ZESZYTY NAUKOWE
WYDZIAŁU NAUK EKONOMICZNYCH

Tomasz SOBESTIAŃCZYK*

METODYKA RUP JAKO NAJLEPSZE DOPEŁNIENIE ZARZĄDZANIA PROJEKTAMI INFORMATYCZNYMI

Zarys treści: Ta publikacja opisuje metodykę RUP – jej zalety oraz słabości w zarządzaniu projektami IT oraz propaguje metodykę jako kompleksowe podejście do zarządzania projektami informatycznymi.
Słowa kluczowe: RUP, Rational, projekt, proces, metodyka, zarządzanie projektami, proces informatyczny, audyt.

Wprowadzenie

Początki RUP sięgają swoimi korzeniami do oryginalnego modelu spiralnego Barrego Boehma. Podejście to zostało opracowane przez National Software w latach osiemdziesiątych i dziewięćdziesiątych XX wieku.

W roku 2000 Rational Software przejęło szwedzką firmę Objectory AB. Zunifikowany proces RUP (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 opublikowana została w 1998 roku.

Istota metodyki RUP

RUP (Rational Unified Process) jest iteracyjną i przyrostową metodyką wytwarzania oprogramowania opracowaną przez firmę Rational Software Corporation. Jest to w pełni konfigurowalna platforma obsługi procesu tworzenia oprogramowania. Wspomaga zdyscyplinowane podejście do rozwoju oprogramowania. Celem tej metodyki jest produkcja oprogramowania wysokiej jakości w przewidywanym dla klienta czasie i budżecie.

* Uniwersytet Jana Kochanowskiego, Kielce

Proces RUP jest szablonem procesu. Został zaprojektowany w celu przystosowania do charakteru konkretnej organizacji, konkretnego zespołu projektowego lub nawet charakteru konkretnego projektu.

RUP jest procesem iteracyjnym zakładającym budowanie systemu w kolejnych przebiegach. Po zakończeniu iteracji produkowany jest fragment systemu spełniający wymagania klienta, jest on następnie udostępniany. W ten sposób zespół analityczno-projektowy otrzymuje szybko sygnał zwrotny od klienta, który umożliwia bieżącą ocenę ryzyka niepowodzenia projektu jak również pozwala stwierdzić czy zespół analityczno-projektowy dobrze zrozumiał wymagania klienta wobec systemu. W razie wystąpienia problemów zostaną one szybko wykryte i zespół analityczno-projektowy będzie mógł wdrożyć odpowiednie postępowanie zaradcze. Podejście iteracyjne powoduje gwałtowną redukcję ryzyka niepowodzenia projektu już po zakończeniu pierwszej iteracji[1].

Z uwagi na duży nakład na dokumentację projektową (do 50% kosztów), RUP zaliczany jest do tzw. ciężkich metodologii.

Metodyka RUP jest ukształtowana w sposób umożliwiający dopasowanie procesu tworzenia systemu do potrzeb konkretnego przedsięwzięcia na podstawie pełnego spektrum swoich możliwości. W efekcie RUP jest elastyczny – znajduje zastosowanie zarówno przy mniejszych, jak i dużych projektach informatycznych. Jest oparta na doświadczeniach największych firm w branży informatycznej[2].

RUP został wykorzystany jak dotąd przez więcej niż 1000 organizacji (dane z 2000 roku) dla różnych zastosowań, dla małych i dużych projektów:

- telekomunikacja: Ericsson, Alcatel, MCI, TP;
- transport, lotnictwo, obrona: Lockheed-Martin, British Aerospace;
- produkcja: Xerox, Volvo, Intel;
- finance: Visa, Merrill Lynch, Schwab;
- inne: Ernst & Young, Oracle, Deloitte & Touche.

Więcej niż 50% użytkowników albo już wykorzystywało RUP do e-biznesu albo planuje wykorzystać w najbliższej przyszłości. Metodyka Rational Unified Process jest oparta na pomysłach i doświadczeniu najlepszych firm w branży informatycznej, partnerów i tysięcy rzeczywistych projektów, doskonale połączonych w efektywny zestaw najlepszych praktyk, schematów przepływu pracy i artefaktów umożliwiających iteracyjne tworzenie oprogramowania. RUP szybko staje się standardem iteracyjnego tworzenia oprogramowania. Korzysta z niego „wielka piątka" światowych integratorów

[1] http://www.cyberwaretechnology.pl, 2012.
[2] http://math.uni.lodz.pl/~robpleb/wyklad10_14.pdf, 2012.

systemów, 8 z 10 największych banków USA. Tysiące projektów na świecie opartych jest na RUP[3].

Niektóre z organizacji – ściśle w oparciu o RUP – budowały własny proces dostosowany do ich specyficznych potrzeb, a niektóre wykorzystywały RUP bardziej nieformalnie traktując jak repozytorium rad, wytycznych i szablonów. RUP to także szkielet, rama (framework), która może być przystosowana (również rozszerzana) stosownie do specyficznych potrzeb adaptującej ją organizacji. RUP oparty został o zbiór sześciu najlepszych praktyk: iteracyjny rozwój, zarządzanie wymaganiami, architektura oparta o komponenty, wizualne modelowanie, systematyczna weryfikacja jakości i zarządzanie zmianami. Eliminują pierwotne przyczyny problemów z oprogramowaniem, pomagając uniknąć typowych pułapek związanych z użyciem nowych technologii i narzędzi. Używając sprawdzonych metod i wspólnie korzystając z jednego, spójnego procesu, zespół programistów może wydajniej komunikować się i pracować. Jedną z głównych metod RUP jest pojęcie tworzenia iteracyjnego[4]. RUP obejmuje całość czynności składających się na proces wytwarzania oprogramowania. Poprzez proces rozumie się zbiór częściowo uporządkowanych kroków mających na celu osiągnięcie pewnego celu. W inżynierii oprogramowania celem tym jest wytworzenie oprogramowania lub ulepszenie już istniejącego. RUP ma na celu usystematyzowanie czynności związanych z produkcją oprogramowania, a także rozdział tych czynności pomiędzy członków zespołu zajmującego się wytwarzaniem oprogramowania. RUP ma za zadanie zapewnić odpowiedni standard wytworzonego oprogramowania, które jest realizowane według ustalonego harmonogramu prac i w ramach przewidzianych środków finansowych. Dzięki bogatym możliwościom konfiguracji, RUP może zostać przystosowany do potrzeb konkretnego przedsiębiorstwa bądź projektu[5]. Za użyciem RUP w trakcie wytwarzania oprogramowania przemawiają przede wszystkim: wprowadzenie jednolitej metodyki wytwarzania oprogramowania, formalizacja zagadnień związanych z procesem wytwarzania oprogramowania. Ryzyko niepowodzenia przedsięwzięcia zostaje zmniejszone poprzez: realizację prac według harmonogramu przy jasno określonych środkach finansowych przeznaczonych na ten cel, określenie osób odpowiedzialnych za poszczególne etapy produkcji, możliwość śledzenia postępów nad pracami. RUP pozwala całkowicie lub częściowo wyeliminować główne przyczyny stojące za fiaskiem projektów

[3]http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.

[4]http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.

[5] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.

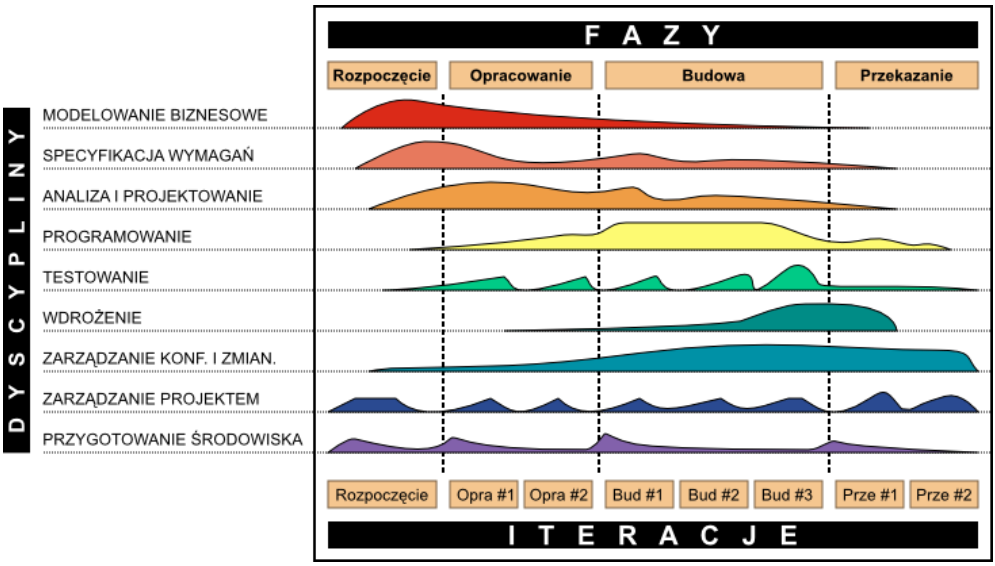
informatycznych: niezrozumienie potrzeb użytkownika, brak elastyczności w uwzględnieniu zmieniających się wymagań użytkownika, problemy z integracją poszczególnych części systemu w całość, problemy z utrzymaniem i skalowalnością systemu, późne wykrycie poważnych błędów projektowych w oprogramowaniu, niedostateczna jakość oprogramowania, brak koordynacji pomiędzy osobami i zespołami pracującymi nad wytwarzaniem oprogramowania[6]. Rational Unified Process (RUP) to także nazwa oprogramowania, opracowanego przez Rational Software (obecnie dostępnego w IBM). Dostarczany jest jako gotowy produkt będący zbiorem dokumentacji wyposażonym w interaktywną przeglądarkę. Aplikacja RUP jest zintegrowana z pozostałymi składnikami wchodzącymi w skład pakietu Rational, jest w pełni konfigurowalna oraz na bieżąco aktualizowana przez producenta. Oprócz dokumentacji procesu wytwarzania, w aplikacji zawarto również obszerną dokumentację dla całego pakietu Rational wraz z przykładowymi projektami opracowanymi w w/w pakiecie. Rational Unified Process – jest hipertekstową bazą wiedzy, która dostarcza metodykę pracy nad projektem w postaci wskazówek, wzorców i nauczycieli narzędzi. Baza wiedzy została zaprojektowana pod kątem łatwości nauki. Pozwala na stopniowe budowanie wiedzy zespołu i uzyskanie wszystkich korzyści z zastosowania podejścia obiektowego i notacji UML. Rational Unified Process organizuje projekty w ramach dziedzin i etapów, z których każdy składa się z jednej lub wielu iteracji. Dzięki podejściu iteracyjnemu nacisk na poszczególne schematy przepływu pracy zmienia się na kolejnych etapach cyklu istnienia produktu. Dzięki możliwości ilustrowania postępów pracy i częstemu tworzeniu wersji wykonywalnych, podejście iteracyjne umożliwia wczesne i ciągłe eliminowanie ryzyka.

Struktura RUP

Strukturę RUP można analizować z dwóch perspektyw, zwanych tu „wymiarami”:

- wymiar statyczny, związany ze statycznymi aspektami procesu, takimi jak np.: dyscypliny, aktywności, artefakty i role. Wymiar statyczny procesu jest reprezentowany przez oś pionową na poniższym rysunku. Na osi pionowej zostały oznaczone główne przepływy prac, grupujące aktywności zgodnie z ich wewnętrzną naturą;
- wymiar dynamiczny, reprezentujący aspekty dynamiczne procesu i opisywany w terminach, takich jak: fazy, iteracje i kamienie milowe. Oś pozioma rysunku reprezentująca ten wymiar odzwierciedla upływ czasu.

[6] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.



Rysunek 1. Struktura RUP
Źródło: http://www.iicmagazine.pl, 2012.

RUP jest metodyką tworzenia oprogramowania powiązaną ze spiralnym modelem cyklu życia oprogramowania oraz z wcześniejszymi metodami, które legły u podstaw powstania języka UML (OMT, OOSE). Definiuje szkielet postępowania, który należy dostosować do uwarunkowań konkretnego projektu programistycznego. Powstał na bazie analizy najczęstszych przyczyn niepowodzeń istniejących procesów wytwarzania oprogramowania. Oparty jest o pewne podstawowe zasady oraz fazy wytwarzania oprogramowania[7].

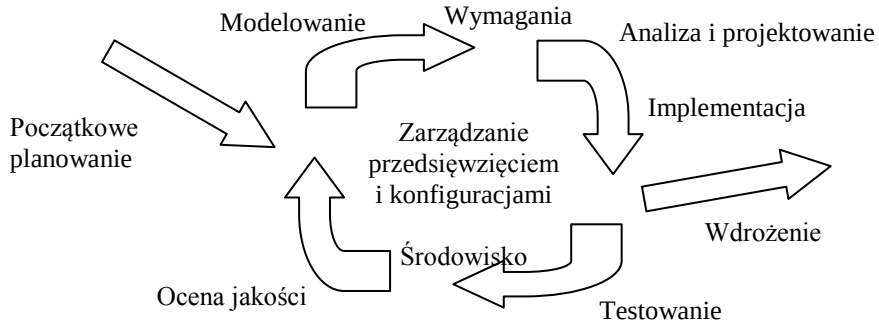
Zasady RUP:

- iteracyjne i przyrostowe tworzenie oprogramowania; sterowane ryzykiem i priorytetami, ułatwiające integrację całości kodu i dostosowanie do zmieniających się wymagań.

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

[7] http://www.brasil.cel.agh.edu.pl, 2012.

opinie od odbiorców oprogramowania a programistom pozwalamy skupić się na węższej dziedzinie.



Rysunek 2. Iteracyjność prac
Źródło: http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.

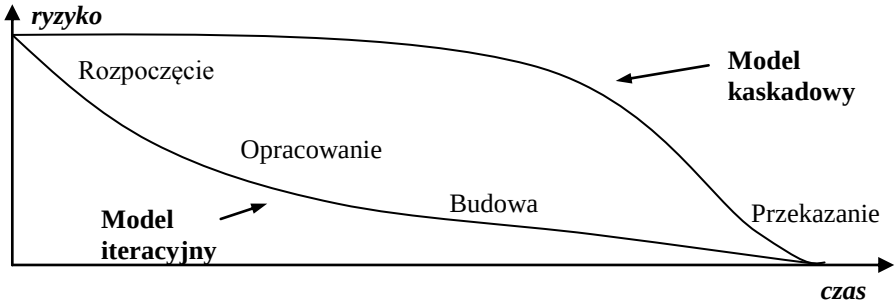
RUP używa podejścia iteracyjnego i przyrostowego z następujących powodów [Rys. 2]:

- integracja oprogramowania, robiona krok po kroku podczas wytwarzania oprogramowania, ogranicza go do mniejszej liczby elementów;
- integracja jest prostsza i mniej kosztowna;
- składowe oprogramowania są projektowane oddzielnie i łatwiej użyć je ponownie;
- łatwiej wykrywać zmiany wymagań i łatwiej nimi zarządzać;
- zagrożenia identyfikowane i atakowane są wcześnie, ponieważ każda iteracja pozwala wykryć kolejne zagrożenia;
- w iteracjach ulepszana jest architektura oprogramowania.

Głównymi czynnikami przemawiającymi za podejściem iteracyjnym w wytwarzaniu oprogramowania są:

- elastyczność w przypadku zmian wymagań lub pojawieniu się nowych wymagań jakie powinien spełniać produkt;
- redukcja ryzyka niepowodzenia projektu;
- integracja poszczególnych części systemu przebiega stopniowo, podzielona jest na kilka etapów w odróżnieniu od metod typu „big bang”;
- ułatwienie identyfikacji reużywalnych składników systemu;
- elastyczność przy zmianie planów strategicznych odnośnie produktu;
- zrównoleglenie prac poszczególnych zespołów pracujących nad projektem;

- zwiększenie jakości wytwarzanego oprogramowania;
- ciągłego usprawniania procesu wytwarzania oprogramowania[8].



Rysunek 3. Ryzyko niepowodzenia projektu a model wytwarzania oprogramowania
Źródło: http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.

1. Zarządzanie wymaganiami; we współpracy z klientem i w oparciu o przypadki użycia.

Zarządzanie wymaganiami to usystematyzowane podejście do wyszukiwania, organizowania i monitorowania zmian wymagań do tworzonego oprogramowania. Główną trudnością w zarządzaniu wymaganiami jest ich dynamiczny charakter. Wymagania bowiem ulegają zmianom, a mianowicie w całym cyklu prac nad systemem dodawane są nowe. Z wyjątkiem najbardziej trywialnych projektów informatycznych, określenie wszystkich wymagań przed rozpoczęciem prac projektowych jest praktycznie niemożliwe[9]. Określanie wymagań ma na celu: lepszą kontrolę złożonych projektów, uzyskanie produktu o lepszej jakości, zwiększenie satysfakcji klienta, poprawę komunikacji w zespole projektowym.

Wymagania stawiane przed systemem informatycznym dzielone są na funkcjonalne i niefunkcjonalne. Wymagania funkcjonalne określają jakie wymogi musi spełniać system bez uwzględnienia dodatkowych ograniczeń nałożonych na niego. Wymagania te określają zachowanie systemu przy wyspecyfikowaniu dla niego warunków wejściowych i wyjściowych[10].Wymagania, które nie kwalifikują się do zbioru wymagań funkcjonalnych, nazywane są wymaganiami niefunkcjonalnymi. Dzielą się one na cztery grupy:

- użytkowanie systemu, m.in. ergonomiczność, łatwość pracy, wymogi odnośnie dokumentacji;

[8] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.
[9] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.
[10] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.

- niezawodność systemu, m.in. dopuszczalny procent błędów w systemie, dokładność obliczeń, przewidywalność zachowania;
- wydajność systemu, m.in. szybkość, czas reakcji, stopień zużycia zasobów systemu operacyjnego;
- utrzymanie i konserwacja systemu. Są to przede wszystkim wymogi stawiane procesowi wytwarzania oprogramowania, np. ograniczenia odnośnie języka implementacji systemu.

2. Stosowanie architektury opartej na komponentach.

W metodyce RUP opracowanie architektury systemu ma miejsce w początkowych fazach cyklu pracy nad oprogramowaniem. Dąży się do stworzenia działającego systemu z częściową implementacją jego funkcji. Pozwala to na wcześniejsze zidentyfikowanie możliwości systemu w zakresie ryzyka, zawodności, przepustowości, pojemności itp. Funkcjonalność systemu jest uzupełniania w fazie budowy. W kolejnych iteracjach budowa systemu ewoluuje.

RUP opiera się na koncepcji projektowania architektury systemu zwanej „4+1". Koncepcja ta polega na przedstawieniu architektury systemu z pięciu perspektyw: przypadków użycia, projektowej, implementacyjnej, procesów, dostarczenia. 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[11].

3. Graficzne modelowanie oprogramowania; różne perspektywy spojrzenia na system, użycie UML.

W RUP do modelowania systemu informatycznego wykorzystywany jest UML (ang. Unified Modeling Language). UML jest graficznym językiem modelowania służącym do wizualizacji, specyfikacji, dokumentacji jednostek

[11] http://elartu.tntu.edu.ua/bitstream/123456789/1455/19/Rational_Unified%20Process__Katarzyna_Pcion.pdf, 2012.

informacji występujących w procesie produkcji systemu informatycznego. Umożliwia standaryzację sposobu opracowywania przekrojów systemu, obejmujących obiekty pojęciowe, takie jak procesy przedsiębiorstwa i funkcje systemowe, a także obiekty konkretne, takie jak klasy zaprogramowane w ustalonym języku, schematy baz danych i komponenty programowe nadające się do ponownego użycia[12].

Przez modelowanie osiągane zostają następujące cele:

- łatwiejsze zrozumienie systemu, który ma zostać wyprodukowany;
- rozważenie alternatyw projektowych systemu niskim kosztem;
- wyspecyfikowanie struktury i zachowania systemu stanowiące fundament dla jego implementacji;
- dokumentacja decyzji podjętych przez osoby zaangażowane w produkcję oprogramowania;
- dokładniejsza specyfikacja wymagań odnośnie powstającego systemu.

Z RUP można wyróżnić zbiór modeli, które obejmują wszystkie ważne aspekty dotyczące obrazowania, specyfikowania, tworzenia i dokumentowania systemu informatycznego. Należą do nich m.in. modele: przypadków użycia, analizy, projektu, procesów, wdrożenia, implementacji oraz testów.

4. Kontrola i weryfikacja jakości oprogramowania przez cały czas procesu wytwarzania.

RUP koncentruje się na dwóch aspektach kontroli jakości: oprogramowaniu oraz procesie wytwórczym.

Pierwszy aspekt obejmuje kontrolę jakości produktu wytworzonego z pomocą RUP – działającej wersji systemu, instrukcji obsługi, materiałów treningowych i dydaktycznych. Kontrolowane są:

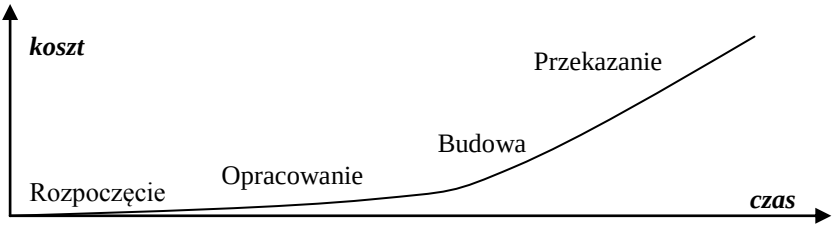
- niezawodność dostarczonego kodu wykonywalnego – jego podatność i odporność na błędy w trakcie wykonania m.in. błędy przydziału pamięci, zawieszanie się systemu itp.;
- spełnienie przez system wymagań funkcjonalnych;
- wydajność – czy system zachowuje się zgodnie z oczekiwaniami i założeniami w różnych warunkach pracy;
- spójność poszczególnych elementów systemu m.in.: poprawność terminologii, semantyki;
- zgodność systemu z wytycznymi procesu produkcji[13].

Drugi aspekt kontroli jakości obejmuje dostosowanie procesu wytwarzania do produkcji konkretnego oprogramowania. Kryteriami pomiaru jakości są:

- zarządzanie opłacalnością i zasobami;

[12] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.
[13] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.

- zarządzanie ryzykiem i jego identyfikacja;
- spełnienie ograniczeń budżetowych, czasowych oraz jakościowych;
- identyfikacja nowych elementów, które mają wpływ na ulepszenie procesu produkcji[14].

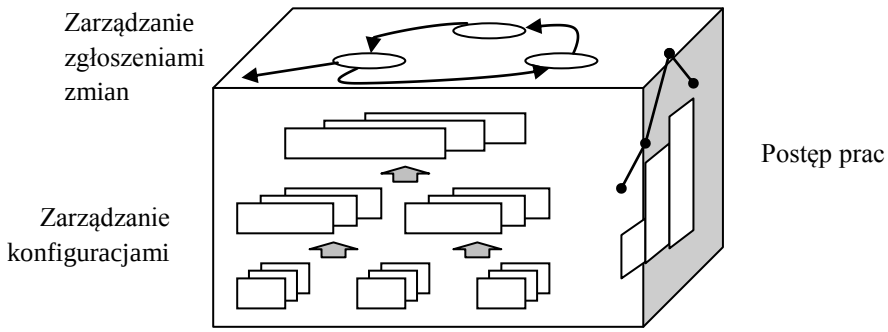


Rysunek 4. Koszty związane z poprawą błędów w systemie na poszczególnych etapach jego wytwarzania
Źródło: http://www.sun.aei.polsl.pl, 2012.

5. Zarządzanie zmianami w oprogramowaniu.

Kontrola zmian ma za zadanie utrzymywanie integracji pomiędzy składnikami tworzonego systemu. Kontrola zmian obejmuje trzy zagadnienia [Rys. 5]:

- zarządzanie konfiguracją. Ma za zadanie kontrolować i zarządzać wersjami poszczególnych elementów systemu,
- zarządzanie zgłoszeniami dotyczącymi zmian w systemie,
- ocenę stanu zaawansowanie prac nad systemem[15].



Rysunek 5. Trójwymiarowy aspekt konfiguracji i kontroli zmian systemu
Źródło: http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.

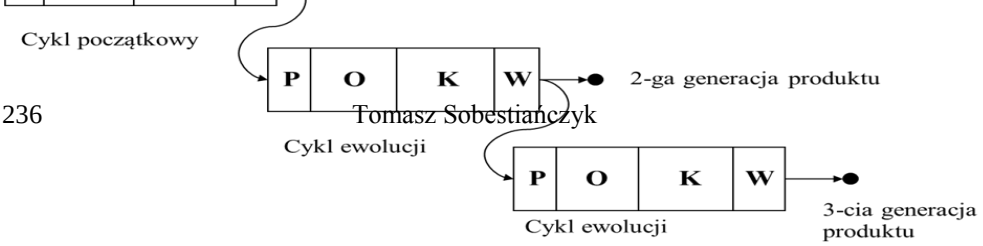
[14] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.
[15] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.

Kontrola zmian przyczynia się do zapewnienia spójności systemu oraz gwarantuje, że osoby zaangażowane w cykl produkcyjny będą poinformowane o aktualnym stanie wytwarzanego oprogramowania.

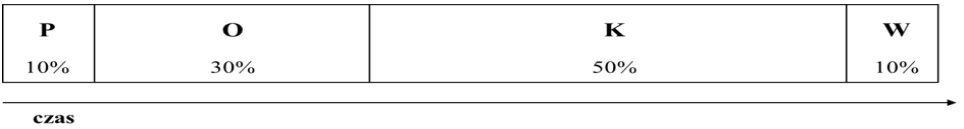
Cykl produkcji oprogramowania według RUP podzielony został na cztery sekwencyjne fazy. Po zakończeniu każdej z faz przeprowadzana jest jej weryfikacja, której pozytywny wynik pozwala rozpocząć kolejną fazę prac. Czas i zasoby przeznaczone na pracę nad poszczególnymi fazami projektu są zróżnicowane. Przejście przez wszystkie fazy projektu kończy pojedynczą iterację prac nad systemem, w trakcie której powstaje wersja działającego systemu. W trakcie kolejnej iteracji wersja ta jest udoskonalana i wyposażana w dodatkowe funkcjonalności.

Fazy składające się na proces wytwórczy oprogramowania nie rozkładają się równomiernie ani pod względem czasochłonności, ani pod względem potrzebnych do ich realizacji zasobów. Lata doświadczeń metodyków wykazały, że w czasie projektowania systemów na realizację poszczególnych faz należy zarezerwować odpowiednią ilość czasu. W swoim artykule David West określa, że na fazę Rozpoczęcia należy przeznaczyć 10% czasu projektu. Faza Opracowania wymaga 30% czasu. Najwięcej czasu, bo aż około 50%, potrzeba na fazę Budowy. Ostanie 10% procent należy przeznaczyć na fazę Przekazania. Odmiennie od rozkładu czasu wygląda rozkład potrzebnych zasobów do realizacji faz. I tak faza Rozpoczęcia wymaga użycia około 10% zasobów. Etap opracowania pochłania zazwyczaj około 20% środków. Najbardziej zasobożerny jest etap budowy, który na swoje potrzeby zużywa około 60% zasobów. Na ostatnią fazę Przekazania zostaje około 10% środków[16]. Cztery fazy (P, O, K, W) są ułożone w następujące po sobie cykle: cykl początkowy i kolejne cykle ewolucji. Cykle mogą nieznacznie zachodzić na siebie. Cykl początkowy kończy się wytworzeniem pierwszej wersji produktu; następne cykle służące wytwarzaniu kolejnych generacji produktu są realizowane przez powtarzanie czterech faz, z różnym naciskiem na różne fazy [Rys. 6].

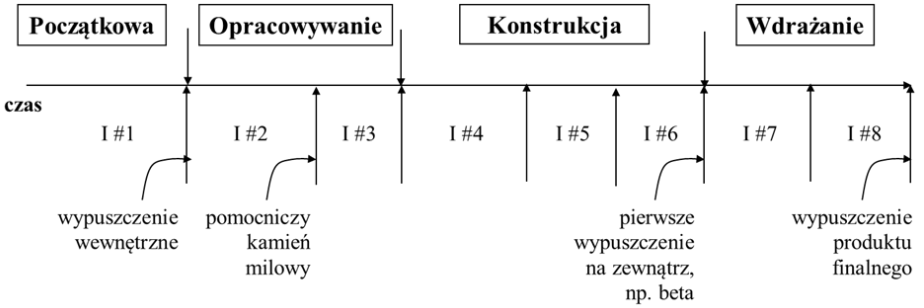
[16] http://www.michalwolski.pl/page/42/, 2012.



Rysunek 6. Cykle wytwarzania oprogramowania
Źródło: http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.



Rysunek 7. Typowy rozkład czasu dla cyklu początkowego
Źródło: http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.



Rysunek 8. Cykl wraz z iteracjami, wytworzonymi produktami i kamieniami milowymi
Źródło: http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.

Przedstawione liczby są tylko orientacyjnymi danymi, gdyż każdy projekt z racji swej niepowtarzalności może mieć inny rozkład faz w czasie i wymagać dla każdego etapu innych zasobów.

Fazą (ang. phase) w metodyce RUP jest okres między kolejnymi punktami przeglądu cyklu iteracyjno-przyrostowego, w którym zrealizowano niezbędne czynności i opracowano adekwatne artefakty. Metodyka RUP wprowadza cztery fazy. Fazy projektu RUP:

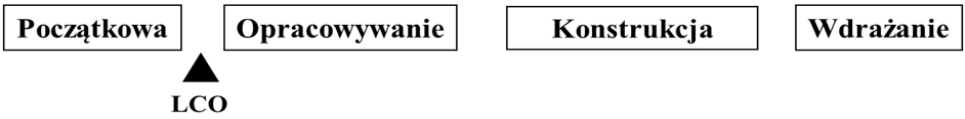
I. faza początkowa (inception) – wstępne określenie wymagań, ryzyka, kosztu, harmonogramu, a także architektury systemu, wypracowanie ogólnej wizji przedsięwzięcia oraz osiągnięcie zrozumienia i akceptacji projektu ze strony wszystkich jego uczestników[17]. Podstawowe zadanie:

[17] https://icis.pcz.pl/~mciesiel/psi/PSI%20wyklad7.pdf, 2012.

- osiągnąć konsensus wśród uczestników projektu;
- ustalić kontekst, zakres odpowiedzialności (najważniejsze wymagania), ograniczenia oraz kryteria akceptacji dla produktu finalnego;
- wyróżnić krytyczne – z punktu widzenia projektu architektury – przypadki użycia (główne scenariusze);
- zaproponować (nawet zademonstrować) architekturę umożliwiającą ich realizację;
- oszacować ryzyka: źródła i prawdopodobieństwa wystąpienia;
- opracować przypadek biznesowy, przygotować plan zarządzania ryzykiem, harmonogram, propozycje kadrowe – dla całości projektu (bardziej szczegółowo potraktować fazę opracowywania, następną w kolejności);
- w oparciu o analizę planów, kosztów i zasobów podjąć decyzje typu: co budować od nowa/ co kupić/ co ponownie wykorzystać[18]?

Podstawowe artefakty wytwarzane w fazie początkowej: dokument wizji – główne wymagania, własności i ograniczenia; przegląd przypadków użycia (w postaci listy); słownik pojęć; przypadek biznesowy (kontekst biznesowy, kryteria sukcesu – dochód, rozpoznanie rynku, itd., prognoza finansowa); szacunki dla ryzyka; plan projektu, z uwzględnieniem faz oraz iteracji[19].

Inne artefakty, wytwarzane w fazie początkowej: model przypadków użycia (10-20% całości – dla cyklu początkowego), model dziedziny, model biznesowy (o ile potrzeba), specyfikacje procesów, jeden lub kilka prototypów.



Rysunek 9. Kamień milowy LCO
Źródło: http://www.si.pjwstk.edu.pl, 2012.

Kryteriami oceny są: stopień zrozumienia wymagań; rzetelność szacunków dla kosztów, priorytetów, ryzyk, procesów, itp.; jakość pierwotnej architektury (prototypu); wydatki rzeczywiste kontra wydatki planowane.

II. faza opracowania (elaboration) – ustalenie wymagań (większości przypadków użycia), architektury systemu oraz planu całego procesu wytwarzania systemu.

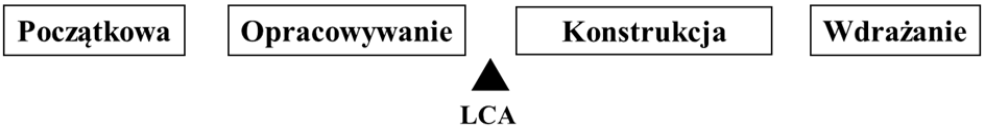
Podstawowe aktywności:

[18] http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.
[19] http://pjwstk.mykhi.org/0sem/PRI/RUP%2004.ppt, 2012.

- solidne zrozumienie krytycznych przypadków użycia – stanowiących bazę dla decyzji architektonicznych i dla planowania;
- przygotowanie procesów i infrastruktury niezbędnej dla ich realizacji;
- wypracowanie fundamentów dla architektury: selekcja komponentów niezbędnych dla realizacji głównych scenariuszy (być może trzeba będzie dokonywać zmian w architekturze lub w wymaganiach);
- decyzje, co budować od nowa/co kupić/co ponownie wykorzystać tak przemyślane, by dało się oszacować koszt i czas fazy konstrukcji[20].

Artefakty:

- model przypadków użycia (kompletny w 80%), co oznacza, że: wszystkie przypadki użycia znajdują się w przeglądzie, którego opracowanie rozpoczęto w poprzedniej fazie, wszyscy aktorzy są zidentyfikowani, dla większości przypadków są opracowane szczegółowe specyfikacje;
- dodatkowe wymagania (funkcjonalne i niefunkcjonalne nie ujęte w żadnym z przypadków użycia);
- opis architektury;
- prototyp;
- plan dla całości projektu (z uwzględnieniem iteracji i kryteriów przechodzenia między nimi);
- specyfikacje procesów, które będą wykorzystywane;
- wstępną wersję podręcznika dla użytkownika (opcjonalnie)[21].



Rysunek 10. Kamień milowy LCA
Źródło: http://www.si.pjwstk.edu.pl, 2012.

Kryteria oceny:

- czy wizja produktu jest stabilna?
- czy architektura jest stabilna?
- czy zidentyfikowano i poprawnie rozwiązano problemy związane z głównymi ryzykami (bazując na demonstracji oprogramowania w oparciu o zbudowany prototyp)?
- czy plan dla fazy konstrukcji jest odpowiedni i wystarczająco szczegółowy?

[20] http://pjwstk.mykhi.org/0sem/PRI/RUP%2004.ppt, 2012.
[21] http://pjwstk.mykhi.org/0sem/PRI/RUP%2004.ppt, 2012.

- czy wszyscy uczestnicy projektu są zgodni, że o ile zostanie wykonany aktualny plan w oparciu o aktualną architekturę, to aktualna wizja produktu finalnego jest realna?
- czy stosunek aktualnych wydatków do wydatków zaplanowanych jest akceptowalny?

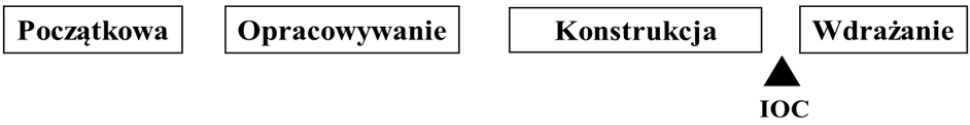
Jeśli projekt nie przejdzie pomyślnie kamienia milowego LCA, prace mogą zostać zakończone, a co najmniej poważnie przemyślane.

III. faza konstrukcji (construction) – tworzenie systemu (kolejnych komponentów), w trakcie następuje oddanie pierwszej (i być może dalszych) wersji użytkownikowi. Podstawowe aktywności:

- zarządzanie zasobami, optymalizacja procesów;
- budowa, rozwój i testowanie komponentów w oparciu o zdefiniowane kryteria;
- budowa użytecznej wersji produktu (alfa, beta czy innej) – zgodnej z wizją produktu finalnego.

Artefakty:

- produkt, zintegrowany na odpowiedniej platformie;
- podręcznik użytkownika;
- opis aktualnego wypuszczenia[22].



Rysunek 11. Kamień milowy IOC
Źródło: http://www.si.pjwstk.edu.pl, 2012.

Kryteria oceny:

- czy wypuszczany produkt jest dostatecznie stabilny i dojrzały, by można było przekazać go użytkownikowi?
- czy wszyscy uczestnicy projektu są na to gotowi?
- czy poziom wydatków jest akceptowalny w porównaniu do wydatków zaplanowanych?

IV. faza przekazania (transition) – system jest przekazywany użytkownikowi, wdrażany, szkoleni są pracownicy obsługi systemu, następuje walidacja i końcowe sprawdzenie jakości.

Kryteria oceny:

- czy użytkownik jest usatysfakcjonowany?
- czy poziom wydatków jest akceptowalny w porównaniu do wydatków

[22] http://pjwstk.mykhi.org/0sem/PRI/RUP%2004.ppt, 2012.

zaplanowanych?

Czwarty, ważny kamień milowy w realizacji projektu oznacza taki punkt w czasie, gdy trzeba zadecydować czy oczekiwania użytkownika zostały zaspokojone i czy należy rozpoczynać kolejny cykl ewolucji.

W niektórych przypadkach, czwarty kamień milowy zbiega się z końcem fazy początkowej następnego cyklu.

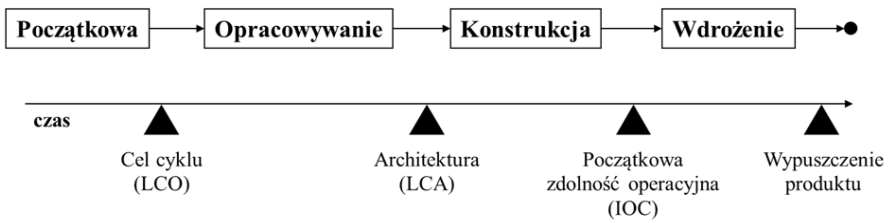
Wymienione powyżej fazy składają się z iteracji, czyli działań produkcyjnych zmierzających do stworzenia funkcjonalnej części systemu. Przejście przez wszystkie fazy stanowi cykl wytworzenia oprogramowania. Każde przejście przez wymienione fazy daje nową generację oprogramowania.

System informatyczny jest rozwijany w kolejnych iteracjach (ang. iterations), występujących w ramach każdej z wymienionych faz. Przejście pomiędzy iteracjami poprzedza przyrostowa integracja artefaktów otrzymanych we wszystkich dotychczasowych iteracjach. Iteracja w metodyce RUP to pojedynczy cykl w ramach fazy, polegający na realizacji czynności poszczególnych dyscyplin; jej efektem jest kolejny przyrost systemu[23].

W jednej iteracji zachodzi:

- zdefiniowanie celów iteracji i jej zaplanowanie,
- wymagania są zdefiniowane i przeanalizowane,
- projekt zostaje poszerzony,
- kod jest napisany i przetestowany,
- system jest integrowany i testowany,
- iteracja jest zamykana i wnioski z niej przydają się w kolejnym etapie.

Z perspektywy zarządzania projektem: trzeba zorganizować sekwencję iteracji – odpowiednio do krótkoterminowych celów. Postęp powinien być mierzony, np.: ilością zrealizowanych use-case, przeprowadzonych testów czy wyeliminowanych ryzyk. Należy zdefiniować punkty w czasie, kiedy będą podejmowane decyzje typu: kontynuujemy, kończymy prace czy też zmieniamy kurs. Takie punkty w RUP noszą nazwę kamieni milowych[24].



Rysunek 12: Kamienie milowe

[23] https://icis.pcz.pl/~mciesiel/psi/PSI%20wyklad7.pdf, 2012.
[24] http://pjwstk.mykhi.org/0sem/PRI/RUP%2004.ppt, 2012.

Źródło: http://www.si.pjwstk.edu.pl, 2012.

Kamienie milowe:

- Faza początkowa (P): specyfikacja wizji produktu finalnego i związanego z nim przypadku biznesowego; określenie celów projektu (kamień milowy: LCO);
- Faza opracowywania (O): planowanie niezbędnych aktywności wraz z określeniem niezbędnych dla ich zrealizowania zasobów; specyfikacja architektury (kamień milowy: LCA);
- Faza konstrukcji (K): budowa produktu; rozwijanie wizji, architektury, planów – do momentu, gdy produkt będzie gotowy do oddania dla użytkowników (kamień milowy: IOC);
- Faza wdrożenia (W): wdrożenie u użytkownika; trening i utrzymanie (kamień milowy: Wypuszczenie produktu).

Dyscyplina (ang. discipline), jest kolekcją powiązanych czynności, artefaktów, ról oraz przepływów pracy odpowiadających tematycznie głównym obszarom tworzenia systemów informatycznych. Dyscypliny podstawowe stanowią rdzeń procesu tworzenia systemu. RUP wyróżnia także „dyscypliny", grupy zadań wykonywanych przez pracowników[25]:

- modelowanie biznesowe (ang. – business modeling) – zawiera wizję nowej, docelowej organizacji, definicji występujących w ramach jej procesów, ról i zakresów odpowiedzialności. Informacje te przedstawia się w postaci biznesowych diagramów;
- specyfikacja wymagań (ang. requirements) – oznacza opracowanie wizji systemu, modelu przypadków użycia i zdefiniowanie wymagań niefunkcjonalnych;
- analiza i projektowanie (ang. analysis and design) – zawiera analizę i projekt systemu z wykorzystaniem całego spektrum diagramów UML;
- implementacja (ang. implementation) – pozwala na opracowanie kodu źródłowego w wybranym języku programowania oraz kompilację kodu i integrację komponentów;
- testowanie (ang. test) – oznacza planowanie testów oraz ocenę systemu poprzez wykonanie szeregu testów;
- wdrożenie (ang. deployment) – dotyczy instalacji oprogramowania systemu i formalną akceptację kolejnych wersji systemu przez klienta czy użytkownika.

Dyscypliny wspomagające realizują funkcje zarządcze i konfiguracyjne w procesie tworzenia systemu. Można do nich zaliczyć:

- zarządzanie konfiguracją i zmianami (ang. configuration and change

[25] https://icis.pcz.pl/~mciesiel/psi/PSI%20wyklad7.pdf, 2012.

menagement) – obejmuje kontrole wersji artefaktów opracowanych podczas kolejnych iteracji;
- zarządzanie projektem (zarządzanie ryzykiem, planowanie iteracji, monitorowanie postępów), (ang. project menagement) – oznacza planowanie i kontrolę realizacji projektu;
- organizacja środowiska (m.in. narzędzi), (ang. environment) – obejmuje przygotowanie infrastruktury dla skutecznej realizacji projektu[26].

Rola oznacza obowiązki i kompetencje osoby lub zespołu zaangażowanego w proces tworzenia systemu informatycznego lub jego wycinka. W wyniku realizacji roli powstają oczekiwane artefakty. Spektrum ról, odpowiadających często współczesnym zawodom informatycznym, jest bardzo szerokie i wiąże się z możliwościami zaawansowanych technologii informatycznych. Dana osoba może mieć przypisanych kilka ról w zależności od kontekstu, w którym występuje, np. osoba zajmująca się pracami projektowymi może brać udział również w implementacji kodu[27].

Przykłady ról:
- Analityk: odpowiada za określenie wymagań. Buduje model use-case, specyfikując funkcjonalność i ograniczenia nakładane na oprogramowanie;
- Projektant: specyfikuje odpowiedzialności, operacje, atrybuty i związki jednej lub kilku klas, a następnie dopasowuje projekt klas(y) do środowiska implementacji;
- Projektant testów: odpowiada za planowanie (plan testów), model testów, implementację i ewaluację pokrycia testów, wyników i efektywności.

Aktywność specyfikuje jednostkową pracę, którą pracownik ma do wykonania i która ma być uwieczniona rezultatem „znaczącym" w kontekście projektu. Każda aktywność musi mieć jasno określony cel. Rozmiar aktywności jest określany za pomocą czasu potrzebnego na jej wykonanie: zazwyczaj kilka godzin do kilku dni. Aktywność z reguły związana jest z jednym pracownikiem i pracą nad jednym lub niewielką liczbą artefaktów. Może stanowić element ponownego użycia dla procesów planowania i rozwoju. Związana z danym artefaktem może być wielokrotnie powtarzana, szczególnie gdy przechodzi się do kolejnych iteracji w procesie rozszerzania i udoskonalania produktu[28].

Przykłady aktywności:
- planuj iterację – wykonywane przez pracownika Kierownik projektu;
- znajdź przypadki użycia – wykonywane przez pracownika Analityk;
- zrób przegląd projektu – wykonywane przez pracownika Recenzent

[26] http://www.math.uni.lodz.pl, 2012.
[27] https://icis.pcz.pl/~mciesiel/psi/PSI%20wyklad7.pdf, 2012.
[28] http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.

projektu;
- przeprowadź test wydajnościowy – wykonywane przez pracownika Tester wydajności.

Artefakt jest jednostką informacji wytworzoną przez pewien proces. Artefakty służą jako dane wejściowe i wyjściowe dla działań. Jednostki z przypisanymi im rolami wykorzystują wejściowe artefakty do swych działań, w rezultacie czego wytwarzane są nowe artefakty lub modyfikowane już istniejące. Przykładem artefaktów mogą być: dokument wymagań użytkownika, diagram klauzul użycia, komponent oprogramowania[29].

RUP promuje ideę: „każda porcja informacji związana jest z konkretną, odpowiedzialną za nią osobą", co oznacza przyporządkowanie artefaktów do osób, w roli ich „właścicieli". Inne osoby mogą wykorzystywać artefakt, ale aktualizacja zawsze wymaga zgody właściciela.

Artefakty mogą przyjmować różną postać, np.: model, np. model use-case czy też model pojęciowy czy logiczny; element modelu, np. przypadek użycia czy klasa; dokument, np. przypadek biznesowy czy dokument specyfikacji architektury; kod źródłowy; kod wynikowy.

Artefakty mogą być złożone z innych artefaktów (zagnieżdżane), np. model pojęciowy składa się z wielu klas. Artefakty w RUP zostały pogrupowane w pięć kategorii: związane z biznesem i zarządzaniem projektem; artefakty związane z planowaniem projektu (SDP Software Development Plan), przypadki biznesowe, wystąpienie procesu dla danego projektu, itd., artefakty operacyjne, np. artefakty związane z wypuszczaniem produktu czy ewaluacją statusu, dokumenty wdrożeniowe, itd.; związane z wymaganiami, dokument wizji, wymagania w postaci potrzeb uczestników projektu, gdzie przez uczestnika projektu rozumie się zarówno klienta, użytkownika końcowego, jak i członka zespołu projektowego, model przypadków użycia wraz z uzupełniającą specyfikacją, model biznesowy, o ile jest niezbędny dla zrozumienia procesów biznesowych wspieranych przez oprogramowanie; związane z projektowaniem, modelu projektowego, opisu architektury, modelu testów; związane z implementacją, kod źródłowy, kod wynikowy, pliki z danymi i pliki potrzebne do ich generowania; związane z wdrażaniem, materiał instalacyjny, dokumentacja użytkownika, materiał treningowy.

[29] http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.

Mocne i słabe strony RUP

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:

- zarządzanie wymaganiami *ad hoc* (najczęściej brak zarządzania nimi);
- niejednoznaczna, nieprecyzyjna komunikacja;
- architektura oprogramowania nieodporna na obciążenia (ang. Brittle architecture);
- zbytnia, niepotrzebna złożoność oprogramowania;
- niewykryte niespójności w wymaganiach, projekcie oraz implementacji;
- brak lub niewystarczające testowanie;
- subiektywna ocena postępu projektu;
- brak zarządzania ryzykiem;
- brak automatyzacji prowadzenia projektu.

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.

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. Zalety RUP:

- RUP może być z powodzeniem wykorzystany w postaci: „as is” dla małych i średnich organizacji;
- RUP przykrywa całość cyklu życiowego SI;
- RUP wykorzystuje najnowsze trendy i technologie: obiektowe podejście, architektura oparta o komponenty;
- RUP jest systematycznie rozwijany i ulepszany;
- RUP posiada „solidną” architekturę, która może być przystosowana do konkretnych potrzeb użytkownika, RUP wspiera rozwój oprogramowania w oparciu o sześć najlepszych praktyk: iteracyjny rozwój, zarządzanie wymaganiami, architektura oparta o komponenty, wizualne modelowanie, systematyczna weryfikacja jakości i zarządzanie zmianami;
- RUP posiada całą paletę wspierających narzędzi.

Podsumowanie

Rational Unifield Process spośród pozostałych metodyk programowania wyróżnia się w szczególności:

- bliską współpracą zespołu do spraw rozwoju projektu z klientami i partnerami. Ta współpraca ma za zadanie zagwarantowanie prawidłowego przebiegu procesu tworzenia oprogramowania a zarazem zapewnia jego zgodność z zaleceniami klienta;
- podnosi efektywność zespołu, zapewniając każdemu programiście łatwy dostęp do bazy danych ze wskazówkami i szablonami. Dzięki temu każdy członek zespołu ma łatwy dostęp do tej samej bazy wiedzy;
- projektuje, testuje lub konfiguruje;
- tworzy i utrzymuje wzorce, zamiast skupiać się na produkcji dużej ilości dokumentacji, kładąc zarazem nacisk na rozwój modeli.

To wszystko sprawia że RUP jest uniwersalnym i idealnym procesem do wytwarzania oprogramowania. Zalety i wady RUP potwierdzają dwie poniższe wypowiedzi:

- „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”[30];
- „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. Zresztą opisano ten problem w artykule powyżej)”[31].

[30] http://www.martinfowler.com, 2012.
[31] http://www.uml.com.pl, 2012.

Bibliografia

1. http://www.martinfowler.com, 2012.
2. http://www.uml.com.pl, 2012.
3. http://www.gnu.univ.gda.pl, 2012.
4. https://www.github.com, 2012.
5. http://www.brasil.cel.agh.edu.pl, 2012.
6. http://pjwstk.mykhi.org/0sem/ZPR/prezentacje/ZPR%20RUP%20Rational%20Unified%20Process.ppt, 2012.
7. http://sun.aei.polsl.pl/~jszedel/sds/IP2-GD/mat/RUP.doc, 2012.
8. http://www.brasil.cel.agh.edu.pl, 2012.
9. http://elartu.tntu.edu.ua/bitstream/123456789/1455/19/Rational_Unified%20Process_Katarzyna_Pcion.pdf, 2012.
10. http://www.michalwolski.pl/page/42/, 2012.
11. https://icis.pcz.pl/~mciesiel/psi/PSI%20wyklad7.pdf, 2012.
12. http://www.si.pjwstk.edu.pl, 2012.
13. http://pjwstk.mykhi.org/0sem/PRI/RUP%2004.ppt, 2012.
14. http://math.uni.lodz.pl/~robpleb/wyklad10_14.pdf, 2012.
15. http://www.cyberwaretechnology.pl, 2012.
16. http://www.iicmagazine.pl, 2012.
17. „Rational Unified Process od strony teoretycznej", Philippe Kruchten, WNT, 2006 r.
18. „Rational Unified Process od strony praktycznej", Per Kroll, Philippe Kruchten, WNT, 2006 r.

RUP IN IT PROJECT MANAGEMENT

This publication describes the best practice for IT Software Development Management. The main purpose of this article is a presentation RUP practices as the method of managing IT software development and characterizing strong and weak sides. The Rational Unified Process is an iterative software development process framework created by the Rational Software Corporation. RUP is not a single concrete prescriptive process, but rather an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs.