Modele wytwarzania oprogramowania

Przemysław Wesołek Andrzej Jaszkiewicz

Instytut Informatyki
Politechnika Poznańska

1 marca 2004

Spis treści

Spis treści

Kontakt

- mgr inż. Przemysław Wesołek
- pok. nr 4, Polsko-Niemieckie Centrum Akademickie, Piotrowo
- tel. 665 26 28, w.15
- przemyslaw.wesolek@cs.put.poznan.pl
- www.cs.put.poznan.pl/pwesolek

Kontakt

- mgr inż. Przemysław Wesołek
- pok. nr 4, Polsko-Niemieckie Centrum Akademickie, Piotrowo
- tel. 665 26 28, w.15
- przemyslaw.wesolek@cs.put.poznan.pl
- www.cs.put.poznan.pl/pwesolek

Kontakt

- mgr inż. Przemysław Wesołek
- pok. nr 4, Polsko-Niemieckie Centrum Akademickie, Piotrowo
- tel. 665 26 28, w.15
- przemyslaw.wesolek@cs.put.poznan.pl
- www.cs.put.poznan.pl/pwesolek

Lata 50-te

- Sprzęt o bardzo ograniczonych możliwościach
- Ograniczone zastosowania (nauka)
- Małe programy
- Pisane dla siebie lub znajomych
- Dobrze wyspecyfikowane zadania

Lata 50-te

- Sprzęt o bardzo ograniczonych możliwościach
- Ograniczone zastosowania (nauka)
- Małe programy
- Pisane dla siebie lub znajomych
- Dobrze wyspecyfikowane zadania

Lata 50-te

- Sprzęt o bardzo ograniczonych możliwościach
- Ograniczone zastosowania (nauka)
- Małe programy
- Pisane dla siebie lub znajomych
- Dobrze wyspecyfikowane zadania

Lata 50-te

- Sprzęt o bardzo ograniczonych możliwościach
- Ograniczone zastosowania (nauka)
- Małe programy
- Pisane dla siebie lub znajomych
- Dobrze wyspecyfikowane zadania

Lata 50-te

- Sprzęt o bardzo ograniczonych możliwościach
- Ograniczone zastosowania (nauka)
- Małe programy
- Pisane dla siebie lub znajomych
- Dobrze wyspecyfikowane zadania

Lata 60-te

- Profesjonalni programiści
- Nowe języki programowania: COBOL, Fortran, Algol
- Sprzęt o dużo większych możliwościach
  np. pamięć wirtualna
- Nowe zastosowania (biznes)
- Próba realizacji wielu dużych
  przedsięwzięć programistycznych

Lata 60-te

- Profesjonalni programiści
- Nowe języki programowania: COBOL, Fortran, Algol
- Sprzęt o dużo większych możliwościach
  np. pamięć wirtualna
- Nowe zastosowania (biznes)
- Próba realizacji wielu dużych
  przedsięwzięć programistycznych

Lata 60-te

- Profesjonalni programiści
- Nowe języki programowania: COBOL, Fortran, Algol
- Sprzęt o dużo większych możliwościach
  np. pamięć wirtualna
- Nowe zastosowania (biznes)
- Próba realizacji wielu dużych
  przedsięwzięć programistycznych

Lata 60-te

- Profesjonalni programiści
- Nowe języki programowania: COBOL, Fortran, Algol
- Sprzęt o dużo większych możliwościach
  np. pamięć wirtualna
- Nowe zastosowania (biznes)
- Próba realizacji wielu dużych
  przedsięwzięć programistycznych

Lata 60-te

- Profesjonalni programiści
- Nowe języki programowania: COBOL, Fortran, Algol
- Sprzęt o dużo większych możliwościach
  np. pamięć wirtualna
- Nowe zastosowania (biznes)
- Próba realizacji wielu dużych
  przedsięwzięć programistycznych

Kryzys oprogramowania

- Rozwój technik wytwarzania oprogramowania nie nadąża za rozwojem sprzętu komputerowego
- Czy kryzys oprogramowania trwa do dzisiaj?
  - Większość przedsięwzięć przekracza czas lub budżet
  - Około 25% przedsięwzięć nie jest kończona
  - 90% firm przyznaje, że często zdarzają się im opóźnienia przedsięwzięć
  - Powszechna akceptacja kiepskiej jakości oprogramowania (w pewnych obszarach)

Kryzys oprogramowania

- Rozwój technik wytwarzania oprogramowania nie nadąża za rozwojem sprzętu komputerowego
- Czy kryzys oprogramowania trwa do dzisiaj?
  - Większość przedsięwzięć przekracza czas lub budżet
  - Około 25% przedsięwzięć nie jest kończona
  - 90% firm przyznaje, że często zdarzają się im opóźnienia przedsięwzięć
  - Powszechna akceptacja kiepskiej jakości oprogramowania (w pewnych obszarach)

Kryzys oprogramowania

- Rozwój technik wytwarzania oprogramowania nie nadąża za rozwojem sprzętu komputerowego
- Czy kryzys oprogramowania trwa do dzisiaj?
  - Większość przedsięwzięć przekracza czas lub budżet
  - Około 25% przedsięwzięć nie jest kończona
  - 90% firm przyznaje, że często zdarzają się im opóźnienia przedsięwzięć
  - Powszechna akceptacja kiepskiej jakości oprogramowania (w pewnych obszarach)

Kryzys oprogramowania

- Rozwój technik wytwarzania oprogramowania nie nadąża za rozwojem sprzętu komputerowego
- Czy kryzys oprogramowania trwa do dzisiaj?
  - Większość przedsięwzięć przekracza czas lub budżet
  - Około 25% przedsięwzięć nie jest kończona
  - 90% firm przyznaje, że często zdarzają się im opóźnienia przedsięwzięć
  - Powszechna akceptacja kiepskiej jakości oprogramowania (w pewnych obszarach)

Kryzys oprogramowania

- Rozwój technik wytwarzania oprogramowania nie nadąża za rozwojem sprzętu komputerowego
- Czy kryzys oprogramowania trwa do dzisiaj?
  - Większość przedsięwzięć przekracza czas lub budżet
  - Około 25% przedsięwzięć nie jest kończona
  - 90% firm przyznaje, że często zdarzają się im opóźnienia przedsięwzięć
  - Powszechna akceptacja kiepskiej jakości oprogramowania (w pewnych obszarach)

Kryzys oprogramowania: przyczyny

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

Kryzys oprogramowania: przyczyny

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

Kryzys oprogramowania: przyczyny

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

Kryzys oprogramowania: przyczyny

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

Kryzys oprogramowania: przyczyny

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

Kryzys oprogramowania: przyczyny

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

Kryzys oprogramowania: przyczyny

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

Kryzys oprogramowania: przyczyny

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

Inżynierii oprogramowania

Definicja

Wiedza techniczna dotycząca wszystkich **faz cyklu życia oprogramowania**, której celem jest uzyskanie wysokiej jakości **produktu – oprogramowania**.

Jakość oprogramowania

- Użyteczność
- Niezawodność
- Ergonomia
- Efektywność
- Łatwość konserwacji
- Bezpieczeństwo użytkownika

Jakość oprogramowania

- Użyteczność
- Niezawodność
- Ergonomia
- Efektywność
- Łatwość konserwacji
- Bezpieczeństwo użytkownika

Jakość oprogramowania

- Użyteczność
- Niezawodność
- Ergonomia
- Efektywność
- Łatwość konserwacji
- Bezpieczeństwo użytkownika

Jakość oprogramowania

- Użyteczność
- Niezawodność
- Ergonomia
- Efektywność
- Łatwość konserwacji
- Bezpieczeństwo użytkownika

Jakość oprogramowania

- Użyteczność
- Niezawodność
- Ergonomia
- Efektywność
- Łatwość konserwacji
- Bezpieczeństwo użytkownika

Jakość oprogramowania

- Użyteczność
- Niezawodność
- Ergonomia
- Efektywność
- Łatwość konserwacji
- Bezpieczeństwo użytkownika

Plan zajęć

- Wprowadzenie i podstawowe modele cyklu życia oprogramowania
- Analiza i modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Niezawodność oprogramowania
- Dokumentacja techniczna i użytkowa
- Narzędzia oprogramowania

Plan zajęć

- Wprowadzenie i podstawowe modele cyklu życia oprogramowania
- Analiza i modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Niezawodność oprogramowania
- Dokumentacja techniczna i użytkowa
- Narzędzia oprogramowania

Plan zajęć

- Wprowadzenie i podstawowe modele cyklu życia oprogramowania
- Analiza i modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Niezawodność oprogramowania
- Dokumentacja techniczna i użytkowa
- Narzędzia oprogramowania

Plan zajęć

- Wprowadzenie i podstawowe modele cyklu życia oprogramowania
- Analiza i modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Niezawodność oprogramowania
- Dokumentacja techniczna i użytkowa
- Narzędzia oprogramowania

Plan zajęć

- Wprowadzenie i podstawowe modele cyklu życia oprogramowania
- Analiza i modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Niezawodność oprogramowania
- Dokumentacja techniczna i użytkowa
- Narzędzia oprogramowania

Plan zajęć

- Wprowadzenie i podstawowe modele cyklu życia oprogramowania
- Analiza i modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Niezawodność oprogramowania
- Dokumentacja techniczna i użytkowa
- Narzędzia oprogramowania

Plan zajęć

- Wprowadzenie i podstawowe modele cyklu życia oprogramowania
- Analiza i modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Niezawodność oprogramowania
- Dokumentacja techniczna i użytkowa
- Narzędzia oprogramowania

Zaliczenie przedmiotu

Laboratoria

Seria kilku mniejszych ćwiczeń

Wykłady

Pisemne sprawdzenie zrozumienia zagadnień

Modele cyklu życia oprogramowania

- Uporządkowanie prac związanych z produkcją oprogramowania
- Ustalenie kolejności prac
- Planowanie i monitorowanie realizacji

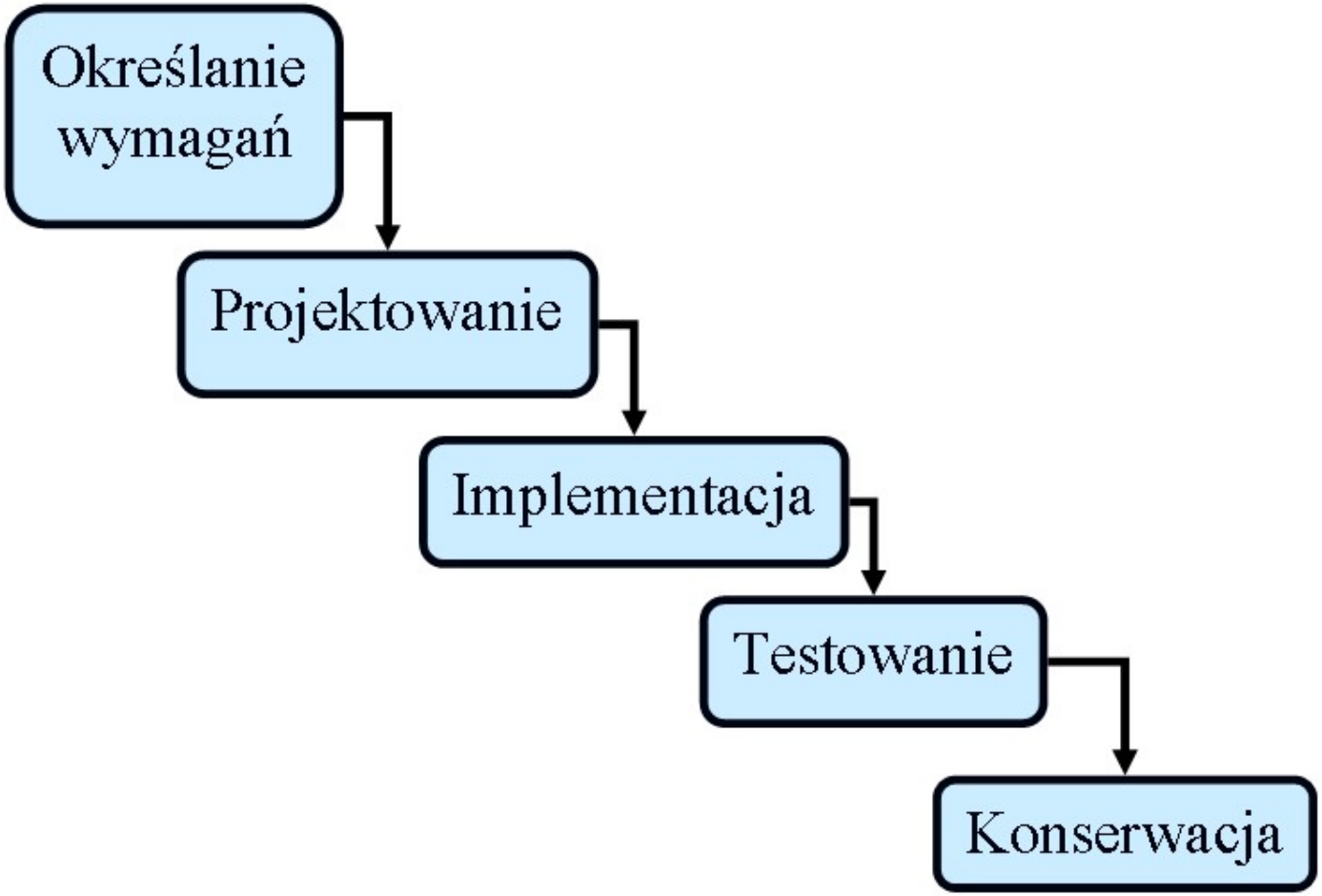
Modele cyklu życia oprogramowania

- Uporządkowanie prac związanych z produkcją oprogramowania
- Ustalenie kolejności prac
- Planowanie i monitorowanie realizacji

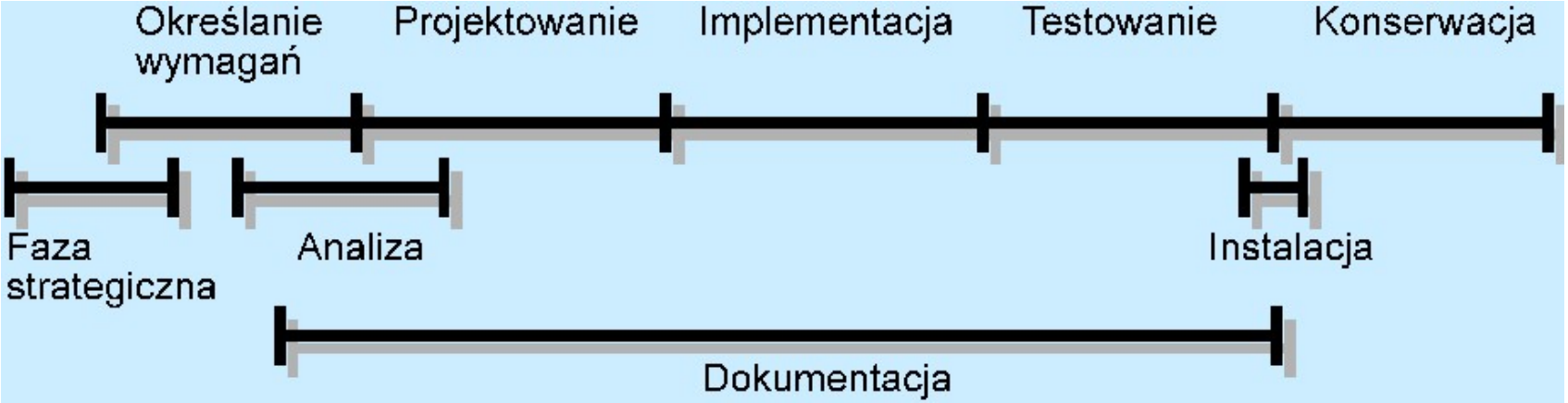
Modele cyklu życia oprogramowania

- Uporządkowanie prac związanych z produkcją oprogramowania
- Ustalenie kolejności prac
- Planowanie i monitorowanie realizacji

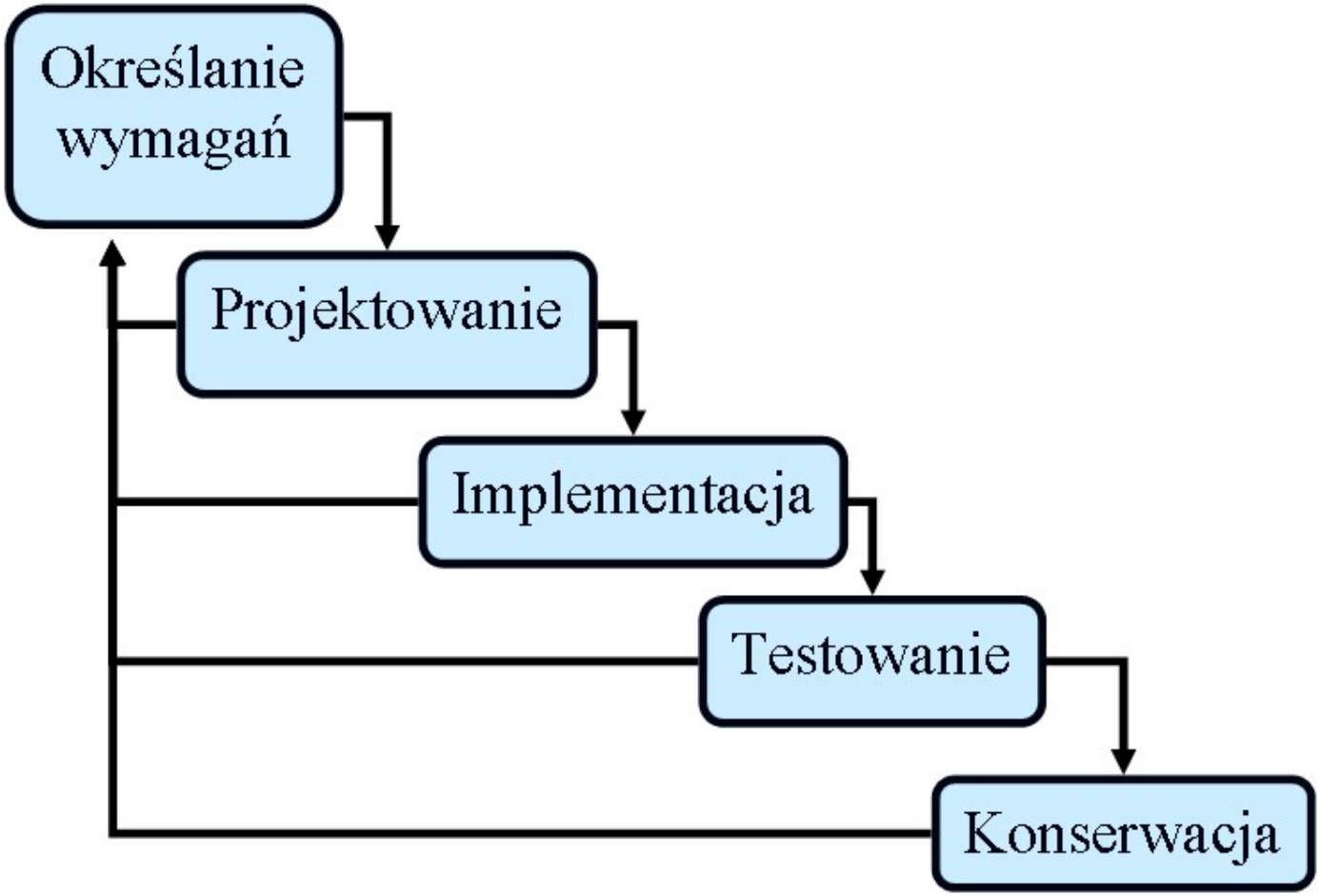
Model kaskadowy



Dodatkowe fazy w modelu kaskadowym



Ścisłe a elastyczne rozumienie modelu kaskadowego



Model kaskadowy: wady i zalety

Zalety

- Łatwość zarządzania – planowanie i monitorowanie

Wady

- Wysoki koszt błędów popełnionych we wstępnych fazach
- Długa przerwa w kontaktach z klientami
- Nie lubiany przez wykonawców (duży rygor)

Model kaskadowy: wady i zalety

Zalety

- Łatwość zarządzania – planowanie i monitorowanie

Wady

- Wysoki koszt błędów popełnionych we wstępnych fazach
- Długa przerwa w kontaktach z klientami
- Nie lubiany przez wykonawców (duży rygor)

Model kaskadowy: wady i zalety

Zalety

- Łatwość zarządzania – planowanie i monitorowanie

Wady

- Wysoki koszt błędów popełnionych we wstępnych fazach
- Długa przerwa w kontaktach z klientami
- Nie lubiany przez wykonawców (duży rygor)

Model kaskadowy: wady i zalety

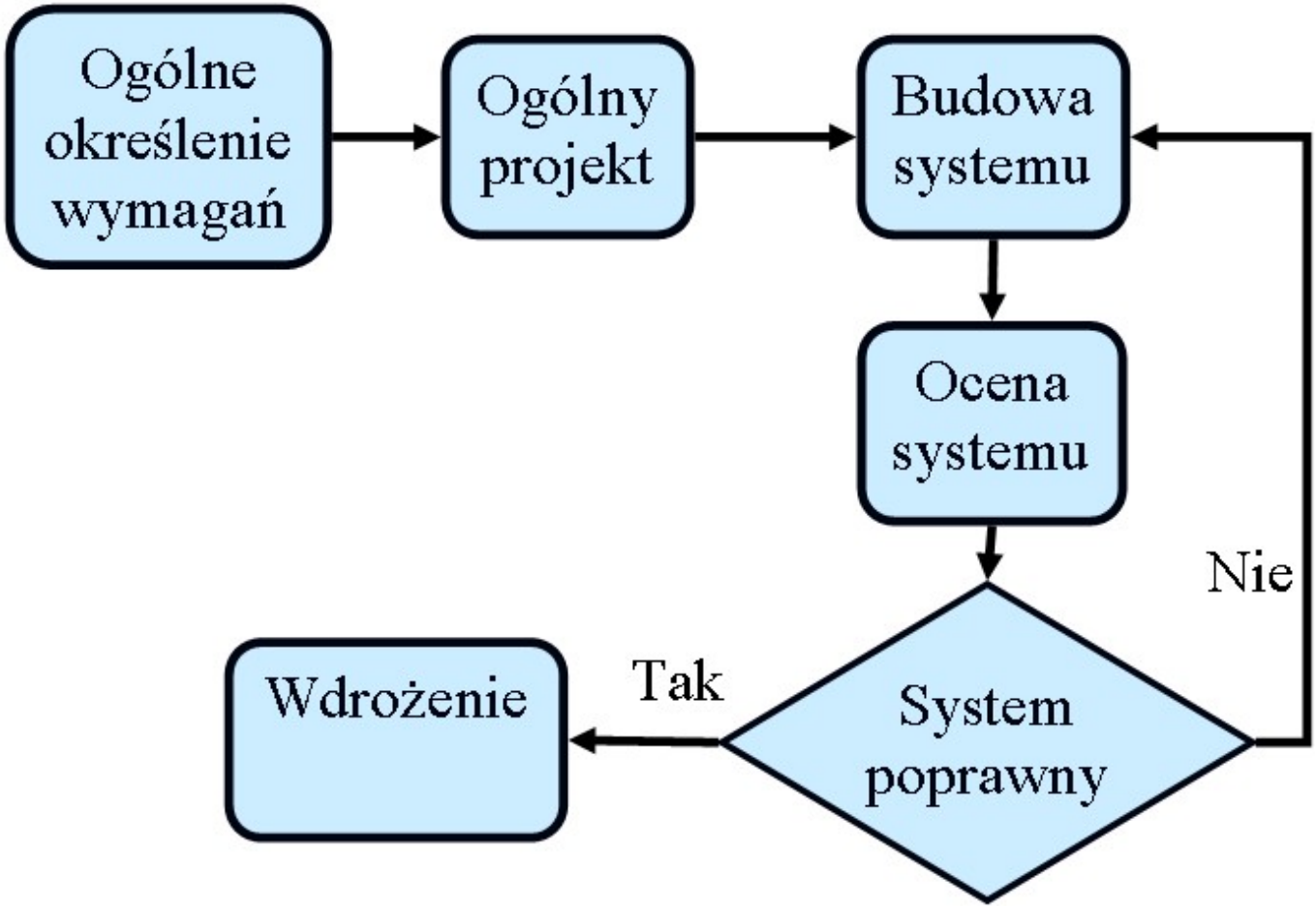
Zalety

- Łatwość zarządzania – planowanie i monitorowanie

Wady

- Wysoki koszt błędów popełnionych we wstępnych fazach
- Długa przerwa w kontaktach z klientami
- Nie lubiany przez wykonawców (duży rygor)

Programowanie odkrywcze



Programowanie odkrywcze: wady i zalety

Zalety

- Możliwość stosowania przy trudnych do określenia wymaganiach klienta

Wady

- Bardzo trudne zachowanie sensownej struktury systemu
- Konieczność testowania wyłącznie z klientem
- Uznawany za „amatorski” sposób tworzenia oprogramowania. Praktycznie stosowany tylko przy tworzeniu prototypu

Programowanie odkrywcze: wady i zalety

Zalety

- Możliwość stosowania przy trudnych do określenia wymaganiach klienta

Wady

- Bardzo trudne zachowanie sensownej struktury systemu
- Konieczność testowania wyłącznie z klientem
- Uznawany za „amatorski” sposób tworzenia oprogramowania. Praktycznie stosowany tylko przy tworzeniu prototypu

Programowanie odkrywcze: wady i zalety

Zalety

- Możliwość stosowania przy trudnych do określenia wymaganiach klienta

Wady

- Bardzo trudne zachowanie sensownej struktury systemu
- Konieczność testowania wyłącznie z klientem
- Uznawany za „amatorski” sposób tworzenia oprogramowania. Praktycznie stosowany tylko przy tworzeniu prototypu

Programowanie odkrywcze: wady i zalety

Zalety

- Możliwość stosowania przy trudnych do określenia wymaganiach klienta

Wady

- Bardzo trudne zachowanie sensownej struktury systemu
- Konieczność testowania wyłącznie z klientem
- Uznawany za „amatorski” sposób tworzenia oprogramowania. Praktycznie stosowany tylko przy tworzeniu prototypu

Prototypowanie

- Cel: lepsze określenie wymagań
- Fazy prototypowania
  - Ogólne określenie wymagań
  - Budowa prototypu
  - Weryfikacja prototypu przez klienta
  - Pełne określenie wymagań
  - Realizacja pełnego systemu zgodnie z modelem kaskadowym

Prototypowanie

- Cel: lepsze określenie wymagań
- Fazy prototypowania
  - Ogólne określenie wymagań
  - Budowa prototypu
  - Weryfikacja prototypu przez klienta
  - Pełne określenie wymagań
  - Realizacja pełnego systemu zgodnie z modelem kaskadowym

Prototypowanie

- Cel: lepsze określenie wymagań
- Fazy prototypowania
  - Ogólne określenie wymagań
  - Budowa prototypu
  - Weryfikacja prototypu przez klienta
  - Pełne określenie wymagań
  - Realizacja pełnego systemu zgodnie z modelem kaskadowym

Prototypowanie

- Cel: lepsze określenie wymagań
- Fazy prototypowania
  - Ogólne określenie wymagań
  - Budowa prototypu
  - Weryfikacja prototypu przez klienta
  - Pełne określenie wymagań
  - Realizacja pełnego systemu zgodnie z modelem kaskadowym

Prototypowanie

- Cel: lepsze określenie wymagań
- Fazy prototypowania
  - Ogólne określenie wymagań
  - Budowa prototypu
  - Weryfikacja prototypu przez klienta
  - Pełne określenie wymagań
  - Realizacja pełnego systemu zgodnie z modelem kaskadowym

Prototypowanie

- Cel: lepsze określenie wymagań
- Fazy prototypowania
  - Ogólne określenie wymagań
  - Budowa prototypu
  - Weryfikacja prototypu przez klienta
  - Pełne określenie wymagań
  - Realizacja pełnego systemu zgodnie z modelem kaskadowym

Prototypowanie: metody tworzenia

Konieczność zbudowania prototypu szybko i niskim kosztem.
Po realizacji prototypu powinien on zostać wyrzucony!

- Niepełna realizacja
- Języki wysokiego poziomu
- Korzystanie z gotowych komponentów
- Generowane interfejsy użytkownika
- Szybkie programowanie (ang. quick-and-dirty)
- Papier
- Programowanie odkrywcze

Prototypowanie: metody tworzenia

Konieczność zbudowania prototypu szybko i niskim kosztem.
Po realizacji prototypu powinien on zostać wyrzucony!

- Niepełna realizacja
- Języki wysokiego poziomu
- Korzystanie z gotowych komponentów
- Generowane interfejsy użytkownika
- Szybkie programowanie (ang. quick-and-dirty)
- Papier
- Programowanie odkrywcze

Prototypowanie: metody tworzenia

Konieczność zbudowania prototypu szybko i niskim kosztem.
Po realizacji prototypu powinien on zostać wyrzucony!

- Niepełna realizacja
- Języki wysokiego poziomu
- Korzystanie z gotowych komponentów
- Generowane interfejsy użytkownika
- Szybkie programowanie (ang. quick-and-dirty)
- Papier
- Programowanie odkrywcze

Prototypowanie: metody tworzenia

Konieczność zbudowania prototypu szybko i niskim kosztem.
Po realizacji prototypu powinien on zostać wyrzucony!

- Niepełna realizacja
- Języki wysokiego poziomu
- Korzystanie z gotowych komponentów
- Generowane interfejsy użytkownika
- Szybkie programowanie (ang. quick-and-dirty)
- Papier
- Programowanie odkrywcze

Prototypowanie: metody tworzenia

Konieczność zbudowania prototypu szybko i niskim kosztem.
Po realizacji prototypu powinien on zostać wyrzucony!

- Niepełna realizacja
- Języki wysokiego poziomu
- Korzystanie z gotowych komponentów
- Generowane interfejsy użytkownika
- Szybkie programowanie (ang. quick-and-dirty)
- Papier
- Programowanie odkrywcze

Prototypowanie: metody tworzenia

Konieczność zbudowania prototypu szybko i niskim kosztem.
Po realizacji prototypu powinien on zostać wyrzucony!

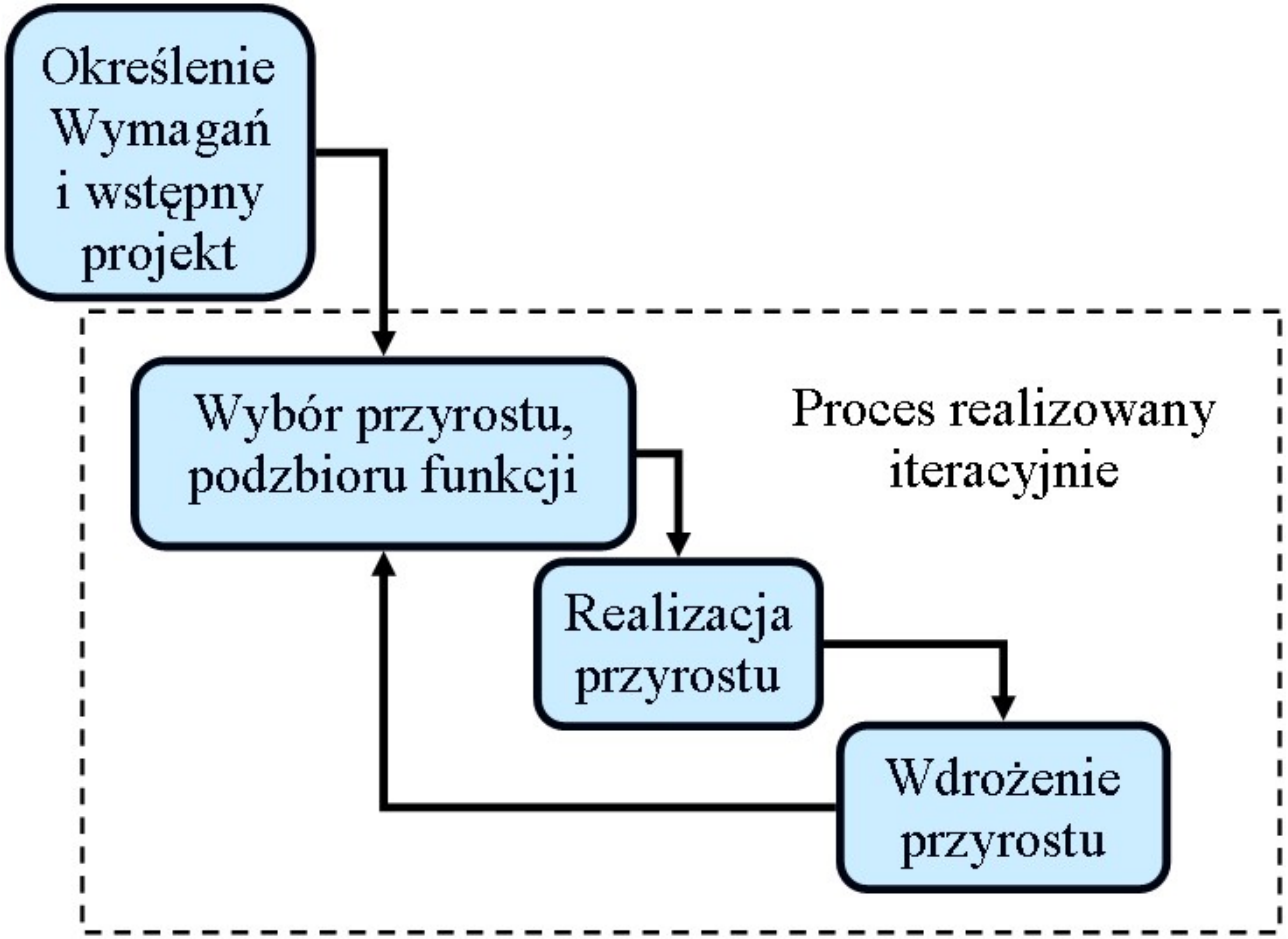
- Niepełna realizacja
- Języki wysokiego poziomu
- Korzystanie z gotowych komponentów
- Generowane interfejsy użytkownika
- Szybkie programowanie (ang. quick-and-dirty)
- Papier
- Programowanie odkrywcze

Prototypowanie: metody tworzenia

Konieczność zbudowania prototypu szybko i niskim kosztem.
Po realizacji prototypu powinien on zostać wyrzucony!

- Niepełna realizacja
- Języki wysokiego poziomu
- Korzystanie z gotowych komponentów
- Generowane interfejsy użytkownika
- Szybkie programowanie (ang. quick-and-dirty)
- Papier
- Programowanie odkrywcze

Realizacja przyrostowa



Realizacja przyrostowa: wady i zalety

Zalety

- Skrócenie przerw w kontaktach z klientami
- Możliwość wczesnego korzystania z części funkcji systemu
- Możliwość elastycznego reagowania na powstałe opóźnienia

Wady

- Dodatkowy koszt związany z niezależną realizacją fragmentów systemu

Realizacja przyrostowa: wady i zalety

Zalety

- Skrócenie przerw w kontaktach z klientami
- Możliwość wczesnego korzystania z części funkcji systemu
- Możliwość elastycznego reagowania na powstałe opóźnienia

Wady

- Dodatkowy koszt związany z niezależną realizacją fragmentów systemu

Realizacja przyrostowa: wady i zalety

Zalety

- Skrócenie przerw w kontaktach z klientami
- Możliwość wczesnego korzystania z części funkcji systemu
- Możliwość elastycznego reagowania na powstałe opóźnienia

Wady

- Dodatkowy koszt związany z niezależną realizacją fragmentów systemu

Realizacja przyrostowa: wady i zalety

Zalety

- Skrócenie przerw w kontaktach z klientami
- Możliwość wczesnego korzystania z części funkcji systemu
- Możliwość elastycznego reagowania na powstałe opóźnienia

Wady

- Dodatkowy koszt związany z niezależną realizacją fragmentów systemu

Literatura

A. Jaszkiewicz, *Inżynieria oprogramowania*, Helion, Gliwice, 1997.
F.P. Brooks, *Mityczny osobomiesiąc*, WNT, Warszawa, 2000.

Za tydzień...

Analiza i modelowanie systemów z wykorzystaniem języka UML