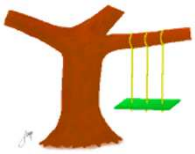
Inżynieria Oprogramowania

Wprowadzenie

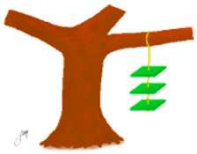
Z życia wzięte (www.dilbert.com)

http://dilbert.com/strips/comic/2008-11-09/

Pół żartem, pół serio

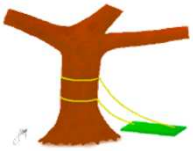


klient zamówił

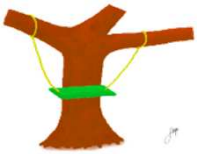


analityk zrozumiał

projekt opisywał



programiści wykonali

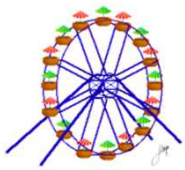


wykorzystano materiały: Patryk Sielski, Politechnika Warszawska

Inżynieria Oprogramowania

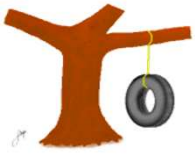
Pół żartem, pół serio

projekt po wdrożeniu



klient zapłacił za

klient potrzebował



zastosowanie praktyczne

Tworzenie oprogramowania – nauka czy sztuka?

„Software engineering is, in fact, a form of engineering.”
[David Parnas]

„It is not, but that it should be.”
[Steve McConnell]

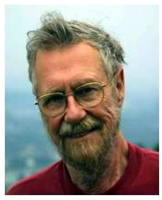
„Programming is an art.”
[Donald Knuth]

Inżynieria Oprogramowania

Tworzenie oprogramowania – nauka czy sztuka?

„It is not exactly art, but it needs an artist in order to exist.”
[Yves Saint Laurent – o modzie ;-)]

Tworzenie oprogramowania – nauka czy sztuka?



„The major cause of the software crisis is that the machines have become several orders of magnitude more powerful!

To put it quite bluntly:

as long as there were no machines, programming was no problem at all;

when we had a few weak computers, programming became a mild problem;

and now we have gigantic computers, programming has become an equally gigantic problem.”
[Edsger Dijkstra]

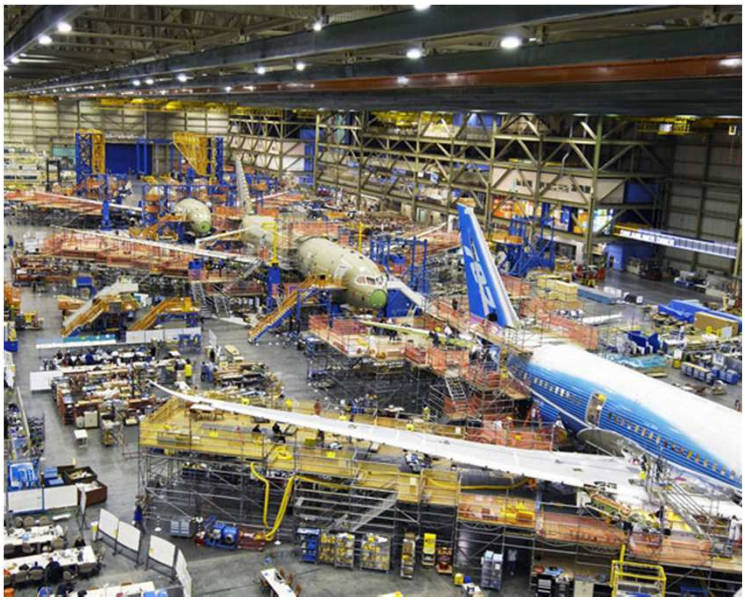
Definicja

inżynieria oprogramowania

=

{ludzie, architektura, proces }

Metafora

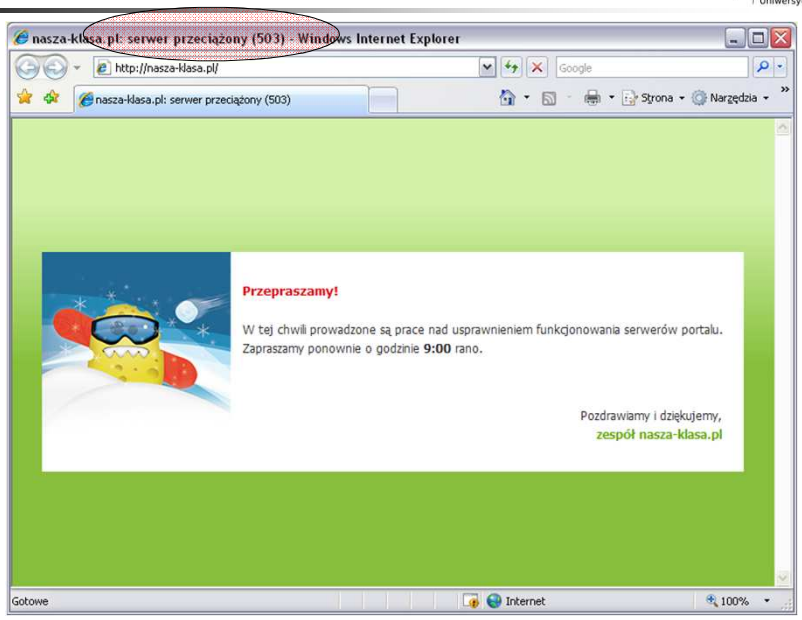


Inżynieria Oprogramowania

Skala ma znaczenie

http://dilbert.com/strips/comic/2010-10-26/

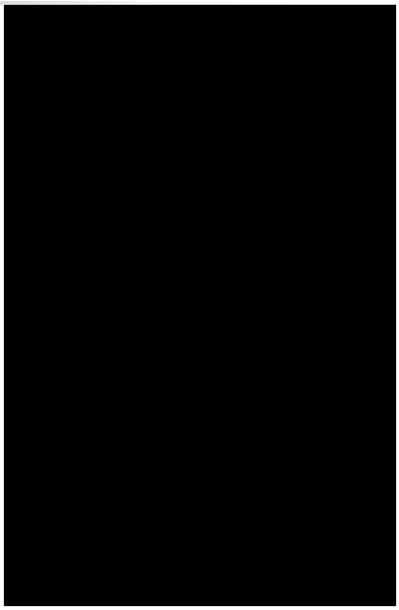
Niby to wiemy, ale …



Inżynieria Oprogramowania

Niby to wiemy, ale …

- Projekt „Ariane” v4
  - zakończony wielkim sukcesem
- Projekt „Ariane” v5
  - kolejny etap po Ariane-4, ale nie będący jego sukcesorem
  - przeznaczony dla większych ładunków
- Etap pierwszy projektu
  - około 10 lat, €7 miliardów
  - początkowo projektowany na potrzeby wynoszenia wahadłowców Hermes
  - projekt Hermes zaniechano
- Etap drugi projektu
  - Ariane przekształcono w rakietę komercyjną
  - Głównym celem dostarczanie satelitów na orbitę geostacjonarną
- 4 czerwca 1996
  - 37 sekund po starcie Ariane uległa samozniszczeniu
  - Zawiódł system komputerowy odpowiadający za wyrównywanie platformy startowej
  - Utrata ładunku wartego około $500M (podobno ładunek nie był ubezpieczony)
  - Jeden z najbardziej znanych błędów komputerowych w historii

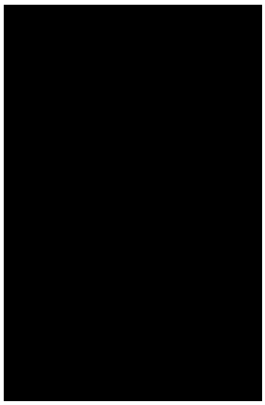


Niby to wiemy, ale …

- 30 sekund po starcie
  - wyłączył się system zapasowy platformy startowej
  - 50ms później wyłączył się system główny platformy startowej
- 31 sekund po starcie
  - komputer pokładowy odbiera błędną informację z systemu platformy
  - komunikat diagnostyczny błędnie zinterpretowany jako dane lotu
  - komputer pokładowy uruchamia pełen ciąg, rakieta zbacza z kursu
- 33 sekundy po starcie
  - następuje rozpad rakiety
  - zostaje uruchomiona procedura samozniszczenia
- Bezpośrednia przyczyna katastrofy
  - w języku Ada nie obsłużony wyjątek „Converting 64-bit floating point to 16-bit signed integer”
  - wymagania określały, że w przypadku nieobsłużonego wyjątku komputer powinien się wyłączyć
- Dodatkowe czynniki
  - System miał podawać informacje poprawne jedynie przed startem
  - System działał jeszcze przez 40 sekund po starcie podając błędne informacje

Inżynieria Oprogramowania

Niby to wiemy, ale …



```
...
declare
  vertical_veloc_sensor: float;
  horizontal_veloc_sensor: float;
  vertical_veloc_bias: integer;
  horizontal_veloc_bias: integer;
  ...
begin
  declare
    pragma suppress(numeric_error, horizontal_veloc_bias);
  begin
    sensor_get(vertical_veloc_sensor);
    sensor_get(horizontal_veloc_sensor);
    vertical_veloc_bias := integer(vertical_veloc_sensor);
    horizontal_veloc_bias := integer(horizontal_veloc_sensor);
    ...
  exception
    when numeric_error => calculate_vertical_veloc();
    when others => use_irs1();
  end;
end irs2;
```

Niby to wiemy, ale …

- Dlaczego system miał działać po starcie?
  - oprogramowanie z Ariane-4
  - 40-sekundowe opóźnienie na wypadek przestoju umożliwiało uniknięcie restartowania całości systemów komputerowych
  - ta cecha przydała się raz (w 1989)
- Dlaczego nie było obsługi wyjątków?
  - obniżenie obciążenia procesora poniżej 80%
  - analiza dla Ariane-4: przepełnienie nie było fizycznie możliwe
  - Ariane-5 miało inną trajektorię
- Dlaczego nie zmodyfikowano oprogramowania dla Ariane-5?
  - uznano za nierozsądne modyfikowanie oprogramowania, które sprawdziło się przy Ariane-4
- Dlaczego system wyłączył się przedwcześnie?
  - błędy uznane za sprzętowe, stąd przełączenie się pomiędzy systemem zapasowym a głównym
- Dlaczego błędu nie wykryto w testach jednostkowych?
  - trajektoria Ariane-5 nie została określona w wymaganiach systemu
- Dlaczego błędu nie wykryto w testach integracyjnych?
  - pełne testy integracyjne uznane za zbyt trudne i kosztowne
- Dlaczego błąd nie został wychwycony podczas inspekcji kodu?
  - założenia implementacyjne nie zostały udokumentowane
- Dlaczego komputer pokładowy uznał dane diagnostyczne za dane lotu?
  - nie wiadomo; może założono, że taka sytuacja nigdy się nie zdarzy???

Inżynieria Oprogramowania

Czym się zajmiemy na IO?

Q:

czym się zajmujemy na wykładzie?

A:

modele wytwórcze i dobre praktyki

wzorce architektoniczne i projektowe

projektowanie, skalowalność i niezawodność

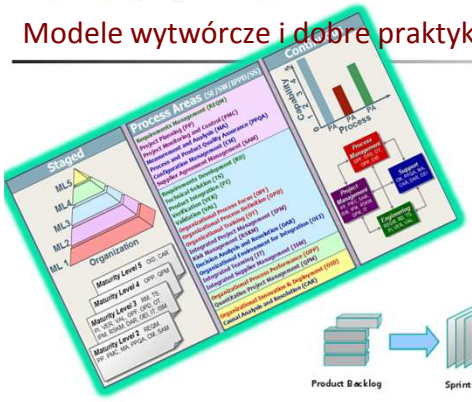
optymalizacja i refaktoring

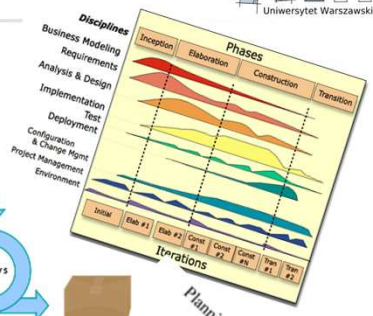
testowanie i jakość procesu i produktu

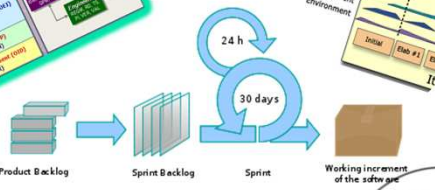
zarządzanie konfiguracją i zmianą

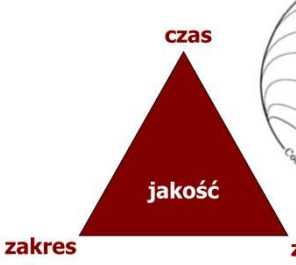
Inżynieria Oprogramowania

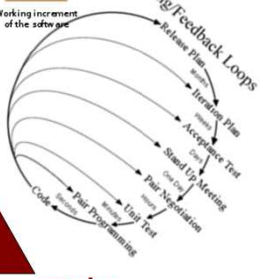
Modele wytwórcze i dobre praktyki





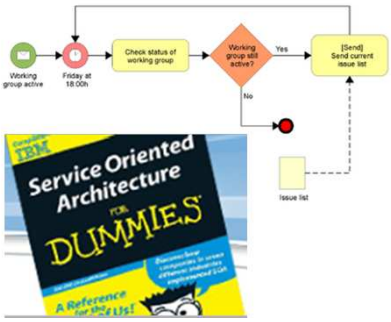






Inżynieria Oprogramowania

Wzroce architektoniczne i projektowe

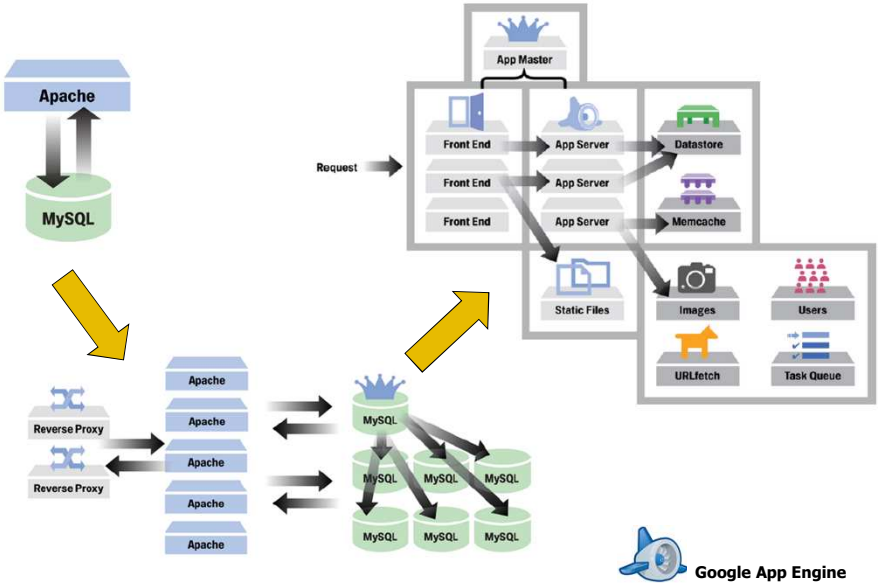


Microsoft | Cloud Services

We're all in.

Google App Engine

Projektowanie, skalowalność, niezawodność

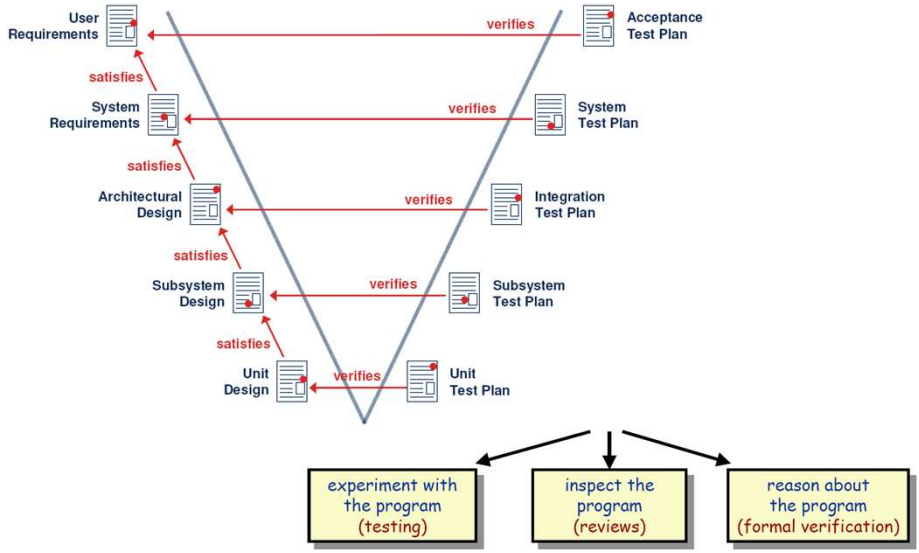


Optymalizacja i refaktoring

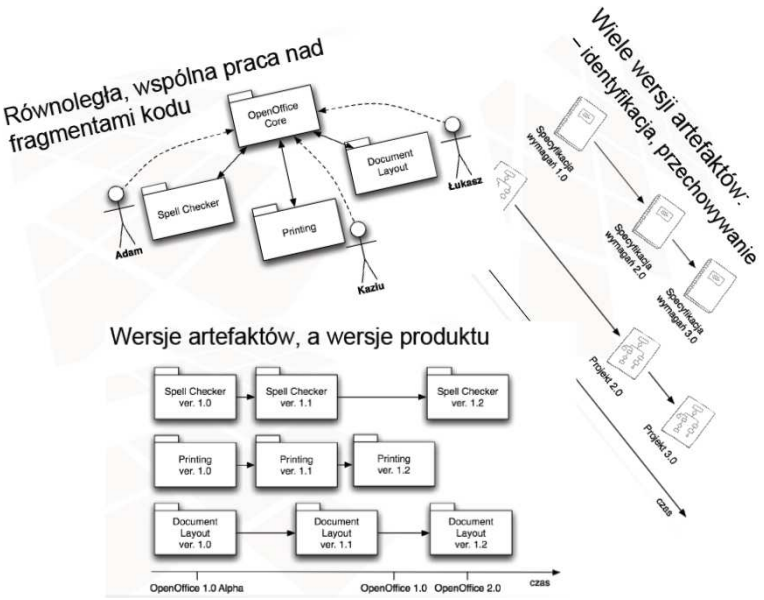
- Jak zapewnić wysoką jakość produktu, gdy zmienia się on cały czas?
- Poprawiać kod czy przepisywać?
- Narzędzia?

- *Instrumentation framework or building dynamic analysis tools*.

Testowanie i jakość produktów i procesu



Zarządzanie konfiguracją i zmianą



Czym się zajmiemy na IO?

Q:

Czym się zajmujemy na laboratorium?

A:

przemyślana i opisana architektura

kompilowalny prototyp

wszystkie ważne decyzje podjęte i odnotowane

(co do zakresu, technologii, itp.)

Nie będzie łatwo

http://dilbert.com/strips/comic/1999-09-07/

Organizacja zajęć

- Wykład
  - moodle.mimuw.edu.pl
    - klucz = IO0216
    - prezentacje z wykładu
    - przykładowe egazaminy
    - ankieta
  - dev.mimuw.edu.pl
  - svn.mimuw.edu.pl
  - ankieta oceniająca wykłady
    - link na moodle
- Laboratorium
  - projekty zespołowe
  - rozwinięcie wykładu
- Zespoły
  - ±4-osobowe
- Tematy
  - projekty rzeczywiste
  - kontynuacja na ZPP (lic)
  - kontynuacja na WSI (mgr)

Inżynieria Oprogramowania

Zasady oceniania

- Ocena z laboratorium – architektura i implementacja
- Ocena z egzaminu – zadania testowe
  - pytania wielokrotnego wyboru
  - punkty ujemne za złe odpowiedzi
  - pytania do uzupełniania
  - zadania z projektowania
- Ocena końcowa to średnia ważona
  - 45% – ocena z egzaminu (praca indywidualna)
  - 55% – ocena z laboratorium (praca zespołowa)

Harmonogram zajęć (przybliżony)

Nr	Wykład	Laboratorium
01	Wprowadzenie	Ustalenie zespołów i projektów
02	Architektura	Wizja
03	Projektowanie	Narzędzia
04	Specyfikowanie	Specyfikacja funkcjonalna
05	Wzorce	Specyfikacja uzupełniająca / pozafunkcjonalna
06	Wzorce	Scenariusze testowe
07	Jakość	Model dziedziny
08	Konfiguracja	Model fizyczny
09	Zmiana	Prezentacje zespołów cz. 1
10	Planowanie	Prezentacje zespołów cz. 1
11	Metodyki	Architektura techniczna (uaktualnienie)
12	Metodyki	Architektura funkcjonalna (uaktualnienie)
13	Zarządzanie	Plan (pracochłonność, zasoby, harmonogram)
14	Podsumowanie	Prezentacje zespołów cz. 2
15	Podsumowanie	Prezentacje zespołów cz. 2

Inżynieria Oprogramowania

Kontekst

Bazy Danych
Programowanie Obiektowe
Indywidualny Projekt Programistyczny
↓
Inżynieria Oprogramowania
↓
Zespołowy Projekt Programistyczny (licencjat)
↓
Zarządzanie Projektami Informatycznymi (monograf)
Wytwarzanie Systemów Informatycznych (seminarium)

Kontakt

- Poczta elektroniczna
  - r.dabrowski@mimuw.edu.pl
  - W tytule prefiks „[IO2011]” (bez cudzysłowów)
- Strona przedmiotu
  - moodle.mimuw.edu.pl
  - literatura na stronie
- Pokój
  - 4270

Inżynieria Oprogramowania