INŻYNIERIA OPROGRAMOWANIA

wykład 2:
MODELE PROCESU WYTWARZANIA OPROGRAMOWANIA

(na podstawie wykładów prof. K. Subiety, Instytut Informatyki PAN)

dr inż. Leszek Grocholski

Zakład Inżynierii Oprogramowania
Instytut Informatyki
Uniwersytet Wrocławski

PROCES WYTWARZANIA OPROGRAMOWANIA

Klasyczne elementy procesu :

- Faza wstępna: określenie strategicznych celów, planowanie i definicja projektu
- Określenie wymagań klienta
- Analiza: określenie dziedziny, wymagań systemowych
- Projektowanie: projektowanie pojęciowe, projektowanie logiczne
- Implementacja/konstrukcja: rozwijanie, testowanie, dokumentacja
- Testowanie integracyjne
- Dokumentacja
- Instalacja
- Przygotowanie użytkowników, akceptacja, szkolenie
- Działanie, włączając wspomaganie tworzenia aplikacji
- Utrzymanie, konserwacja, pielęgnacja
- Wycofanie oprogramowania

Wytwarzania oprogramowania a procesy w życiu oprogramowania

Uwaga:
Wytwarzanie oprogramowania to tylko jedna z grup procesów w życiu oprogramowania!

Dokładnie procesy w cyklu życia oprogramowania wymienia i opisuje norma ISO/IEC 12207.

Z kolei procesy w cyklu życia systemów wymienia i opisuje norma ISO/IEC 15288.

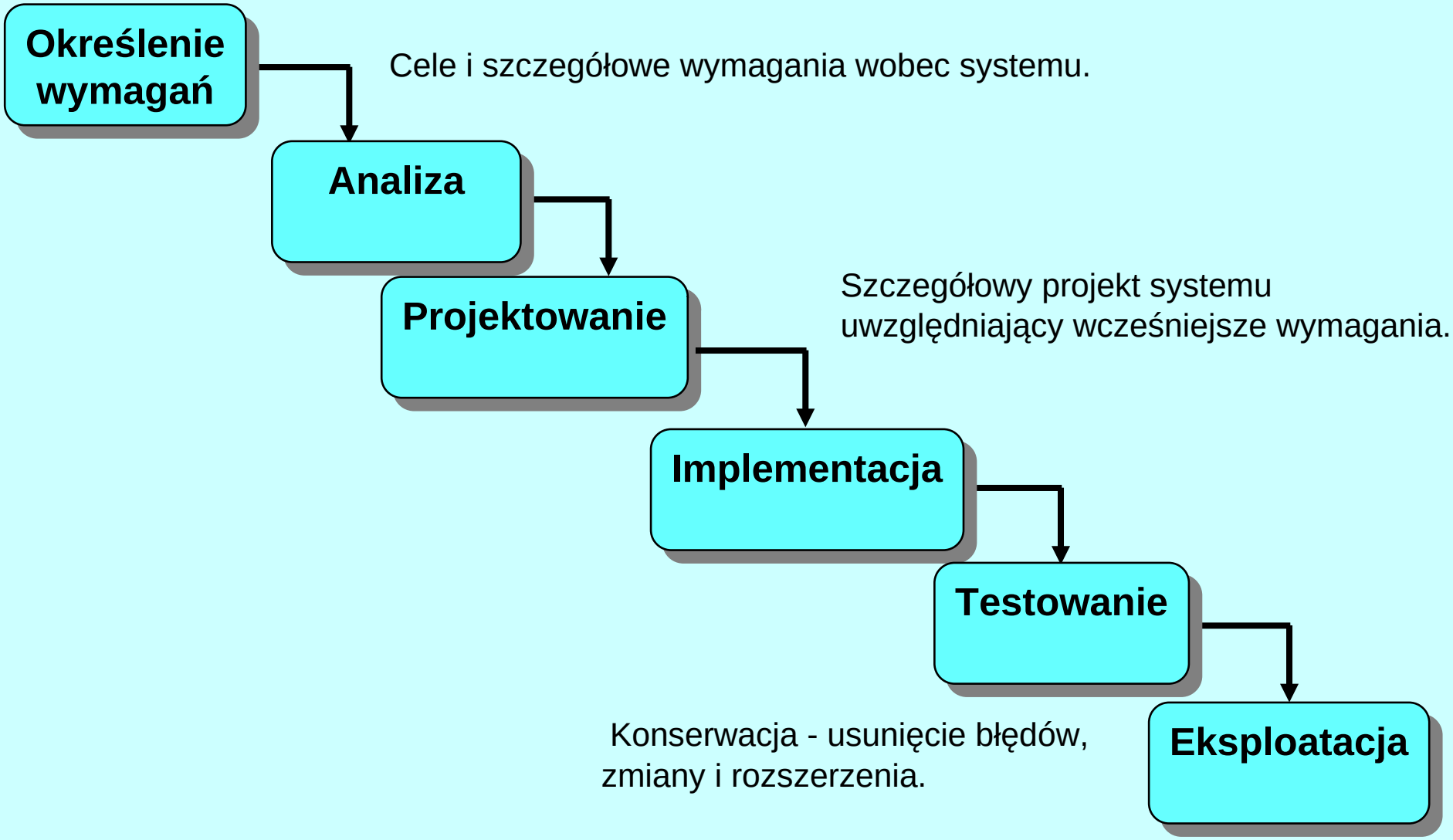
Modele cyklu wytwarzania oprogramowania

- Model kaskadowy (wodospadowy)
- Modele iteracyjne
- Model spiralny
- Prototypowanie
- Montaż z gotowych komponentów

Tego rodzaju modeli (oraz ich mutacji) jest bardzo dużo.

waterfall model

Model kaskadowy (wodospadowy)

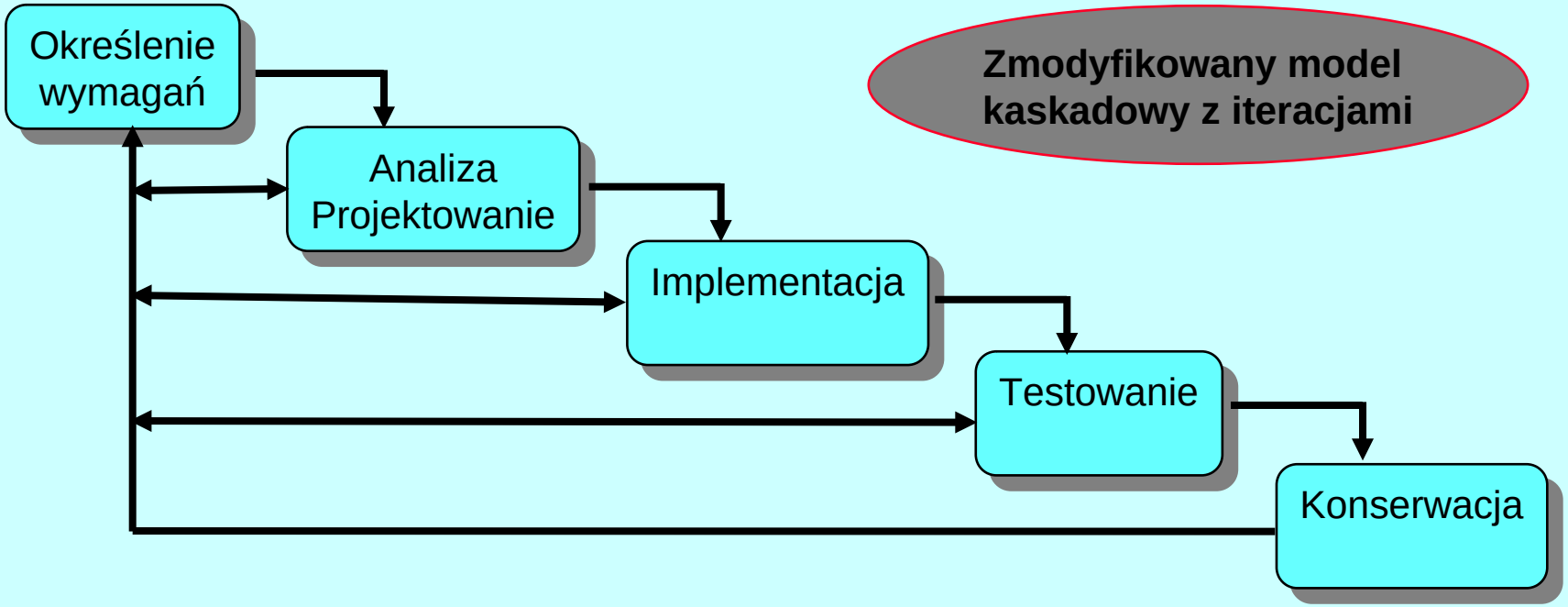


KRYTYKA MODELU KASKADOWEGO

Istnieją zróżnicowane poglądy co do przydatności praktycznej modelu kaskadowego. Podkreślane są następujące wady:

- Niemożliwość określenia (na początku i nie tylko)wszystkich wymagań
- Brak możliwości uwzględniania zmian wymagań
- Wysoki koszt błędów popełnionych we wczesnych fazach

Z drugiej strony, jest on niezbędny dla planowania, harmonogramowania, monitorowania i rozliczeń finansowych.



Model spiralny

spiral model

Planowanie: Ustalenie celów produkcji kolejnej wersji systemu

Analiza ryzyka
(ew. budowa prototypu)

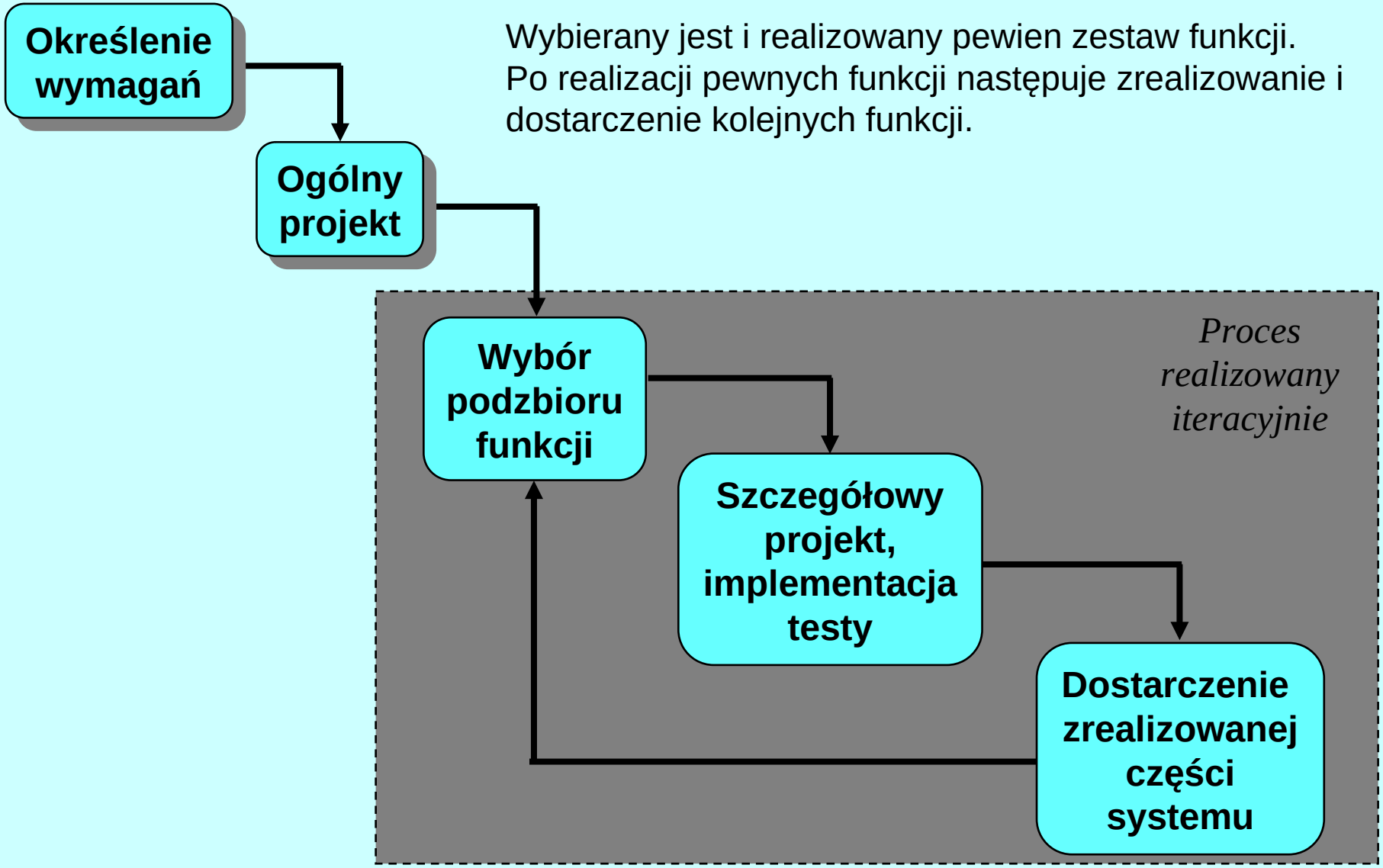
Konstrukcja
(model kaskadowy)

Atestowanie (przez klienta). Jeżeli ocena jest w pełni pozytywna, rozpoczynany jest kolejny cykl.

Istnieje wiele wariantów tego modelu.

Realizacja przyrostowa (odmiana m. spiralnego)

incremental development



Wybierany jest i realizowany pewien zestaw funkcji.
Po realizacji pewnych funkcji następuje zrealizowanie i dostarczenie kolejnych funkcji.

Prototypowanie

prototyping

Sposób na uniknięcie zbyt wysokich kosztów błędów popełnionych w fazie określania wymagań. Zalecany w przypadku, gdy określenie początkowych wymagań jest stosunkowo łatwe. **Przykłady: tworzenie makiet formatek (ekranów)**

Fazy:
- ogólne określenie wymagań
- budowa prototypu
- weryfikacja prototypu przez klienta
- pełne określenie wymagań
- realizacja pełnego systemu zgodnie z modelem kaskadowym/iteracyjnym

Cele
- wykrycie nieporozumień pomiędzy klientem a twórcami systemu
- wykrycie brakujących funkcji
- wykrycie trudnych usług
- wykrycie braków w specyfikacji wymagań

Korzyści:
- możliwość demonstracji makiety systemu
- możliwość szkoleń zanim zbudowany zostanie pełny system

Cechy prototypowania

- **Niepełna realizacja**: objęcie tylko części funkcji

- **„Wyklikiwanie”**: np. Oracle Express

- **Wykorzystanie gotowych komponentów**

- **Generatory interfejsu użytkownika**: wykonywany jest wyłącznie interfejs, wnętrze systemu jest “puste”.

- **Szybkie programowanie** (*quick-and-dirty*): normalne programowanie, ale bez zwracania uwagi na niektóre jego elementy, np. zaniechanie testowania.

Podczas implementacji następuje stopniowe, ewolucyjne przejście od prototypu do końcowego systemu. To oczywiście trwa. Należy starać się nie dopuścić do sytuacji, aby klient miał wrażenie, że prototyp jest prawie ukończonym produktem.
Po fazie prototypowania najlepiej prototyp skierować do archiwum.

Montaż z gotowych komponentów

Kładzie nacisk na możliwość redukcji nakładów poprzez wykorzystanie podobieństwa tworzonego oprogramowania do wcześniej tworzonych systemów oraz wykorzystanie gotowych komponentów dostępnych na rynku. **Klasyka dla systemów ERP.**

Temat jest określany jako ponowne użycie (reuse)

Metody
- zakup elementów ponownego użycia od dostawców
- przygotowanie elementów poprzednich przedsięwzięć do ponownego użycia

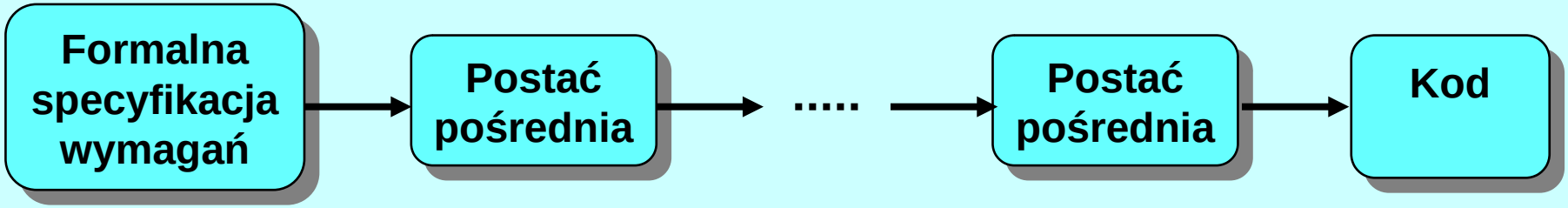
Zalety
- wysoka niezawodność
- zmniejszenie ryzyka
- efektywne wykorzystanie specjalistów
- narzucenie standardów

Wady
- dodatkowy koszt przygotowania elementów ponownego użycia
- ryzyko uzależnienia się od dostawcy elementów
- niedostatki narzędzi wspomagających ten rodzaj pracy.

Formalne transformacje

formal transformations

Postulowany w ramach tzw. nurtu formalnego w inżynierii oprogramowania.



Transformacje są wykonywane bez udziału ludzi (czyli w istocie, język specyfikacji wymagań jest nowym “cudownym” językiem programowania).

Tego rodzaju pomysły nie sprawdziły się w praktyce. Nie są znane szersze (są znane pewne) ich zastosowania. Metody matematyczne nie są w stanie utworzyć pełnej metodyki projektowania, gdyż metodyki włączają wiele elementów (np. psychologicznych) nie podlegających formalnemu traktowaniu. Metody matematyczne mogą jednak wspomagać pewne szczegółowe tematy (tak jak w biologii, ekonomii i innych dziedzinach), np. obliczanie pewnych mierzalnych charakterystyk oprogramowania.

RUP – Rational Unified Process

RUP – proces wytwarzania oprogramowania opracowany przez firmę Rational Software obecnie IBM Rational.

Obecnie zapomniany.

Firma IBM Rational jest autorem popularnego oprogramowania IBM Rational Architect stosowanego w bardzo wielu firmach, też we Wrocławiu
-Siemens, VOLVO IT, Asseco, Signity

Oprogramowanie to zawiera m in. pakiety do:
- zarządzania wymaganiami (Requisite Pro),
- zarządzania wersjami (Clear Case),
- edytor UML.

FAZY RUP


Początkowa
Opracowywanie
Konstrukcja
Wdrażanie
10%
30%
50%
10%
Planowanie
Wymagania
Architektura
Projekt implementacji
Implementacja
Integracja
Testowanie, weryfikowanie
Iter. wstępna
Iter. #1, Iter. #2
Iter.#n, Iter. #..., Iter.#m
Iter. #m+1, Iter .#m+2

Działania a fazy i iteracje (1)

W każdej z faz położony jest różny nacisk na różne działania:

- **Faza początkowa (wstępna) :** tu wysiłki koncentrują się na określeniu kontekstu systemu (aktorów), wymagań funkcjonalnych, zakresu odpowiedzialności (głównych wymagań), ograniczeń (wymagań niefunkcjonalnych), kryteriów akceptacji dla produktu finalnego oraz zgrubnego planu realizacji projektu.

- **Faza opracowywania:** poświęca się wiele uwagi wymaganiom, ale też prowadzone są prace projektowe i implementacyjne (w celu określenia projektu architektury i wytworzenia prototypu, stanowiącego podstawę dla dalszych prac) oraz planuje się MITYGACJE technicznych ryzyk poprzez testowanie różnych rozwiązań i narzędzi.

- **Faza budowy (konstrukcji):** nacisk jest położony na prace projektowe i implementacyjne. W tej fazie prototyp jest rozwijany, aż do uzyskania pierwszego operacyjnego produktu.

- **Faza przekazania (wdrażania):** prace są skoncentrowane na zapewnieniu produktowi odpowiedniej jakości: usuwa się błędy, ulepsza i/lub uzupełnia o brakujące elementy, wypełnia bazę, trenuje użytkowników.

Działania a fazy i iteracje (2)

FAZA OPRACOWANIA

✓Podstawowe zadania: (1) przeprowadzić „bardziej dokładną” analizę, (2) wypracować fundamenty dla architektury, (3) wyeliminować elementy obarczone największym ryzykiem, (4) uszczegółowić plan projektu (czyli skonstruować szczegółowe plany dla iteracji).

Decyzje architektoniczne muszą bazować na zrozumieniu całości systemu: jego funkcjonalności i ograniczeń (“zrozumienie z perspektywy na milę szerokiej i na cal głębokiej”).

✓ **<u>Faza opracowywania – najbardziej krytyczna z czterech faz</u>** (wysoki koszt, wysokie ryzyko): wymagania, architektura, plany powinny osiągnąć stabilną postać, ryzyka muszą być zmitygowane, tak aby była możliwa realna werfikacja zaplanowanych kosztów i czasu potrzebnego na ukończenie projektu.

✓ Bada się różne rozwiązania budując prototyp(y) – w jednej lub kilku iteracjach (zakres, rozmiar, nowości, inne ryzyka). Budowa prototypów ułatwia mitygowanie ryzyk oraz ustanawianie kompromisów między wymaganiami a możliwościami środowiska implementacji.

Planowanie z uwzględnieniem iteracji (1)

Planowanie projektu z uwzględnieniem iteracji wymaga rozstrzygnięcia kwestii, które nie istniały przy zastosowaniu tradycyjnego podejścia kaskadowego, jak np.:

- Jak wiele iteracji będzie potrzebne?
- Jak długo powinna trwać każda z nich?
- W jaki sposób należy ustalić cele i „zawartość” każdej iteracji?
- Jak należy śledzić postęp prac w trakcie realizacji iteracji?

Planowanie dużych, złożonych projektów

Ambitne próby planowania realizacji całości dużych, złożonych projektów z dokładnością „co do minuty”, wyróżniania działania i przypisywania osób do zadań na kilka miesięcy czy kilka lat do przodu, jak wykazała praktyka − są z góry skazane na niepowodzenie.
Tworzono diagramy Gantt’a (czy semantyczne sieci działania), które pokrywały całe ściany, a mimo to projekty kończyły się niepowodzeniem.
Skuteczna realizacja szczegółowych planów wymaga posiadania zarówno dobrego zrozumienia problemu, jak i stabilnych wymagań, stabilnej architektury oraz ukończonego choć jednego podobnego projektu,

Planowanie z uwzględnieniem iteracji (2)

Innymi słowy, jak można planować zbudowanie przez osobę ***X*** w tygodniu ***n*** modułu ***Y*** skoro w momencie planowania, w żaden sposób nie da się w tym czasie w ogóle wydedukować potrzeby posiadania modułu ***Y***? Planowanie szczegółowe jest możliwe w tych dziedzinach inżynieryjnych, gdzie wymagania są mniej lub bardziej zestandaryzowane i stabilne, gdzie kolejność zadań jest wystarczająco uporządkowana. Np. budując budynek nie można wznosić jednocześnie piętra pierwszego i czwartego. Po pierwszym musi powstać drugie, potem trzecie, aby można było zająć się czwartym.

RUP rekomenduje, by realizacja projektu była opierana na dwóch rodzajach planów, różniących się stopniem szczegółowości: **plan faz (zgrubny – nie więcej niż jedna, dwie strony opisu) i seria planów dla iteracji (szczegółowych)**.

Plan faz

Tworzony jest jeden plan faz, **jeden na potrzeby całego projektu**. Plan faz obejmuje z reguły jeden cykl − czasami, stosownie do potrzeb, kilka kolejnych cykli. Tworzony jest bardzo wcześnie, w fazie początkowej. W każdej momencie, może być poddawany modyfikacji.

Planowanie z uwzględnieniem iteracji (3)

Plan faz zawiera poniższe elementy:

- **Daty głównych** kamieni milowych:
  - **LCO** – określenie celów danego cyklu rozwijania projektu (koniec fazy początkowej; projekt ma dobrze określony zakres odpowiedzialności i zapewnione środki na realizację).
  - **LCA** – specyfikacja architektury (koniec fazy opracowywania; stabilizacja architektury).
  - **IOC** – osiągnięta początkowa zdolność operacyjna (koniec fazy konstrukcji; pierwsze beta).
  - Wypuszczenie produktu (koniec fazy wdrażania i koniec danego cyklu).

- **Profile zespołowe** --> jakie zasoby ludzkie są wymagane na poszczególnych etapach cyklu.
- **Specyfikację mniej ważnych kamieni milowych** --> data zakończenia każdej iteracji wraz ze specyfikacją jej celów (o ile można je zidentyfikować).

Planowanie z uwzględnieniem iteracji (4)

Plan iteracji
Tworzony jest **jeden plan dla każdej iteracji** (plan szczegółowy). Projekt z reguły posiada dwa plany iteracji aktywne w danym momencie:

- Plan dla bieżącej iteracji: jest wykorzystywany do śledzenia postępów prac w trakcie realizacji iteracji.
- Plan dla następnej iteracji: jest tworzony począwszy od połowy bieżącej iteracji i jest prawie (?) gotowy przy jej końcu.

Plan iteracji tworzony jest za pomocą tradycyjnych technik (diagramy Gantt'a, itp.) i narzędzi do planowania (Microsoft Project, itp.). Cel – definiowanie zadań, przypisanie zadań do osób i zespołów. Plan zawiera ważne daty, takie jak np.: zakończenie budowy „czegoś", dostarczenie komponentów z innej organizacji czy terminy głównych przeglądów.

RUP – przyjmując za podstawę proces iteracyjny i niekończącą się akomodację zmian celów i taktyki – zakłada tym samym, że nie istnieją racjonalne powody, by marnować czas na przygotowywanie szczegółowych planów dla dłuższych horyzontów czasowych: takie plany są trudne do zarządzania, szybko stają się nieaktualne oraz z reguły są ignorowane przez organizacje wykonawcze).

Planowanie z uwzględnieniem iteracji (5)

Iteracje w fazie początkowej

- W fazie początkowej często uważa się, że iteracji nie ma, tzn. nie ma czegoś, co można by nazwać „prawdziwą iteracją” (mini-projektem): żaden kod nie jest produkowany, wykonywane są przede wszystkim działania związane z rozpoznawaniem, planowaniem i marketingiem.

Czasami jednak iteracja jest potrzebna, po to by:

- Zbudować prototyp, by przekonać siebie czy sponsora, że proponowane rozwiązanie (lepiej: pomysł na rozwiązanie) jest dobre.

- Zbudować prototyp, w celu mitygowania ryzyk technicznych (np. eksploracja nowych technologii czy nowych algorytmów) czy udowodnienia, że cele wydajnościowe są osiągalne.

- Przyspieszyć wdrażanie narzędzi i procesów w organizacji.

Planowanie z uwzględnieniem iteracji (6)

Iteracje w fazie opracowywania

Plany dla iteracji w fazie opracowywania muszą być budowane w oparciu o trzy, najważniejsze w tej fazie elementy: ryzyka, własności krytyczne dla osiągnięcia sukcesu i pokrycie :

- Plany powinny być budowane tak, by większość ryzyk została zmitygowana lub „wysłana na emeryturę” właśnie w fazie opracowywania.

- Ryzyka są bardzo ważne, ale to nie mitygacja ryzyk jest celem ostatecznym. **Ostatecznym celem jest uzyskanie produktu, posiadającego własności uznane za krytyczne dla osiągnięcia sukcesu.**

- Ponieważ podstawowym celem fazy opracowywania jest budowa stabilnej architektury, plany muszą być konstruowane tak, by architektura adresowała wszystkie aspekty oprogramowania (pokrycie). Jest to ważne, bo architektura stanowi bazę dla dalszego planowania.

Podsumowanie (1)

Proces sekwencyjny jest odpowiedni dla realizacji małych projektów, czy tych z niewielką liczbą ryzyk (mało nowości, znane technologie, znana dziedzina problemowa, doświadczeni ludzie), natomiast niezbyt nadaje się dla projektów dużych, z dużą liczbą ryzyk.

Proces iteracyjny jest realizowany w oparciu o sekwencję iteracji. W trakcie każdej iteracji, opartej o model kaskadowy, wykonywane są działania związane z wymaganiami, tworzeniem projektu implementacji, implementacją i szacowaniem rezultatów.

W RUP w celu ułatwienia zarządzania procesem realizacji, iteracje zostały pogrupowane w cztery fazy: początkową, opracowywania, konstrukcji i wdrażania. W każdej z faz położono różny nacisk na różne działania.

Podejście iteracyjne ułatwia dokonywanie zmian (związanych z użytkownikiem, rynkiem czy technologiami), mitygację ryzyk, wprowadzanie technologii ponownego użycia, podnoszenie umiejętności członków zespołu, ulepszanie procesu jako takiego – wszystko to w efekcie skutkuje uzyskaniem lepszej jakości produktu finalnego.

Podsumowanie (2)

RUP − jako proces “prowadzony” w oparciu o przypadki użycia − uczynił z przypadków bazę dla całego procesu budowania produktu tworząc w oparciu o nie rodzaj języka stanowiącego podstawę do komunikacji między wszystkimi osobami zaangażowanymi w realizację projektu.

W RUP przypadki użycia wspomagają członków zespołu przy:

- planowaniu iteracji,
- budowie i walidacji stabilnej architektury systemu,
- definiowaniu przypadków i procedur testowych w modelu testów,
- tworzeniu dokumentacji użytkownika,
- rozmieszczaniu aplikacji (ang. deployment).

INŻYNIERIA OPROGRAMOWANIA

Dziękuję za uwagę