RATIONAL UNIFIED PROCESS

Opis metodyki i procesu produkcji oprogramowania

RATIONAL UNIFIED PROCESS

- **Rational Unified Process** (**RUP**) to proces iteracyjnego wytwarzania oprogramowania opracowany przez firmę Rational Software Corporation (firma została przejęta przez IBM).
- Proces RUP nie jest pojedynczym, ściśle określonym procesem, ale raczej szablonem procesu. Został on zaprojektowany w celu przystosowania do charakteru konkretnej organizacji (przedsiębiorstwa), konkretnego zespołu projektowego lub nawet charakteru konkretnego projektu. Z szablonu RUP można wybrać elementy w zależności od konkretnych potrzeb.
- Rational Unified Process (RUP) to także nazwa oprogramowania, opracowanego przez Rational Software (obecnie dostępnego w IBM). Produkt ten zawiera hipertekstową bazę wiedzy z przykładowymi artefaktami oraz szczegółowymi opisami wielu typów czynności. Process RUP definiowany jest także w produkcie Rational Method Composer (RMC), który pozwala na tworzenie spersonalizowanych wersji RUP.

HISTORIA

- Proces Rational sięga swoimi korzeniami do oryginalnego modelu spiralnego Barrego Boehma - jeden z głównych autorów RUP, Ken Hartman, prowadził razem z Boehmem badania. Podejście Rational (Rational Approach) zostało opracowane przez Rational Software w latach osiemdziesiątych i dziewięćdziesiątych.
- W roku 2000 Rational przejęło szwedzką firmę Objectory AB. Zunifikowany proces Rational (Rational Unified Process) był rezultatem połączenia podejścia Rational oraz metodyki Objectory zdefiniowanej przez jej założyciela Ivara Jacobsona. Początkowo powstał proces nazwany Rational Objectory Process, który był podejściem firmy Objectory przystosowanym do narzędzia Rose. Kiedy połączenie obydwu metodyk zostało ostatecznie osiągnięte, zmieniono nazwę na obecną. Pierwsza wersja RUP 5.0 opublikowana została w 1998 roku. Głównym architektem był Philippe Kruchten.

BUDOWA

- Autorzy procesu skupili się na diagnozowaniu charakterystyk projektów, które zakończyły się fiaskiem. Postępując w ten sposób, próbowali poznać przyczyny owych niepowodzeń. Przyglądali się również ówcześnie istniejącym procesom inżynierii oprogramowania i sposobom, w jaki rozwiązywały one problemy.
- Lista najczęstszych błędów zawierała następujące rzeczy:
  1. Zarządzanie wymaganiami *Ad hoc* (najczęściej brak zarządzania nimi)
  2. Niejednoznaczna, nieprecyzyjna komunikacja
  3. Architektura oprogramowania nieodporna na obciążenia (ang. Brittle architecture)
  4. Zbytnia, niepotrzebna złożoność oprogramowania
  5. Niewykryte niespójności w wymaganiach, projekcie oraz implementacji
  6. Brak lub niewystarczające testowanie
  7. Subiektywna ocena postępu projektu
  8. Brak zarządzania ryzykiem
  9. Brak automatyzacji prowadzenia projektu

BUDOWA

- Niepowodzenie projektu było spowodowane kombinacją wielu czynników, w każdym projekcie w specyficzny sposób. Rezultatem badań firmy Rational było opracowanie zbioru dobrych praktyk, które nazwane zostały właśnie Rational Unified Process.
- Proces RUP został opracowany z użyciem tych samych technik, których zespół Rational używał do modelowania systemów - języka UML. Język UML powstawał równolegle z RUP (również jako połączenie doświadczenia w modelowaniu firm Objectory i Rational).

PODSTAWY I NAJLEPSZE PRAKTYKI

- RUP bazuje na zbiorze zasad inżynierii programowania oraz najlepszych praktykach, na przykład:

1. Iteracyjnym wytwarzaniu oprogramowania (Iterative Development)
2. Zarządzaniu wymaganiami (Requirement Management)
3. Używaniu architektury bazującej na komponentach (Component-based architecture)
4. Graficznym projektowaniu oprogramowania
5. Kontroli jakości oprogramowania (Quality Assurance)
6. Procesu kontroli zmian w oprogramowaniu (Change Management)

ITERACYJNE WYTWARZANIE OPROGRAMOWANIA

- Wymagania podczas procesu wytwarzania oprogramowania ulegają częstym zmianom, z powodu ograniczeń architektury, zmiany potrzeb użytkownika lub lepszego zrozumienia problemu. Wytwarzanie oprogramowania w kolejnych iteracjach, pozwala skupić się w pierwszej kolejności na obszarach najbardziej ryzykownych (np. najmniej rozpoznanych). W idealnym przypadku każda iteracja kończy się stworzeniem wykonywalnego artefaktu - pomaga to zredukować ryzyko w projekcie, otrzymujemy szybciej opinie od odbiorców oprogramowania a programistom pozwalamy skupić się na węższej dziedzinie.
- RUP używa podejścia iteracyjnego i przyrostowego z następujących powodów:
  - Integracja oprogramowania robiona krok po kroku podczas wytwarzania oprogramowania, ograniczając go do mniejszej liczby elementów
  - Integracja jest prostsza i mniej kosztowna
  - Składowe oprogramowania są projektowane oddzielnie i łatwiej poddają się reużywalności
  - Łatwiej wykrywać zmiany wymagań i łatwiej nimi zarządzać
  - Ryzyka identyfikowane i atakowane są wcześnie ponieważ każda iteracja pozwala wykryć kolejne ryzyka
  - W iteracjach ulepszana jest architektura oprogramowania
- Projekt wykorzystujący model iteracyjny będzie posiadał jeden główny plan faz, a zarazem wiele planów iteracji. Włączenie się udziałowców (stakeholders) jest często praktykowane przy każdym kamieniu milowym. W tej sytuacji, kamienie milowe stanowią zachętę dla udziałowców oraz dostarczają informację dotyczącą spełnienia wymagań przez system oraz gotowości organizacji do jego wdrożenia.

ZARZĄDZANIE WYMAGANIAMI

- Zarządzanie wymaganiami w RUP jest skupione na zaspokojeniu oczekiwań użytkowników końcowych systemu poprzez identyfikację i specyfikację ich potrzeb oraz **wykrywanie zmiany** tych wymagań. Zalety zarządzania wymaganiami:
  - Poprawnie zidentyfikowane wymagania tworzą prawidłowy produkt, potrzeby użytkownika są zaspokojone.
  - Tworzymy istotną dla użytkowników funkcjonalność, redukując późniejsze koszty dobudowywania zapomnianej (niezidentyfikowanej podczas tworzenia) funkcjonalności.

ZARZĄDZANIE WYMAGANIAMI

- RUP sugeruje, że zarządzanie wymaganiami składa się z następujących czynności:

1. **Analiza problemu** - uzgodnienie problemu i stworzenie miar, które dowiodą jego istotności dla klienta.
2. **Zrozumienie potrzeb udziałowców (stakeholders -** są to odbiorcy i użytkownicy oprogramowania na różnych szczeblach w organizacji, w innych metodykach zarządzania projektami nazywa się ich Interesariuszami) - konsultacja problemu i jego wartości z głównymi udziałowcami (stakeholders) i rozpoznanie w jaki sposób koncepcja rozwiązania zaspokaja ich potrzeby.

ZARZĄDZANIE WYMAGANIAMI

3. **Definicja systemu** - tworzenie projektu funkcjonalności na podstawie potrzeb użytkowników, identyfikacja przypadków użycia - które prezentują ogólne wymagania (high-level requirements) i użyteczność modelu systemu.
4. **Zarządzanie zakresem systemu** (Scope Management) - modyfikowanie zakresu prac nad systemem bazując na analizie wymagań, wybór kolejności realizacji (atakowania) przypadków użycia.
5. **Zawężanie definicji systemu** - uszczegóławianie scenariuszy przypadków użycia razem z użytkownikami systemu w celu stworzenia dokładnej Specyfikacji wymagań (ang. Software Requirements Specification - SRS), która może służyć (i na ogół służy) jako umowa pomiędzy wykonawcą systemu a klientem. Na podstawie dokumentu SRS tworzony jest projekt systemu oraz scenariusze testów.
6. **Zarządzanie zmianami wymagań** - zarządzanie zmianami wymagań lub nowozidentyfikowanymi wymaganiami w czasie trwania projektu.

ARCHITEKTURA BAZUJĄCA NA KOMPONENTACH

- Użycie architektury bazującej na komponentach pozwala na stworzenie systemu, który jest łatwo rozszerzalny, intuicyjnie zrozumiały i wspomaga reużywalność. Komponentem nazywamy zbiór powiązanych obiektów (w sensie programowania obiektowego).
- Architektura oprogramowania zyskuje na znaczeniu w miarę jak systemy informatyczne stają się coraz większe i bardziej złożone. RUP skupia się na stworzeniu prostej architektury w początkowych iteracjach. Staje się ona prototypem dla pierwszej fazy implementacji (development). Ewoluuje ona w każdej iteracji zbliżając się do architektury finalnej. RUP zakłada reguły i ograniczenia projektowe w celu uchwycenia reguł architektury. Poprzez iteracyjne wytwarzanie oprogramowania zyskujemy możliwość stopniowej identyfikacji komponentów, które mogą być w dalszej części: zakupione, zbudowane, lub użyte ponownie. Komponenty są często budowane na bazie istniejących technologii typu CORBA, COM, JEE.

WIZUALNE MODELOWANIE OPROGRAMOWANIA

- Abstrakcja projektowania od kodu i przedstawienie koncepcji za pomocą bloków graficznych może być efektywnym sposobem aby pokazać perspektywę rozwiązania. Używając takiej reprezentacji, techniczni członkowie zespołu mają możliwość wybrania najlepszego sposobu implementacji zbioru powiązanej funkcjonalności. Reprezentacja graficzna jest także produktem pośrednim pomiędzy analizą procesu biznesowego, a implementacją. Model w tym kontekście jest formą wizualizacji oraz uproszczeniem bardziej skomplikowanego projektu. RUP specyfikuje wymagane modele i opisuje dlaczego są wymagane.

KONTROLA I WERYFIKACJA JAKOŚCI OPROGRAMOWANIA

- Ocena jakości jest najczęstszym słabym punktem projektów programistycznych ponieważ jest często planowana po fakcie budowy systemu i czasami obsługiwana przez inny zespół. RUP pomaga w planowaniu kontroli jakości i jej ocenie poprzez wbudowanie jej w cały proces i zaangażowanie w nią wszystkich członków zespołu. Nie ma pracowników przypisanych tylko do jakości - RUP zakłada, że każdy członek zespołu jest odpowiedzialny za jakość w ciągu całego procesu. Proces koncentruje się na spełnieniu wymaganego poziomu jakości i zapewnia mechanizmy (workflows) do pomiaru tego poziomu.

ZARZĄDZANIE ZMIANAMI W OPROGRAMOWANIU

- We wszystkich projektach programistycznych pojawiają się z czasem zmiany i są one nieuniknione. RUP definiuje metody śledzenia, ewidencji i kontroli zmian. Zdefiniowane są także tzw. *secure workspaces* (bezpieczne przestrzenie robocze), które pozwalają na zagwarantowanie, że zmiany w innych systemach nie wpłyną na system tworzony. Koncepcja ta jest ściśle powiązana z tworzeniem architektury zorientowanej komponentowo.

CYKL ŻYCIA PROJEKTU

- Cykl życia w RUP bazuje na modelu spiralnym. RUP jest dostępny jako struktura prowadzenia projektu, która może być personalizowana w celu przystosowania do specyficznych potrzeb projektowych. Cykl życia w RUP układa zadania w fazy i iteracje.
- Projekt został podzielony na cztery fazy:

1. Faza rozpoczęcia (Inception phase)
2. Faza opracowywania (Elaboration phase)
3. Faza konstrukcji (Construction phase)
4. Faza przekazania systemu (Transition phase)

FAZA ROZPOCZĘCIA

- W fazie tej formułowany jest problem - zagadnienie biznesowe (business case). Przy opracowaniu tego zagadnienia określa się jego kontekst (business context); czynniki wpływające na jego powodzenie (success factors) - na przykład spodziewany zwrot z inwestycji, zwiększenie udziału w rynku; oraz prognozę finansową. Dodatkowo uzupełnia się go o prosty model przypadków użycia, plan projektu, wstępną analizę ryzyka oraz opis projektu (główne wymagania, ograniczenia, główna funkcjonalność).

FAZA ROZPOCZĘCIA

- Po stworzeniu powyższych dokumentów projekt sprawdza się według następujących kryteriów:
  - Zgoda użytkowników na oszacowany koszt/czas wykonania.
  - Zrozumienie wymagań poprzez ocenę jakości głównych przypadków użycia.
  - Wiarygodność szacowanych kosztów, priorytetów, ryzyka i planu procesu wytwarzania.
  - Rozmiar stworzonego prototypu architektury.
  - Wydatki rzeczywiste względem wydatków planowanych.
  - Jeżeli wstępny projekt nie osiągnie kamienia milowego (ang. milestone), nazywanego **Lifecycle Objective Milestone**, może być albo zakończony, albo faza ta może zostać jeszcze raz powtórzona (w celu ulepszenia projektu wstępnego).

FAZA OPRACOWANIA

- W tej fazie projekt systemu nabiera kształtów. Przeprowadzona jest analiza dziedziny zagadnienia (ang. Domain Analysis - nazywana też w literaturze polskiej Analizą/Modelem Domeny) i budowana podstawowa architektura systemu.
- Zakończenie tej fazy wiąże się z osiągnięciem kamienia milowego **Lifecycle Architecture Milestone** poprzez spełnienie kryteriów:

1. Stworzony został model przypadków użycia - zidentyfikowani zostali aktorzy i większość przypadków. Model jest kompletny w 80%.
2. Została opracowana architektura systemu.
3. Architektura ta pozwala realizować główne przypadki użycia.
4. Sprawdzona została zgodność zagadnienia biznesowego oraz listy ryzyk.
5. Stworzony został plan prac dla całego projektu.

- Jeżeli projekt nie może przejść tej fazy, ciągle mamy czas na jego zaniechanie, lub ponowne opracowanie. Przechodząc do następnej fazy przechodzimy w obszar większego ryzyka, w którym zmiana (np. wymagań) jest dużo trudniejsza i znacząca.

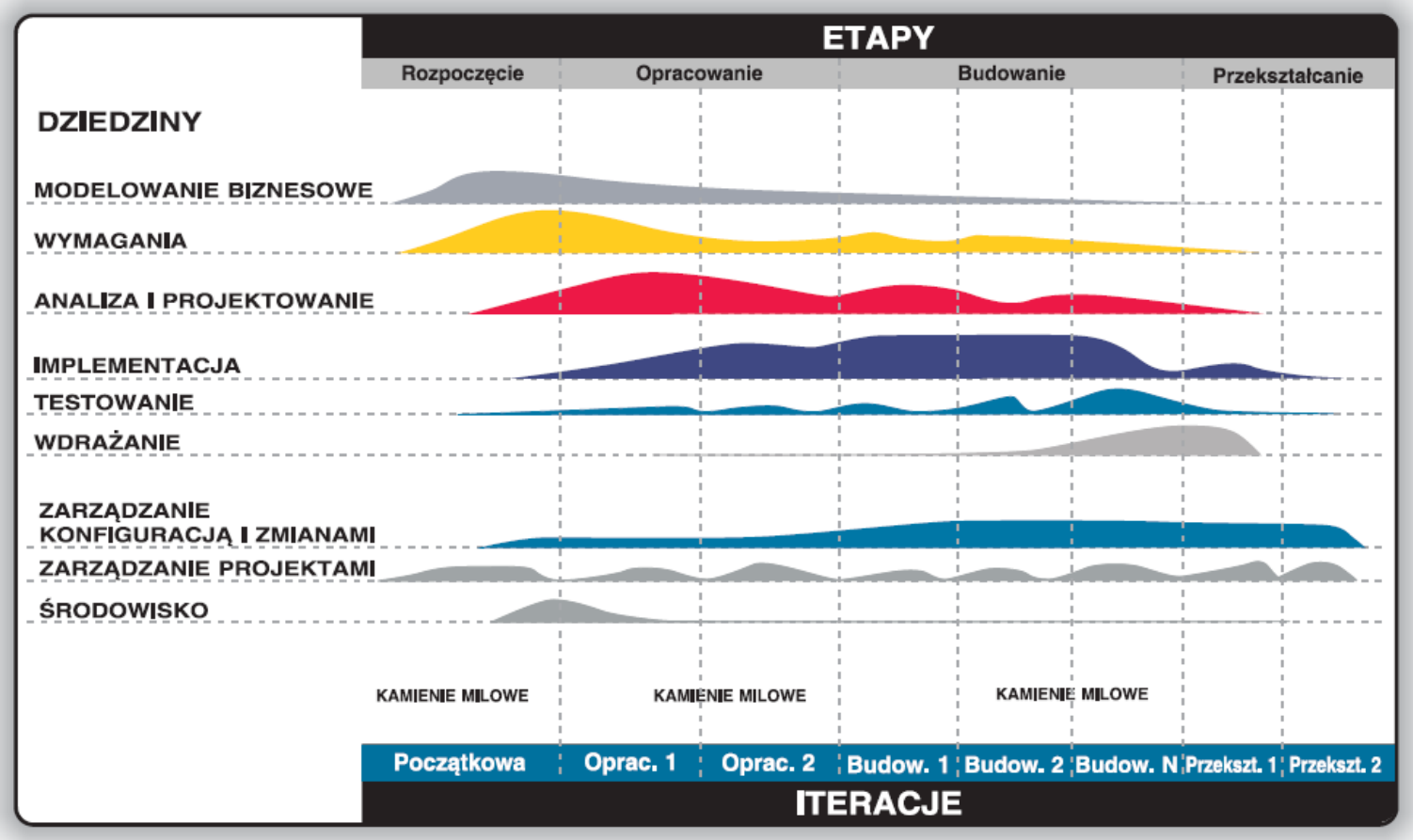
FAZA KONSTRUKCJI

- W fazie tej główny nacisk położony jest na budowę komponentów i innych funkcjonalności opracowywanego systemu. W tej fazie odbywa się większość prac programistycznych. W większych projektach może być wiele iteracji konstrukcji, w celu podzielenia dziedziny przypadków użycia na mniejsze, zarządzalne poddziedziny. Pozwala to także na szybsze przekazywanie części prac (lub prototypów).
- W tej fazie tworzona jest pierwsza wersja oprogramowania do wglądu użytkowników zewnętrznych. Zakończenie fazy wiąże się z osiągnięciem **Initial Operational Capability Milestone**.

FAZA PRZEKAZANIA SYSTEMU

- W tej fazie produkt przekazywany jest od zespołu programistycznego do użytkowników końcowych (potocznie mówiąc: do produkcji). W tej fazie znajdują się takie czynności jak: trening użytkowników końcowych i administratorów, testy akceptacyjne (testy beta). Sprawdzana jest zgodność produktu z miarami jakości określonymi w pierwszej fazie.
- Spełnienie celów jest tożsame z osiągnięciem **Product Release Milestone** i zakończeniem cyklu wytwarzania oprogramowania.

FAZY PROJEKTOWANIA I PRZEKSZTAŁCANIA W METODYCE RUP



DYSCYPLINY I POSTĘP PRAC

- RUP bazuje na zbiorze klocków (**building blocks**, content elements). Opisują one, co ma zostać stworzone, jakie umiejętności są do tego wymagane oraz, krok po kroku, jak powinien wyglądać proces wytwarzania.
- Główne klocki:
  - **Rola** (Roles) - *Kto?* – Rola definiuje zbiór wymaganych umiejętności, kompetencji i odpowiedzialności.
  - **Produkt** (Work Products) - *Co?* – Produkt reprezentuje wynik zadania oraz wszystkie dokumenty i modele utworzone w czasie procesu.
  - **Zadanie** (Tasks) - *Jak?* – Zadanie opisuje jednostkę pracy przypisaną do roli.

DYSCYPLINY I POSTĘP PRAC

- W ramach każdej iteracji zadania podzielone są na dziewięć dyscyplin (disciplines):
  - Dyscypliny inżynierskie (Engineering Disciplines):
    - **Modelowanie biznesowe** (Business modeling)
    - **Wymagania** (Requirements)
    - **Analiza i projekt** (Analysis and design)
    - **Implementacja** (Implementation)
    - **Testowanie** (Test)
    - **Wdrożenie** (Deployment)
  - Dyscypliny pomocnicze (Supporting Disciplines):
    - **Zarządzanie zmianami oraz konfiguracją** (Configuration and change management)
    - **Zarządzanie projektem** (Project management)
    - **Środowisko** (Environment)

MODELOWANIE BIZNESOWE

- Z biegiem czasu przedsiębiorstwa i inne organizacje stają się coraz bardziej zależne od systemów informatycznych. Wymusza to w sposób oczywisty na inżynierach tworzących oprogramowanie wiedzę, w jaki sposób ich systemy wpasowują się w procesy zachodzące w administracji i jakie jej wymogi adresują. Z kolei firmy inwestują na ogół w systemy informatyczne na podstawie racjonalnych przesłanek - wtedy, kiedy widzą wartość dodaną wynikającą ze stworzenia takiego systemu.
- Celem modelowania biznesowego jest przede wszystkim zapewnienie komunikacji i lepsze zrozumienie pomiędzy biznesem (inżynieria biznesowa) a IT (inżynieria oprogramowania). Zrozumienie biznesu oznacza, że inżynierowie oprogramowania muszą zrozumieć strukturę i dynamikę organizacji swojego klienta, jego bieżące problemy i możliwe usprawnienia. Muszą także zapewnić wspólne zrozumienie celów pomiędzy klientami, użytkownikami końcowymi a programistami.
- Modelowanie biznesowe tłumaczy w jaki sposób opisać wizję organizacji, w której będzie wdrożony system i jak później użyć jej do opisania procesów, ról i odpowiedzialności w organizacji.

WYMAGANIA

- Celem Wymagań jest opisanie tego, co system powinien robić. Wymagania zbierane są przez analityków, którzy odkrywają je, klasyfikują i dokumentują. Proces zbierania wymagań polega na dyskusji i uzgodnieniach pomiędzy tworzącymi system a klientem.

ANALIZA I PROJEKT

- Celem Analizy i projektu jest zobrazowanie sposobu w jaki będzie tworzony system w fazie implementacji. Ma to być system, który:
  - Zapewnia w specyficznym środowisku realizację zadań i funkcji opisanych w przypadkach użycia.
  - Spełnia wszystkie swoje wymagania.
  - Jest łatwy do zmiany, gdy zmienią się wymagania funkcjonalne.
- **Analiza i projekt** tworzy model projektowy i opcjonalnie model analityczny systemu. Model projektowy zapewnia abstrakcję od kodu źródłowego - to znaczy, służy on jako wytyczne do stworzenia tego kodu. Model projektowy składa się z klas zorganizowanych w pakiety i podsystemy z dobrze określonymi interfejsami. Służy to wyodrębnieniu komponentów w fazie implementacji. Zawiera także opis, które obiekty klas współpracują w celu realizacji przypadków użycia.

IMPLEMENTACJA

Celami implementacji są:

- Zdefiniowanie organizacji kodu systemu, w sensie podsystemów zorganizowanych w warstwy.
- Stworzenie klas i obiektów w sensie komponentów (pliki źródłowe, binaria, pliki wykonywalne i inne)
- Testowanie tworzonych komponentów jako jednostki (testami jednostkowymi).
- Integracja wyników tworzonych przez poszczególne osoby lub zespoły do pełnego systemu.

- Systemy realizowane są poprzez implementację swoich komponentów. Proces opisuje w jaki sposób zapewnić reużywalność istniejących komponentów albo implementować nowe komponenty ze zdefiniowaną odpowiedzialnością tworząc system łatwiejszy do utrzymania i zwiększając reużywalność.

TESTOWANIE

- Celami dyscypliny testowania są:
  - Weryfikacja interakcji pomiędzy obiektami.
  - Weryfikacja poprawnej integracji komponentów.
  - Sprawdzenie, czy wszystkie wymagania zostały zaimplementowane w sposób poprawny.
  - Identyfikacja i sprawdzenie, że defekty zostały usunięte przed wdrożeniem oprogramowania.
- Proces RUP proponuje podejście iteracyjne, które oznacza testowanie od początkowych faz projektu. Pozwala to na szybsze wykrywanie defektów i ograniczenie kosztów ich usunięcia. Testy są prowadzone w ramach wymiarów jakości: wiarygodności, funkcjonalności, osiągów pojedynczych aplikacji oraz systemu (performance). RUP opisuje w jaki sposób testować w każdym z tych wymiarów w czasie trwania projektu.

WDROŻENIE

- Celem wdrożenia (deployment) jest udane wytwarzanie dystrybucji produktu i dostarczanie oprogramowania końcowym użytkownikom. Pokrywa ono szeroki zbiór czynności włączając:
  - Produkcja zewnętrznych dystrybucji oprogramowania
  - Pakowanie oprogramowania
  - Dystrybucja oprogramowania
  - Instalacja oprogramowania
  - Zapewnienie pomocy i wsparcia użytkownikom
- Jakkolwiek czynności wdrożenia są skoncentrowane głównie na fazie przekazania (transition), wiele z nich musi być włączone we wcześniejsze fazy w celu przygotowania do wdrożenia na końcu fazy budowy(construction). Procesy (workflows) *Deployment and Environment* w procesie RUP zawierają mniej szczegółów niż inne procesy.

ZARZĄDZANIE ZMIANĄ I KONFIGURACJĄ

- Dyscyplina zarządzania zmianą (change management) w RUP dotyka trzech obszarów:
  - Zarządzania konfiguracją (configuration management) - jest odpowiedzialne za systematyczne strukturalizowanie produktów. Artefakty takie jak dokumenty i modele muszą być wersjonowane (version control) a zmiany muszą być widoczne. W skład zarządzania konfiguracją wchodzi także utrzymywanie rejestru zależności pomiędzy artefaktami, tak, aby wszystkie powiązane części były uaktualniane wraz ze zmianami.
  - Zarządzanie zleceniami zmian(Change request management) - w czasie opracowywania oprogramowania istnieje wiele artefaktów z różnymi wersjami. Zarządzanie polega na trzymaniu rejestru propozycji lub zleceń zmian.
  - Zarządzanie stanem i miarami (Status and measurement management) - zlecenia zmian (change requests) mają stany takie jak nowy, zalogowany, zatwierdzony, przypisany i zakończony. Zlecenia zmian mają także atrybuty takie jak przyczyna (root cause) oraz natura (jak defekt lub rozszerzenie), priorytet itp. Te stany i atrybuty powinny być przechowywane w bazie danych, tak aby umożliwić tworzenie użytecznych raportów na temat postępów prac. Firma Rational posiada produkt, który umożliwia utrzymywanie takiego rejestru ClearQuest. Czynność ta wiąże się z procedurami, które trzeba wykonywać.

ZARZĄDZANIE PROJEKTEM

- Planowanie projektu w RUP występuje na dwóch poziomach - zgrubnym (coarse-grained) zwanym planem faz, który opisuje cały projekt oraz serii szczegółowych planów iteracji, które opisują iteracje.
- Ta dyscyplina skupia się głównie na ważnych aspektach iteracyjnego procesu wytwarzania oprogramowania. **Nie próbuje objąć** natomiast wszystkich aspektów zarządzania projektami, na przykład:
  - Zarządzania zespołem: zatrudniania, szkoleń, opieki (coaching)
  - Zarządzania budżetem: definiowania, alokowania itp.
  - Zarządzania umowami ze sprzedawcami i klientami
  - Główne obszary dyscypliny:
  - Zarządzanie ryzykiem
  - Planowanie projektu iteracyjnego, w ramach całego cyklu i pojedynczych iteracji
  - Monitorowanie postępu projektu iteracyjnego, miary

ZARZĄDZANIE PROJEKTEM

- Dyscyplina zarządzania projektem zawiera również inne **Plany** i **Artefakty**, które są używane do kontrolowania projektu i monitorowania jego postępu. Do planów należą:
  - **Plan faz** (The Software Development Plan)
  - **Plan iteracji**
- **Plan faz**

  Każda faza traktowana jest jako projekt, kontrolowany i mierzony poprzez **Software Development Plan** pogrupowany w podzbiór planów kontrolnych:
  - **Plan miar** (Measurement Plan) - definiuje cele pomiarów, skojarzone miary, i proste miary, które będą gromadzone w projekcie w celu monitorowania jego postępu.
  - **Plan zarządzania ryzykiem** (Risk Management Plan) - uszczegóławia w jaki sposób zarządzać ryzykami związanymi z projektem. Wymaga uszczegółowienia zadań zarządzania ryzykami, które będą wykonywane, przypisania do nich odpowiedzialności oraz dodatkowych wymaganych zasobów. W projektach mniejszej skali plan może być powiązany z Software Development Plan.
  - **Lista ryzyk** (Risk list) - posortowana lista znanych i otwartych ryzyk posortowanych według ważności i skojarzonych z akcjami minimalizacji oraz planami awaryjnymi (mitigation and contingency actions).

OPINIE

RUP jest często błędnie postrzegany jako ciężki i kosztowny proces. Tymczasem RUP nie był opracowany, pozycjonowany i promowany jako gotowy proces "prosto z pudełka". Na przystosowywanie procesu do własnych potrzeb pozwala produkt Rational Method Composer. W chwili obecnej jest on rozwijany na bazie produktu Eclipse Process Framework powstającego w ramach Eclipse. W ramach tego projektu udostępniona została za darmo wersja Open Unified Process.

OPINIE

Doświadczenia programistów z użytkowania metodyki RUP są pozytywne i negatywne. Poniżej wypowiedzi programistów, którzy mieli styczność z tą właśnie metodyką programowania:

"Moje doświadczenie z RUP jest takie, że jego nieograniczona dostosowywalność stwarza problemy. Napotykałem przypadki użycia RUP od modelu kaskadowego z iteracjami analitycznymi, do pełnego procesu Agile. Uderzyło mnie to, że promowanie RUP jako pojedynczego procesu doprowadziło do tego, że ludzie mogą zrobić wszystko i nazwać to RUP - co prowadzi do tego, że RUP staje się nic nie znaczącym słowem."

"Moje doświadczenie natomiast jest takie, że muszą istnieć pewne wzorce w kierunku których prace mają podążać. Problem różnej interpretacji RUP, de facto, jest jego zaletą. Negatywne skutki, które zauważamy w aktualnych projektach, wywodzą się z "cięcia kosztów" nie w tym miejscu organizacji projektu - co potrzeba. Brakuje w nim najczęściej zapomnianej roli "Inżyniera procesu wytwórczego". Być może ktoś nazwałby go kierownikiem projektu(Ale zwykle kierownika projektu postrzega się inaczej i ocenia inne kompetencje projektu)."

BIBLIOGRAFIA

- http://pl.wikipedia.org/wiki/RUP

RATIONAL UNIFIED PROCESS

Koniec ☺

Łukasz Kluk