Politechnika Warszawska

Wprowadzenie do inżynierii oprogramowania

dr hab. inż. Michał Śmiałek, prof. uczelni
dr inż. Kamil Rybiński

Wydział Elektryczny
POLITECHNIKA WARSZAWSKA

Inżynieria oprogramowania

1

Czym jest inżynieria oprogramowania

- Termin po raz pierwszy użyty podczas konferencji NATO w 1967 roku
- Podstawowe cechy inżynierii oprogamowania
  - Inżynierskie podejście do tworzenia oprogramowania
  - Budowa oprogramowania w ramach typowych dla innych dziedzin inżynierii
  - Stosowanie wiedzy naukowej, technicznej i doświadczenia w celu projektowania, implementacji, walidacji i dokumentowania oprogramowania

2

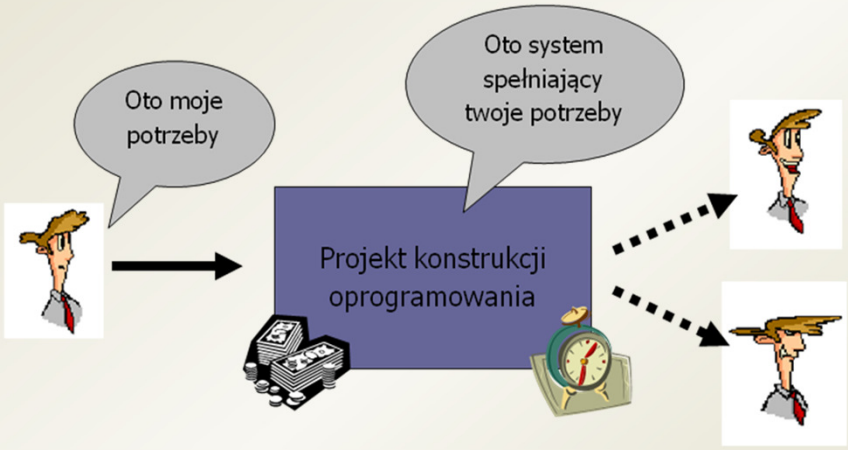
Dyscypliny inżynierii oprogramowania

- Wymagania
  - Uzyskanie od osób zainteresowanych systemem (klienta) informacji o ich potrzebach
- Projektowanie
  - Stworzenie „rysunków technicznych” budowanego systemu
- Implementacja
  - Stworzenie kodu systemu (programu) na podstawie projektu
- Walidacja
  - Sprawdzenie, czy system odpowiada potrzebom klienta
- Nadzór
  - Zapewnienie przestrzegania procedur i reguł postępowania
- Środowisko pracy
  - Wybór oraz stosowanie odpowiednich narzedzi pracy

Złożoność inżynierii oprogramowania

- Typowy system oprogramowania zawiera setki tysięcy lub nawet miliony wierszy kodu
  - Przykład: system Red Hat Linux (wersja 7.1) zawiera ponad 30 milionów wierszy kodu
- Złożoność różnych dziedzin wspomaganych systemami oprogramowania
- Różnorodność możliwych funkcjonalności
  - Pytania: ilu różnych funkcjonalności dostarcza typowy procesor tekstu, aplikacja bankowa, …?
- Systemy oprogramowania występują we wszystkich praktycznie dziedzinach współczesnego życia
  - Oprogramowanie aplikacyjne
  - Oprogramowanie wbudowane (telewizory, maszyny do szycia, samochody, maszynki do golenia, …)

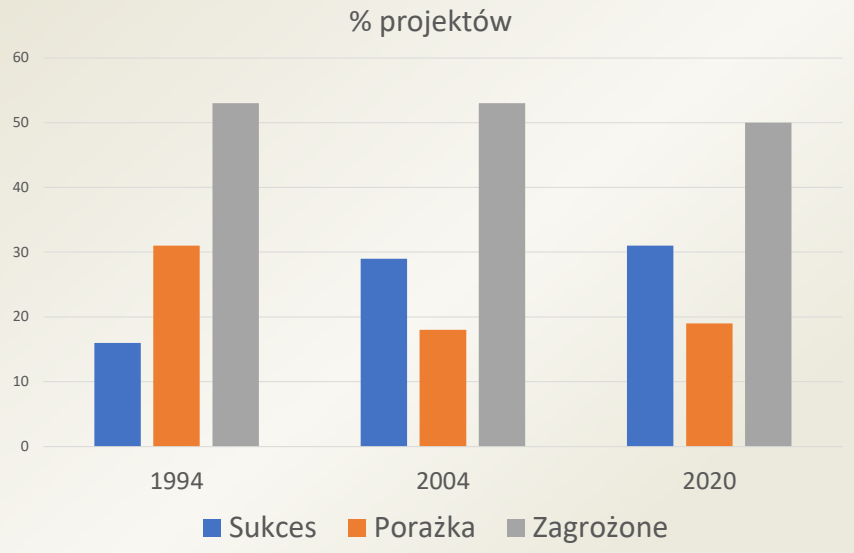
Kryteria powodzenia projektu?



Projekt powinien: mieścić się w budżecie, być zakończony o czasie i spełniać rzeczywiste potrzeby klienta

5

A jak jest w rzeczywistości?



6

Kryzys czy choroba przewlekła?

- Coroczny „raport chaosu”
  - Znaczna część projektów kończy się niepowodzeniem (porażka lub poważne problemy)
  - Budżety przekroczone w znacznym stopniu (np. w 2004 roku średnio o 82%)
  - Niespełnione wymagania (średnio 52%)
- Konferencja NATO: jest kryzys!
  - Ta konferencja odbyła się w 1967 roku!
- Roger Pressmann: to jest choroba przewlekła!

7

Objawy „przewlekłej choroby”

- Niezadowoleni klienci
  - Brak akceptacji dla systemu
- Niezadowoleni dostawcy
  - Zdziwienie niezadowoleniem klienta
- Kłótnie o zakres
  - Niejasno zapisane wymagania i warunki kontraktu
- Chaotyczne zmiany wymagań
  - Brak zasad w przypadku zmian
- Programiści pracują 24/7
  - Rozmiar systemu przerasta możliwości zespołu
- Stres na koniec projektu
  - Termy, terminy, …
- Brak stabilności rezultatów
  - W każdym projekcie jest inaczej
- Syndrom systemu „prawie gotowego”
  - System jest już w zasadzie gotowy, ale nie działa …

8

Przyczyny problemów (1)

- Nieprecyzyjne specyfikowanie
  - Wymagania formułowane jak beletrystyka
- Zła komunikacja
  - Uczestnicy projektu rozmawiają różnymi językami
  - Użycie nieprecyzyjnych notacji
- Brak projektowania architektonicznego
  - Pomijanie dyscypliny projektowania
  - Pisanie kodu bezpośrednio na podstawie (nieprecyzyjnych) wymagań
- Brak zarządzania złożonością
  - Brak dobrze określonego podziału systemu na mniejsze fragmenty
  - Konieczność czasochłonnego odtwarzania struktury kodu podczas poprawek i zmian

Przyczyny problemów (2)

- Późne odkrywanie nieporozumień
  - Nieporozumienia odkrywane dopiero podczas walidacji systemu
  - Złe zrozumienie wymagań przez deweloperów
- Brak zarządzania zmianami
  - Brak procedur w przypadku wystąpienia zmiany wymagań czy technologii
  - Niemożność oceny zakresu systemu, który się zmienia
- Nieużywanie narzędzi wspomagających
  - Tworzenie oprogramowania jedynie przy pomocy środowiska programistycznego
  - Brak narzędzi do zarządzania wymaganiami, projektowania, wdrażania, …

Ocena dojrzałości do tworzenia oprogramowania

- Model CMM-SW (Capability Maturity Model – Software)
  - Powstały w latach 80. na uniwersytecie Carnegie-Mellon
  - Poziom 1 (initial): brak procesu, ad-hoc
  - Poziom 2 (repeateable): powtarzalne praktyki, często nieformalne
  - Poziom 3 (defined): zdefiniowany i przestrzegany standard procesu wytwórczego
  - Poziom 4 (managed): zarządzany
  - Poziom 5 (optimizing): optymalizujący

Najlepsze praktyki (1)

- Produkuj iteracyjnie
  - System oddawany klientowi w sposób przyrostowy
- Stosuj architektury komponentowe
  - Podział systemu na komponenty
  - Panowanie nad złożonością poprzez określenie ram dla projektantów i programistów
- Stale kontroluj jakość
  - Poprawność systemu sprawdzana od początku projektu
  - Ciągłe sprawdzanie satysfakcji klienta

Najlepsze praktyki (2)

- Zarządzaj wymaganiami
  - Podział wymagań na zarządzalne jednostki
  - Tworzenie śladów dla wymagań
  - Stosowanie procedur zarządzania wymaganiami
- Zarządzaj zmianami
  - Ustanowienie oficjalnego sposobu zgłaszania zmian
  - Stosowanie procedur obsługi zmian
- Modeluj wizualnie
  - Tworzenie diagramów pomagających zrozumieć skomplikowany system
- Uwaga: najlepsze praktyki wzajemnie się uzupełniają

Manifest zwinnego (agile) wytwarzania oprogramowania

„Odkrywamy nowe metody programowania dzięki praktyce w programowaniu i wspieraniu w nim innych. W wyniku naszej pracy, zaczęliśmy bardziej cenić:

- Ludzi i interakcje od procesów i narzędzi,
- Działające oprogramowanie od szczegółowej dokumentacji,
- Współpracę z klientem od negocjacji umów,
- Reagowanie na zmiany od realizacji założonego planu.

Oznacza to, że elementy wypisane po prawej są wartościowe, ale większą wartość mają dla nas te, które wypisano po lewej.”

Metodyki wytwarzania oprogramowania

- Metodyki zwinne
  - Przykłady: Scrum, eXtreme Programming
- Metodyki sformalizowane
  - Przykłady" RUP, OpenUP
- Opis metodyki
  - Proces techniczny – organizacja technik w spójny ciąg czynności
  - Notacja – sposób dokumentowania wszystkich decyzji (tekst, grafika)
  - Techniki – sposoby działania w poszczególnych dyscyplinach