Inżynieria oprogramowania (IO)

Wykłady:
mgr inż. Sławomir Wróblewski

Godziny przyjęć:
wtorki 10-11, środy 15-16
pokój nr 19 (6 piętro)
Katedra Mikroelektroniki
i Technik informatycznych
Politechniki Łódzkiej,
al. Politechniki 11
Łódź

Kontakt:
Telefon: (42) 631-26-53
E-mail: swroble@dmcs.p.lodz.pl

HTTP:
Materiały umieszczono na stronie:
lux.dmcs.p.lodz.pl

Cele wykładu, laboratorium

Celem wykładu jest przedstawienie wybranych aspektów tworzenia oprogramowania od początkowej fazy specyfikacji systemu aż do jego pielęgnacji po dacie rozpoczęcia jego użytkowania.

Celem laboratorium jest zapoznanie się z językiem UML służącym do zapisywania projektu systemu.

Literatura obowiązkowa:

[1] Ian Sommerville: Inżynieria Oprogramowania, WNT 2003

[2] G.,Rumbaugh J., Jacobson I.: UML podręcznik użytkownika, WNT 2001

Plan wykładu IO

Inżynieria oprogramowania:

- Wprowadzenie
- Wymagania stawiane oprogramowaniu
- Projektowanie oprogramowania
- Weryfikacja i zatwierdzanie oprogramowania
- Ewolucja oprogramowania

Plan wykładu UML

Wprowadzenie do języka UML (ang. Unified Modeling Language) Ujednolicony Język Modelowania

- Diagramy języka UML:
  - klas, obiektów
  - stanów
  - inne
- Klasy, obiekty – programowanie obiektowe
- Modelowanie artefaktów systemu

Słowo wstępne

- Gospodarki wszystkich rozwiniętych krajów zależą od oprogramowania.
- Obecnie wytwarzanie oprogramowania jest poważną gałęzią gospodarki narodowej rozwiniętego kraju.
- Coraz więcej i więcej systemów wymaga niezawodnego oprogramowania.
- Wytwarzanie oprogramowania nie jest prostym zagadnieniem (system informatyczny dla ZUS ;)).
- Wraz ze wzrostem złożoności oprogramowania pojawia się większa liczba błędów.
- Wyłoniła się dziedzina wiedzy nazwana inżynierią oprogramowania.

Kryzys oprogramowania, narodziny IO

lata	programy	błędy	Inżynieria oprogramowania
1950-1960	programy pisane dla siebie	niewielkie błędy	nie istnieje
1960-1970	duże systemy	kosztowne błędy początki kryzysu	rodzi się
1970-1990	olbrzymie systemy oraz komputery dla mas	wyniszczające błędy uznanie błędów za rzecz zwykłą	inżynieria oprogramowania w rozkwicie
1990-2004	nic już nie możemy bez komputerów	kryzys oprogramowania w rozkwicie	inżynieria oprogramowania nadal kwitnie
Przyszłość	błędnie zaprogramowane komputery przeprogramowują ludzi do własnych potrzeb ;)	trudności z błędami w ludziach	powstanie inżynieria oprogramowania genetycznego ludzi

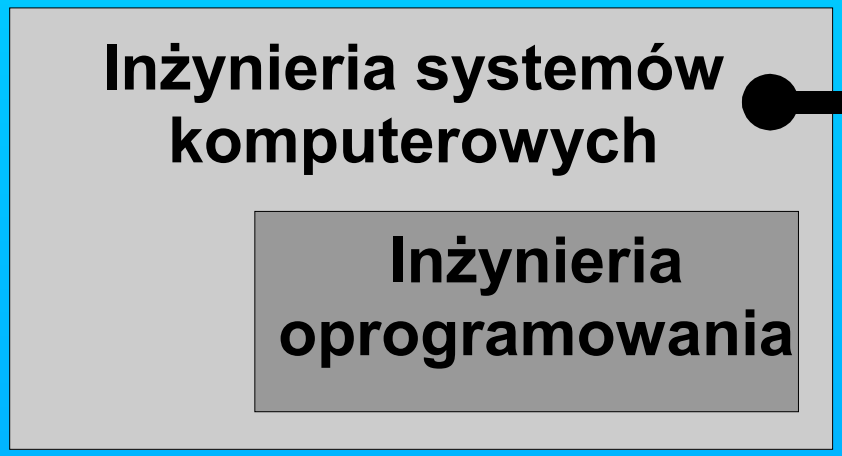
Co to jest nżynieria oprogramowania?

Dziedzina inżynierii obejmująca wszystkie aspekty tworzenia oprogramowania:

- techniczny proces tworzenia oprogramowania
- zarządzanie przedsięwzięciem programistycznym
- wypracowanie standardów jakości porównywalnych do obowiązujących w innych dziedzinach inżynierii
- wypracowanie procedur postępowania sprzyjających wysokiej jakości

Produktem inżynierii oprogramowania jest oprogramowanie.

Inżynieria oprogramowania a inżynieria systemów komputerowych



Inżynieria systemów komputerowych obejmuje wszystkie aspekty tworzenia i ewolucji systemów komputerowych, w których oprogramowanie odgrywa główną rolę.

Inżynierowie systemów biorą udział w specyfikacji systemu i definiowaniu jego architektury.

Inżynieria oprogramowania a informatyka

Zasadniczo informatyka obejmuje teorie i podstawowe zasady działania komputerów. Inżynieria oprogramowania obejmuje praktyczne problemy związane z tworzeniem oprogramowania.

Byłoby idealnie, gdyby inżynier oprogramowania znał teorie informatyczne, jednakże nie zawsze teorie można zastosować do rzeczywistych złożonych zadań.

Wyzwania inżynierii oprogramowania

Wyzwanie dziedzictwa

Pielęgnacja i modyfikacja działających starych systemów, pełniących poważne funkcje gospodarcze.

Wyzwanie różnorodności

Wymóg działania oprogramowania w systemach rozproszonych przy rożnych typach komputerów i systemów wspomagających. Elastyczność oprogramowania.

Wyzwanie doręczenia

Wymóg dostarczania gotowego oprogramowania w szybkim czasie bez utraty jakości. Odpowiedź na gwałtowne reakcje gospodarki na zmiany rynku i jej szybką ewolucję.

Odpowiedzialność etyczna i zawodowa

- **Zachowywanie tajemnicy**

  Inżynierowie powinni zawsze dochowywać tajemnic powierzonych przez pracodawców i klientów, niezależnie od tego czy podpisano formalną umowę o ochronie tajemnicy.

- **Kompetencje**

  Inżynierowie nie powinni zawyżać poziomu swoich kompetencji. Nie powinni świadomie przyjmować prac, które przekraczają ich możliwości.

Odpowiedzialność etyczna i zawodowa

- **Prawo własności intelektualnej**

  Inżynierowie powinni znać miejscowe prawo regulujące korzystanie z własności intelektualnej jak patenty, prawa autorskie itd. Powinni szczególnie dbać o poszanowanie intelektualnej własności swoich pracodawców i klientów.

- **Niewłaściwe użycie komputera**

  Inżynierowie oprogramowania nie powinni używać swoich umiejętności do niewłaściwego używania cudzych komputerów. Niewłaściwe użycie może być dość banalne (np. granie na maszynie pracodawcy) lub skrajnie poważne (rozsiewanie wirusów).

Narzędzia CASE

CASE – Computer-Aided Software Engineering (Inżynieria oprogramowania wspomagana komputerowo) obejmuje rozmaite programy wykorzystywane do wspomagania czynności procesu tworzenia oprogramowania.

- ***upper-CASE*** – początkowa faza: edytory notacji używanej w metodzie, weryfikacja poprawności modelu, generatory raportów itp.
- ***lower-CASE*** – końcowa faza: implementacja, wyszukiwanie błędów, testowanie.

“Dziecko” IO: oprogramowanie

To nie tylko programy ale także jego dokumentacja (systemowa, użytkownika), dane konfiguracyjne niezbędne do poprawnego działania systemu. Oprogramowanie jest produktem, który można sprzedać klientowi. Można wyróżnić 2 kategorie oprogramowania:

- oprogramowanie powszechne
- oprogramowanie pisane na zamówienie

Cechy oprogramowania jako produktu

- **Zgodne z wymaganiami klienta**
- **Niezawodne**

  Nie powinno powodować fizycznych lub ekonomicznych katastrof w przypadku awarii.
- **Efektywne**

  Nie powinno marnotrawić zasobów systemu takich jak pamięć czy czas procesora.
- **Łatwe w pielęgnacji**

  Zdolność do ewolucji zgodnie z potrzebami klientów.
- **Ergonomiczne**

  Powinno być użyteczne, bez zbędnego wysiłku ze strony użytkownika (np. interfejsy).

Proces tworzenia oprogramowania

Czynności zmierzające do wyprodukowania systemu. Istnieje wiele różnych sposobów (procesów) tworzenia oprogramowania, posiadają one jednak wspólne czynności.

Proces tworzenia oprogramowania

- **Specyfikacja oprogramowania**
  Gromadzenie wymogów definiujących, co system powinien robić.
  Analiza pozwalająca zrozumieć wymogi.
- **Projektowanie i implementowanie oprogramowania**
  Ustalenie, w jaki sposób system będzie spełniał narzucone wymogi.
  Budowanie systemu.
- **Zatwierdzanie oprogramowania**
  Testowanie, weryfikacja, czy system spełnia wymogi.
- **Wdrożenie**
  Udostepnienie systemu użytkownikom.
- **Ewolucja oprogramowania**
  Rozwój oprogramowania. Dostosowanie do zmieniających się wymagań klienta

Modele procesów tworzenia oprogramowania

To uproszczona prezentacja procesu tworzenia oprogramowania. Jest to abstrakcja konkretnego procesu, który ma być opisany. Model może zawierać:

- czynności składające się na proces
- produkty programowe
- role osób biorących udział w tworzeniu oprogramowania
- inne

Modele procesów tworzenia oprogramowania

Istnieje kilka ogólnych modeli (paradygmatów) tworzenia oprogramowania:

- Model kaskadowy

  W tym modelu podstawowe czynności specyfikowania, tworzenia, zatwierdzania i ewolucji są odrębnymi fazami procesu.
- Tworzenie ewolucyjne

W tym procesie czynności specyfikowania, projektowania i zatwierdzania przeplatają się.

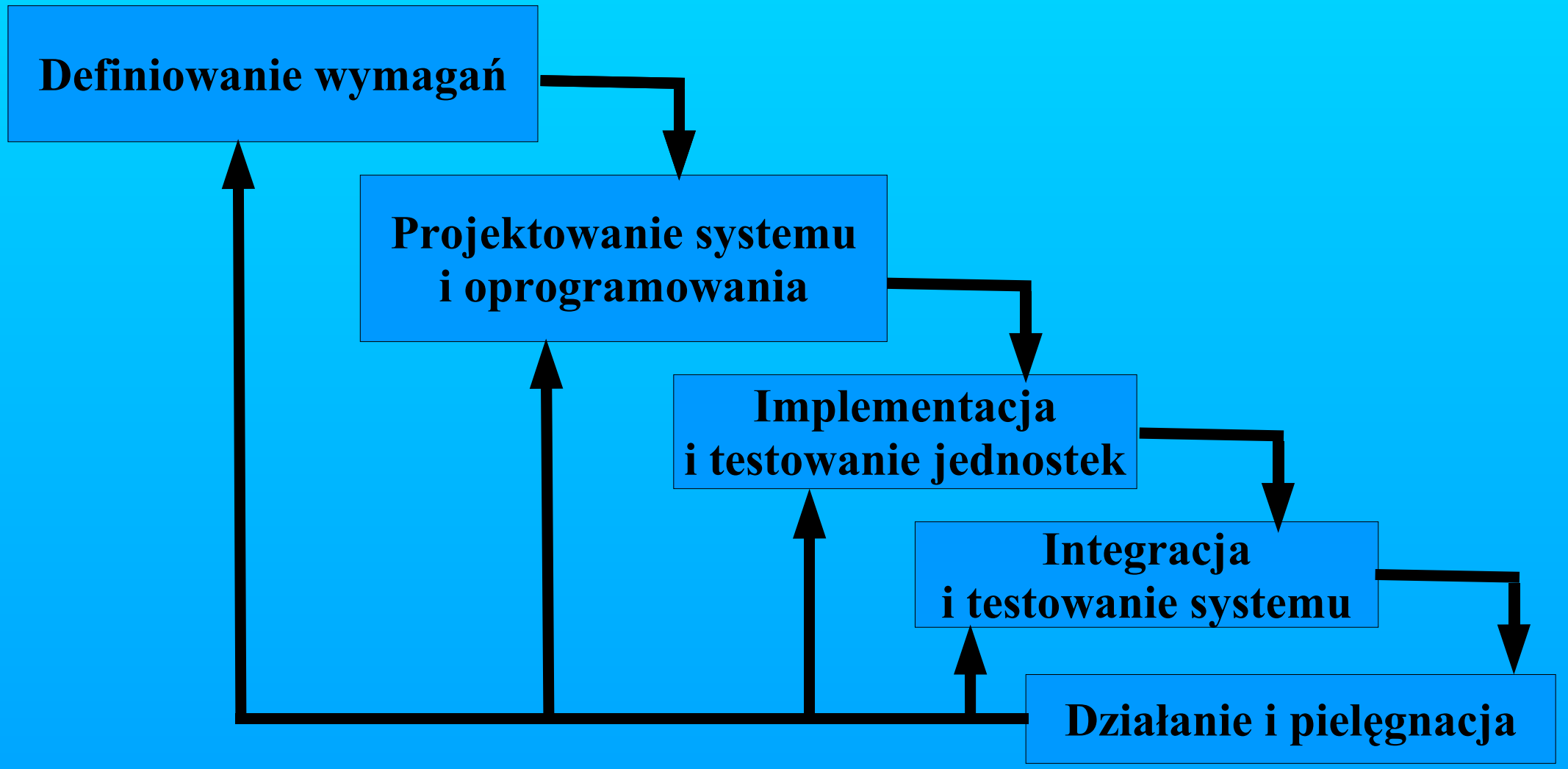
Modele procesów tworzenia oprogramowania

- Formalne przekształcenia

  To podejście jest oparte na budowaniu formalnych matematycznych specyfikacji systemu i przekształcaniu tych specyfikacji w program za pomocą metod matematycznych.
- Składanie systemu z komponentów ponownego użycia

  W tym podejściu zakłada się istnienie dużej liczby komponentów zdatnych do ponownego użycia.

Model kaskadowy



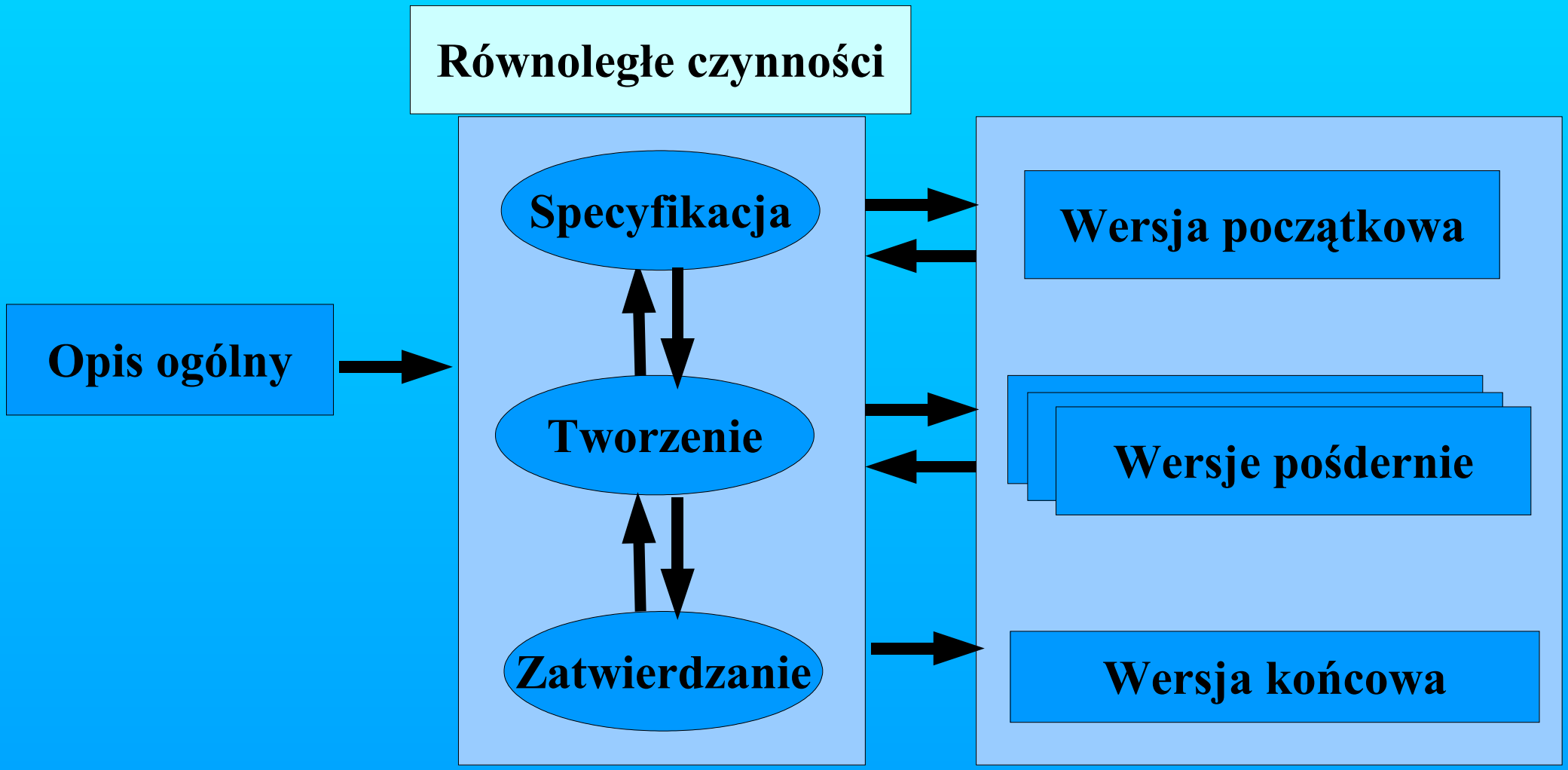
Wady modelu kaskadowego

- Następnej fazy nie powinno się rozpoczynać, jeśli poprzednia się nie zakończy.
- Koszty opracowania i akceptacji dokumentów są wysokie i dlatego iteracje są również kosztowne oraz wymagają powtarzania wielu prac.
- Nieelastyczny podział na rozłączne etapy.
- Model kaskadowy powinien być używany jedynie wówczas, gdy wymagania są jasne i zrozumiałe.

Stosowanie modelu kaskadowego

- Systemy duże (powyżej 500 000 wierszy kodu)

Tworzenie ewolucyjne



Typy tworzenia ewolucyjnego

1. **Tworzenie badawcze.** Celem procesu jest praca z klientem, polegająca na badaniu wymagań i dostarczeniu ostatecznego systemu. Tworzenie rozpoczyna się od tych części systemu, które są dobrze rozpoznane. System ewoluuje przez dodawanie nowych cech, które proponuje klient.
2. **Prototypowanie z porzuceniem.** Celem procesu tworzenia ewolucyjnego jest zrozumienie wymagań klienta i wypracowanie lepszej definicji wymagań stawianych systemowi. Budowanie prototypu ma głównie na celu eksperymentowanie z tymi wymaganiami użytkownika, które są niejasne.

Wady modelu ewolucyjnego

- Proces nie jest widoczny. Menedżerowie potrzebują regularnych wyników, aby mierzyć postę prac.Jeśli proces tworzony jest szybko, nie opłaca się generować dokumentów opisujących każdą wersję systemu.
- System ma złą strukturę. Ciągłe modyfikacje przyczyniają się do psucia struktury oprogramowania. Wprowadzenie nowych zmian staje się coraz trudniejsze i bardziej kosztowne.
- Konieczne mogą być specjalne narzędzia i techniki. Ułatwiają one szybkie tworzenie, ale mogą być niekompatybilne z innymi narzędziami i technikami.

Stosowanie modelu ewolucyjnego

- Systemy małe (mniej niż 100 000 wierszy kodu)
- Systemy średnie (do 500 000 wierszy kodu)

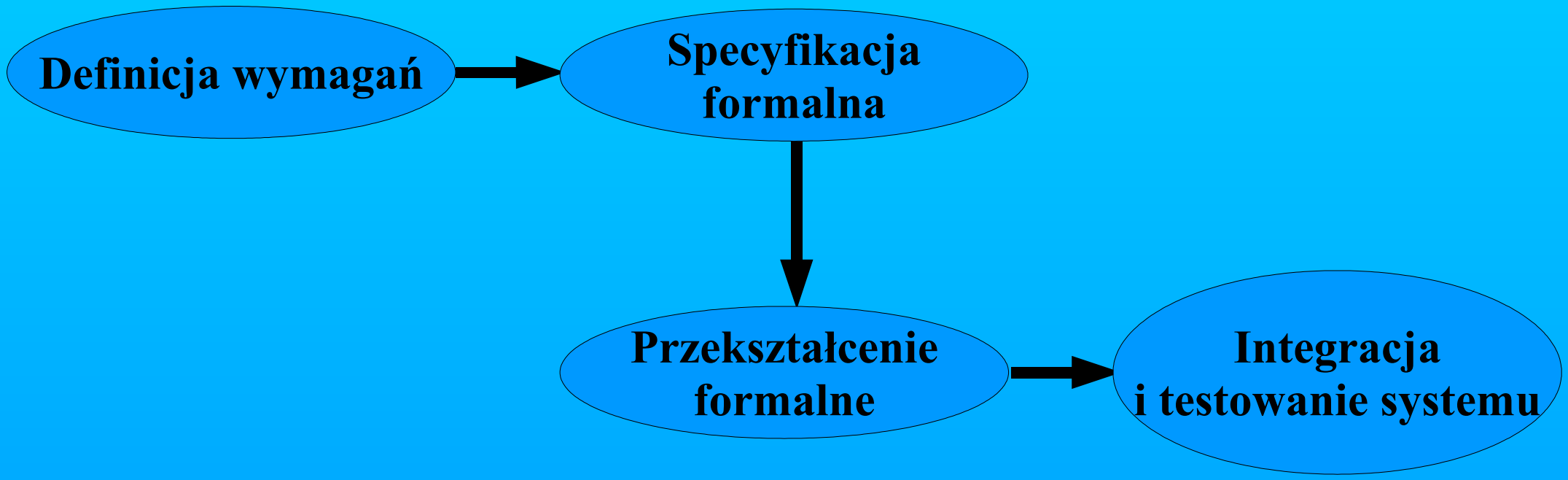
Model formalny

- Tworzenie formalne systemów jest podejściem, które ma wiele wspólnego z modelem kaskadowym. Proces tworzenia jest tu jednak oparty na matematycznych przekształceniach specyfikacji systemu w program wykonywalny.
- W procesie przekształcania formalna matematyczna reprezentacja systemu jest metodycznie przekształcana w bardziej szczegółowe, ale wciąż matematycznie poprawne reprezentacje systemu.

Model formalny

- Podejście z przekształceniem złożone z ciągu małych kroków jest łatwiejsze w użyciu. Wybór, które przekształcenie zastosować, wymaga jednak dużych umiejętności.
- Najlepiej znanym przykładem takiego formalnego procesu tworzenia jest Cleanroom, pierwotnie opracowany przez IBM (Proces Cleanroom jest oparty na przyrostowym tworzeniu oprogramowania, gdy formalnie wykonuje się każdy krok i dowodzi jego poprawności.)

Tworzenie formalne



Model formalny - uwagi

- Oprócz specjalistycznych dziedzin procesy oparte na przekształceniach formalnych są używane rzadko.
- Wymagają specjalistycznej wiedzy i w praktyce okazuje się, że w wypadku większości systemów nie powodują zmniejszenia kosztów lub polepszenia jakości w porównaniu z innymi podejściami.
- Interakcje systemów nie poddają się łatwo specyfikowaniu formalnemu.

Tworzenie z użyciem wielokrotnym

- W większości przedsięwzięć programistycznych występuje użycie wielokrotne oprogramowania.
- Zakłada się istnienie wielkiego zbioru dostępnych komponentów programowych użycia wielokrotnego oraz integrującej je struktury. Niekiedy systemy te są same w sobie systemami (Commercial Off-The-Shelf, COTS), które dostarczają specyficznej funkcjonalności.
- Etapy procesu:
  - analiza komponentów,
  - modyfikacja wymagań,
  - projektowanie systemu z użyciem wielokrotnym,
  - tworzenie i integracja

Etapy tworzenia z użyciem wielokrotnym

Początkowa faza specyfikacji wymagań i faza zatwierdzania są podobne do innych procesów; etapy pośrednie są inne.

Etapy pośrednie tworzenia z użyciem wielokrotnym

- **Analiza komponentów.** Na podstawie specyfikacji wymagań poszukuje się komponentów implementujących tę specyfikację.
- **Modyfikacja wymagań.** Analizuje się wymagania pod kątem uzyskanych komponentów, następnie modyfikuje się wymagania, aby odzwierciedlały dostępne komponenty.
- **Projektowanie systemu.** Projektuje się zrąb systemu lub ponownie wykorzystuje się instniejące zręby.Mogą być potrzebne nowe fragmenty oprogramowania.
- **Tworzenie i integracja.** Integruje się w system komponenty i zakupione systemy COTS.

Tworzenie z użyciem wielokrotnym