i produktu
4	[CM] Zarządzanie konfiguracją	[SAM] Zarządzanie dostawcami	[RD] Wydobywanie wymagań	[TS] Projektowanie i budowanie rozwiązań	[PI] Integracja produktów
3	[VER] Weryfikacja	[WAL] Walidacja	[OPF] Poprawa procesów organizacji	[OPD] Definiowanie procesów organizacji	[OT] Szkolenia w organizacji
2	[IPM] Zintegrowane zarządzanie projektami	[RM] Zarządzanie ryzykiem	[DAR] Wnioskowanie i analiza decyzji	[OEI] Integracja środowisk	[IT] Integracja zespołów
1	[ISM] Zintegrowane zarządzanie dostawcami	[OPP] Wydajność procesów w organizacji	[QPM] Ilościowe zarządzanie projektem	[OID] Wdrażanie innowacji w organizacji	[CAR] Analizy przyczyn źródłowych

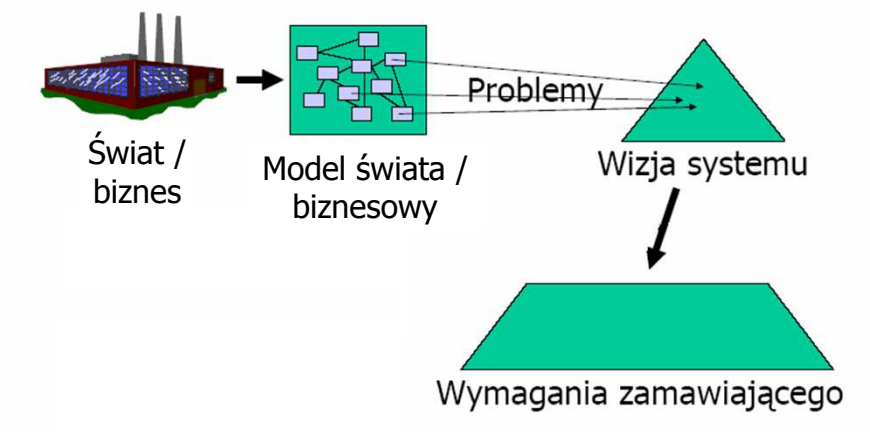
11

Przyczyny udanych projektów informatycznych



12

Podejście



Dobra praktyka: wszerz, nie wgłąb

Wizja

↓

Procesy biznesowe

- czynności poza systemem
- automatyzacja

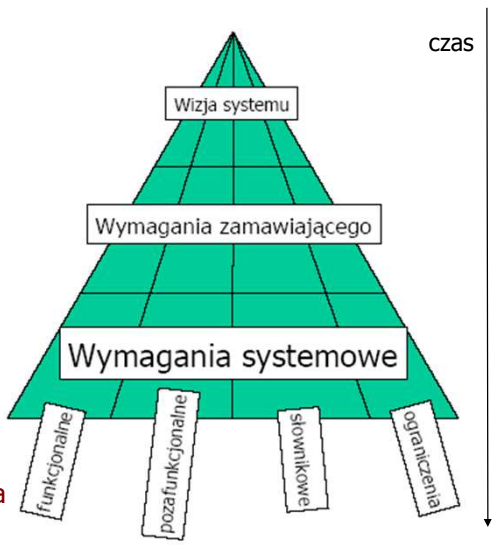
↓

Przypadki użycia

- aktorzy
- scenariusz główny
- scenariusze alternatywne
- model zależności

↓

Specyfikacja pozafunkcjonalna

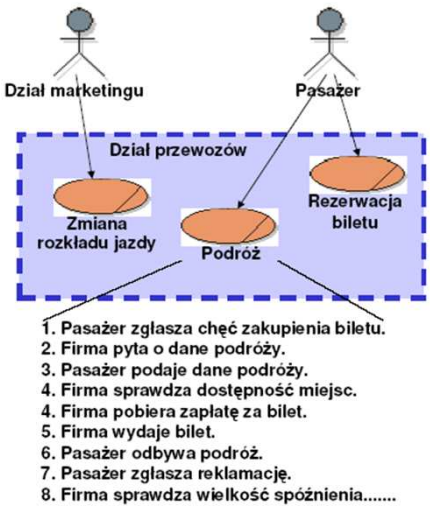


Wizja systemu

- Składniki wizji systemu
  - Postawienie problemu
  - Motto dla systemu
  - Interesariusze / osoby zainteresowane
  - Kluczowi użytkownicy
  - Cechy oprogramowania
  - Ograniczenia środowiska

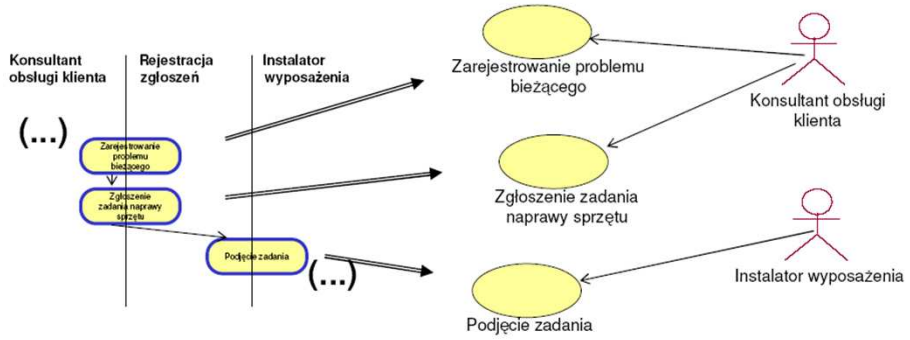
15

Proces biznesowy



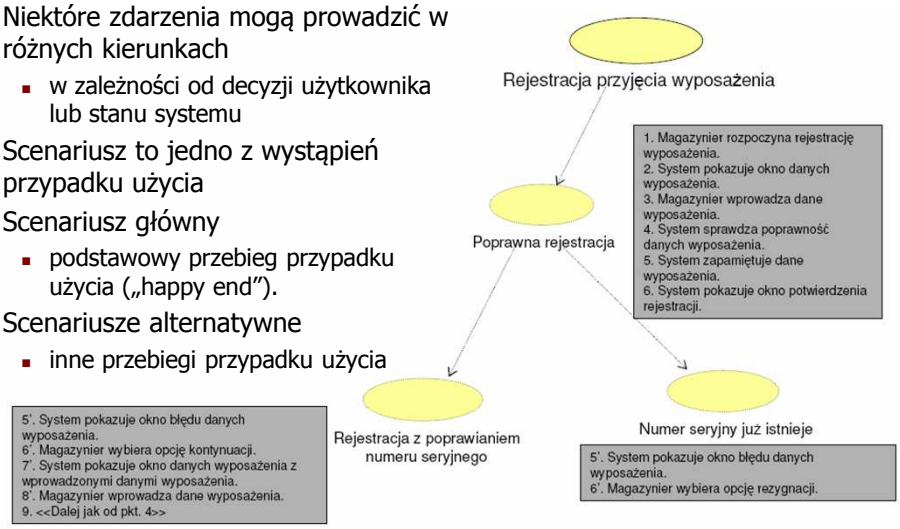
16

Przypadek użycia



Scenariusz

- Niektóre zdarzenia mogą prowadzić w różnych kierunkach
  - w zależności od decyzji użytkownika lub stanu systemu
- Scenariusz to jedno z wystąpień przypadku użycia
- Scenariusz główny
  - podstawowy przebieg przypadku użycia („happy end”).
- Scenariusze alternatywne
  - inne przebiegi przypadku użycia



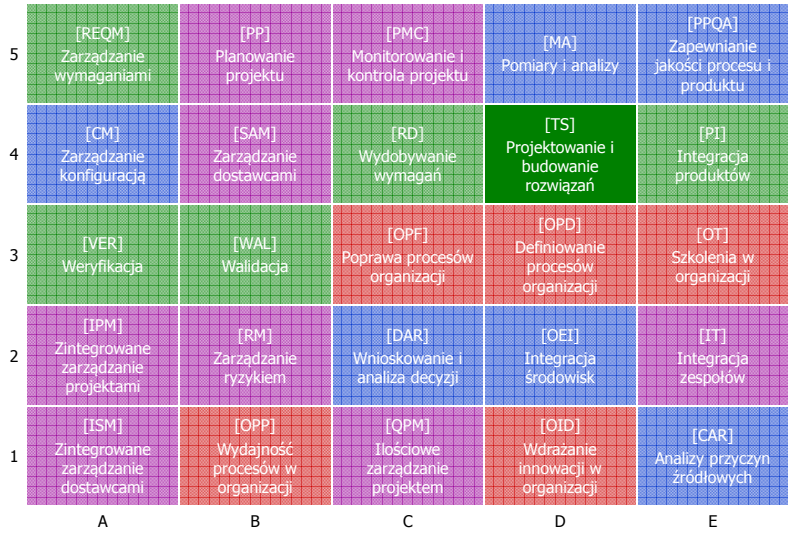
Specyfikacja uzupełniająca

- Wygląd zewnętrzny
  - poziom zgodności ze standardem korporacyjnym
  - stopień czytelności napisów
  - stopień spełnienia zasad ergonomii
- Łatwość użycia
  - czas potrzebny na naukę systemu
  - czas potrzebny na wykonanie typowych czynności
- Wydajność
  - czas trwania operacji
  - czas niedostępności systemu w miesiącu
- Łatwość serwisowania
  - czas poświęcany na administrowanie systemem
- Bezpieczeństwo
  - system nie pozwoli na nieautoryzowany dostęp do danych przy próbie włamania przez firmę hakerską
- ...

Cechy dobrej specyfikacji

- Poprawność
  - Czy rzeczywiście tak ma działać system?
- Jednoznaczność
  - Czy wymagania mają tylko jedną interpretację?
- Kompletność
  - Czy zamieszczono wszystkie wymagania funkcjonalne i pozafunkcjonalne?
- Spójność
  - Czy wymagania zamawiającego są zgodne z wizją i niesprzeczne ze sobą?
- Możliwość porządkowania wymagań
  - Czy każde wymaganie ma atrybuty umożliwiające określenie wagi i znaczenia?
- Weryfikowalność
  - Czy każde wymaganie ma przyporządkowane kryterium jakości, które można przetestować?
- Modyfikowalność
  - Czy można w łatwy sposób wprowadzać zmiany do specyfikacji wymagań?
- Utrzymywanie śladu
  - Czy wymagania posiadają identyfikatory i wynikają bezpośrednio z wymagań wyższego poziomu?

Praktyki wytwórcze – Projektowanie



Abstrakcja i dekompozycja

- Przypomnienie
  - Jak radzić sobie ze złożonymi systemami?
- Podejście
  - Abstrakcja
    - uprość i uogólnij
  - Dekompozycja
    - podziel i zwyciężaj

Abstrakcja

- Wnioskowanie na temat problemu
  - upraszcza problem
  - nie rozwiązuje problemu
- Dobra abstrakcja
  - ukrywa/ignoruje niepotrzebne szczegóły
  - upraszcza analizę
  - znajduj analogie pomiędzy różnymi bytami

Dekompozycja

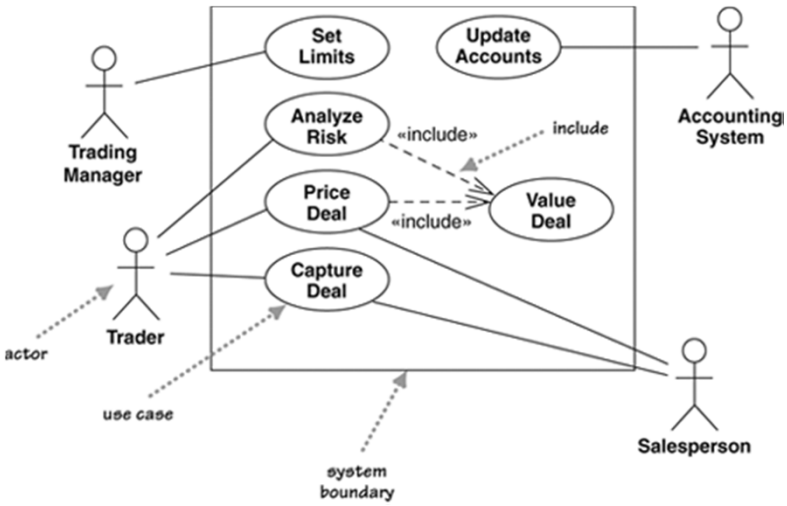
- Rozwiązywanie dużych problemów
  - metoda dziel i zwyciężaj
- Dobra dekompozycja
  - każdy podproblem jest podobnej wielkości
  - podproblemy można rozwiązywać niezależnie
  - z rozwiązań podproblemów można uzyskać rozwiązanie całości

UML

- Perspektywy w modelach obiektowych
  - Wymagania
  - Przypadki użycia
  - Aktywność
  - Klasy
  - Stany
  - Kooperacje / interakcje
  - Komponenty
  - Wdrożenie

25

Przypadki użycia



26

Scenariusze

Buy a Product

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address; next-day or 3-day delivery)
4. System presents full pricing information, including shipping
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming e-mail to customer

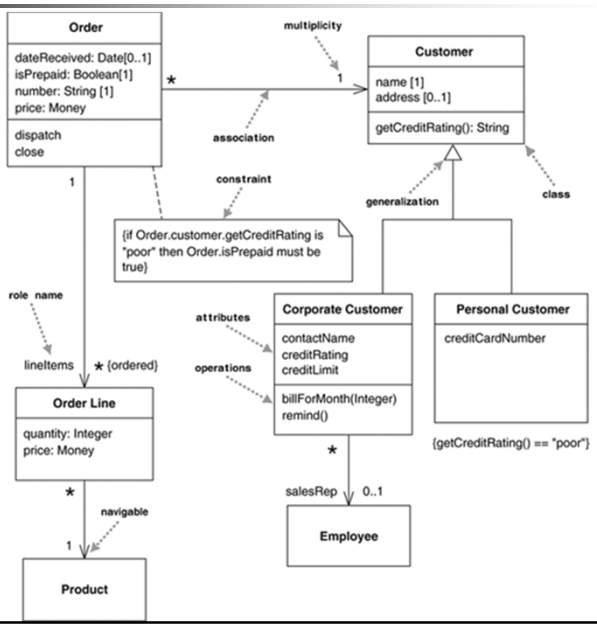
Extensions:

3a: Customer is regular customer
.1: System displays current shipping, pricing, and billing information
.2: Customer may accept or override these defaults, returns to MSS at step 6

6a: System fails to authorize credit purchase
.1: Customer may reenter credit card information or may cancel

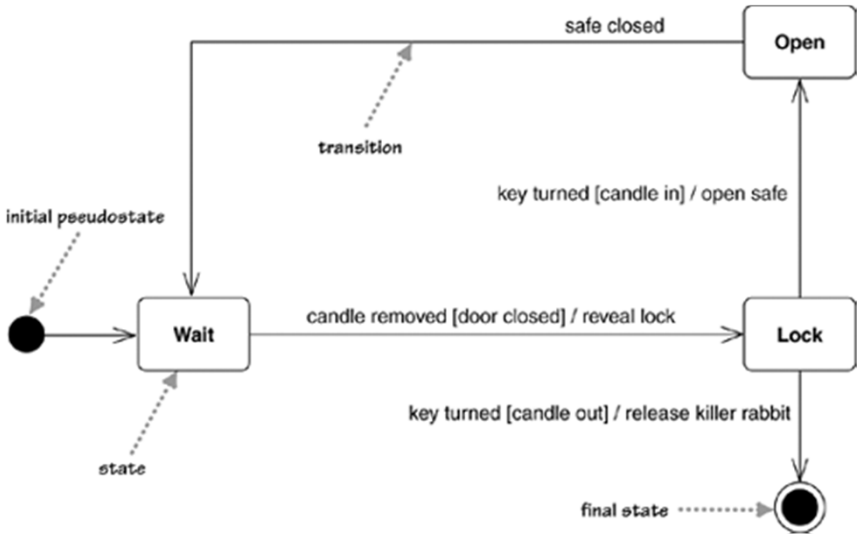
27

Diagramy klas



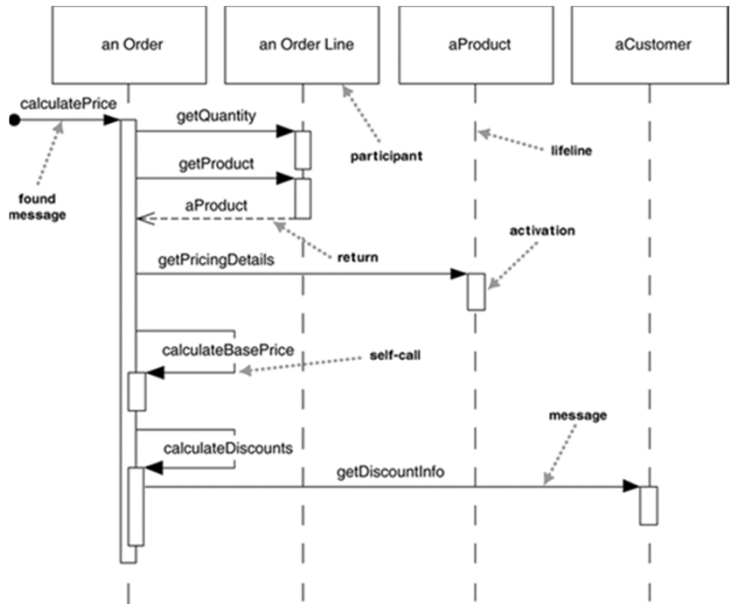
28

Diagramy stanów



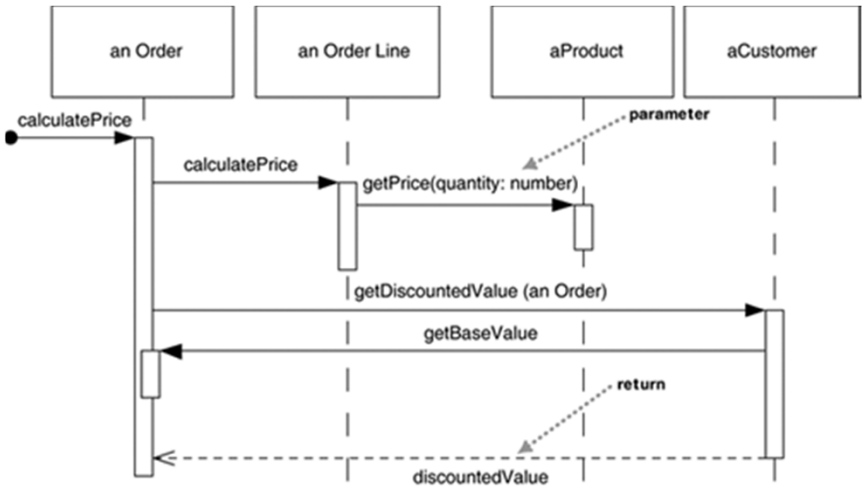
29

Diagramy interakcji – scentralizowane



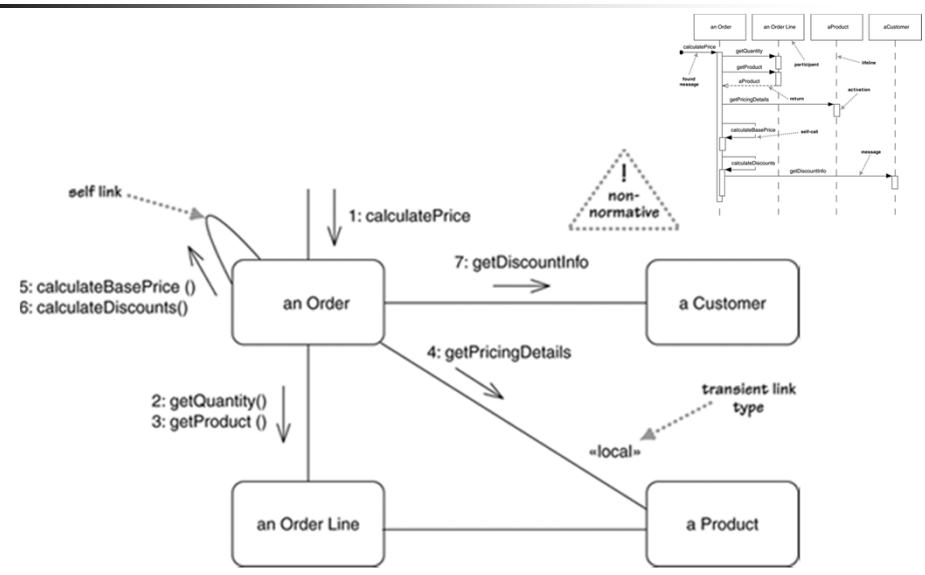
30

Diagramy interakcji – delegowane



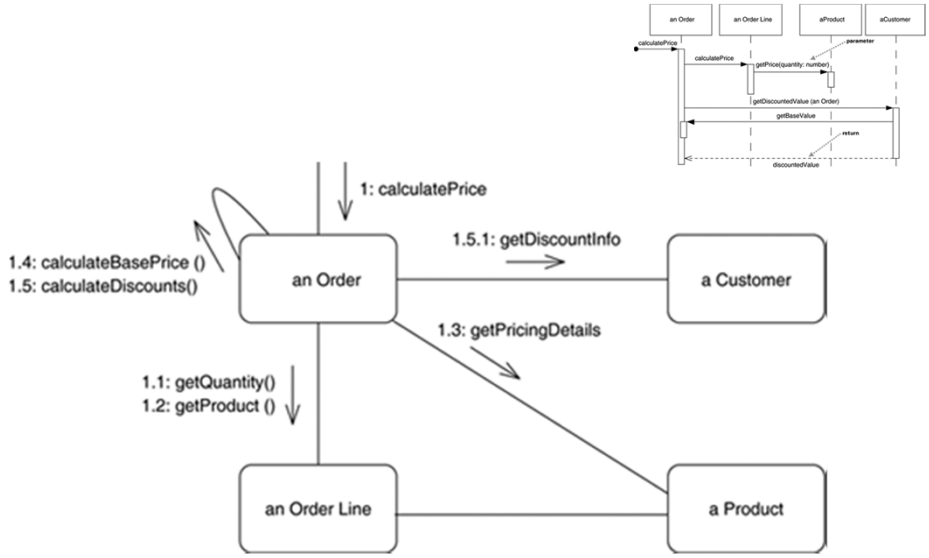
31

Diagramy kooperacji – scentralizowane



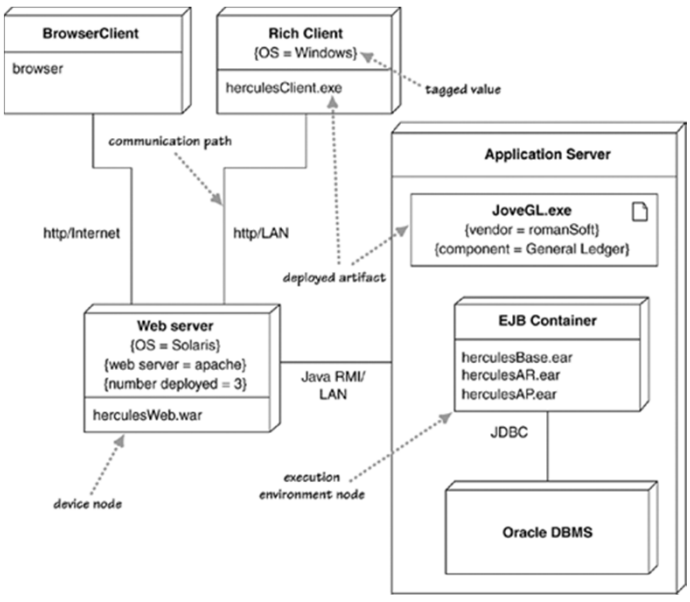
32

Diagramy kooperacji – delegowane



33

Diagramy wdrożenia



34

Zasady / wzorce projektowe (projektowania)

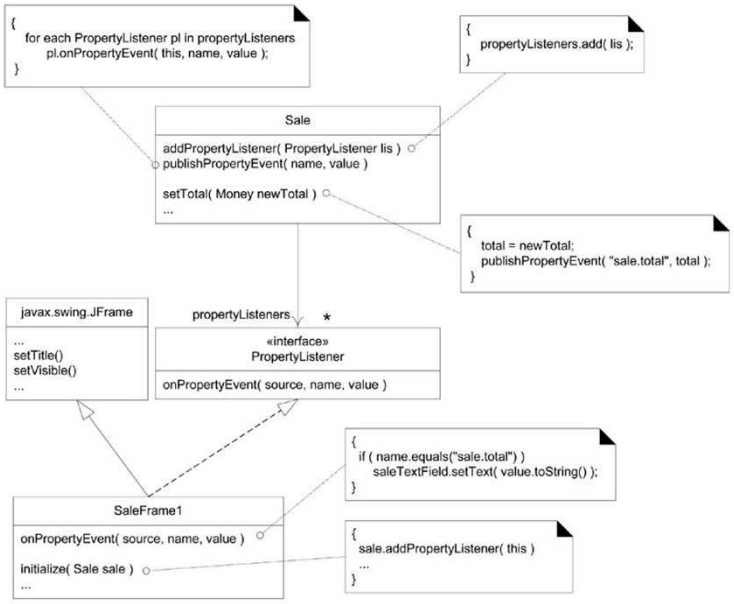
- Wzorce
  - Posiadające nazwę i dobrze znane pary {problem/rozwiązania}, które można zastosować w nowym kontekście.
    - Zazwyczaj również informacje jak je dostosować oraz omówienie kompromisów, implementacji, odmian, itp.
    - Czasami mówi się również o antywzorcach
- Zbiory wzorców
  - Gang-of-Four (GoF)
    - Erich Gamma, Richard Helm, Ralph Johnson i John Vlissides: *Wzorce projektowe*
    - coś co dla jednych jest wzorcem dla innych może być podstawowe
  - General Responsibility Assignment Patterns/Principles (GRASP)
    - Craig Larman: *Applying UML And Patterns*
    - prościej napisana; zwiera również wiele wzorców GoF
  - ...

Przykład: OBSERVER – Problem

- Problem:
  - Różne obiekty (obserwatorzy) są zainteresowane obserwowaniem zmian stanu obiektów (obserwowanych).
  - Chcą po swojemu reagować na te zdarzenia, jednocześnie zachowując luźne sprzężenie.

Inżynieria Oprogramowania, MIMUW

Przykład: OBSERVER – Rozwiązanie



37

Inżynieria Oprogramowania, MIMUW

Praktyki wytwórcze – Zarządzanie konfiguracją

	A	B	C	D	E
5	[REQM] Zarządzanie wymaganiami	[PP] Planowanie projektu	[PMC] Monitorowanie i kontrola projektu	[MA] Pomiary i analizy	[PPQA] Zapewnianie jakości procesu i produktu
4	[CM] Zarządzanie konfiguracją	[SAM] Zarządzanie dostawcami	[RD] Wydobywanie wymagań	[TS] Projektowanie i budowanie rozwiązań	[PI] Integracja produktów
3	[VER] Weryfikacja	[WAL] Walidacja	[OPF] Poprawa procesów organizacji	[OPD] Definiowanie procesów organizacji	[OT] Szkolenia w organizacji
2	[IPM] Zintegrowane zarządzanie projektami	[RM] Zarządzanie ryzykiem	[DAR] Wnioskowanie i analiza decyzji	[OEI] Integracja środowisk	[IT] Integracja zespołów
1	[ISM] Zintegrowane zarządzanie dostawcami	[OPP] Wydajność procesów w organizacji	[QPM] Ilościowe zarządzanie projektem	[OID] Wdrażanie innowacji w organizacji	[CAR] Analizy przyczyn źródłowych

38

Typowe oczekiwania od Zarządzania Konfiguracją

- Jaka platforma jest wymagana dla danej wersji systemu?
- Które wersje systemu są zależne od zmiany danego komponentu?
- Ile błędów zgłoszono do danej wersji?
- Jakie komponenty zmodyfikowano przy realizacji danej zmiany?
- ...

39

Uszczegółowienie oczekiwań od Zarządzania Konfiguracją

- Każdy komponent oprogramowania jest jednoznacznie identyfikowany
- Oprogramowanie jest zbudowane ze spójnego zestawu komponentów
- Zawsze wiadomo, która wersja komponentu oprogramowania jest najnowsza
- Zawsze wiadomo, która wersja dokumentacji pasuje do której wersji komponentu oprogramowania
- Komponenty oprogramowania są zawsze łatwo dostępne
- Komponenty oprogramowania nie zostaną stracone (np. wskutek awarii nośnika lub błędu operatora)
- Każda zmiana oprogramowania jest zatwierdzona i udokumentowana
- Zmiany oprogramowania nie zaginą (np. wskutek jednoczesnych sprzecznych aktualizacji)
- Zawsze istnieje możliwość powrotu do poprzedniej wersji
- Historia zmian jest przechowywana, co umożliwia odtworzenie kto i kiedy zrobił zmianę, i jaką zmianę

40

Jakie działania w tym celu realizujemy?

- Kontrola wersji
- Zarządzanie zmianą
- Budowanie wydań

41

Kontrola wersji

- Identyfikacja wersji i wydań systemu
  - System przydziela identyfikatory automatycznie podczas zgłaszania nowej wersji pliku do systemu
- Kontrolowanie zmian
  - Tylko jedna wersja w danej chwili może podlegać zmianie. Można równolegle pracować nad różnymi wersjami.
- Zarządzanie przestrzenią dyskową
  - Przechowywanie różnić a nie pełnych nowych wersji
  - Dla danych tekstowych jak też binarnych
- Rejestrowanie historii zmian
  - Zapamiętuje uzasadnienia wprowadzenia zmian

42

Hierarchia konfiguracji

- Element konfiguracji
  - pojedynczy, możliwy do odseparowania komponent projektu lub produktu programistycznego
    - dokumentacja: wymagań, analityczna, projektowa, testowania, użytkownika
    - moduły z kodem źródłowym, kody do konsolidowania, kody binarne
    - ekrany interfejsu użytkownika
    - pliki z danymi tekstowymi (np. komunikatami systemu), bazy danych, słowniki
    - kompilatory, konsolidatory, interpretery, biblioteki, protokoły, narzędzia CASE, konfiguracje sprzętowe
    - oprogramowanie testujące, dane testujące
    - serwery WWW wraz z odpowiednimi stronami HTML i oprogramowaniem

Budowanie wydania

- Łączenie wszystkich komponentów w jeden wykonywalny system
- W przypadku dużych systemów naiwna kompilacja i linkowanie wszystkich komponentów może trwać godzinami/dniami

Zarządzanie wydaniami

- Wydania muszą uwzględniać zmiany
  - wymuszone w systemie w związku z wykryciem przez użytkowników błędów
  - związane z nową funkcjonalnością
- Planowanie wydań jest związane z
  - terminem udostępnienia nowej wersji
  - zakresem zmian włączanych do nowej wersji

45

Zarządzanie wydaniami (dobre praktyki)

- Przyrostowa zmiana systemu jaka może być włączona do nowego wydania jest w przybliżeniu stałego rozmiaru
- Jeśli zbyt wiele nowych cech zostanie włączonych razem z poprawkami błędów, wówczas koszt przygotowania nowego wydania znacząco wzrasta
- Jeśli do wydania włączono wiele zmian, musi nastąpić po nim wydanie poprawiające błędy wynikające ze zmian wprowadzonych w pierwszym wydaniu

46

Zarządzanie zmianą

- żądanie zmiany w postaci wniosku zmiany
- analiza wniosku zmiany
- **jeżeli** (wniosek zasadny) **wtedy**
  - analiza sposobu implementacji
  - analiza wpływu na harmonogram i kosztorys
  - przygotowanie propozycji zmiany
  - **jeżeli** (propozycja przyjęta) **wtedy**
    - **powtarzaj**
      - wprowadź zmiany
      - przedstaw wynik do weryfikacji
    - **dopóki** (jakość wyniku jest adekwatna)
    - utwórz nowe wydanie systemu
  - **w przeciwnym przypadku**
    - odrzuć wniosek zmiany
- **w przeciwnym przypadku**
  - odrzuć wniosek zmiany

47

Narzędzia zarządzania konfiguracją

- Rozwiązanie optymalne:
  - zintegrowane środowisko wspierające wszystkie etapy życia projektu
- Rozwiązanie akceptowalne:
  - środowisko deweloperskie zintegrowane z narzędziami projektowania wizualnego
  - system kontroli wersji
  - system zarządzania zmianą
  - rejestr wydań
- Rozwiązanie często spotykane:
  - brak systemowych rozwiązań

48

Praktyki wytwórcze – Jakość

	A	B	C	D	E
5	[REQM] Zarządzanie wymaganiami	[PP] Planowanie projektu	[PMC] Monitorowanie i kontrola projektu	[MA] Pomiary i analizy	[PPQA] Zapewnianie jakości procesu i produktu
4	[CM] Zarządzanie konfiguracją	[SAM] Zarządzanie dostawcami	[RD] Wydobywanie wymagań	[TS] Projektowanie i budowanie rozwiązań	[PI] Integracja produktów
3	[VER] Weryfikacja	[WAL] Walidacja	[OPF] Poprawa procesów organizacji	[OPD] Definiowanie procesów organizacji	[OT] Szkolenia w organizacji
2	[IPM] Zintegrowane zarządzanie projektami	[RM] Zarządzanie ryzykiem	[DAR] Wnioskowanie i analiza decyzji	[OEI] Integracja środowisk	[IT] Integracja zespołów
1	[ISM] Zintegrowane zarządzanie dostawcami	[OPP] Wydajność procesów w organizacji	[QPM] Ilościowe zarządzanie projektem	[OID] Wdrażanie innowacji w organizacji	[CAR] Analizy przyczyn źródłowych

Walidacja

- Czy system realizuje to co trzeba?
  - "Czy budujemy właściwy system?"
- Trudno to stwierdzić
- Oceny są zazwyczaj subiektywne

Weryfikacja

- Czy oprogramowanie jest zgodne ze specyfikacją?
  - "Czy prawidłowo budujemy system?"
- Może być obiektywne
- Specyfikacje muszą być wystarczająco precyzyjne

Trzy techniki

- Techniki

  - Testowanie
  - Przeglądy / inspekcje
  - Metody formalne
- Weryfikować należy wszystko
  - Także sam proces weryfikacji

Testowanie: Czarna skrzynka

- Generowanie przypadków testowych na podstawie specyfikacji
  - Nie patrzymy na kod programu
- Zalety:
  - Unikamy przyjmowania tych samych założeń co programista
  - Dane testowe są niezależne od implementacji
  - Wyniki można interpretować bez wnikania w szczegóły implementacyjne
- Trzy sugestie wyboru przypadków testowych
  - Ścieżki w specyfikacji
    - wybierz przypadki testowe pokrywające każdą z klauzul „wymaga”, „modyfikuje”, „wpływa na” w specyfikacji
  - Warunki brzegowe
    - Wybierz przypadki testowe dla warunków brzegowych zakresu danych wejściowych (lub blisko nich)
    - Szukaj błędów w aliasach (dwa parametry odnoszące się do tego samego obiektu)
  - Przypadki nienominalne
    - Wybieraj testy, które próbują każdego typu niepoprawnych danych wejściowych (program powinien elegancko obsłużyć taki przypadek, bez utraty danych)

Testowanie: Przezroczysta skrzynka

- Badanie kodu i testowanie ścieżek w kodzie
  - …ponieważ testowanie czarnej skrzynki nigdy nie gwarantuje, że wykonaliśmy wszystkie fragmenty kodu
- Kompletność ścieżek:
  - Zestaw testów pokrywa komplet ścieżek jeśli każda ścieżka w kodzie jest wykonana co najmniej raz w zestawie testów
    - Uwaga: To nie to samo, co stwierdzić, że każda instrukcja kodu jest wykonana!!

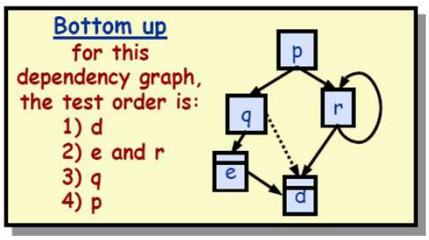
Testowanie: Testy komponentów / modułów

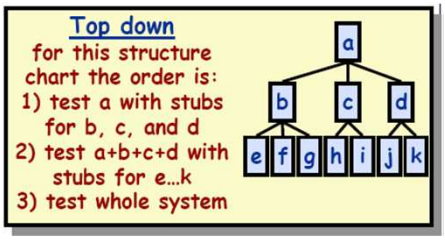
- Spopularyzowane przez Extreme Programming
- Każdy moduł jest testowany oddzielnie
  - Sprawdzamy czy spełnia specyfikację
  - Nie ma modułu bez testu
- Pisz test, zanim napiszesz moduł

55

Testowanie: Testy integracji

- Testy integracji
  - Testujemy czy moduły współpracują ze sobą
  - Dwie strategie:





- Testowanie integracji jest trudne:
  - Dużo trudniej zidentyfikować klasy równoważności
  - Pojawiają się problemy skali
  - Często wykrywamy błędy specyfikacji a nie błędy integracji

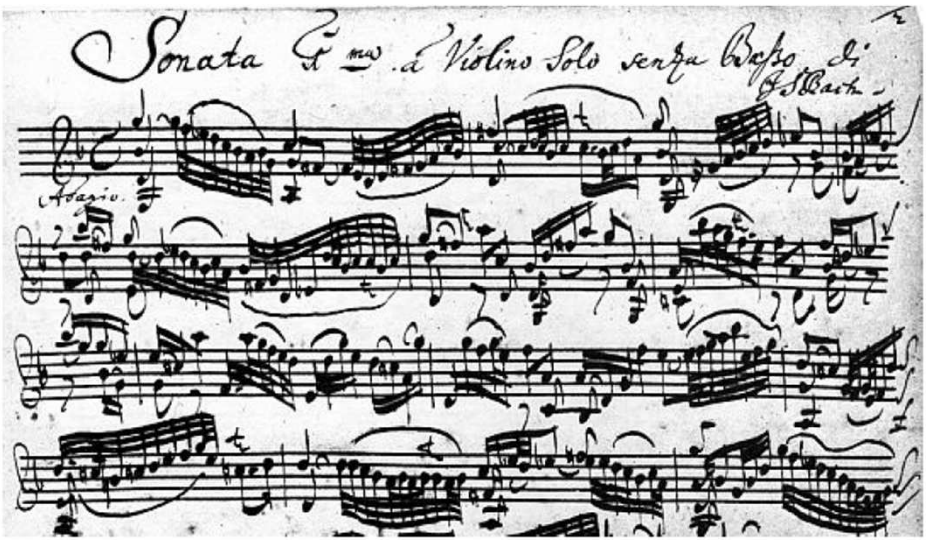
56

Testowanie: Testy systemowe

- Inne typy testów
  - facility testing – Czy system zapewnia wszystkie wymagane funkcje?
  - volume testing – Czy system radzi sobie z dużą ilością danych?
  - stress testing – Czy system radzi sobie z dużym obciążeniem?
  - endurance testing – Czy system zachowuje parametry w długim okresie?
  - usability testing – Czy system jest łatwy w użyciu?
  - security testing – Czy system wytrzymuje ataki?
  - performance testing – Jak dobry jest czas reakcji systemu?
  - storage testing – Czy pojawiają się problemy ze składowaniem danych?
  - configuration testing – Czy system działa na wszystkich platformach?
  - installability testing – Czy da się skutecznie zainstalować system?
  - reliability testing – Jak zmienia się niezawodność systemu w czasie?
  - recovery testing – Jak skutecznie system podnosi się po awarii?
  - serviceability testing – Czy daje się pielęgnować system?
  - documentation testing – Czy dokumentacja jest dokładna? Użyteczna?
  - operations testing – Czy instrukcje operatorów są poprawne?
- Regresja
  - Powtarzanie wszystkich testów po każdej modyfikacji

57

Zapewnianie jakości



58

Jakość w małej skali

- Jakość oprogramowania w ogólności oznacza zgodność z oczekiwaniami
  - Trzeba wiedzieć jakie są oczekiwania…
  - …jakie funkcje powinny być dostępne
  - …jakie inne własności muszą być zapewnione (modyfikowalność, niezawodność, użyteczność, …)
- Nie wszystkie atrybuty jakości można mierzyć w trakcie projektowania
  - ponieważ jakość nie jest atrybutem oprogramowania wyizolowanego ze środowiska
  - ale możemy patrzeć na wskaźniki ułatwiające prognozowanie
- Niezawodność, wydajność, pielęgnowalność, użyteczność
  - to zazwyczaj cztery najważniejsze czynniki jakości
  - …chociaż wiele opracowań podaje inne listy
- Stopień modularności jest często dobrym wskaźnikiem prognozy jakości
  - measure it by looking at cohesion and coupling

Jakość w dużej skali

- Skala problemu
  - (zapewnienia jakości wytwarzanego oprogramowania)
- rośnie wraz ze skalą organizacji
- Bliski związek pomiędzy jakością produktu a jakością procesu
- Aby wyprodukować dobry produkt potrzeba dobrego procesu
- Dla dóbr wytwarzanych ręcznie proces ma kluczowe znaczenie
- Dla czynności opartych na projektach, inne czynniki są istotne, w tym zdolności projektantów

Inżynieria Oprogramowania, MIMUW

Praktyki wytwórcze – Planowanie, monitorowanie

	A	B	C	D	E
5	[REQM] Zarządzanie wymaganiami	[PP] Planowanie projektu	[PMC] Monitorowanie i kontrola projektu	[MA] Pomiary i analizy	[PPQA] Zapewnianie jakości procesu i produktu
4	[CM] Zarządzanie konfiguracją	[SAM] Zarządzanie dostawcami	[RD] Wydobywanie wymagań	[TS] Projektowanie i budowanie rozwiązań	[PI] Integracja produktów
3	[VER] Weryfikacja	[WAL] Walidacja	[OPF] Poprawa procesów organizacji	[OPD] Definiowanie procesów organizacji	[OT] Szkolenia w organizacji
2	[IPM] Zintegrowane zarządzanie projektami	[RM] Zarządzanie ryzykiem	[DAR] Wnioskowanie i analiza decyzji	[OEI] Integracja środowisk	[IT] Integracja zespołów
1	[ISM] Zintegrowane zarządzanie dostawcami	[OPP] Wydajność procesów w organizacji	[QPM] Ilościowe zarządzanie projektem	[OID] Wdrażanie innowacji w organizacji	[CAR] Analizy przyczyn źródłowych

61

Inżynieria Oprogramowania, MIMUW

Przypomnienie...

62

Punkty funkcyjne

- Chcemy określić pracochłonność projektu
- Bazujemy na funkcjonalności systemu
- Uwzględniamy aspekty
  - Technologiczne
  - Środowiskowe
  - Funkcjonalne
    - Przypadki użycia
    - Aktorów

63

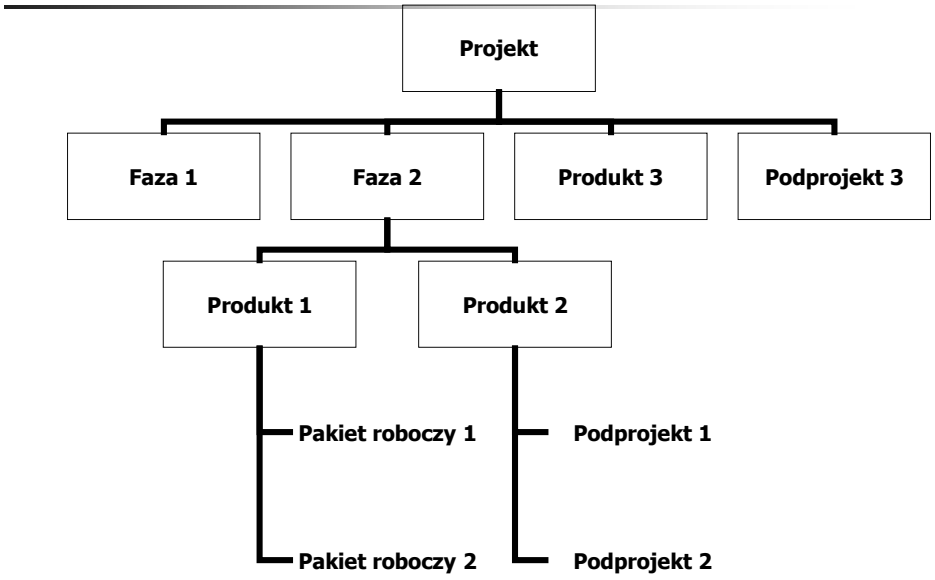
Punkty obiektowe

- Łatwiej oszacować na podstawie specyfikacji niż punkty funkcyjne
  - Liczone na podstawie elementów zawartych w specyfikacji
    - Ekrany
    - Raporty
    - Moduły
- Możliwość oszacowania na wstępnym etapie projektu
- Wtedy trudno jeszcze np. oszacować ilość linii kodu

64

Inżynieria Oprogramowania, MIMUW

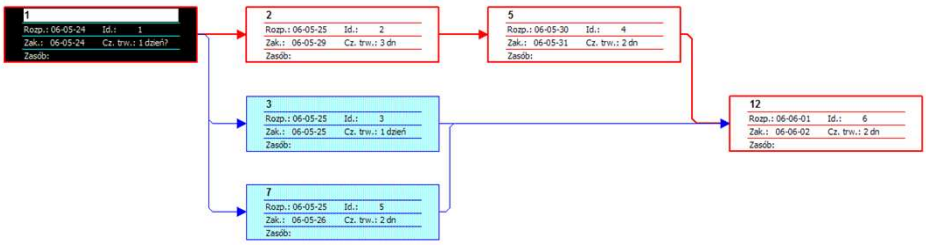
Struktura podziału pracy



65

Inżynieria Oprogramowania, MIMUW

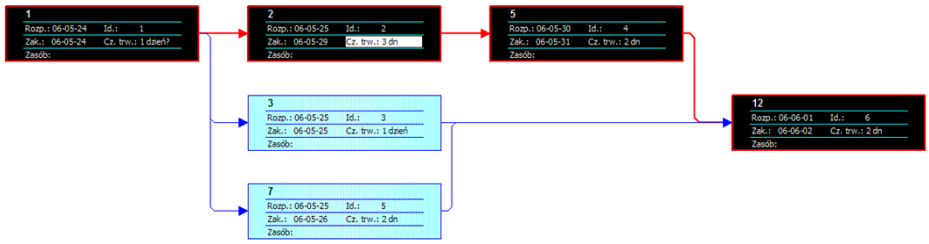
Diagram sieciowy



66

Inżynieria Oprogramowania, MIMUW

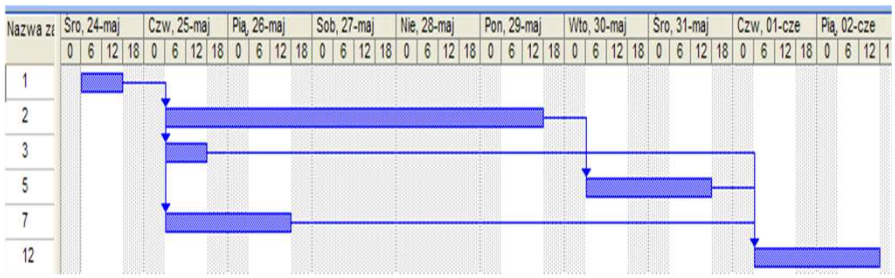
Ścieżka krytyczna



67

Inżynieria Oprogramowania, MIMUW

Wykres Gantta



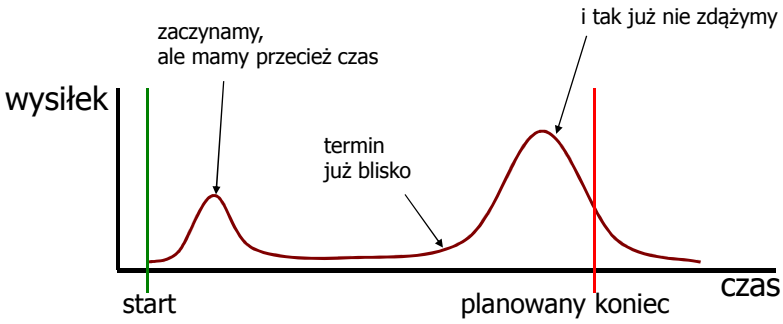
68

Prawo Parkinsona

- Praca trwa tyle, ile na nią przeznaczymy
  - Koszt jest często oszacowywany na podstawie dostępnych zasobów a nie celów
    - Jeśli 5 osób ma wytworzyć system w 12 miesięcy, to koszt wyniesie 60 osobomiesięcy

69

Syndrom Studenta



70

Podsumowanie – Co było celem projektów zespołowych?