Inżynieria oprogramowania II

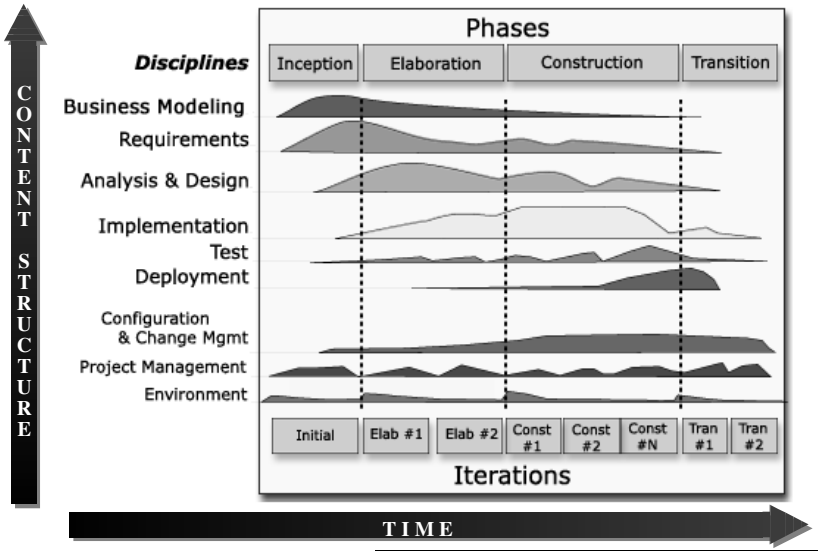
Wykład 3:

"Cykl życia projektu RUP"

Marek Krętowski
pokój 206
e-mail: mkret@ii.pb.bialystok.pl
http://aragorn.pb.bialystok.pl/~mkret

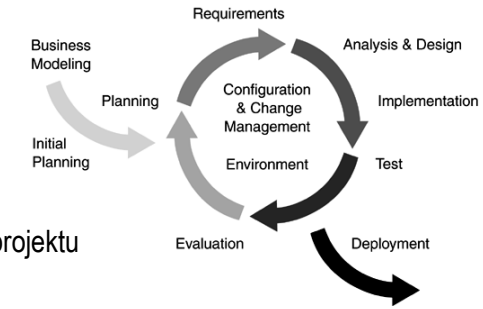
Wersja 1.05

Cykl życia projektu RUP

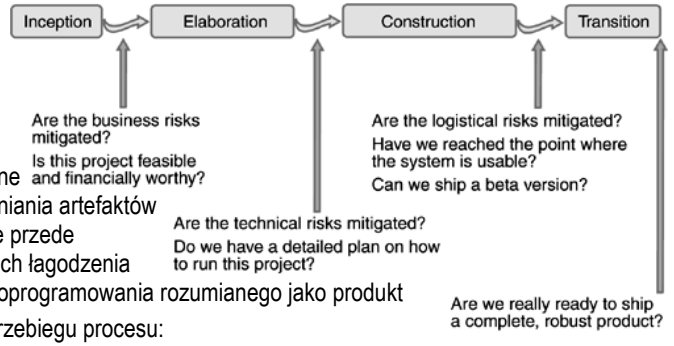


Fazy cyklu RUP

- 4 fazy cyklu życia: Rozpoczęcie, Rozwinięcie, Konstrukcja i Przekazanie
- Pojawia się błędne przekonanie, że sekwencyjnie przeprowadzane 4 fazy procesu odpowiadają fazom klasycznego modelu wodospadowego
  - rozumując w ten sposób całkowicie tracimy zalety metody iteracyjnej
- Fundamenty RUP => **iteracyjność** i **sterowanie ryzykiem**
- W ramach każdej z faz iteracje
  - zawierają różnorodne czynności procesu wytwórczego,
  - ich natężenie ulega zmianie w zależności od fazy
- Celem fazy jest wykonanie tych czynności, które pozwolą na osiągnięcie określonego stanu projektu
  - kamienie milowe



Kamienie milowe



- Kamienie milowe nie powinny być określane w kategoriach wypełniania artefaktów czy dokumentów, ale przede wszystkim w terminach łagodzenia ryzyka i ukończenia oprogramowania rozumianego jako produkt
- Brak narzuconego przebiegu procesu:
  - RUP oferuje paletę rozwiązań, z których należy wybierać te, które w danym kontekście (rodzaj cyklu, typ projektu, ...) są potrzebne do osiągnięcia celów konkretnej fazy
  - najgorsza sytuacja => zespół "ślepo" realizuje wszystkie możliwe czynności i artefakty
- Unikanie "zamrażania" artefaktów
  - celem nie jest zakończenie artefaktu, ale osiągnięcie takiego stanu, który pozwoli na podejmowanie decyzji

Rozpoczęcie (ang. inception)

Ustalenie celów i zakresu projektu oraz zdobycie wystarczającej ilości informacji, aby podjąć decyzję o przystąpienie do (bądź odstąpienie od) realizacji przedsięwzięcia.

Wyróżniamy 5 podstawowych celów fazy:

- **Zrozumienie co ma być tworzone**
  - określenie wizji, zakresu systemu i jego granic; zidentyfikowanie kto chce tego systemu i co jest on warty dla nich
  - ustalenie aktorów (użytkownicy i inne systemy) i sposobów ich interakcji z systemem (przypadki użycia)
  - opracowanie prototypów interfejsów użytkownika
- **Ustalenie kluczowej funkcjonalności**
  - rozstrzygnięcie które przypadki użycia są najbardziej krytyczne lub istotne z architektonicznego punktu widzenia
  - do 20% wszystkich + szczegółowy opis
- **Określenie co najmniej jednego możliwego rozwiązania**
  - zidentyfikowanie co najmniej jednej potencjalnej architektury, która umożliwia stworzenie systemu przy akceptowalnym poziomie ryzyka i rozsądnych kosztach
  - Wybierając potencjalne architektury analizujemy oczekiwaną funkcjonalność (zarówno w pierwszej jak i kolejnych wersjach), kompatybilność z innymi aplikacjami oraz wymagania dotyczące obsługi i pielęgnacji

Rozpoczęcie (2)

Pomocne mogą być pytania o:

- Jakie inne, podobne systemy zostały zbudowane (wykorzystana architektura, technologia), przy jakim koszcie?
- w sytuacji rozwoju istniejącego systemu: czy aktualna architektura jest nadal satysfakcjonująca?
- Jakie technologie musiałaby być wykorzystane w systemie? Czy niezbędne byłoby przyswojenie (nabycie) nowej technologii? Jakie są tego koszty i ryzyko z tym związane?
- Jakie komponenty programowe są potrzebne (bazy danych, middleware, ...) ? Czy mogą być nabyte? Czy mogą być ponownie wykorzystane z innych użytkowanych systemów? Jaki szacunkowy koszt i związane z tym ryzyko?

- **Zrozumienie kosztów, harmonogramu oraz ryzyk związanych z projektem**
  - uzasadnienie biznesowe (Business Case) w celu uzyskania odpowiednich funduszy, zwrot z inwestycji (ROI - return on investment)
  - w przypadku narzuconego budżetu projektu, co może być dostarczone w ramach budżetu i terminów
- **Zdecydowanie jaki proces zastosować i jakie narzędzia wykorzystać**
  - dostosowanie (okrojenie) procesu do specyficznych potrzeb projektu => eliminacja zbędnych narzutów, przygotowanie wzorców artefaktów
  - IDE, narzędzia zarządzania wymaganiami i modelowania, narzędzia zarządzania wersjami i konfiguracją, ...

Rozwinięcie (ang. elaboration)

Inna nazwa: opracowanie

Podstawowe cele fazy:

- **Uzyskanie bardziej szczegółowego zrozumienia wymagań**
  - dokładne opisanie większości przypadków użycia (do 80%),
  - stworzenie interfejsów użytkownika oraz przejście z użytkownikami przez każdy z nich w celu potwierdzenia funkcjonalności (jakie informacje wprowadzane, jakie wyświetlane, ...)
  - częściowa implementacja kluczowych scenariuszy
  - ew. pojawiają się nowi aktorzy i przypadki użycia.
  - część przypadków użycia (najprostsze lub analogiczne do innych, ale wykorzystujące inne dane) nie musi być formalnie opracowana => ich opisanie nie wpływa na łagodzenie ryzyka
  - “time-box” - ograniczenie czasu opracowywania przypadku użycia w celu uniknięcia zbyt szczegółowego opisu => zwłaszcza przy ograniczonych projektach

Rozwinięcie: architektura

Zaprojektowanie, zaimplementowanie oraz walidacja architektury

- umożliwienie przetestowania szkieletu systemu (także w. niefunkcjonalne)
- krytyczne decyzje projektowe (wybór technologii, główne komponenty i ich interfejsy)
- zaprojektowanie i implementacja mechanizmów architektonicznych (rozwiązanie typowych problemów np. trwałość danych, komunikacja z systemami zewnętrznymi przy użyciu protokołów, identyfikacja użytkownika, ...)

- W uproszczeniu architektura obejmuje kilka kluczowych decyzji projektowych
  - strukturalizacja systemu (podział na części i ich interfejsy); podjęcie decyzji o sposobie uzyskania (budowa, zakup, ponowne wykorzystanie)
  - sposób interakcji elementów podczas realizacji najistotniejszych scenariuszy (dynamika systemu)
  - implementacja i przetestowanie osiągów, skalowalności i kosztu =>eliminacja podstawowego ryzyka technicznego
- Zaleca się wykorzystanie szkieletów architektonicznych (np. aplikacje ubezpieczeniowe => IAAA firmy IBM) lub zaadaptowanie wcześniej opracowanych i sprawdzonych rozwiązań

Rozwinięcie: architektura

- **Wykonywalna architektura** (ang. executable architecture) - częściowa implementacja systemu zbudowana, aby pokazać, że zaproponowane rozwiązanie architektoniczne wspiera kluczową funkcjonalność oraz wykazuje oczekiwane własności (wydajność, niezawodność, skalowalność, ...)
  - ewolucyjny prototyp => wykorzystanie w docelowym systemie
- Na zakończenie architektura powinna osiągnąć stabilność pozwalającą traktować ją jako element odniesienia podczas dalszej budowy systemu (baseline)
  - ograniczenie zmian => tylko uzasadnione, wyjątkowe sytuacje

Rozwinięcie (4)

- Łagodzenie zasadniczych ryzyk oraz opracowanie bardziej dokładnych oszacowań harmonogramu i kosztu
  - ryzyka związane z wymaganiami => uzyskanie akceptacji użytkowników w wyniku prezentacji opracowanych przypadków użycia (interfejsy użytkownika)
  - ryzyka techniczne => zaimplementowana architektura
  - ryzyka związane z ustawieniem procesu i jego środowiskiem => sprawdzenie efektywności pracy zespołu, narzędzi i technologii w działaniu (pełna iteracja)
  - redukcja ryzyka pozwala na znaczące zbliżenie dolnych i górnych oszacowań
- Poprawienie dostosowanych do aktualnego projektu elementów procesu wytwórczego i sposobów ich wykorzystania (ang. development case) oraz udoskonalenie środowiska wytwórczego

Konstrukcja (ang. construction)

- Inna nazwa: budowa
- Obejmuje efektywne z punktu widzenia kosztów tworzenie wysokiej jakości kodu, prowadzącego do powstania kompletnej, operacyjnej wersji systemu, która może być przekazana do użytkowników
- Największa faza projektu (65% całościowych nakładów, 50% czasu)

Podstawowe cele fazy:

- Minimalizowanie kosztów rozwoju oraz zrównoleglenie przynajmniej części prac w zespole wykonawczym
  - optymalizacja wykorzystania zasobów oraz unikanie zbędnych powtórzeń (organizowanie prac w ramach architektury)
  - nawet w niewielkich projektach występują komponenty, które mogą być tworzone niezależnie

Konstrukcja (2)

- Iteracyjny rozwój kompletnego produktu, który jest gotowy do przekazaniu docelowemu użytkownikowi
  - opracowanie pierwszej operacyjnej wersji systemu (beta) poprzez opisywanie pozostałych przypadków użycia i innych wymagań, uzupełnienie szczegółów projektowych, kończenie implementacji komponentów, przeprowadzanie scalania systemu oraz testowanie oprogramowania
  - ustalenie czy oprogramowanie, docelowe miejsca wykorzystania oraz użytkownicy są gotowi do wdrożenie aplikacji

Przekazanie (ang. transition)

- Zapewnienie, że stworzone oprogramowanie w pełni spełnia potrzeby i oczekiwania użytkowników

Podstawowe cele fazy:

- Przeprowadzenie testów beta
  - zwykle wiąże się z dopracowaniem aplikacji (usuwanie błędów, poprawa osiągów i użyteczności)
  - metryki defektów i pokrycia testów
- Przeszkolenie użytkowników oraz administratorów w celu osiągnięcia przez nich pewności i samodzielności
- Przygotowanie miejsca wdrożenia oraz przeniesienie niezbędnych danych
  - może wymagać zakupu i odpowiednie-go dostosowania sprzętu i infrastruktury
  - okres równoległego wykorzystania starego i nowego systemu
- Przygotowanie do wprowadzenia na rynek
  - opakowania, nośniki, premiera i oprawa marketingowa, przekazanie do dystrybucji i sprzedaży, przeszkolenie konsultantów
  - głównie systemy komercyjne

Przekazanie (2)

- Uzyskanie akceptacji wszystkich zainteresowanych, że produkt jest gotowy i zgodny z określonymi w wizji kryteriami oceny
  - testy akceptacyjne (formalne, nieformalne, beta testy)
  - obejmuje nie tylko oprogramowanie, ale również inne el . np. dokumentację
- Ulepszenie realizacji przyszłych projektów poprzez wykorzystanie doświadczenia zdobytego podczas realizacji projektu
  - wymaga udokumentowania i przeanalizowania co się sprawdziło a które elementy zawiodły
  - poprawa procesu oraz środowiska wytwórczego
  - warto również rozważyć ponowne wykorzystanie części projektu => może wymagać to prac przygotowawczych