Inżynieria oprogramowania
Wykład 1

Wprowadzenie do inżynierii oprogramowania

Bartosz Walter
<Bartek.Walter@man.poznan.pl>

Prowadzący

- **dr inż. Bartosz Walter**
- **Instytut Informatyki PP**
- **Pokój: Centrum Polsko-Niemieckie p. 2**
- **Email: Bartosz.Walter@cs.put.poznan.pl**
- **WWW: http://www.se.cs.put.poznan.pl/**

Agenda

- **Historia i geneza inżynierii oprogramowania**
- **Rola inżynierii**
- **Aspekty inżynierii oprogramowania**
- **Plan wykładów**

Rynek oprogramowania

- Świat 207 miliardów euro (USA 97 miliardów, UE 63 miliardy)
  - Bez oprogramowania wytwarzanego na własne potrzeby
- Wzrost 5.1% rocznie
- + 125 miliardów euro dodatkowych usług
- W UE 60-70% oprogramowania jestwytwarzane w firmach, dla których nie jest to główną działalnością

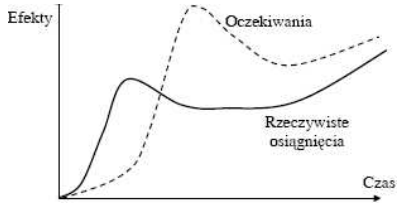
Historia: początki

- Sprzęt o bardzo ograniczonych możliwościach
- Ograniczone zastosowania
- Małe programy
- Programy pisane często dla własnych potrzeb lub potrzeb dobrze znanych osób
- Dobrze wyspecyfikowane zadania

Historia: rozwój

- Nowy zawód: programista
- Nowa rewolucja przemysłowa (Osborne 1979)
- Specjalizowane języki programowania – COBOL, Fortran, Algol
- Sprzęt o dużo większych możliwościach, np. pamięć wirtualna, pamięć masowa, czytniki kart i taśm
- Nowe, poza naukowe i poza militarne zastosowania – np. w biznesie
- **Próba realizacji wielu dużych przedsięwzięć programistycznych**

Objawy kryzysu oprogramowania



Rozwój technik wytwarzania oprogramowania nie nadąża za rozwojem sprzętu komputerowego i potrzeb klientów

Metody tworzenia oprogramowania się nie skalują

Historia: kryzys oprogramowania

- Czy kryzys oprogramowania trwa do dzisiaj?
  - Nadal większość przedsięwzięć przekracza czas i/lub budżet
  - Około 25% przedsięwzięć programistycznych nie jest kończona
  - 90% firm przyznaje, że dość często zdarzają im się opóźnienia przedsięwzięć
  - Powszechna akceptacja kiepskiej jakości oprogramowania (w pewnych obszarach)
  - Brak powszechnie akceptowanych i stosowanych standardów

Przyczyny kryzysu

- Duża złożoność systemów informatycznych
- Złożoność, zmienność, nieadekwatność wymagań
- Niepowtarzalność poszczególnych przedsięwzięć
- Nieprzejrzystość procesu budowy oprogramowania
- Pozorna łatwość wytwarzania i modyfikowania oprogramowania
- Potrzeba kreatywności
- Czynnik ludzki
- Mało wymagający rynek (?)
- Niedoskonałość narzędzi
- Brak standaryzacji

Historia: próby przełamania kryzysu

- 1968: konferencja NATO w Ga-Pa nt. inżynierii oprogramowania
- Metodyki strukturalne
- Metodyki obiektowe
- Technologie komponentowe
- Programowanie aspektowe
- Standaryzacja technologii

Time to Market

Kryzys czy chroniczne niedomaganie?

Czym jest inżynieria oprogramowania?

- Inżynieria oprogramowania dotyczy tworzenia dużych systemów informatycznych przez wiele osób
- Inżynieria oprogramowania to wiedza techniczna, dotycząca wszystkich faz cyklu życia oprogramowania, której celem jest uzyskanie wysokiej jakości produktu – oprogramowania.
- Inżynieria oprogramowania: zastosowanie systematycznego, zdyscyplinowanego, policzalnego podejścia do tworzenia, wykorzystywania i utrzymania oprogramowainia.
- Inżynieria oprogramowania = proces + metody zarządcze i techniczne + narzędzia

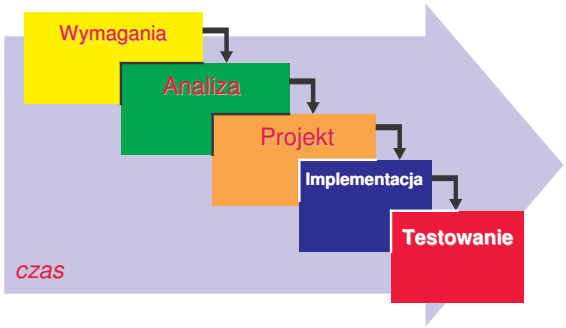
Oprogramowanie

- Oprogramowanie jest budowane lub rozwijane w sposób inżynierski (zdefiniowany, uporządkowany), jednak w zupełnie inny niż w innych dziedzinach inżynierii
- Oprogramowanie się nie zużywa, choć jego jakość się pogarsza
- Choć przemysł skłania się ku tworzeniu oprogramowania z komponentów i reużywalności, to jednak większość systemów jest dedykowana

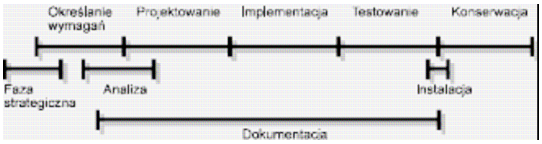
Modele procesu tworzenia oprogramowania

- **Model klasyczny wodospadowy**
  - Zamknięcie jednej fazy otwiera następną
- **Prototypowanie**
  - Pierwszy wstępny prototyp (działający!), potem właściwa implementacja
- **Programowanie odkrywcze**
  - Programiści odgadują potrzeby klienta
- **Model przyrostowy**
  - Kolejne funkcje są implementowane stopniowo

Model klasyczny (wodospad)



Model klasyczny (wodospad)



Wady i zalety modelu wodospadowego

- Wymuszona niezmienność wymagań
- Wysoki koszt błędów popełnionych we wstępnych fazach
  - Koszt błędu w wymaganiach 100-1000 razy większy od kosztu błędu programistycznego!
- Długa przerwa w kontaktach z klientem
- Nie lubiany przez wykonawców
- Wyższe prawdopodobieństwo niepowodzenia w przypadku dużych przedsięwzięć
- Łatwość zarządzania – planowanie i monitorowanie

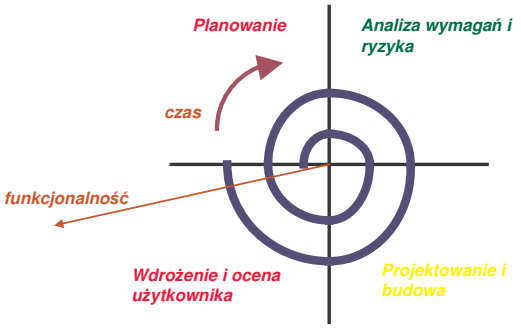
Prototypowanie

- **Cel – lepsze określenie wymagań**
- **Fazy:**
  - Ogólne określenie wymagań.
  - Budowa prototypu.
  - Weryfikacja prototypu przez klienta.
  - Pełne określenie wymagań.
  - Realizacja nowego, pełnego systemu zgodnie z modelem kaskadowym.

Wady i zalety prototypowania

- Zmniejszone ryzyko popełnienia błędu podczas definiowania wymagań i analizy
- Szybka prezentacja klientowi działającego szkieletu aplikacji (szybka reakcja)
- Dodatkowy koszt wytworzenia prototypu
- Możliwość nieporozumienia z klientem

Model spiralny (B. Boehm)



Wady i zalety modelu spiralnego

- Możliwość zmiany wymagań w trakcie realizacji systemu
- Szybka reakcja klienta na pojawiające się zmiany
- Niektóre funkcje są dostępne przed zakończeniem przedsięwzięcia
- Integracja!
- Zarządzanie!

Składowe inżynierii oprogramowania

Modelowanie

- Modelowanie: odtwarzanie rzeczywistości w celu uchwycenia najważniejszych elementów i pomijanie rzeczy nieistotnych
  - Elementami modelu są elementy rzeczywistości
- Modelowanie procesów a modelowanie danych
- Dekompozycja strukturalna
- Modelowanie obiektowe (UML): obiekty, relacje, atrybuty

Projektowanie

- Reprezentacja modelu uwzględniająca ograniczenia techniczne i sposób realizacji
- Ten sam model może być zaprojektowany na różne sposoby
  - Elementami projektu są byty implementacyjne
- Projektowanie strukturalne vs. Obiektowe
- Pojęcia: abstrakcja, uszczegółowienie, modularność, architektura, hermetyzacja
- Projektowanie klas, relacji, interfejsów...

Testowanie

- Błędów nie można całkowicie wyeliminować
  - Testowanie jest czynnością destrukcyjną
  - Dobry test to test, który z dużym prawdopobieństwem pozwala wykryć błąd
  - Skuteczny test to test, który znajduje nowy błąd
- Testowanie a model tworzenia oprogramowania
- Dobór danych testowych i obszaru testowania
  - Zasada Pareto
  - Skalowalność – od szczegółu do ogółu
- Niezależność testowania od tworzenia oprogramowania

Testowanie: zadanie

```
int oblicz(int arg1, int arg2, int operacja) {
  switch (operacja) {
    case DODAWANIE: return arg1 + arg2;
    case ODEJMOWANIE: return arg1 – arg2;
    case MNOŻENIE: return arg1 * arg2;
    case DZIELENIE: return arg1 / arg2;
    default: return –1;
  }
}
```

Szacowanie

- Czy przedsięwzięcie jest wykonalne?
- Czas = pieniądz
- Dekompozycja
  - Funkcjonalna
  - Produktowa
- Modele
  - Punkty funkcyjne
  - COCOMO II
  - PROBE

Dokumentowanie

- Programy żyją dłużej niż pamięć twórców
- Spójność artefaktów (automatyzacja?)
- Dokumentacja użytkowa a dokumentacja techniczna
- Język i psychologia
- Dokumentowanie kosztuje...
- Nikt nie lubi dokumentowania, ale dobrze gdy dokumentacja istnieje

Planowanie

- Zasoby
  - Czas
  - Ludzie
  - Narzędzia
  - Materiał (komponenty)
- Zadania i ich przydział
- Harmonogramy
- Czy przedsięwzięcie jest wykonalne?

Wersjonowanie i zarządzanie zmianą

- Zmiany są nieuniknione
- Jaka wersja modułu X posiadała funkcję A?
- Identyfikacja elementów i ich wersji
- Spójność wersji a praca w zespole
- Wiele wersji jednego oprogramowania
- Formalne zatwierdzanie zmian

Pomiary

- Potrzeba ilościowej informacji nt. produktu i procesu
- Informacja zarządcza
  - nie można sterować niczym, czego nie można zmierzyć
  - Dane historyczne, teraźniejszość i przyszłość
- Rodzaje metryk
  - Produktowe: rozmiar, złożoność komponentów, liczba interfejsów, ekranów, formatek, stron dokumentacji, liczba wykrytych błędów
  - Procesowe: czas, koszt, opóźnienie, stopień zrównoleglenia, liczba wymagań, czas poświęcony na testowanie

Reinżynieria

- Nieustanny pęd ku lepszemu
- Zmiana fundamentów w czasie eksploatacji
- Oblicza zmian
  - Zmiana procesu
  - Zmiana funkcjonalności
  - Zmiana kodu (refaktoryzacja)
- Koszt pielęgnacji = 80% TCO
- Inżynieria odwrotna: odtwarzanie projektu na podstawie programu

Plan – I sem.

- Modelowanie (UML)
- Projektowanie (design patterns)
- Programowanie w języku Java
- Testowanie (scenariusze, Junit)
- Szacowanie (PROBE, Cocomo II)
- Dokumentowanie (użytkowe i techniczne)
- Zarządzanie zmianą i wersjonowanie
- Pomiary
- Narzędzia

Readings

1. Jaszkiewicz A., *Inżynieria oprogramowania*. Helion 1997.
2. Górski J., *Inżynieria oprogramowania w projekcie informatycznym*. Mikom 2000.
3. Fowler M., *UML w kropelce*. Mikom 2001.
4. Szejko St. (red.), *Metody wytwarzania oprogramowania*. Mikom 2002.
5. Pressman R., *Software Engineering. A Practitioner's Approach*. McGraw Hill 2001.
6. Sommerville I., *Software Engineering. Addison-Wesley* 1995.

Q & A