Dr Katarzyna Grzesiak-Kopeć

Inżynieria oprogramowania

Warunki zaliczenia wykładu

Wykład z Inżynierii Oprogramowania kończy się **oceną**, która jest liczona jako ocena zaliczeniowa z przedmiotu i wchodzi w skład średniej ocen ze studiów.

Podstawą uzyskania zaliczenia z wykładu jest **zaliczenie testu** na zakończenie semestru.

Jako ocena końcowa zostanie przepisana ocena zaliczeniowa z ćwiczeń.

Plan wykładu

- Tworzenie oprogramowania
- Najlepsze praktyki IO
- Inżynieria wymagań
- Technologia obiektowa i język UML
- Techniki IO
- Metodyki zwinne
- Refaktoryzacja
- Mierzenie oprogramowania
- Jakość oprogramowania
- Programowanie strukturalne
- Modelowanie analityczne
- Wprowadzenie do testowania

Literatura

- R. S. Pressman, *Praktyczne podejście do inżynierii oprogramowania*, WNT 2004
- D. Hamlet & J. Maybee, *Podstawy techniczne inżynierii oprogramowania*, WNT 2003
- G. Booch, J. Rumbaugh, I. Jacobson, UML przewodnik użytkownika, WNT 2002
- M. Fowler & K. Scott, *UML w kropelce. Wersja 2.0*, LT&P 2005

1. Tworzenie oprogramowania

Plan wykładu

- Tworzenie oprogramowania
- Najlepsze praktyki IO
- Inżynieria wymagań
- Technologia obiektowa i język UML
- Techniki IO
- Metodyki zwinne
- Refaktoryzacja
- Mierzenie oprogramowania
- Jakość oprogramowania
- Programowanie strukturalne
- Modelowanie analityczne
- Wprowadzenie do testowania

Oprogramowanie jako produkt

- Raczej byt logiczny niż fizyczny
  - Wytwarzany a nie fizycznie konstruowane
  - Niewidoczny
- Nie zużywa się tak, jak byt fizyczny
- Ciągła zmiana wymagań (żyje!)
- Złożoność
  - Zarówno produkt, jak i proces

Oprogramowanie to produkt

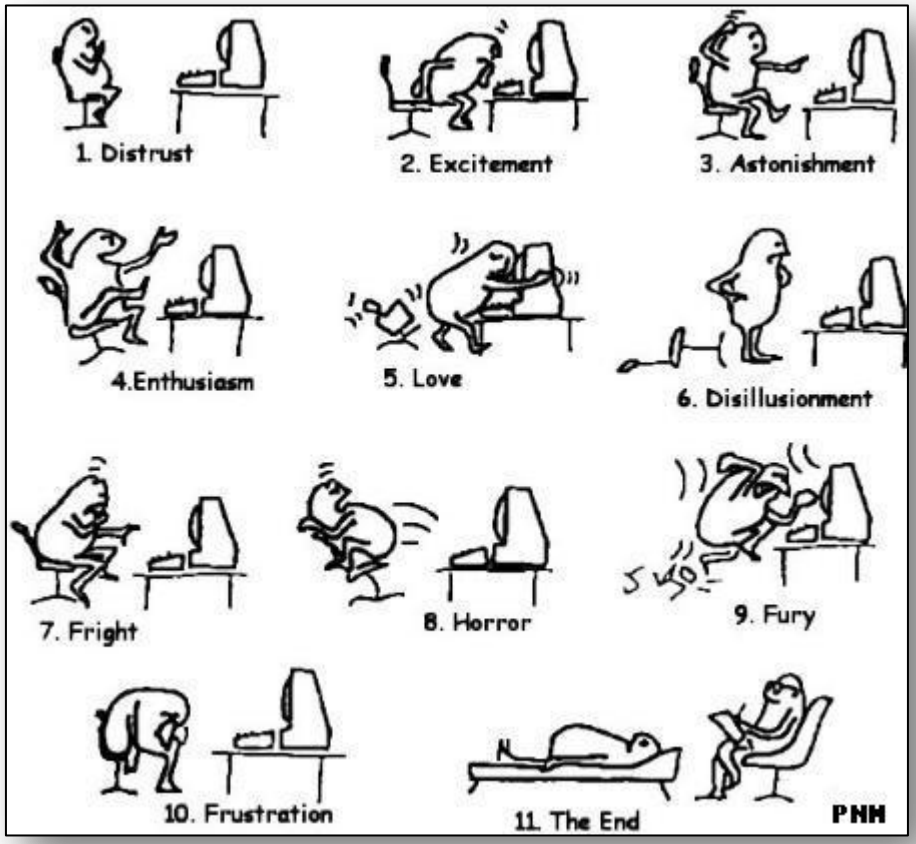
- Wytwarzany przez projektantów, programistów i innych ☺
- Składa się z:
  - programów wykonywanych na komputerach każdej wielkości i rodzaju
  - dokumentów elektronicznych i drukowanych
  - danych

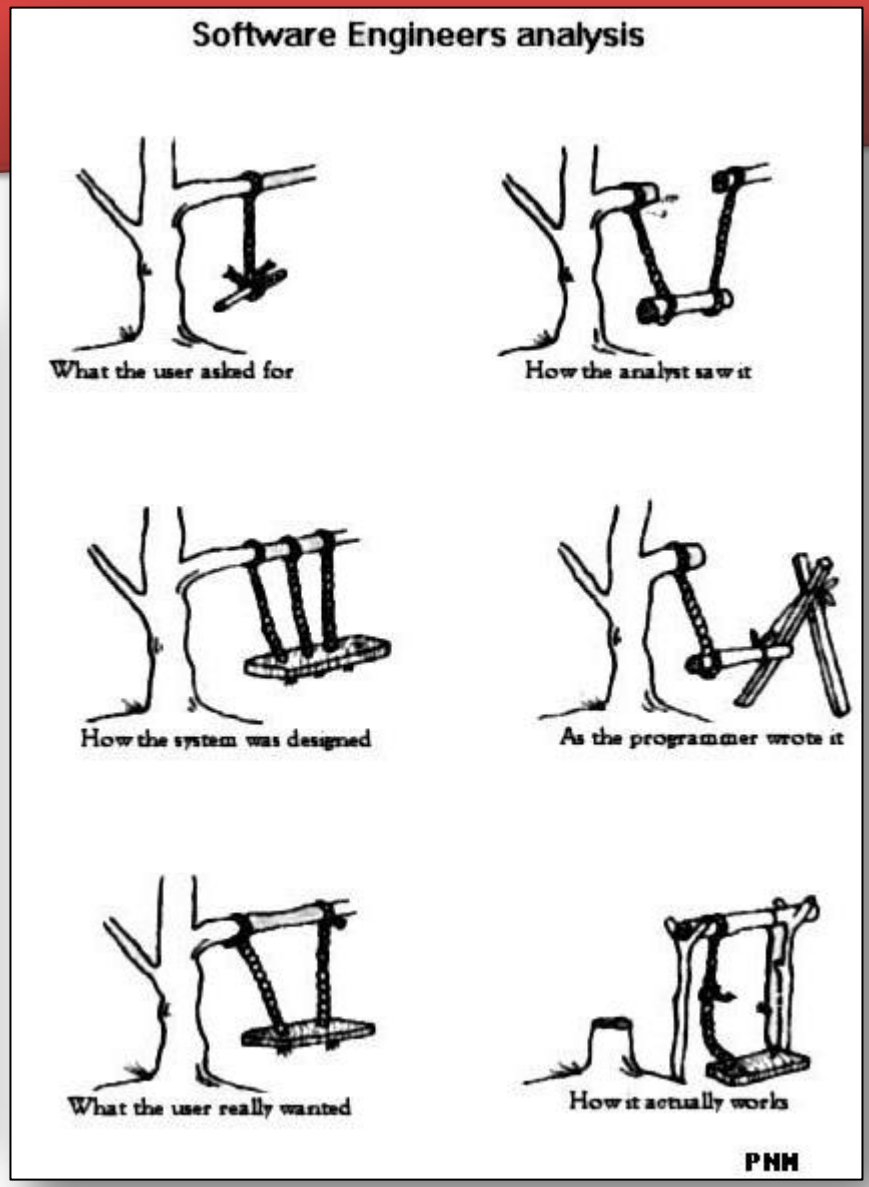
Inżynieria oprogramowania

1. Zastosowanie systematycznego, zdyscyplinowanego, poddającego się ocenie ilościowej podejścia do wytwarzania, stosowania i pielęgnacji oprogramowania, czyli wykorzystanie tradycyjnej inżynierii w informatyce.
2. Dziedzina wiedzy zajmująca się badaniem metod jak w (1).

IEEE

IO w praktyce ☺

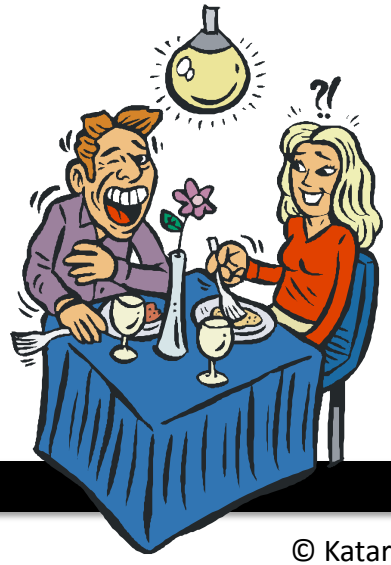


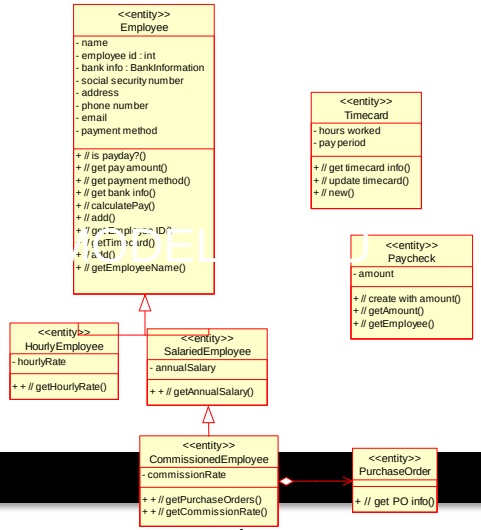


http://www.prashantmhatre.com/fun/software-engineering-classics/

Tworzenie oprogramowania

```
public Vector getPlanesFromPoints(Vector points, Vector lines){
  if(points == null) return null;

  planes = new Vector();
  pointToLines = new HashMap();
  lineToPoints = new HashMap();

  Point2D.Double p = null;
  Line line = null;
  Point2D.Double[] linePoints = null;
  boolean[] isOnLine = null;

  for(int i = 0; i < points.size(); i++){
    p = (Point2D.Double)points.get(i);
    isOnLine = new boolean[lines.size()];

    for(int j = 0; j < lines.size(); j++){
      linePoints = (Point2D.Double[])lines.get(j);
      line = new Line(linePoints[0], linePoints[1]);
      isOnLine[j] = line.isOnLine(p);
      if(isOnLine[j]){
        addPointToLine(p, linePoints);
      }//if on line
    }//for j

    pointToLines.put(p,isOnLine);
  }//for i
```





Co to jest MODEL?

Jest pewnym uproszczeniem rzeczywistości.

Dlaczego modelujemy?

- Lepiej zrozumieć system, który rozwijamy
- Pokazanie zależności między elementami
- Budujemy modele skomplikowanych systemów, ponieważ nie jesteśmy w stanie ogarnąć ich w całości

Dlaczego modelujemy?

- Realizuje 4 cele:
  - Pomaga w wizualizacji
  - Pozwala wyspecyfikować strukturę i zachowanie systemu
  - Daje pewien wzorzec wyznaczający nam kolejne kroki budowy aplikacji
  - Dokumentuje podjęte przez nas decyzje

Tworzenie oprogramowania

- Oprogramowanie może robić wszystko (prawie ☺)
- Jest sztuczne i logiczne
- Chyba najbardziej złożone dzieło człowieka i najbardziej złożony proces wytwarzania
- Młody i gwałtownie rozwijający się przemysł

Przeprowadzka

- Jedna z najbardziej stresujących sytuacji
- Syndrom pierwszych dni po przeprowadzce
  - Gdzie jest moja szczoteczka do zębów?
  - Dlaczego nie przyszedł wyciąg z konta?
  - Jak odebrać awizowaną przesyłkę?
- Jak uniknąć syndromu?
  - Przygotować szczegółowy plan działań
  - Wynająć profesjonalną firmę

Tworzenie oprogramowania

- Nowy projekt to przeprowadzka
  - Nowa dziedzina i nowe wymagania
  - Nowa technologia
  - Nowy klient i nowi współpracownicy
- Syndrom nowego projektu
  - Jak rozmawiać z klientem?
  - Jak opisać programistom architekturę?
  - Co i jak dokumentować?
  - Jak opisać interfejs użytkownika?
  - Jak zweryfikować system? ...

Przeprowadzka vs oprogramowanie

- Podobieństwa
  - Nie zbieramy doświadczeń – zawsze rewolucja
  - Nie potrafimy oszacować terminu zakończenia
  - Zakres prac drastycznie się zwiększa
  - Każda większa zmiana to katastrofa
  - Zorientujemy się w czym tkwił błąd – **KONIEC projektu!**

Przeprowadzka vs oprogramowanie

- Różnice
  - Złożoność problemu
  - Po przeprowadzce nie zaczynamy następnej (zazwyczaj)

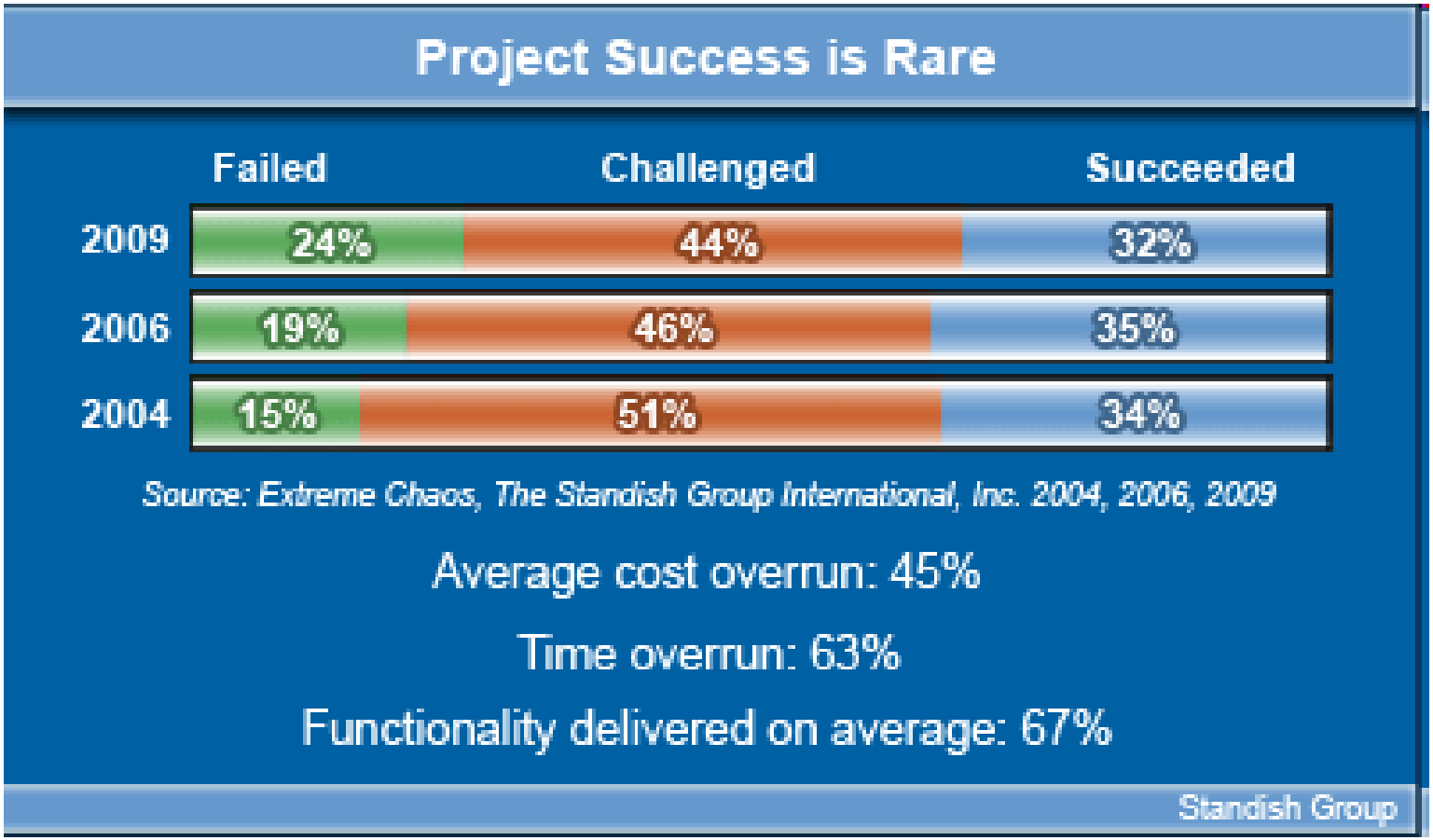
Powodzenie?

- Kiedy osiągnęliśmy sukces?
  - Zmieściliśmy się w zaplanowanym budżecie
  - Zmieściliśmy się w zaplanowanym czasie
  - Spełniliśmy rzeczywiste wymagania klienta
- To nie jest proste ☹

Sukces w liczbach

- Chaos Report – Standish Group
  - Rynek amerykański, 40000 projektów, 10 lat
  - 1994 – 16% zakończonych sukcesem
  - 2004 – 34% zakończonych sukcesem
- Nadal
  - 2004 – 52% wymagań nie zrealizowanych
  - 2004 – 82% przekroczony czas

2009 Chaos Report



2011 Dr Dobbs, How Successful are IT Projects, Really?

- Paradygmat IO, proces:
  - Ad hoc, iteracyjny, zwinny, wodospad, lean
- Ocena projektu
  - Sukces – dostarczony i spełnione kryteria sukcesu w zakresie akceptowalnym przez organizację
  - Zakwestionowany – dostarczony , ale nie spełniający wszystkich kryteriów
  - Porażka – niedostarczono projektu

2011 Dr Dobbs, Wyniki dla różnych procesów

- Iteracyjny: 69% sukces, 25% zakwestionowane
- Zwinny: 67% sukces, 27% zakwestionowane
- Wodospad: 50% sukces, 36% zakwestionowane
- Ad hoc: 49% sukces, 38% zakwestionowane
- Lean: 62% sukces, 30% zakwestionowane

2011 Dr Dobbs, Co jest sukcesem?

- Czas/harmonogram:
  - 20% zgodnie z harmonogramem
  - 26% wtedy, kiedy system jest gotowy
  - 51% oba czynniki równie ważne
- Return on investment (ROI)
  - 15% dostarczone w budżecie
  - 60% wysoki ROI
  - 25% oba czynniki równie ważne

2011 Dr Dobbs, Co jest sukcesem?

- Zadowolenie klienta:
  - 4% zgodnie ze specyfikacją
  - 80% rzeczywiste wymagania klienta
  - 16% oba czynniki równie ważne
- Jakość
  - 4% dostarczone w czasie i w budżecie
  - 57% rozwiązanie wysokiej jakości łatwe w utrzymaniu
  - 40% oba czynniki równie ważne

Uwaga na objawy

- Niezadowolony klient
  - Nie o taki system chodziło!
- Niezadowolony wykonawca
  - Czego oni od nas chcą?
- Kłótnie o zakres systemu
  - Żądacie, a tego nie było w umowie!
- Chaotyczna obsługa zmian
  - Tutaj wstawisz, tam przesuniesz....

Uwaga na objawy

- Niewyspani programiści
  - Pizza i cola pod drzwi
- Stres związany z końcem projektu
  - To nie tak miało być, zupełnie nie tak…
- Brak powtarzalności procesu
  - To o ile tym razem przekroczymy budżet?
- Marsz śmierci (Ed Yourdon)
  - Musicie zrobić to dwa razy szybciej niż konkurencja.

Problemy

- Nie spełnia oczekiwań klienta
- Zmiana wymagań
- Moduły nie są zintegrowane
- Trudności w utrzymaniu
- Późna identyfikacja „dziur”
- Kiepska jakość lub niezadowolenie klienta
- Brak koordynacji w zespole oprogramowania
- Trudności podczas wdrożenia

Zasad kilka ☺

- Kontrola
  - Złożone problemy
  - Uwaga – język nas zawodzi!
- Dziel i zwyciężaj
  - Rozwiązanie części nie zawsze gwarantuje rozwiązania całości
  - Niezależność części
  - Reguła 7 ± 2
  - Hierarchie
  - Wiedzieć, kiedy się zatrzymać

Zasad kilka ☺

- Od rozmycia do skupienia
  - Uwaga na abstrakcje – dysonans poznawczy
  - Ważny jest model
- Udokumentuj to
  - Plany testów
  - Własne założenia
  - Dokumentacja to nie powieść
- Wejście/wyjście jest podstawą oprogramowania

Zasad kilka ☺

- Nie przesadzaj z inżynierią
  - Lepsze jest wrogiem dobrego
  - Nie zaskakuj mile klienta
- Przygotuj się na zmiany
- Nie wynajduj koła
- Przyjmij odpowiedzialność

Na pomoc!

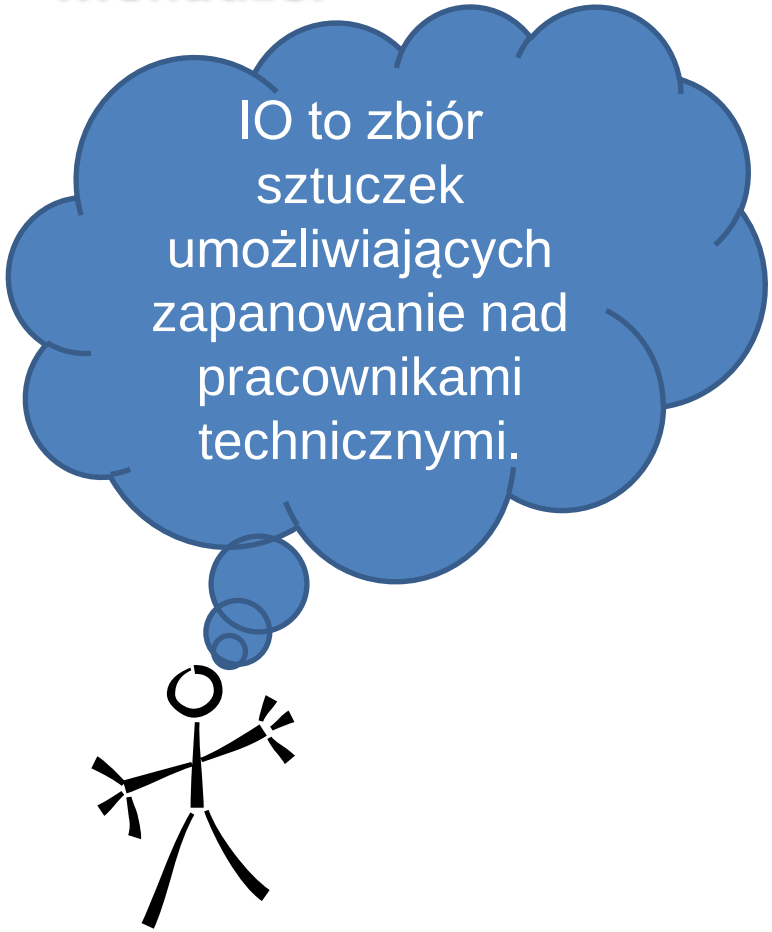
- Internet, literatura
- Potrzebny nam proces?
- Ocena procesu i standardy IO
- Najlepsze praktyki

Metodyka (proces)

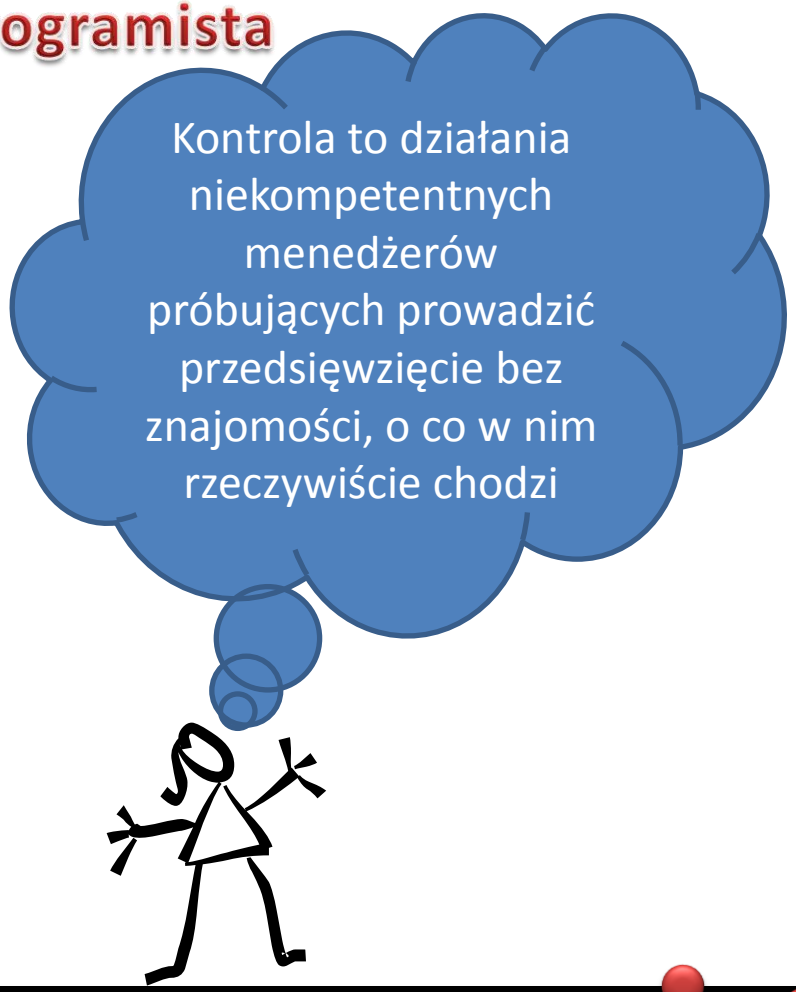
Definiuje KTO robi CO, KIEDY oraz JAK, żeby osiągnąć zamierzony cel.

Metodyka

Menadżer



Programista



Metodyka

© Scott Adams, Inc./Dist. by UFS, Inc.

Metodyka

- Można kontrolować
- Można mierzyć
- Można zmieniać w miarę poznawania
- Można poprawiać!
- Orientacja na PRODUKT

Cele wprowadzenia procesu

- Efektywność
  - Zrozumieć i wyprodukować to, czego oczekuje klient
  - Zweryfikować, czy o taki produkt chodziło
- Pielęgnacja
- Przewidywalność
- Powtarzalność
- Jakość
- Doskonalenie
- Śledzenie

Ocena procesu

- Model dojrzałości Capability Maturity Model
  - Software Engineering Institute (SEI)
  - Pięciostopniowa miara dojrzałości organizacji do produkcji oprogramowania
  - Ocena zakłada, że zawsze stosujemy te same procedury
  - Brak procedury oceny CMM – jest subiektywna
  - 2003 zmierzch modelu CMM

Standardy IO

- IEEE (The Institute of Electrical and Electronics Engineers)
- ISO (International Organization for Standardization)
- DoD (the US Department of Defense)

Recepty na dobry proces

- Wzorce procesów
- Modele procesów
  - technika wodospadu, model spiralny, model oparty na metodach formalnych, prototypowanie, Rapid Application Development (RAD), Component Based Development (CBD), Concurrent Development, Disciplined Software Development, Aspect-Oriented Software Development, Agile Process Models, The Rational Unified Process (RUP)…

KONIEC

CMM

- Ocena poziomu CMM mówi dużo o przedsiębiorstwie, a czynniki, które muszą być uwzględnione w celu osiągnięcia wyższych poziomów, odzwierciedlają ogólną wiarę w najlepszy sposób tworzenia oprogramowania.
- SW-CMM
  - Software Capability Maturity Model

Poziomy CMM

- Poziom 1 – początkowy
  - Brak procesu, działania ad-hoc, czasem chaotyczne; sukces zależy od wysiłku poszczególnych pracowników
- Poziom 2 – powtarzalny
  - Podstawowe procesy kontrolowania kosztów, harmonogramów i możliwości produktów; pozwalają na powtórzenie poprzednich sukcesów z podobnymi produktami

Poziomy CMM

- Poziom 3 – zdefiniowany [Poziom 2⊂ Poziom 3]
  - Ustalony, zintegrowany i udokumentowany schemat procesu wytwórczego stosowany we wszystkich projektach
- Poziom 4 – zarządzany [Poziom 3⊂ Poziom 4]
  - Pomiary kontrolujące proces i produkt
- Poziom 5 – optymalizowany [Poziom 4⊂ Poziom 5]
  - Proces jest stale udoskonalany dzięki nowym pomysłom i technikom oraz doświadczeniu

CMM – 18 cech kluczowych

- Poziom 2
  1) Zarządzanie konfiguracją
  2) Zapewnianie jakości
  3) Zarządzanie współpracą z podwykonawcami
  4) Śledzenie i nadzór nad projektem
  5) Planowanie projektu
  6) Zarządzanie wymaganiami

CMM – 18 cech kluczowych

- Poziom 3
  7) Wewnętrzne recenzje i przeglądy
  8) Zarządzanie współpracą między zespołami
  9) Zastosowanie metod IO
  10) Zintegrowane zarządzanie oprogramowaniem
  11) Program szkoleń
  12) Ustalenie schematu procesu
  13) Duże znaczenie procesu w działalności firmy

CMM – 18 cech kluczowych

- Poziom 4
  14) Zarządzanie jakością oprogramowania
  15) Zarządzanie procesami określone ilościowo
- Poziom 5
  16) Zarządzanie zmianami procesu
  17) Zarządzanie zmianami technologicznymi
  18) Zapobieganie błędom

Krytyka CMM

- Opis pełen pojęć mających na celu sprzedanie go menadżerom
- Wysiłki nie powinny koncentrować się na procesie kosztem produktu
- Koncepcje CMM muszą być stosowane w kolejności

Uwaga!

- 2003 zmierzch modelu CMM
  - Software Engineering Institute (SEI) przestał
    - certyfikować audytorów,
    - prowadzić kursy CMM,
    - doradzać przejście na któryś z modeli CMMI (Integration) (następca CMM)
- CMMI ???