WYPRÓBUJ ROZWIĄZANIA IBM RATIONAL – ZESTAW PROGRAMÓW NA 2 PŁYTACH DVD

SDJ EXTRA

ZESPÓŁ PRZEDE WSZYSTKIM

ROZWIĄZANIA IBM RATIONAL DLA DEWELOPERÓW, TESTERÓW, PROJEKTANTÓW APLIKACJI

Publikacje IBM Redbooks
źródło wiedzy i doświadczenia

IBM

Budowanie rozwiązań SOA na bazie Rational SDP

ibm.com/redbooks

Redbooks

Niniejsza publikacja (część serii IBM® Redbooks®) wyjaśnia podstawowe pojęcia i praktyki związane z tworzeniem rozwiązań bazujących na architekturze zorientowanej na usługi w oparciu o rozwiązanie IBM Rational® Software Delivery Platform (SDP). Wykorzystuje w tym celu najnowszą wersję metodologii IBM Rational® Unified Process (RUP), w której występuje zawartość dotycząca modelowana architektury zorientowanej na usługi (SOMA), udostępniana przez IBM Global Business Services.

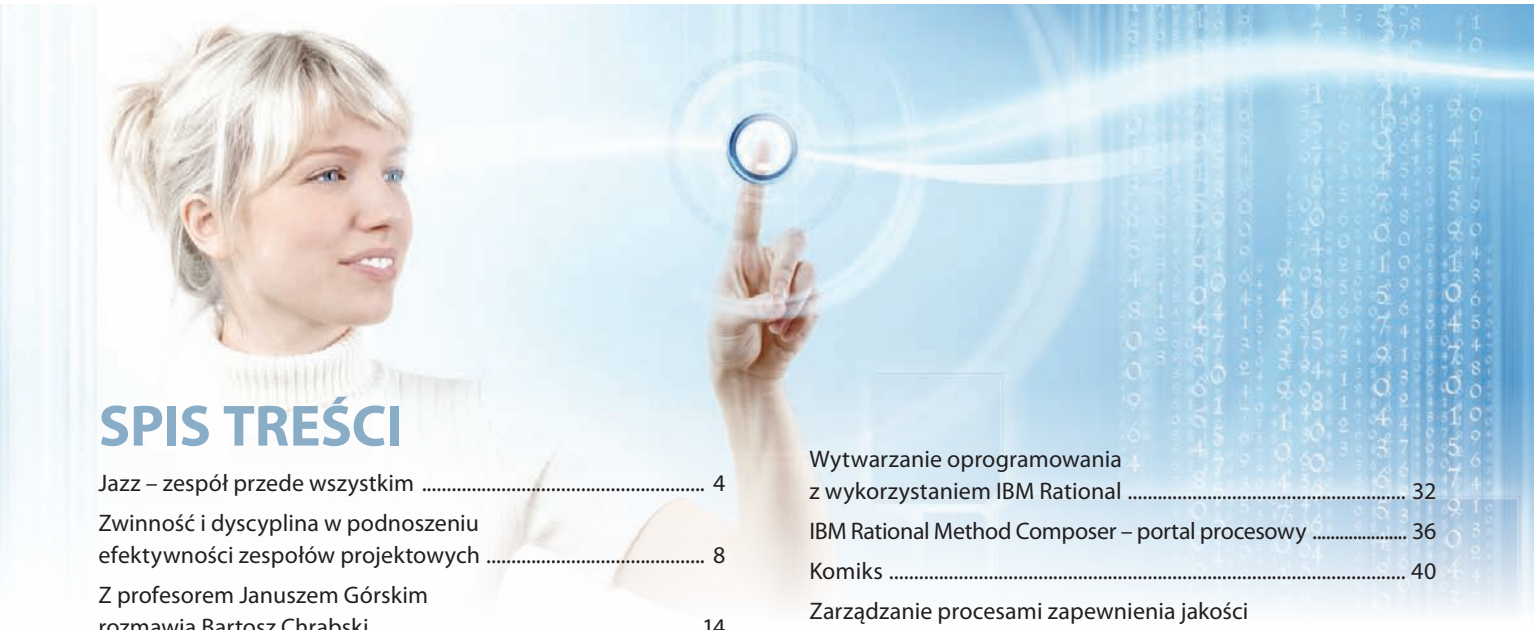
Książka ta stanowi nieocenioną pomoc dla praktyków, którzy rozwijają projekty oparte na architekturze SOA. Czytając ją można poznać kluczowe pojęcia związane z SOA i nauczyć się jak wykorzystać oprogramowanie narzędziowe w celu automatyzacji zadań związanych z rozwojem rozwiązań bazujących na tej architekturze.

ibm.com/redbook

ZESPÓŁ PRZEDE WSZYSTKIM

Zespół – efektywny dzięki właściwemu doborowi pracowników, kompetencji, wyznaczonym rolom: obserwatora, kreatora, lidera, analityka, itd. Na ile wszystkie te elementy mogą wystarczyć by zrealizować w wyznaczonym czasie stawiane przez organizację zadania, osiągnąć cele? W dobie pędu za zaspokojaniem potrzeb klientów wewnętrznych czy zewnętrznych bez właściwych narzędzi informatycznych praca żadnego zespołu nie jest możliwa. Specjalny numer Software Developers' Journal poświęcony jest właśnie narzędziom wspierającym pracę zespołów programistycznych, deweloperskich, testerskich. Zachęcam do lektury artykułów poświęconych platformie do pracy grupowej IBM Rational Jazz, a także tak ważnemu tematowi jak bezpieczeństwo tworzonych aplikacji. Specjalnie dla Państwa prof. Janusz Górski w artykule poświęconym inżynierii oprogramowania zakreślił trendy i wyzwania, z którymi spotykają lub będą się spotykali Państwo w najbliższych latach. Zachęcam również do lektury wywiadu z czternastokrotnym Mistrzem Europy w Rallycrossie – Kennethem Hansenem, poświęconego istocie pracy zespołu, bo przecież w dzisiejszych czasach każdy sukces to praca zespołu, a nie pojedynczego człowieka. Życzę udanej lektury i zachęcam do kontaktu z nami!

Bartosz Chrabski
IBM Rational Team Leader
Jest starszym specjalistą IT pracującym w grupie oprogramowania IBM Polska. Zajmuje się projektowaniem i wdrażaniem systemów zarządzania pracą zespołów deweloperskich oraz technicznym wsparciem sprzedaży rozwiązań do zarządzania i wytwarzania oprogramowania z rodziny IBM Rational. Specjalizuje się w technologiach middleware oraz modelowaniu architektury SOA. Doktorant na wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Założyciel i lider Łódzkiej Grupy Użytkowników Technologii Java (Lodz JUG).



SPIS TREŚCI

Software Developer's Journal Extra
jest wydawany przez Software-Wydawnictwo Sp. z o.o.

Wydawca: Anna Adamczyk
anna.adamczyk@software.com.pl
Redaktor naczelny: Łukasz Łopuszański
lukasz.lopuszanski@software.com.pl
Redaktor prowadzący: Tomasz Łopuszański
tomasz.lopuszanski@software.com.pl
Korekta: Tomasz Łopuszański
tomasz.lopuszanski@software.com.pl
Koordynatorzy projektu: Bartosz Chrabski, Maciej Mroczek, Joanna Izdebska
Kierownik produkcji: Andrzej Kuca
andrzej.kuca@software.com.pl

DTP: Marcin Ziółkowski, *www.gdstudio.pl*
Projekt okładki: Anna Adamczyk
anna.adamczyk@software.com.pl
Nakład: 6 000 egz.
Dział reklamy: *adv@software.com.pl*

Adres korespondencyjny:
Software Press Sp. z o.o., ul. Bokserska 1, 02-682 Warszawa, Polska
tel. +48 22 427 36 91, fax +48 22 224 24 59
www.sdjournal.org; cooperation@software.com.pl

Jazz

Zespół przede wszystkim

Ułatwienia w komunikacji między zespołami, zwiększenie przewidywalności projektów, integracja narzędzi, obniżenie ryzyka zawsze było wyzwaniem w projektach. Czy już jesteśmy gotowi, aby robić to efektywnie? Nie ma jasnej odpowiedzi na to pytanie, ale już wiemy, jak to zrobić z IBM Rational Jazz.

Wytwarzanie oprogramowania to zawsze złożony i rozbudowany proces, który uważa się za sztukę, naukę, rzemiosło, wymagające tajemnej wiedzy i lat doświadczenia. Często najbardziej niedocenianym obszarem we wszystkich realizowanych projektach IT jest ich wymiar społecznościowy, opisujący, jak ludzie pracują ze sobą przy realizacji wspólnych oraz osobistych celów. Duże przedsięwzięcie jest zależne nie od jednej, a wielu współpracujących i komunikujących się ze sobą osób, w określonym oczywiście porządku – wszystko po to, by końcowy produkt był jak najwyższej jakości. Zespołowe opracowywanie oprogramowania przypomina grę w zespole muzycznym. W obu przypadkach niezbędna jest równowaga między umiejętnościami indywidualnymi i współpracą w zespole.

Aby ułatwić zarządzanie całym procesem, IBM stworzył platformę integracyjną nazwaną Jazz. Dzięki odpowiedniemu podejściu do zagadnienia, praca zespołu może być efektywna na różnych etapach cyklu wytwórczego oprogramowania. Nazwa nawiązuje do faktu, iż muzycy zmieniają swoje uczucia w muzykę, natomiast w przypadku branży IT chodzi o zmianę sztuki dostarczania oprogramowania w będącą bardziej transparentną i produktywną.

Wizja

Platforma Jazz jest inicjatywą firmy IBM, która powstała, aby pomóc zespołom w celu osiągnięcia jak najlepszych wyników w pracy Autorzy,wykorzystując dobre praktyki współpracy z obszaru muzyki, przekształcili je tak, aby były możliwe do zastosowania w procesie wytwarzania oprogramowania, czyniąc je bardziej spójne, wydajne i przejrzyste.

Na fundamenty platformy Jazz wpływają trzy elementy :

- architektura integracji dla cyklu życia produktu,
- wsparcie portfolio produktów,
- społeczność.

W przeciwieństwie do monolitycznych i zamkniętych produktów przeszłości, platforma Jazz jest otwarta i przeznaczona dla każdej osoby zainteresowanej, która chce poprawić cykl życia oprogramowania i przełamać bariery między narzędziami. Produkty Jazz są uosobieniem innowacyjnego podejścia bazującego na otwartych, elastycznych usługach internetowych, a także najlepszych praktykach zgodnych ze standardami Opeb Web i OSGi Alliance.

Platforma Jazz została zaprojektowana tak, by dostarczać organizacjom elastyczność umożliwiającą przygotowanie własnego, idealnego środowiska dostarczania oprogramowania, używając preferowanych narzędzi. Co więcej, pozwala elastycznie rozwijać środowiska organizacji wedle własnych wymagań, własnym tempem, zamiast stać na drodze integracji narzędzi. Platforma definiuje wspólny zbiór usług Jazz Foundation, które można wykorzystać w dowolnym narzędziu implementującym to podejście. Uwzględnia ona również zdefiniowane w OSLC – Open Services for Lifecycle Collaboration specyfikacje, a także niezależne od dostawców zestawy protokołów współdzielenia informacji pomiędzy narzędziami.

Praca zespołowa

Projekt Jazz składa się ze wspólnej platformy i zestawu narzędzi, które umożliwiają wszystkim członkom zespołu rozwojowego łatwiejszą współpracę. Odzwierciedla to pogląd, że najważniejszym elementem w rozwoju oprogramowania nie są jednostki, nie proces, ale współpraca zespołu. W tym celu platforma udostępnia rozszerzalną architekturę zaprojektowaną z myślą o zwiększeniu poziomu współpracy, produktywności i przejrzystości produkcji oprogramowania.

Została ona dostosowana do potrzeb zespołów pracujących w zróżnicowanych lokalizacjach, uzupełniając tą wiedzę dodatkowo o informacje dotyczące użytkowników, projektów i procesów z elementami automatyzacji. Odpowiednie podejście do współpracy i zastosowanie

Rysunek 1. *Strona WWW projektu IBM Rational Jazz*

najlepszych praktyk z obszaru inżynierii oprogramowania pozwoliło na skrócenie cyklu powstawania oprogramowania, podniesienie jakości oraz usprawniło samo zarządzanie projektem.

Na razie platformy Jazz zostały stworzone trzy produkty :

- *Rational Requirements Composer* – rozwiązanie dedykowane do poprawnego definiowania wymagań przy zastosowaniu tekstu oraz rozbudowanych elementów graficznych. Narzędzie umożliwia zapisywanie wyrafinowanych potrzeb biznesowych w jednoznaczne wymagania, co na dalszych etapach prac nad produktem wpływa bezpośrednio na poprawę jakości czy usprawnienia samego procesu wytwórczego.
- *Rational Team Concert* – wspólne środowisko pracy dla programistów, projektantów, architektów i kierowników projektów. Dedykowane narzędzie pozwalające w jednym miejscu na zarządzanie pracą, kontrolę źródeł, zarządzanie systemem budowania, wsparcie dla planowania iteracji oraz proces planowania, obejmuje zwinne metodyki, m.in. Scrum oraz Eclipse Way.
- *Rational Quality Manager* – oparty o interfejs www system zarządzania procesem testowania dla decydentów oraz specjalistów ds. jakości. Dostarcza konfigurowalne rozwiązanie do planowania testów, przepływu kontroli, monitorowania i raportowania, realizacji testów czy zarządzania środowiskami testowymi.

Jazz to nie tylko społeczność praktyków rozwijających oprogramowanie, którzy pomagają innym, to także wpływ klientów i społeczności na kierunek rozwoju produktów przez wczesne, ciągłe i bezpośrednie rozmowy lub wymiany poglądów.

Po dołączeniu do społeczności na stronie *jazz.net*, można komunikować się z zespołami projektowymi, śledzić postęp budowy produktów, kamienie milowe, przekazać informacje o tym, co działa, a co nie oraz zgłaszać i śledzić defekty.

Każda z osób ma pełną przejrzystość szczegółowych planów, stanu, postępu budowy samej platformy lub komercyjnych produktów IBM Rational. Korzyść, jaka płynie z tej przejrzystości i dostępności do danych, to przede wszystkim fakt, iż istnieje możliwość stania się jednym z decydentów wpływających na ich rozwój. Częste dostarczanie trafnych opinii pozwoli zrozumieć i wpłynąć na kierunek kolejnych wydań i priorytetów, zanim decyzje zapadną. Celem takiego podejścia jest stworzenie środowiska pracy, które pomaga zespołom współpracować, być innowacyjnymi i tworzyć oprogramowanie wysokiej jakości. W związku z tym, skupiono się na fundamentach współpracy w zespole, automatyzacji procesów i raportowania cyklu życia oprogramowania. Narzędzia Jazz odzwierciedlają założenia, że najważniejszym obiektem nie jest tu jednostka czy proces, a współpraca. Zauważa się również, że zespół to nie tylko grupa programistów wraz z ich menedżerami, ale także pozostali, którym zależy na sukcesie inicjatywy, czyli klienci, sponsorzy. Celem narzędzi jest więc umożliwienie przejrzystości zespołów i projektów dla ciągłej współpracy, która pozwala na promowanie przełomowych innowacji, budowę spójności zespołu, wykorzystanie zasobów, talentó w i poza przedsiębiorstwem.

Automatyzacja

Badania pokazują, że prawie wszystkie organizacje chcą zmniejszyć liczbę biurokratycznych przeszkód stojących na drodze rozwoju oprogramowania, automatyzując żmudne i podatne na błędy zadania oraz uciążliwe do utrzymania operacje na danych. Jednak oni również muszą utrzymać i poprawiać proces spójności i zarządzania oraz zwiększać wgląd w rzeczywisty postęp projektu. Celem Jazz jest automatyzacja procesów przepływów pracy i zadań, dzięki czemu organizacje mogą przyjąć odpowiednią liczbę zasad rozwoju w tempie, które ma dla nich sens. Otrzymujemy więc poprawę wsparcia i wykonywania procesów, włączając w to procesy zwinne (*Agile*) oraz sformalizowane (*Rational Unified Process*), redukcję żmudnych i czasochłonnych czynności manualnych, przechwytywanie informacji o postępach, wydarzeniach, decyzjach i zezwoleniach bez dodatkowego wprowadzania danych.

Raportowanie

Uzyskanie szybkiego dostępu do informacji opartych na faktach jest niezbędne do planowania dalszej pracy. Zbyt często raporty na temat stanu rozwijanego oprogramowania wiążą się ze żmudną pracą wymagającą wysiłku ludzi odpowiedzialnych za sprawozdania, co wymaga czasu, a to powoduje, że raporty są mniej aktualne. Jazz koncentruje się na dostarczaniu raportów w czasie rzeczywistym, które umożliwią wgląd w programy, projekty i użyteczność zasobów, co znacząco pomaga zespołom projektowym. Identyfikuje i rozwiązuje problemy znacznie wcześniej w cyklu życia oprogramowania, zamiast szacowanych wartości, przedstawia metryki bazujące na faktach, co zwiększa efektywność podejmowanych decyzji, a także pozwala na odpowiednie wykorzystanie metryk do poprawy prac zespołu.

Architektura i implementacja Jazz

Architektura Jazz oparta jest na zasadach będących kluczem do porzucenia podejścia stosowanego w przeszłości. Wszystkie te zasady pozwalają zespołom na korzystanie z sieci WWW, by uzyskiwać dostęp do procesów czy artefaktów w ramach całego procesu wytwórczego.

Oto kilka przykładowych idei:

- Oddziela się wdrażanie narzędzi od definicji i dostępu do danych; semantyka danych nie opiera się na sekretnej wiedzy wbudowanej w kod produktu.
- Jazz może mieć dostęp i integrować dane tam, gdzie się one znajdują – nie ma potrzeby importowania i eksportowania danych pomiędzy narzędziami i repozytoriami.

Rysunek 2. *Produkty bazujące na platformie IBM Rational Jazz*

- Zakłada się, że istnieje otwarty, elastyczny, rozproszony model danych. Nie zakłada się natomiast, że istnieje jeden model danych, który jest centralnie zarządzany, ani że każde narzędzie musi rozumieć cały model danych w celu udziału w projekcie.
- Pozwala narzędziom na to, by zostały zaimplementowane w dowolnym języku programowania na dowolnej platformie. Jednocześnie nie narzuca wzorca implementacji przywiązanego do konkretnego języka programowania czy technologii platformy.
- Obsługuje wiele technologii klienckich. Interfejs użytkownika jest wyborem dla różnych narzędzi Jazz. Wspiera także oprogramowanie bazujące na Eclipse czy Microsoft Visual Studio. Inne platformy mogą zostać łatwo zintegrowane, zależnie od żądania.

Standard OSLC (*Open Services for Lifecycle Collaboration*) to inicjatywa przemysłu mająca na celu umożliwienie interoperacyjności narzędzi i zasobów różnych dostawców. Zwykle klienci korzystają z narzędzi od różnych dostawców w celu wychwytywania i zarządzania danymi w całym cyklu życia oprogramowania. Dane te są zazwyczaj dostępne za pomocą narzędzi, przy pomocy których zostały one stworzone, lub przez dostęp do bazy danych używanej przez to narzędzie. Brak wspólnych protokołów dla wszystkich zasobów sprawia, że trudno jest zarządzać oprogramowaniem.

Architektura definiuje zestaw usług zwanych Jazz Foundation Services (usługi podstawowe Jazz), które mogą korzystać z podstawowych funkcji platformy do realizacji ogólnych zadań między narzędziami różnych dostawców, w tym do administracji projektami, użytkownikami, bezpieczeństwa, współpracy, zapytań. Usługi te umożliwiają narzędziom implementować swe własne funkcjonalności tak, że będą one dobrze współgrać z innymi narzędziami.

W centrum architektury integracji Jazz znajduje się Jazz Team Server. JTS realizuje fundamentalne usługi Jazz opisane przez JIA, które umożliwiają współpracę grupom narzędzi. Jazz Team Server może składać się z jednego lub większej liczby fizycznych serwerów, które działają jako jeden serwer logiczny.

W celu ułatwienia wdrażania produktów zgodnych z JIA dostarczono odpowiednie zestawy narzędzi Jazz. Należą do nich: biblioteki specyficzne dla danego języka umożliwiające dostęp do różnych usług Jazz Foundation oraz dla usług specyficznych dla domeny.

Większość poziomów integracji JIA jest osiągalnych w dowolnym języku programowania zdolnym do przetwarzania zapytań HTTP oraz danych w formacie XML. Oznacza to, że frameworki Jazz mogą być rozwijane przez kogokolwiek i gdziekolwiek.

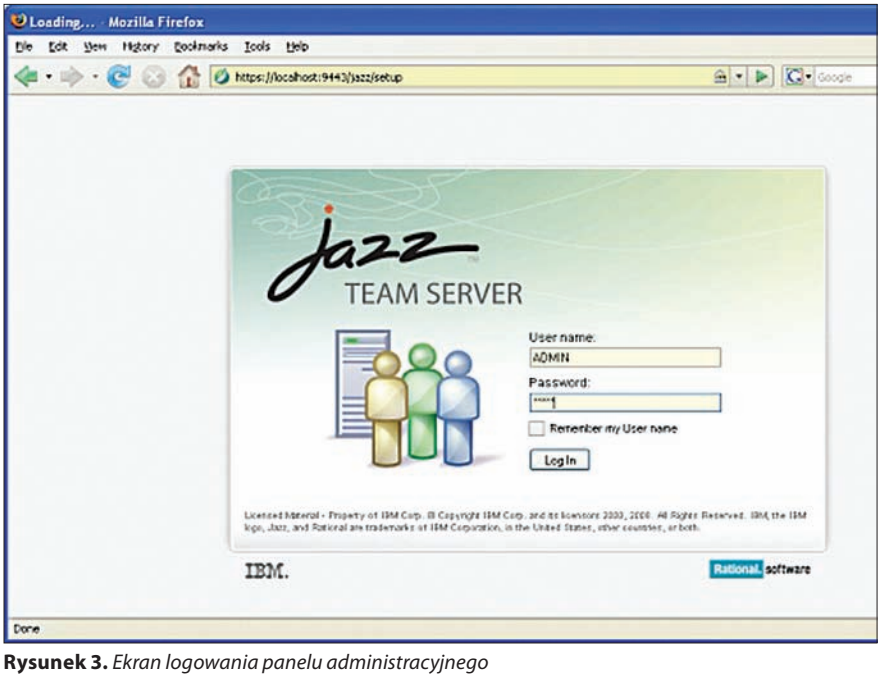
Podsumowanie

IBM Rational wraz z partnerami oferuje produkty współdziałające z Jazz Integration Architecture. Jednak nie każdy produkt Jazz zawiera każdy element z JIA oraz całe podejście Jazz, bowiem całe podejście ciągle ewoluuje. W praktyce, opisywana elastyczność łączenia wersji i tolerancji różnych stopni złożoności jest jedną z najważniejszych pozytywnych cech platformy oraz Jazz Integration Architecture.

Aktualnie w ofercie znajdują się następujące produkty:

- Rational Requirements Composer,
- Rational Team Concert,
- Ration Quality Manager.

Pozostałe produkty z portfolio IBM Rational w przyszłości prawdopodobnie zostaną wcielone do platformy Jazz, co przyczyni się do jeszcze silniejszej ich integracji. Lista produktów z pewnością będzie też poszerzona o dodatkowe nowe projekty.



Rysunek 3. *Ekran logowania panelu administracyjnego*

Odniesienia

- *http://www-01.ibm.com/software/rational/jazz/*
- *http://jazz.net/*
- *http://jazz.net/library/LearnItem.jsp?href=content/docs/platform-overview/index.html*
- *http://www.rodenas.org/blog/2009/04/20/from-the-eclipse-platform-to-the-ibm-rational-jazz-platform/*
- *http://www.youtube.com/watch?v=u_M8jcwuPlg&translated=1*
- *http://www.infoworld.com/d/developer-world/ibm-hails-jazz-collaboration-platform-126*
- *http://publib.boulder.ibm.com/infocenter/repmhelp/v1r0m0/index.jsp?topic=/com.ibm.jazz.platform.doc/topics/c_jazz-architecture.html*
- *http://www-01.ibm.com/software/info/television/html/K936283A96390X11.html*
- *http://www-903.ibm.com/kr/event/download/200807_364_jazz/s364_01.pdf*

W sieci:

- Jazz.net
  http://www.jazz.net/
- Rational Requirements Composer
  http://www-01.ibm.com/software/awdtools/rrc/
- Rational Team Concert
  http://www-01.ibm.com/software/awdtools/rtc/
- Ration Quality Manager
  http://www-01.ibm.com/software/awdtools/rqm/

Bartosz Chrabski
IBM Rational Team Leader
Jest starszym specjalistą IT pracującym w grupie oprogramowania IBM Polska. Zajmuje się projektowaniem i wdrażaniem systemów zarządzania pracą zespołów developerskich oraz technicznym wsparciem sprzedaży rozwiązań do zarządzania i wytwarzania oprogramowania z rodziny IBM Rational. Specjalizuje się w technologiach middleware oraz modelowaniu architektury SOA. Doktorant na wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Założyciel i lider Łódzkiej Grupy Użytkowników Technologii Java (Lodz JUG).

www.forpsi.pl

Profesjonalny Hosting ...dla każdego

Oferujemy:

rejestrację domen
od 8 PLN + vat/rok

www Windows lub Linux
od 2 PLN + vat/miesiąc

pakiet = domena + www
od 30 PLN + vat/rok

serwery vps
od 40 PLN + vat/miesiąc

serwery dedykowane

Teraz! Nowość!

Serwer Basic
od 96 PLN + vat/miesiąc

forpsi

AlphaNet spółka z o. o.
al. Mireckiego 22 - 41-200 Sosnowiec
Tel. 032 763 66 66 Fax. 032 785 33 32

Zwinność i dyscyplina w podnoszeniu efektywności zespołów projektowych

W artykule przedstawiono założenia metodyki projektowania systemów informatycznych łączącą cechy Rational Unified Process oraz metodyk zwinnych takich jak SCRUM i OpenUP. Artykuł przedstawia charakterystykę metodyk Rational Unified Process, SCRUM i OpenUP z uwypukleniem ich cech wspólnych. Artykuł zawiera także analizę wyników zastosowania proponowanej metodyki w projekcie informatycznym w kontekście podnoszenia efektywności zespołu projektowego pracującego zgodnie z tą metodyką.

Podejścia projektowe

Zwinne podejście do projektowania systemów informatycznych określa, że istotne elementy w projektowaniu tego typu systemów to [9]:

- Procesy i narzędzia,
- Szczegółowa dokumentacja,
- Negocjacje kontraktowe,
- Przestrzeganie planu.

Natomiast, zgodnie z założeniami podejścia zwinnego, jeszcze bardziej istotnymi elementami są:

- Osoby i interakcje,
- Funkcjonujące oprogramowanie,
- Współpraca z klientem,
- Dostosowywanie się do zmian.

Widać z tego, że podejście zwinne przenosi punkt skupienia z procesu z przypisanymi rolami, sztywnego trzymania się szczegółowego planu i nadmiernego dokumentowania decyzji projektowych na jak najszybsze uzyskiwanie funkcjonującego oprogramowania spełniającego zmieniające się potrzeby klienta.

W ocenie autora nie oznacza to jednak odrzucenia procesu projektowania i określonych ról, gdyż role są zdefiniowane w metodykach zwinnych, lecz koncentrację na uzyskaniu efektu końcowego, jakim jest funkcjonujące zgodnie z potrzebami klienta oprogramowanie.

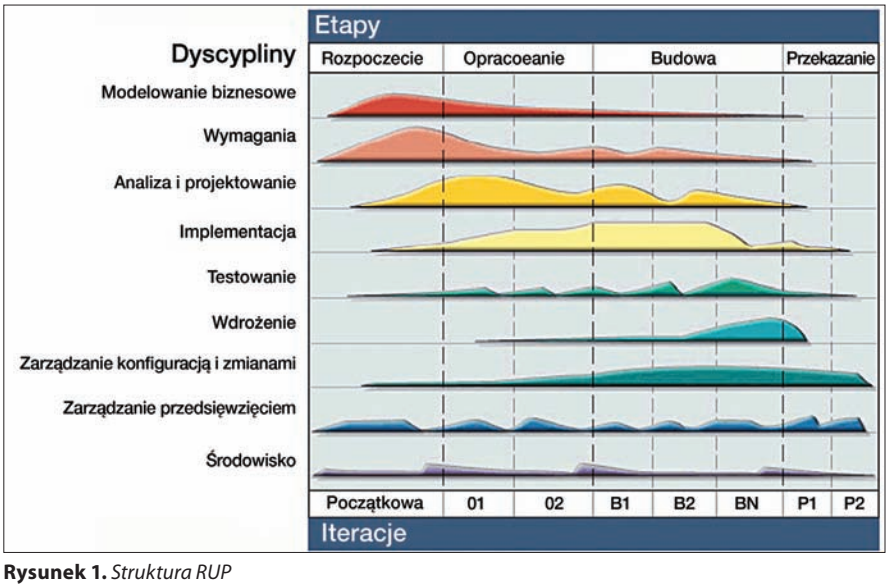
Praktyka projektowa pokazuje, że klienci, podpisując kontrakt, mają jasno zdefiniowane cele tworzenia systemu. Oprogramowanie to ma określony zakres funkcjonalności, którą należy zbudować w określonym czasie i budżecie. Dwa ostatnie elementy w trakcie trwania projektu nie ulegają wydłużeniu ani zwiększeniu.

Klient, podpisując kontrakt, oczekuje określonych korzyści biznesowych, jakie przyniesie mu wdrożenie oprogramowania, które zamawia. Musi ono spełniać określone wymagania funkcjonalne oraz, co bardzo istotne, poza funkcjonalne. W szczególności wymagania związane z wydajnością i bezpieczeństwem.

Zdaniem autora, najlepszy efekt uzyskuje się, łącząc najlepsze cechy metodyk tradycyjnych takich jak IBM Rational Unified Process oraz metodyk zwinnych takich OpenUP czy SCRUM. W ten sposób można uzyskać proces projektowania systemu informatycznego, który zapewni odpowiedni poziom dyscypliny w prowadzeniu projektu z określonymi terminami, budżetem oraz adaptowalność zespołu projektowego do zmieniających się w trakcie projektu wymagań. Przy bliższym spojrzeniu na te metodyki okazuje się, że nie są one od siebie tak odległe jak mogłoby się wydawać.

Rational Unified Process

Rational Unified Process (RUP) jest iteracyjną metodą wytwarzania oprogramowania. Metodyka RUP ma opinię ciężkiej metodyki. Argumentuje się to olbrzymim nakładem prac na



Rysunek 1. *Struktura RUP*

wytwarzanie dokumentacji projektu. Według niektórych [11] prace nad dokumentacją zajmują do 50% wszystkich prac projektowych. Tego typu wyliczenia wskazują na stosowanie metodyki RUP w pełnej postaci, co jest pomysłem delikatnie mówiąc nierozsądnym.

U podstaw RUP leży sześć kluczowych reguł projektowania systemów informatycznych ukierunkowanego na zastosowanie biznesowe:

- dostosowanie procesu,
- bilansowanie przeciwstawnych priorytetów udziałowców,
- współpraca między zespołami,
- demonstrowanie wartości w sposób iteracyjny,
- dostosowanie poziomu abstrakcji,
- ciągłe skupienie na jakości.

Za regułami tymi kryją się podstawowe dla inżynierii oprogramowania praktyki:

- wytwarzaj iteracyjnie,
- zarządzaj wymaganiami,
- używaj architektury komponentowej,
- modeluj wizualnie (UML2.0),
- stale weryfikuj jakość,
- zarządzaj zmianami.

Na podkreślenie zasługują praktyki związane z zarządzaniem wymaganiami, modelowaniem wizualnym z UML oraz używaniem architektury komponentowej.

Struktura procesu RUP jest dwuwymiarowa, co przedstawia Rysunek 1. Wymiar poziomy pokazuje ułożenie projektu w czasie z podziałem na etapy i iteracje. Wymiar pionowy ukazuje dyscypliny procesu RUP. W ramach RUP, występuje dziewięć dyscyplin, począwszy od modelowania biznesowego a skończywszy na utrzymaniu środowiska produkcyjnego. W każdej z dyscyplin określone są czynności i role odpowiedzialne za wykonanie tych czynności, produkty potrzebne do realizacji czynności, a także produkty będące wynikiem realizacji czynności [6].

Z punktu widzenia planowania projektu istotne są dwa plany: *Plan przedsięwzięcia, Plan iteracji*, za które odpowiada Kierownik projektu. *Plan przedsięwzięcia* zawiera takie informacje jak terminy głównych kamieni milowych faz, profil zespołu przedsięwzięcia, terminy pomniejszonych kamieni milowych. *Plan iteracji* jest planem szczegółowym i tyczącym się aktualnej iteracji. *Plan iteracji* zawiera terminy iteracji, jej zakres oraz kryteria odbioru. Podczas każdej iteracji realizowane są czynności związane z dyscyplinami potrzebnymi do jej realizacji. Udział czynności z poszczególnych dyscyplin zmienia się wraz z postępem procesu.

Czas trwania iteracji nie jest ściśle określony. RUP rekomenduje, że iteracja powinna trwać od dwóch do sześciu tygodni [8]. Planowanie iteracji zgodnie z metodyką RUP powinno składać się z czterech kroków:

- określenie zakresu iteracji,
- określenie kryteriów oceny iteracji,
- przypisanie odpowiedzialności do poszczególnych czynności.
- określenie czynności wykonanych podczas iteracji,

Cele iteracji zależą od etapu, w którym rozpatrywana iteracja jest przeprowadzana.

Pierwszym etapem w metodyce RUP jest etap rozpoczęcia. Głównymi celami etapu rozpoczęcia są: określenie zakresu sytemu, przedstawienie architektury kandydującej systemu, określenie listy ryzyk.

W etapie opracowania głównymi celami są zweryfikowanie architektury systemu oraz rozwiązanie głównych ryzyk. Szczegółowemu opisowi i implementacji w etapie opracowania powinny podlegać te przypadki użycia, które są krytyczne z punktu widzenia architektury, oraz te, które realizują funkcjonalność najistotniejszą dla udziałowców.

Celem etapu budowy jest wytworzenie stabilnej wersji systemu z pełną wymaganą funkcjonalnością. Etap budowy jest najbardziej pracochłonnym z etapów i zajmuje przeciętnie 50% czasu trwania całego projektu.

Celem etapu przekazania jest dostarczenie końcowej wersji produktu użytkownikowi.

Metodyka Rational Unified Process uważana jest za metodykę ciężką ze względu na tworzoną dużą liczbę dokumentów. Dotyczy to jednak pełnej metodyki RUP, która nie powinna być w takiej postaci stosowana w projektach. Należy pamiętać o pierwszej zasadzie RUP – dostosowanie procesu – i dobrać poziom dokumentowania procesu do wielkości i złożoności projektu.

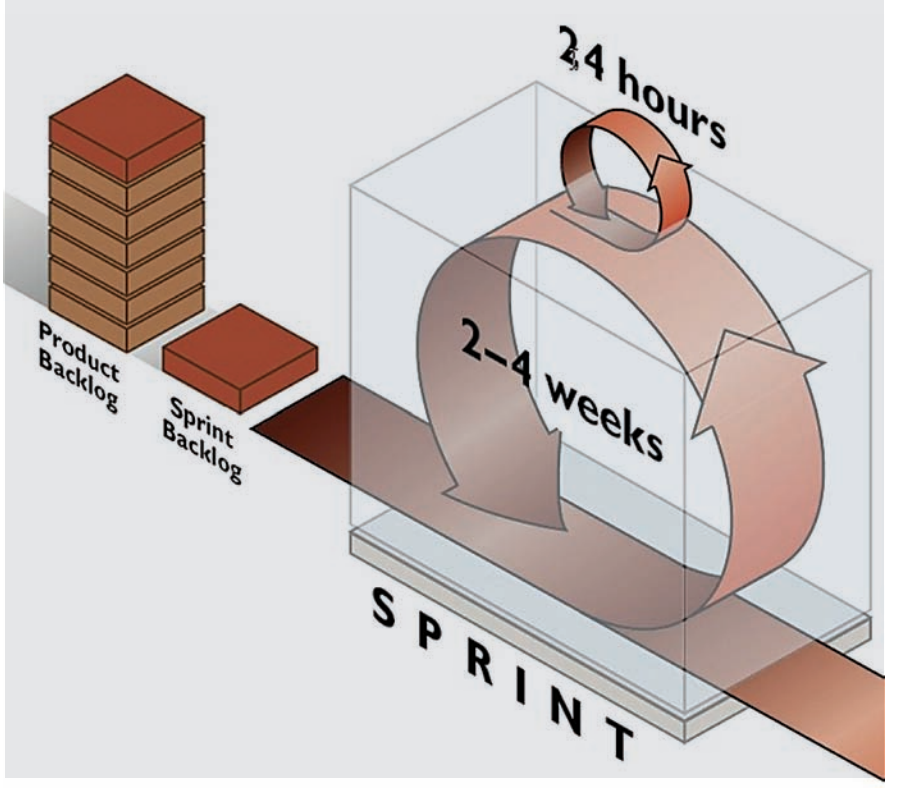
SCRUM

Scrum to, zgodnie z Agile Manifesto, zwinna metodyka prowadzenia projektów. Najczęściej jest ona wykorzystywana w projektach informatycznych. SCRUM używany jest w projektach charakteryzujących się wysokim poziomem zmienności wymagań lub w przypadku przedsięwzięć o wysokim stopniu innowacyjności.

Metodyka skupia się na:

- dostarczaniu kolejnych, coraz bardziej dopracowanych wyników projektu,
- włączaniu się przyszłych użytkowników w proces wytwórczy,
- samoorganizacji zespołu projektowego.

Zazwyczaj zespół SCRUM składa się z 5-9 osób. Dobrze, gdy ma on charakter interdyscyplinarny i składa się z osób reprezentujących różne umiejętności. Osoby uczestniczące w zespole nie mogą uczestniczyć w innych zespołach. Główne role w projekcie to: Mistrz Młyna (ang. *Scrum Master*), Właściciel Produktu (ang. *Product Owner*) i Członkowie Zespołu (ang. *The Team*).



Rysunek. 2 *Przebieg prac zgodnie ze SCRUM [13]*

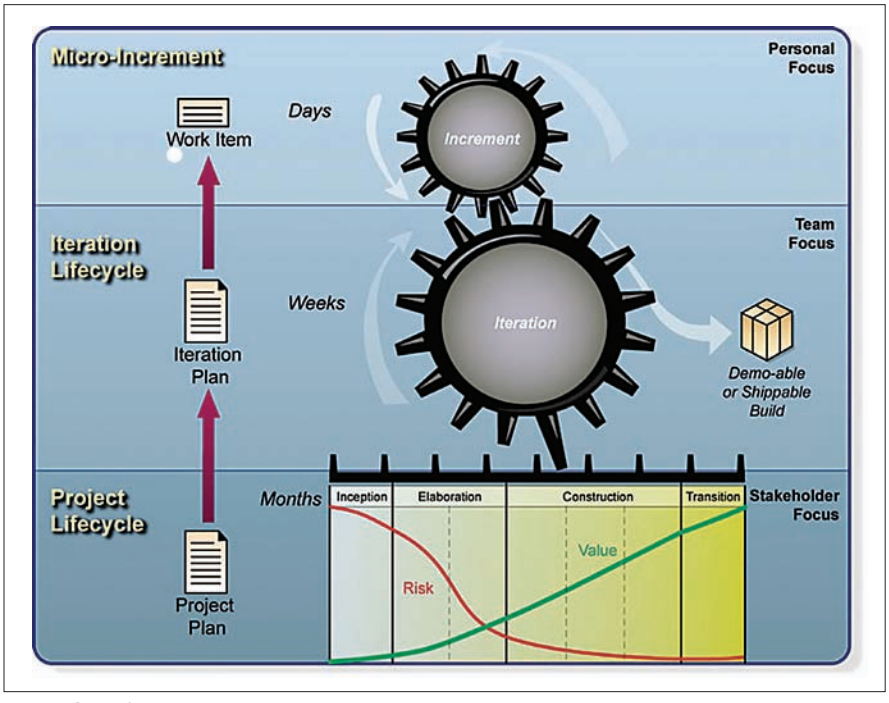
Zespół projektowy pracuje w określonym przedziale czasowym zwanym przebiegiem (ang. *sprint*). Efektem przebiegu za każdym razem powinno być dostarczenie użytkownikom kolejnego działającego produktu. Zasadą jest to, że prace wykonywane w ramach przebiegu muszą skutkować dostarczeniem nowej funkcjonalności. Przebieg może trwać od 2 do 6 tygodni. Zaleca się stosowanie przebiegów *o stałych długościach.*

W pierwszym etapie tworzona jest lista wymagań użytkownika. Właściciel projektu jest też zobowiązany do przedstawienia priorytetów wymagań oraz głównego celu przebiegu. Po tym formułowany jest rejestr wymagań (ang. *Product Backlog*). Następnie wybierane są zadania o najwyższym priorytecie, *a jednocześnie przyczyniające się do realizacji celu projektu. Szacuje się czas realizacji każdego zadania. Lista zadań wraz z oszacowaną czasochłonnością nosi nazwę rejestru zadań przebiegu (ang. Sprint Backlog).* W trakcie realizacji przebiegu Właściciel Produktu nie może ingerować w prace zespołu. Nie powinno się także zmieniać zakresu przebiegu. Przebieg prac zgodnie ze SCRUM został przedstawiony na Rysunku 2.

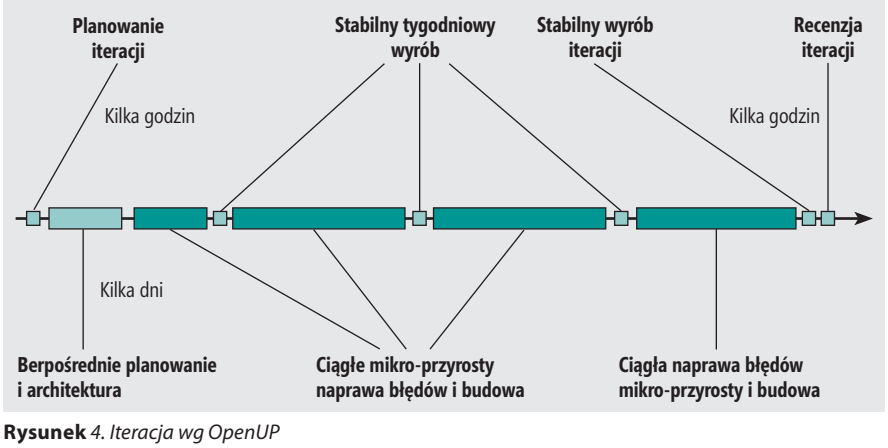
Zespół z założenia jest samoorganizujący się i nie ma odgórnego przypisywania zadań do poszczególnych członków zespołu. Samodzielnie dokonują oni wyboru realizowanych zadań, według wspólnych ustaleń i umiejętności.

Naczelną zasadą metodyki jest przeprowadzanie codziennych (około 15-minutowych) spotkań (ang. *scrum meeting*), na których omawiane są zadania zrealizowane poprzedniego dnia i problemy występujące przy ich realizacji oraz zadania do wykonania w dniu spotkania.

Przebieg kończy się przeglądem przebiegu (ang. *sprint review*), na którym prezentowany jest wynik pracy zespołu przez prezentowanie produktu wykonanego podczas przebiegu. Powinni w nim uczestniczyć wszyscy zainteresowani projektem. Po omówieniu produktu ustalany jest termin spotkania planistycznego do następnego przebiegu.



Rysunek 3. *Obszary OpenUP [2]*



Rysunek *4. Iteracja wg OpenUP*

OpenUP

Metodyka Open Unified Process, w skrócie OpenUP, jest częścią projektu Eclipse Process Framework (EPF). OpenUP dostarcza zbiór dobrych praktyk iteracyjnego wytwarzania oprogramowania. OpenUP proponuje podzielić cały proces wytwarzania oprogramowania na trzy następujące obszary:

- Cykl Życia Projektu (ang. *Project Lifecycle*),
- Cykl Życia Iteracji (ang. *Iteration Lifecycle*),
- Mikro-Przyrosty (ang. *Micro-Increments*).

Podział ten oraz jego najważniejsze aspekty przedstawia Rysunek 3.

Każdy z obszarów posiada artefakt, który opisuje jego zakres oraz prace do wykonania. Najbardziej ogólnym obszarem jest Cykl Życia Projektu, dokumentem z nim związanym jest *Plan projektu.* Cykl Życia Projektu dostarcza udziałowcom, członkom zespołu widocznych punktów synchronizacji oraz punktów decyzyjnych w całym projekcie.

Proces wytwarzania oprogramowania, zgodnie z OpenUP, podobnie jak RUP, składa się z czterech etapów:

- Rozpoczęcia (ang. *Inception*),
- Opracowania (ang. *Elaboration*),
- Budowy (ang. *Construction*),
- Przekazania (ang. *Transition*).

Etapy kończą się kamieniami milowymi. Każdy z etapów składa się z jednej lub wielu iteracji. Na podstawie dokumentu *Plan projektu* tworzony jest dokument *Plan iteracji.* Artefakt ten związany jest z *obszarem OpenUP, jakim jest Cyklu Życia Iteracji. Obszar Cykl Życia Iteracji zapewnia zbiór praktyk opisujących sposób przeprowadzenia iteracji ukierunkowanej na zespół, dostarczającej przyrostowe wartości udziałowcom w przewidywalny sposób. Iteracja wymusza na zespole skoncentrowanie swoich prac na dostarczeniu, co kilka tygodni w pełni przetestowanej, kolejnej części produktu. Przebieg iteracji zgodny z OpenUP przedstawiony jest na* Rysunku 4.

Postęp iteracji widoczny jest przez zakończenie kolejnych mikro-przyrostów (ang. *micro-increments*). Na początku iteracji organizowane jest spotkanie dotyczące planu iteracji. Spotkanie takie trwa do kilku godzin. Inicjalizacja iteracji trwa od jednego do dwóch dni. Czas ten poświęcony jest na uszczegółowienie Planu iteracji, dogłębne zrozumienie zależności logicznych wytwarzanych elementów oraz ich wpływu na architekturę. Większość czasu iteracji poświęcona jest na wykonywanie mikro-przyrostów, które dostarczają przetestowany kod oraz zatwierdzone artefakty. Dla uzyskania większej dyscypliny, pod koniec każdego tygodnia należy zapewnić stabilny kod. Umożliwia to wczesne wykrycie i rozwiązanie

problemów. Ostatni tydzień lub kilka ostatnich dni iteracji zazwyczaj poświęconych jest w dużej mierze na poprawę wcześniej zidentyfikowanych błędów. Celem jest dostarczenie, na koniec iteracji, wysokiej jakości produktu zawierającego wymaganą funkcjonalność. Iteracje kończą się oceną produktu, który został podczas niej wytworzony. W ocenie biorą udział zainteresowane strony. Iteracje w różnych etapach różnią się od siebie wykonywanymi czynnościami oraz celami. Cele w poszczególnych fazach są podobne do celów RUP.

Metodyka OpenUP jest lekką metodyką iteracyjnego wytwarzania oprogramowania. Przeznaczona jest dla zespołów składających się z 3-6 sześciu osób, oraz projektów, które trwają około sześciu miesięcy. Szczegółowe porównanie RUP i OpenUP zawarto w [5].

Zwinny RUP

W projektowaniu systemów informatycznych istotnym jest element spełniania wymagań postawionych przez udziałowców.

Z praktyki projektowej wynikają niezbicie dwa fakty. Zbyt sztywne trzymanie się harmonogramu i skupienie na tworzeniu szerokiej listy dokumentów może prowadzić do olbrzymich kosztów i nikłych efektów w sensie wytworzonego produktu końcowego, jakim jest funkcjonujący system informatyczny. Z drugiej strony brak jakiejkolwiek dokumentacji i duża skłonność do wprowadzania zmian na życzenie udziałowców prowadzi do nadmiernego rozprzestrzeniania się zakresu systemu, wielokrotnego przepisywania kodu i jego testowania i w wyniku także wysokich kosztów wytworzenia systemu informatycznego.

W związku z powyższym istotnym jest odpowiednie skonfigurowanie procesu projektowania systemu informatycznego. Kluczowym, zdaniem autora, jest łączenie zalet podejścia zdyscyplinowanego z podejściem zwinnym. Podstawowym jest zastosowanie głównych założeń metodyk takich jak RUP na poziomie wyższym projektu: etapy. Natomiast na niższym poziomie – iteracji, mikro-przyrostu – należy stosować założenia charakteryzujące metodyki zwinne takie jak SCRUM i OpenUP.

Do zasad RUP, które warto stosować w projektach informatycznych, należą:

- iteracyjne wytwarzanie,
- modelowanie wizualne (UML),
- modelowanie architektury (wybrane widoki),
- zarządzanie wymaganiami w sposób ciągły.

Do zalet RUP należy stosowanie podziału projektu na etapy i iteracje. Zastosowanie czterech etapów z jasno określonymi celami i kryteriami przejścia do kolejnego etapu przyczynia się do wczesnego zdefiniowania zakresu, weryfikacji architektury, rozwiązania ryzyk projektowych, przewidywalnego dostarczania kolejnych przyrostów oraz przygotowanego przekazania systemu użytkownikowi końcowemu.

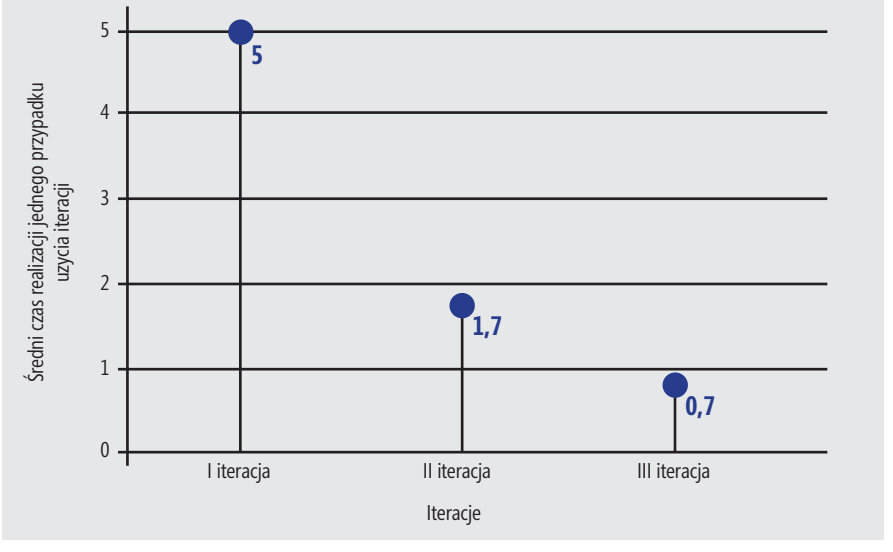
Elementem, jaki należy zrealizować przy zastosowaniu RUP w projektach, jest jasne zdefiniowanie zakresu czynności i produktów wchodzących w zakres metodyki stosowanej w określonym projekcie. Podstawowe w RUP jest ograniczenie zbioru wytwarzanych dokumentów oraz modeli do tych naprawdę istotnych w określonym projekcie. Zapewni to odciążenie zespołu projektowego od nadmiernego dokumentowania i modelowania oraz pozwoli na skupienie się na wytwarzaniu oprogramowania.

Przy projektowaniu systemów informatycznych istotne są kluczowe role definiowane w RUP:

- Kierownik projektu,
- Główny analityk,
- Główny architekt,
- Projektant bazy danych,
- Kierownik testów.

Role te koordynują główne prace w zespole projektowym i mają największy wpływ na kreowanie końcowego rozwiązania, jakim jest system informatyczny. Porównując z metodyką SCRUM, można zauważyć podobieństwo ról Kierownik projektu i Główny Analityk z RUP oraz Mistrz Młyna i Właściciel Produktu ze SCRUM.

SCRUM czy OpenUP może wnieść do projektu, paradoksalnie, zdyscyplinowane wytwarzanie praktycznie codziennych mikro-przyrostów oprogramowania. Efektem takiej organizacji pracy jest bardziej stabilne oprogramowanie na koniec iteracji. Należy jednak pamiętać,



Rysunek 5. *Wykres średniego czasu realizacji przypadku użycia w iteracji*

aby nie popaść w zbyt szczegółowe testowanie poszczególnych mikro-przyrostów. Z drugiej strony testowanie oprogramowania tylko przez tworzących je programistów może prowadzić do pozostawienia w kodzie zbyt wielu ukrytych błędów. Wyjściem jest stosowanie testowania jednostkowego na poziomie przyrostu, a testowania wersji włącznie z testami regresyjnymi na poziomie iteracji.

Kluczowym przy stosowaniu podejścia SCRUM jest znajomość jasno zdefiniowanych wymagań dla przebiegu. W roli Właściciela Produktu ze SCRUM może występować Analityk z RUP zajmujący się określonym wymaganiem funkcjonalnym – przypadkiem użycia. Analityk ściśle współpracuje z Członkami Zespołu w celu doprecyzowania wymagań realizowanych w przebiegu.

Zasada, aby nie przedłużać czasu trwania iteracji, jest wspólna, zarówno dla metodyk RUP, jak i SCRUM. Zarówno RUP, jak i SCRUM podkreślają znaczenie oceny iteracji (przebiegu) dla lepszego dalszego planowania projektu.

Drugim podstawowym elementem procesu wytwórczego jest architektura systemu informatycznego. Jest to element szczególnie podkreślany w metodyce RUP. Metodyka RUP w pełnej postaci zakłada modelowanie systemu informatycznego z punktu widzenia modelu architektonicznego „4+1"[7]. Model ten obejmuje następujące widoki architektoniczne systemu informatycznego: Przypadków użycia, Logiczny, Wdrożeniowy, Procesów, Implementacyjny.

W większości projektów informatycznych istotne jest modelowanie architektury systemu informatycznego z punktu widzenia następujących widoków architektonicznych modelu „4+1": Przypadków użycia, Logicznego, Wdrożeniowego.

Pierwszy z widoków architektonicznych stanowi widok Przypadków użycia. W ramach tego widoku wykonuje się Model przypadków użycia. Model ten stanowi specyfikację wymagań budowanego systemu.

W widoku Logicznym przedstawiana jest struktura oprogramowania oraz sposób jego działania. Budowany jest w nim Model projektowy oraz Model bazy danych.

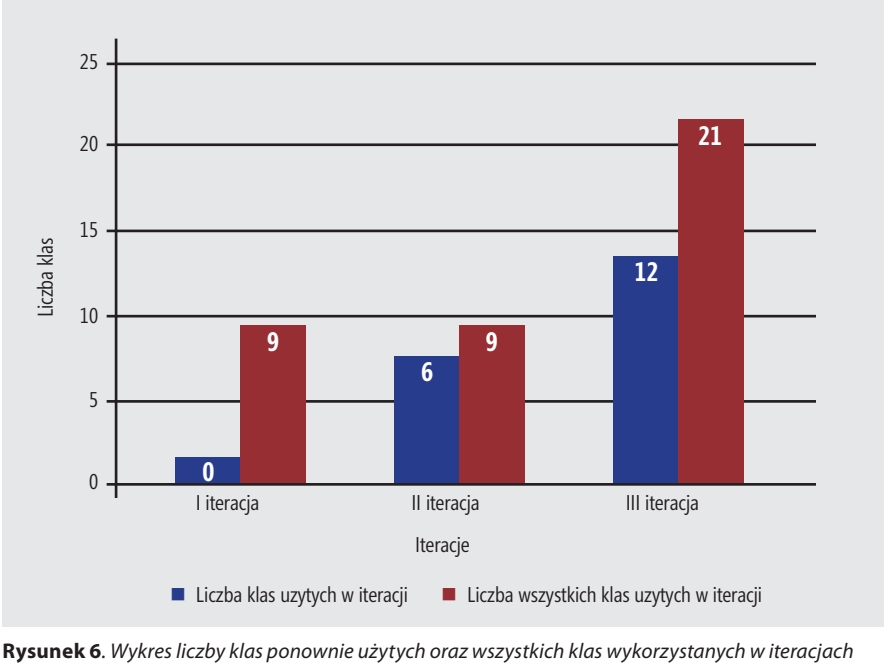
Istotnym jest widok Wdrożeniowy pokazujący system informatyczny na środowisku uruchomieniowym z fizycznymi serwerami oraz środowiskami uruchomieniowymi. Budowany jest w nim Model wdrożeniowy.

Wczesne zdefiniowanie i zweryfikowanie architektury systemu informatycznego zapewnia minimalizację ryzyk projektowych oraz minimalizację kosztów projektu. Zgodnie z RUP, pełen zespół projektowy powoływany jest dopiero w fazie budowy, po wcześniejszej weryfikacji architektury przez niewielki zespół doświadczonych specjalistów.

Zwinny RUP w praktyce

Przedstawione powyżej podejście zostało wstępnie zweryfikowane w projekcie uczelnianym systemu informatycznego dla Kancelarii Prawniczej. Następnie zostało zastosowane w projektach informatycznych istotnie różniących się od siebie:

- Projekt dla Służby Zdrowia RP „Modelowanie repozytorium i analiza efektywności informacyjnej wytycznych i ścieżek klinicznych w służbie zdrowia", nr ref.: POIG. 01.03.01-00-145/08, Program Operacyjny „Innowacyjna Gospodarka" – zespół 30 osób.
- Etap I wdrożenia Zintegrowanego Systemu Wspomagającego Zarządzanie Pojazdami Kolejowymi i Drużynami Trakcyjnymi w PKP CARGO S.A. – zespół 50 osób.



Rysunek 6. *Wykres liczby klas ponownie użytych oraz wszystkich klas wykorzystanych w iteracjach*

Najistotniejszą cechą wspólną tych projektów informatycznych jest to, że zakończyły się w z góry określonym czasie, dostarczono wymaganą funkcjonalność oraz zrealizowano to w ramach ustalonego wcześniej budżetu.

Dalej przedstawiono analizę wyników budowy fragmentu systemu informatycznego dla Kancelarii Prawniczej [10]. System ten tworzony był w architekturze Java EE, przy wykorzystaniu dostosowanej konfiguracji RUP. Cały proces składał się z trzech iteracji. W iteracjach zostały uzyskane wartości następujących wskaźników efektywności zespołu projektowego:

- średni czas realizacji przypadku użycia,
- liczba zaimplementowanych przypadków użycia w iteracji,
- liczba klas wytworzonych w iteracji,
- liczba klas ponownie użytych w iteracji,
- liczba wszystkich klas użytych w iteracji,
- liczba tworzonych metod w iteracji dla realizacji przypadku użycia.

Dzięki wykorzystaniu doświadczenia oraz już istniejących elementów systemu, czas potrzebny na implementacje przypadku użycia w kolejnych iteracjach uległ znacznemu skróceniu. Średni czas realizacji przypadku użycia w kolejnych iteracjach został przedstawiony na Rysunku 5.

Każda z iteracji trwała tydzień (pięć dni roboczych). W pierwszej iteracji został zrealizowany jeden przypadek użycia, w drugiej, w tym samym czasie, zaimplementowane zostały trzy przypadki użycia.

W trzeciej iteracji liczba wykonanych przypadków użycia wzrosła do siedmiu. Oznacza to, że w tym samym okresie możliwe było dostarczenie większego zakresu funkcjonalności systemu, co przekłada się na wzrost postępu prac i tym samym podniesienie poziomu efektywności zespołu projektowego. Efekt ten wynika:

- ze zwiększającego się wskaźnika wtórnego użycia kodu w kolejnych iteracjach,
- z nabywania doświadczenia, umiejętności i wiedzy dotyczącej stosowanej technologii,
- z wczesnej stabilizacji architektury systemu.

Wraz z postępem prac wzrastała także liczba klas powtórnie użytych w iteracji. W drugiej iteracji liczba ta wyniosła 6, a w trzeciej 12. Rysunek 6 przedstawia wykres liczby klas ponownie użytych oraz wszystkich klas wykorzystanych podczas implementacji w poszczególnych iteracjach.

Liczbę klas wytworzonych i powtórnie użytych w iteracji odniesiono także do liczby wszyst-

kich klas użytych w iteracji. Uzyskano procentowy udział obydwu tych liczb w liczbie wszystkich klas użytych w iteracji. Zostało to przedstawione na wykresie na Rysunku 7.

Ciekawe wyniki uzyskano również, obserwując wskaźnik pokazujący nakład pracy, wyrażony za pomocą liczby tworzonych metod w iteracji dla realizacji przypadku użycia. W pierwszej iteracji należało utworzyć 118 metod, aby zrealizować jeden przypadek użycia, w drugiej 42 metody, aby zrealizować 3 przypadki użycia, w trzeciej 71 metod, aby zrealizować 7 przypadków użycia. Liczba tworzonych metod w iteracji dla realizacji przypadku użycia kształtowała się następująco w iteracjach:

- I iteracja - 118 metod,
- II iteracja - 14 metod,
- III iteracja - 10 metod.

Realizacja przypadku użycia w kolejnych iteracjach odbywała się mniejszym kosztem nakładu prac, m.in. dzięki ponownemu użyciu już istniejącego kodu. W pierwszej iteracji stworzona została podstawowa architektura systemu, co pociąga za sobą potrzebę implementacji klas i ich metod stanowiących rdzeń systemu.

Istotnym aspektem jest fakt, że zastosowanie podejścia iteracyjnego umożliwia wczesne nabywanie doświadczenia i umiejętności, co przekłada się na podniesienie efektywności zespołu projektowego. Ponadto, wcześnie zdefiniowana i zweryfikowana architektura systemu niweluje ryzyko jego technicznej realizacji.

Otrzymane wyniki świadczą jednoznacznie o podnoszeniu w trakcie projektu efektywności zespołu projektowego. Postęp prac zwiększał się w każdej kolejnej iteracji przy jednoczesnym zmniejszaniu się nakładu prac.

Bibliografia

- [1] Barnes J., *Implementing the IBM Rational Unified Process and Solutions. A Guide to Improving Your Software Development Capability and Maturity.*, IBM Press, 2007,
- [2] Dokumentacja OpenUP Version 1.5.0.1,
- [3] Fowler M., *UML Distilled Third Edition*, Addison-Wesley, 2005,
- [4] Kan S. H., *Metryki i modele w inżynierii jakości oprogramowania*, Mikom, 2006,
- [5] Kroll P., MacIsaac B., *Agility and discipline made easy*, Addison-Wesley, 2006,
- [6] Kroll P., Krutchten P., *Rational Unified Process od strony praktycznej*, WNT, Warszawa 2007,
- [7] Kruchten P., *The Rational Unified Process, An Introduction*, Addison-Wesley, USA, 1999,
- [8] Krutchten P., *Rational Unified Process od strony teoretycznej*, WNT, Warszawa 2007,
- [9] *Manifesto for Agile Software Development*, www.agilemanifesto.org,
- [10] Olszewski J., *Metoda iteracyjnej budowy systemów informatycznych w architekturze J2EE*, praca magisterska WAT, 2009, kierownik dr inż. Tomasz Górski,
- [11] *RUP (Rational Unified Process)*, www.javatech.com.pl/rup.html,
- [12] Schwaber Ken, *Agile Poject Management with Scrum*, Microsoft Press, Washington 2004,
- [13] *Scrum in five minutes*, www.softhouse.se/Uploades/Scrum_eng_webb.pdf,
- [14] Takeuchi Hirotaka, Nonaka Ikujiro, *The New Product Development Game*, Harvard Business Review, 01-02.1986.

Podsumowanie

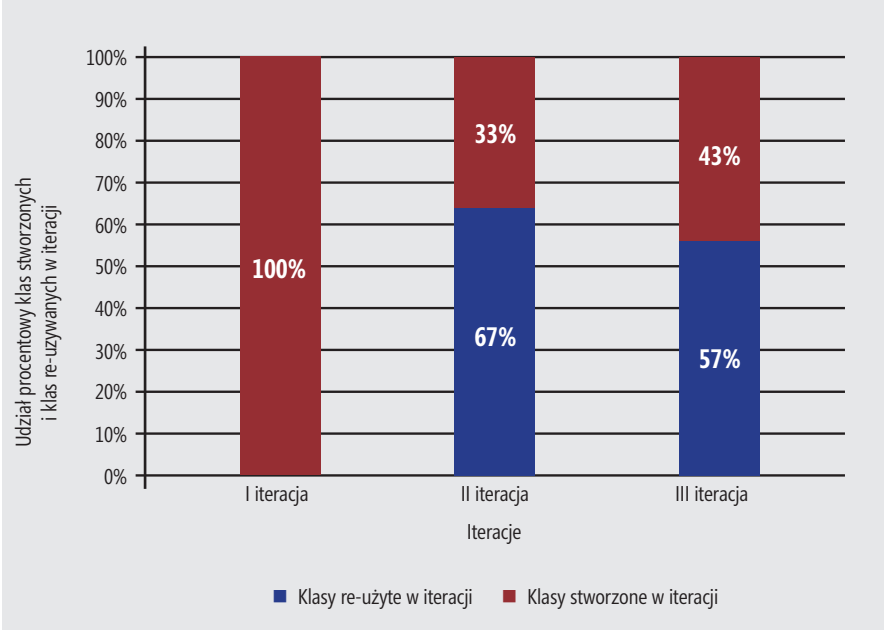
W artykule została zaproponowana metoda iteracyjnej budowy systemów informatycznych łącząca Rational Unified Process i SCRUM. W metodzie tej zaproponowano podejście ukierunkowane na architekturę i łączenie wytwarzania iteracyjnego z zaletami przebiegów SCRUM.

Metodyka ta została wykorzystana podczas budowy fragmentu systemu informatycznego wspomagającego pracę kancelarii prawniczej, ale także podczas realizacji w/w projektów komercyjnych.

Projekty realizowane zgodnie z przedstawioną metodyką charakteryzowały się następującymi cechami:

- podniesienie przewidywalności projektu,
- wczesna stabilizacja architektury systemu informatycznego,
- sukcesywne i przewidywalne dostarczanie produktów projektu,
- podniesienie ponownego użycia klas w systemie informatycznym,
- obniżenie liczby ponownie przepisywanych klas,
- skrócony czas realizacji przypadku użycia,
- zmniejszenie liczby metod potrzebnych do realizacji przypadku użycia.

Łączenie dwóch elementów: elastycznej organizacji zespołu projektowego oraz ścisłej kontroli architektury systemu informatycznego prowadzi do podnoszenia efektywności zespołu projektowego. Należy podkreślić, że podstawowym widokiem architektonicznym jest widok Przypadków użycia. Kluczowym jest utrzymywanie ciągłego porozumienia nad zebranym zestawem wymagań pomiędzy Zespołem projektowym a Zamawiającym i Użytkownikiem końcowym.



Rysunek 7. *Wykres przedstawiający udział procentowy klas wytworzonych oraz klas ponownie użytych w iteracjach*

dr inż. Tomasz Górski

Adiunkt w Instytucie Systemów Informatycznych, Wydziału Cybernetyki, WAT. Prowadzi firmę Right-Solution™ (Autoryzowane Centrum Edukacyjne IBM Rational) zajmującą się projektowaniem systemów informatycznych zgodnie z Rational Unified Process w architekturze SOA. Certyfikowany instruktor RUP. Zainteresowania naukowe i prace komercyjne skoncentrowane na procesie projektowania systemów informatycznych a w szczególności na projektowaniu platform integracyjnych dla organizacji sektora publicznego.

e-mail: *tomasz.gorski@rightsolution.pl*

Wywiad z Januszem Górskim

rozmawia Bartosz Chrabski

Czy inżynieria oprogramowania nadal potrafi nas naprawdę zaskoczyć nowymi odkryciami i podejściami do już znanych od lat problemów?

Celem inżynierii oprogramowania jest skuteczne rozwiązywanie, przy zastosowaniu środków z dziedziny technologii informacyjnych, problemów z praktycznie dowolnych dziedzin aplikacyjnych. A więc na przykład rozwiązanie problemu ułatwienia dostępu klientów do usług bankowych, niezależnie od ich lokalizacji geograficznej względem siedziby i oddziałów tego banku. Rozwiązanie polega tu na wdrożeniu bankowości internetowej. Jeżeli zgodzimy się co do takiego rozumienia inżynierii oprogramowania, to zaskoczenie, o które Pani pyta, może pojawić się z dwóch źródeł: od strony problemów, które są przedmiotem zainteresowania, oraz od strony środków, które są wykorzystywane podczas rozwiązywania tych problemów.

Tendencje po stronie problemów to przede wszystkim poszerzający się ich zakres, zmienność, presja czasu oraz otwartość.

Poszerzanie zakresu oznacza, że rozwiązania informatyczne są stosowane w nowych obszarach i do problemów, które nie były wcześniej objęte informatyzacją. Zmienność oznacza, że identyfikacja problemu praktycznie nigdy się nie kończy – trzeba liczyć się z tym, że możemy jedynie lepiej lub gorzej nadążać za zmianami. Odpowiedzią na to jest położenie większego nacisku na współpracę z udziałowcami reprezentującymi dziedzinę problemową oraz utrzymywanie ciągłości tej współpracy w pełnym cyklu życia oprogramowania. Presja czasu powoduje, że inżynieria oprogramowania szuka metod umożliwiających szybką dostawę rozpoznawalnej dla klienta wartości dodanej, nawet jeżeli będzie ona dostarczana w sposób niepełny i potem uzupełniana kolejnymi przyrostami. Te dwie ostatnie tendencje są odzwierciedlone w tzw. podejściach „zwinnych". Otwartość na integrację oznacza, że powstające rozwiązania techniczne nie mogą być „zamknięte" i użytkowane w izolacji. Dynamika w świecie biznesu, w sferze społecznej czy w gospodarce skutkuje potrzebą integrowania istniejących i nowych systemów.

W wymienionych wyżej obszarach bardziej niż o rewolucji trzeba raczej mówić o ewolucji, ale ewolucji, która ostatnio gwałtownie przyspieszyła.

Wśród tendencji po stronie środków metodologicznych i technicznych można wyróżnić położenie większego nacisku na ciągły kontakt z kluczowymi udziałowcami, zarządzanie projektem w sposób umożliwiający „zwinne" nadążanie za zmianami, architektury SOA i nacisk na integrację procesów biznesowych oraz poszukiwanie abstrakcji umożliwiających reprezentowanie oprogramowania w sposób niezależny od jego technicznej realizacji. W coraz większym stopniu rozpoznawane jest również znaczenie czynnika ludzkiego, zarówno po stronie analizy i oceny systemów, jak i po stronie wytwórczej.

W mojej ocenie po stronie technologicznej jest większa szansa na zaskoczenie. Technologie informacyjne już nie raz dowiodły, że ich rozwój może przyczynić się do zmian o charakterze rewolucyjnym. Przyczyną jest to, że mogą one otwierać nowe i nieprzewidziane wcześniej możliwości w dziedzinach problemowych, tak jak Internet otworzył drogę do nowych modeli świadczenia usług i w znacznym stopniu uwolnił biznes od ograniczeń geograficznych, a telefonia komórkowa zrewolucjonizowała strukturę komunikacyjną w wymiarze indywidualnym i biznesowym. Takie zmiany w infrastrukturze inspirują nowe i gwałtownie rozwijające się potrzeby, a jednocześnie tworzą warunki do nowych i bardziej skutecznych sposobów zaspokajania tych potrzeb. Co będzie takim „zapalnikiem" w najbliższej przyszłości? *SOA* i integracja procesów biznesowych? *Cloud computing* jako nowy i uniwersalny model dostępu do usług *IT*? A może będzie to coś, co się „narodzi" na przecięciu informatyki i biotechnologii? Trudno przesądzać, tym bardziej że ludzie wielokrotnie już dowiedli, że przewidywanie przyszłości nie bardzo im się udaje. Ja przewiduję przesuwanie się inżynierii oprogramowania w stronę „inżynierii systemów" i związany z tym rosnący rynek pracy dla specjalistów łączących kompetencje z dziedzin aplikacyjnych z umiejętnościami obszaru analizy, (re)inżynierii procesów biznesowych oraz architektury oprogramowania i infrastruktury komunikacyjnej. Natomiast kompetencje odnoszące się do wytwarzania oprogramowania w sensie „umiejętności programowania" będą nie tyle zanikać, co lokalizować się w miejscach, gdzie takie oprogramowanie będzie tworzone. Czy zmiany te przybiorą charakter rewolucyjny, czy będzie to ewolucja – zobaczymy.

Czy w ostatnich latach powstały jasno identyfikowalne trendy w dziedzinie inżynierii oprogramowania, na które warto jest zwracać uwagę?

Niewątpliwie jest ich kilka. Wszystkie one w jakiś sposób narastają stopniowo, a więc ich identyfikacja jest bardziej ekstrapolacją historii i stanu obecnego niż wizją jakiejś nowej zaskakującej zmiany. Przewiduję tu narastającą tendencję do poszerzania perspektywy w postrzeganiu oprogramowania w wymiarze systemowym (a więc bardziej *systems engineering* niż *software engineering*) i w związku z tym większy nacisk na poza-funkcjonalne własności związane z gwarancjami takimi jak bezpieczeństwo (*safety*), zabezpieczenie (*security*), prywatność czy zdolność systemów do przeżycia, pomimo niesprzyjających warunków wewnętrznych czy zewnętrznych. W tendencję tę wpisuje się również większa koncentracja na użytkownikach i dostarczanej im wartości dodanej. Spowoduje to rosnący nacisk na przekraczanie barier w komunikacji i współpracy między reprezentantami różnych dziedzin i organizacji oraz rosnące znaczenie pracy grupowej. Szybkość i zakres zmian oraz konieczność nadążania za zmianą utrzymają zainteresowanie metodami „zwinnymi" oraz traktowanie przedsięwzięć informatycznych jako procesów, które bardziej nadążają za „ruchomym celem" niż są realizowane według przygotowanego wcześniej szczegółowego planu. Utrzyma się tendencja do integracji i związany z tym wzrost złożoności systemów. Z jednej strony wymagać to będzie środków technicznych wspomagających taką integrację, z drugiej zaś strony niezbędne będą abstrakcje i metody umożliwiające przygotowanie tworzonych systemów do przyszłej integracji oraz zapanowanie nad wynikającą stąd złożonością. Wiąże się z tym również zagadnienie gotowych komponentów (*COTS – Commercial off-the-shelf*), czy szerzej, wielokrotne wykorzystanie istniejących już komponentów (*software re-use*). Rosnącego znaczenia będzie nabierać integracja systemów (*systems-of-systems*) nie tylko w wymiarze funkcjonalnym, ale również (a może nawet bardziej) w wymiarze związanych z nimi gwarancji poza-funkcjonalnych (takich jak bezpieczeństwo, dostępność czy niezawodność). Integracja ta będzie musiała również uwzględniać istnienie oprogramowania „odziedziczonego" (*legacy systems*).

Jakie obszary według Pana są najmniej zbadane i wymagają ciągłej pracy nad analizą tej dziedziny?

Pole do badań jest bardzo obszerne. W końcu mamy do czynienia z dziedziną, która nie „żyje" jeszcze nawet jednego pełnego stulecia. Której zakres oraz spektrum możliwych zastosowań są wciąż dalekie od pełnej identyfikacji. Która wciąż nie ma wspólnego i powszechnie akceptowanego systemu pojęć i związanego z nim języka. Nie

do końca wiemy, jak powtarzać sukces w budowie oprogramowania, nie rozumiemy i nie uzgodniliśmy kryteriów dla takiego sukcesu, mamy do czynienie z budulcem, który jest ulotny i może zmieniać swoje własności, nie mamy satysfakcjonującej wiedzy na temat procesów, w ramach których powstaje i jest eksploatowane oprogramowanie. Obszarów tych jest tak dużo, że trudno je wszystkie wymienić. Stan dziedziny jest w jakiś sposób odzwierciedlony w corocznych raportach *Standish Group*, które podają statystyki dotyczących udanych i nieudanych przedsięwzięć informatycznych. Dwie obserwacje w stosunku do tych danych są uderzające: (1) w pełni udanych jest około 30% przedsięwzięć (inne mają mniejsze lub większe trudności) oraz (2) proporcje te nie podlegają znaczącej zmianie wraz z upływem czasu. W jakiś sposób mówi to o stanie inżynierii oprogramowania i wskazuje na ogrom problemów czekających rozwiązania.

Osobiście uważam, że potrzebna jest eksploracja obszaru pomiędzy dwoma zauważalnymi tendencjami w podejściu do inżynierii oprogramowania: z jednej strony nadawanie priorytetu metodom i narzędziom, a z drugiej ludziom i ich profesjonalizmowi. Kierunki te w jakiś sposób odzwierciedlają różnice pomiędzy „sztywnym" i „zwinnym" podejściem do wytwarzania oprogramowania. Wydaje mi się, że bardzo ważna jest głębsza eksploracja i zrozumienie obszaru, który mieści się pomiędzy tymi skrajnościami.

Inny obszar, który mnie szczególnie interesuje, to rola zaufania i środki budowy zaufania w odniesieniu do procesów i produktów związanych z oprogramowaniem. Ma to silne przełożenie na takie własności poza-funkcjonalne jak bezpieczeństwo, zabezpieczenie czy prywatność. Wspólnym mianownikiem, który łączy te obszary, jest zarządzanie ryzykiem.

Oba wymienione wyżej obszary są przedmiotem badań w kierowanej przeze mnie grupie badawczej, która działa w Katedrze Inżynierii Oprogramowania Politechniki Gdańskiej.

Czy możliwe jest uporanie się z budową nowoczesnych, skomplikowanych systemów IT bez odpowiedniego warsztatu narzędzi?

Jak w każdej innej dziedzinie, narzędzia są potrzebne i mogą okazać się nadzwyczaj przydatne. Szczególnie gdy rozpatrujemy problemy skali, czy to w odniesieniu do złożoności, czy do wydajności lub kosztu. Narzędzia mogą również skutecznie chronić człowieka przed popełnianiem błędów i zdecydowanie zwiększają szansę na powtarzalność rezultatów. Nie wyobrażam sobie, by skuteczne podjęcie wyzwań, o których wspominałem wcześniej, było możliwe bez wsparcia narzędziowego. Należy jednak pamiętać, że programy budują ludzie, a nie metody i narzędzia. Oznacza to w szczególności, że narzędzi nie można rozpatrywać w izolacji od używających ich ludzi oraz zadań i procesów, w które są oni zaangażowani.

Jakie wyzwania według Pana stają przed nowoczesnymi narzędziami wspierającymi proces wytwórczy na rynku?

Oczywistym wydaje się postulat, by narzędzia wspierały osiąganie celów, do których dążą ich użytkownicy. Tak więc cele wydają się być nadrzędne nad narzędziami, tzn. narzędzie, które nie wspiera zidentyfikowanego celu, jest bezużyteczne. Jednak związek ten nie jest aż tak wyraźnie jednokierunkowy. Zauważmy, że niezwykle użyteczne narzędzie, młotek, przydaje się w bardzo wielu zastosowaniach, niekoniecznie oczywistych wtedy, gdy narzędzie to powstało po raz pierwszy. Istnieje więc jeszcze inne kryterium oceny narzędzi, które można by nazwać ich uniwersalnością, a więc zdolnością do użycia w różnych celach, niekoniecznie z góry przewidzianych. Uważam to za bardzo ważne, gdyż chroni to narzędzie przed moralnym starzeniem się i zwiększa jego zdolność do „przeżycia". W przeciwieństwie do narzędzi zbyt wyspecjalizowanych, które się szybko dezaktualizują. Na gruncie informatyki, ze względu na zmienność w tej dziedzinie, istnieje wcale pokaźne „cmentarzysko" takich narzędzi.

Kolejny dylemat, to złożoność funkcjonalna. Przez analogię, czy lepiej używać młotka z końcówką do wyciągania gwoździ, czy też używać młotka „konwencjonalnego" i obcęgów? W pierwszym wypadku mamy dwie funkcje złożonym narzędziu, w drugim dwa narzędzia, każde do realizacji jednej prostej funkcji. Zauważmy, że w tym wypadku odpowiedź zależy od tego, jak wygląda proces użytkowania tych narzędzi. Bez analizy tego procesu trudno jest dać jednoznaczną odpowiedź. Jednak z punktu widzenia przyszłej ewolucji, zmian funkcji już istniejących lub dodawania nowych, rozwiązanie drugie, w duchu *unixowym*, wydaje się mieć przewagę. Tym bardziej podkreśla to konieczność zadbania o to, by narzędzia mogły ze sobą współpracować. A to z kolei zwraca uwagę na standaryzację dotyczącą takiej współpracy. I na konieczność właściwego rozumienia pojęcia „standard". W pojęciu tym kryje się uzgodnienie i akceptacja, bez których „standard" standardem nie jest. Na tym tle dość chwiejne są tezy o tym, że nowo pojawiające się na rynku narzędzie „wprowadza standard" w dotyczącym go zakresie. Przy braku powszechnej akceptacji okres życia takich „standardów" jest czasem zaskakująco krótki.

Uniwersalność nie pozostaje w sprzeczności ze specjalizacją, jeżeli w architekturze narzędzia wyraźnie oddzielimy to, co uniwersalne, od tego, co wyspecjalizowane. W części wyspecjalizowanej można dopuścić do wyraźnego ukierunkowania na określoną metodę, funkcjonalność czy sprofilowanie użytkowników. Rozróżnienie takie wspomaga przystosowywanie narzędzia do zmian i w efekcie jego przeżywalność na rynku.

Narzędzi, za wyjątkiem być może tych najbardziej uniwersalnych, nie można rozpatrywać w oderwaniu od kontekstu wspierającego ich rozwój i zastosowania. Wielokrotnie obserwowałem narzędzia (i związane z nimi metody), które szybko obumierały, gdy kończyło się związane z nimi wsparcie. Oprócz tych kryteriów ogólnych jest oczywiście bardzo ważną sprawą, by narzędzia odpowiadały na aktualne potrzeby. Niektóre z tych potrzeb wynikają z odpowiedzi, które udzieliłem na poprzednie pytania.

No i jeszcze jedna sprawa, którą uważam za ważną. Ze względu na rosnące znaczenie pracy grupowej i rozproszenie użytkowników, uważam, że planując rozwój narzędzi, trzeba brać pod uwagę ich wersje „internetowe" z cienkim lub pogrubionym klientem. Niewpisanie się w ten trend grozi utratą rynku lub zdecydowanym jego zawężeniem. Warto również poważnie rozważać model biznesowy polegający na udostępnianiu usług związanych z narzędziami, a nie model zakładający sprzedaż narzędzi jako produktów. Ten sposób myślenia o narzędziach może okazać się bardzo trafny w kontekście (być może) nadciągającej rewolucji związanej z *cloud computing*.

Janusz Górski (prof. dr hab. inż., prof. zw. PG) kieruje Katedrą Inżynierii Oprogramowania na Politechnice Gdańskiej. W pierwszej połowie lat 90. utworzył pierwszą w kraju specjalizację z zakresu inżynierii oprogramowania na studiach wyższych. W drugiej połowie lat 90. utworzył pierwsze w kraju studium podyplomowe inżynierii oprogramowania. W roku 1999 był inicjatorem i pierwszym przewodniczącym Krajowej Konferencji Inżynierii Oprogramowania, która do chwili obecnej jest główną platformą grupującą badaczy tej tematyki w kraju. Prowadził i konsultował wiele projektów rozwojowych i badawczych zarówno w kraju, jak i na forum międzynarodowym, w tym również w programach ramowych UE (programy ENVIRONMENT, COPERNICUS, 5. i 6. Programy Ramowe). Jest ekspertem Komisji Europejskiej w 5. 6. i 7. Programach Ramowych. Jest również doradcą Dyrektora European Network and Security Agency (ENISA) w zakresie kierunków rozwoju Agencji oraz oceny wybranych obszarów badawczych. Jego obecne zainteresowania dotyczą inżynierii oprogramowania (w szczególności problemów związanych ze skutecznym pozyskiwaniem oprogramowania przez klientów) oraz zrządzania ryzykiem dotyczącym procesów i produktów programistycznych. W szczególności w odniesieniu do ryzyka związanego z bezpieczeństwem, zabezpieczeniem i prywatnością. W kierowanym przez niego zespole (http://iag.pg.gda.pl) rozwijana jest innowacyjna w skali międzynarodowej metodologia Trust-IT, ukierunkowana na wspomaganie zarządzania argumentacją i wykorzystanie argumentów w budowie zaufania i w innych obszarach zastosowań. Jednym z takich obszarów jest wspomaganie procesów dochodzenia i oceny zgodności z normami i standardami (projekt NOR-STA, www.nor-sta.eu realizowany w ramach Programu Operacyjnego Innowacyjna Gospodarka).

O zwinnym tworzeniu oprogramowania

Bardzo miło wspominam film Christophera Nolana „Batman – Początek", który zgłębia genezę legendy o Batmanie i narodziny Mrocznego Rycerza jako stróża porządku w Gotham. Znajomość *początku* jest ważna, porządkuje wiele rzeczy, pozwala zrozumieć całość. Dzisiaj będzie o tym, jak narodziły się „zwinne" metody tworzenia oprogramowania – będzie o tym, jak narodził się *agile*.

Z artykułu dowiesz się:

- Czym różnią się „zwinne" metody tworzenia oprogramowania od tradycyjnych podejść?
- Jak myślenie odwrotne wpływa na rozwój oprogramowania?
- Dlaczego agile jest filozofią?
- O korzyściach płynących z wdrożenia „zwinnych" metod w organizacji.

Powinieneś wiedzieć...

...że zwinność wymaga dyscypliny

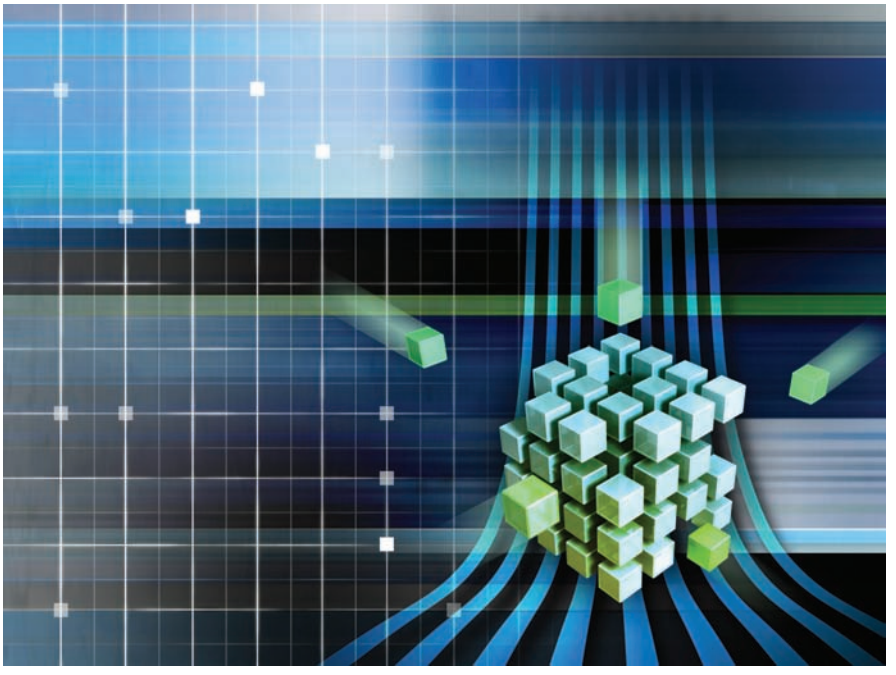
Tradycyjne podejście

Z powstaniem „zwinnych" (ang. *agile*) metod tworzenia oprogramowania jest trochę tak jak z powstaniem życia na ziemi. Podstawowe idee tych metod, wyznawane wartości i zasady ewoluowały wraz z dyscypliną inżynierii oprogramowania, a więc od początku jej istnienia (przełom lat 1950/ 1960). Niewątpliwym katalizatorem, który przyczynił się do ich rozwoju, był świat tradycyjnych metod wytwórczych, które najpełniej wyraża podejście zwane kaskadowym (od ang. *waterfall*). Polega ono na tym, że aktywności projektowe realizowane są liniowo (sekwencyjnie) – płyną niczym Nil, tworząc imponujące kaskady wodne (dwie z nich mają postać potężnych wodospadów – nazywanych Ripon i Owena). Cykl życia projektu dzielony jest na określone fazy, które wzajemnie od siebie zależą. Najpierw zbierane są potrzeby klienta (wymagania biznesowe), potem tłumaczy się je na język zrozumiały dla programistów (wymagania funkcjonalne), żeby można było zaprojektować określone rozwiązania techniczne (projekt systemu). Kolejna faza to implementacja wybranych rozwiązań, które są kolejno: integrowane, testowane i podtrzymywane (ang. *maintenance*), w wyniku wdrożenia. Proszę zwrócić uwagę, że każda faza w tym podejściu stanowi domkniętą całość. Jej produkty wyjściowe (*outputs*) stanowią coś, bez czego nie jest w stanie ruszyć praca w fazie następnej (*inputs*). Metodyka kaskadowa opiera się jednak na jednym, bardzo zgubnym – jak się okazuje– założeniu, że proces tworzenia oprogramowania jest procesem, który niczym nie różni się od procesu produkcyjnego. Jest linia produkcyjna, są stanowiska robocze (maszynowe, ręczne lub mieszane), pogrupowane wg kolejnych operacji procesu technicznego. Idea „linii produkcyjnej", w momencie powstania, była alternatywą dla dotychczasowej produkcji rzemieślniczej i jej powstanie stanowiło niekwestionowaną zasługę amerykańskiego koncernu Ford. Inżynierowie oprogramowania popełnili jednak zasadniczy błąd, próbując przenieść ten pomysł na grunt projektów softwarowych.

Bardzo dobrze pamiętam problemy, które wynikają ze stosowania tej metody, a którym jako początkujący Kierownik Projektów IT musiałem stawić czoło. Przede wszystkim dość szybko doszedłem do wniosku (jestem przekonany, że znacie to z własnego podwórka), że *stałe wymagania w projekcie są tak rzadkie jak role oscarowe u Arnolda Schwarzeneggera*. Wyobraźcie sobie sytuację, że skończyliście prace, związane z przygotowaniem niskopoziomowego projektu systemu (ang. *Low Level Design*). Nagle dzwoni klient i mówi, że dwie ostatnie funkcjonalności, o których rozmawialiście 5 dni temu, chciałby jednak trochę rozbudować i dodatkowo dorzucić jedną nową. Do dzisiaj myśleliście, że wszystko jest pod kontrolą. Nagle okazuje się, że grawitacja zmienia swój bieg, a czasoprzestrzeń zakrzywia się w nieprawdopodobny wręcz sposób. Scena żywcem z „Incepcji" Christophera Nolana. Chciałoby się włamać przez sen do świadomości klienta i zaszczepić mu ideę cierpliwego czekania i nie „wrzucania" nam dodatkowej roboty. Ale... nie ma sprawiedliwości na tym świecie. Musimy kilka rzeczy przeprojektować, przeplanować i starać się jakoś podnieść morale sfrustrowanego zespołu. Nie muszę już chyba dodawać, jak bardzo takie „wrzutki" zwiększają koszty prowadzonego projektu.

Kolejny problem, który napotkałem przy okazji stosowanie modelu kaskadowego, to: *formalne pozbawienie prawa głosu naszego klienta w trakcie prac rozwojowych*. Celowo napisałem „formalne", gdyż klient i tak kontaktował się z nami na wiele różnych możliwych sposobów i zgłaszał swoje widzimisię. I nic nie mogliśmy zrobić. Nie pomagała eskalacja tego problemu do wyższego kierownictwa, które jak jedna z zabawek kupionych w „Smyku" natrętnie powtarzało:

– Takie jest życie! Dobrze wiesz, że jest to nasz strategiczny partner (czyt. dojna krowa) i musimy to jakoś znieść. Głowa do góry! Następnym razem będzie lepiej. Nie było.



Dużym problemem w korzystaniu z metodyki kaskadowej jest również *ucieczka błędów*. Wyobraźcie sobie, że będąc na etapie testów systemowych, nagle dowiadujecie się, że określone rozwiązanie zostało źle zaprojektowane, i musicie wrócić, rozgrzebać architekturę systemu, zaimplementować odpowiednie poprawki, zrobić testy i finalnie wdrożyć wszystko na środowisko produkcyjne. W tym momencie wasze plany biorą w łeb. To jest trochę tak, jak z wyjazdem na weekendowy biwak na Mazury. Pakujemy się: śpiwór, karimata, ubrania, ręczniki (oczywiście wszystko razy cztery, chyba że nie bierzemy żony i dzieci), buty, kosmetyczka, latarka, zapałki, saperka, ładowarka do Nokii, konserwy... Wreszcie wyjeżdżamy z Krakowa. Nawet nam sprawnie poszło, mimo że piątek i 17.00. Ujechaliśmy może ze dwie godziny, nagle żona, patrząc błędnym wzrokiem na przejeżdżające sąsiednim pasem auta, pyta: – Krzysiek, a namiot? Możecie sobie wyobrazić, jaki jest dalszy ciąg tej historii. Z modelem kaskadowym jest podobnie. Im później się zorientujemy, że nie mamy namiotu, tym powrót po niego jest droższy.

Winston Royce, który jako pierwszy w artykule *Managing the Development of Large Software System* (1970) opisał model kaskadowy (choć nie użył wtedy ani razu słowa *waterfall*), komentując to podejście, powiedział bardzo ciekawą rzecz: „Podoba mi się ten pomysł, ale jego implementacja jest ryzykowna i skłonna do błędów". Jest swoistym paradoksem, że wiele osób uważa tego człowieka za twórcę metodyki kaskadowej – człowieka, który widział w niej duże zagrożenie.

Myślenie odwrotne

Metody agile'owe powstały w wyniku próby spojrzenia na proces tworzenia oprogramowania w nieco inny – odwrotny sposób. Na myśl przychodzi mi tutaj historia, którą przeczytałem w książce Paula Ardena *Cokolwiek myślisz, pomyśl odwrotnie*. Rzecz miała miejsce przed olimpiadą w Meksyku, która odbyła się w 1986 roku. Był to czas, kiedy technika skoków wzwyż polegała na tym, że skoczkowie ustawiali się równolegle do poprzeczki, „przerzucając" swoje ciało w trakcie skoku. Na wspomnianej olimpiadzie pojawił się mało znany wtedy zawodnik Dick Fosbury, który podbiegł do poprzeczki, ustawionej na rekordowej wysokości 2 metrów 24 centymetrów. Wybił się, i zamiast skoczyć jak wszyscy, ustawił swoje ciało w kierunku poprzeczki, obracając się do niej plecami. Unosząc nogi, pokonał poprzeczkę tyłem. Jego styl skakania obowiązuje do dzisiaj i zasłynął jako „flop Fosbury'ego". Dick skoczył wyżej, niż ktokolwiek przedtem, bo odważył się (!) myśleć inaczej niż wszyscy. Metody agile to właśnie przykład „myślenia odwrotnego" względem pokonywania poprzeczki w tradycyjny sposób, czyli stosując podejście kaskadowe. Dzisiaj, przeglądając zdjęcia osób wykonujących skoki wzwyż sprzed 1986 roku, od razu uderza nas sztuczność techniki „przerzutowej". Jak można było tak nienaturalnie skakać? A jednak. Ktokolwiek prowadził projekty metodą kaskadową i „przeskoczył" na „zwinne" metody pracy, bardzo szybko dojdzie do podobnych wniosków, jak przy oglądaniu zdjęć skoczków sprzed słynnego „flopu Fosbury'ego".

„Zwinne" metody tworzenia oprogramowania wbrew pozorom wcale nie są nowe. Można powiedzieć, że pewne idee, które tworzą zrąb tego wszystkiego, co określamy słowem „agile", powstawały równolegle do tradycyjnego podejścia tworzenia softwaru. Nie można tutaj nie wspomnieć o takich osobach jak Gerald M. Weinberg, Frederick P. Brooks, Tom De Marco, Timothy Lister, którzy już począwszy od lat 70. podkreślali znaczenie „myślenia odwrotnego" w tworzeniu oprogramowania i odejścia od mentalności, zaczerpniętej ze świata produkcji, która tworzenie softwaru każe nam porównywać z produkcją hamburgerów. „Myślenie odwrotne" w inżynierii oprogramowania to także, a może przede wszystkim, zwrócenie uwagi na LUDZI – analityków, programistów, testerów, kierowników projektów, od których koniec końców zależy sukces prowadzonych projektów.

Manifest Agile

Równolegle z kształtowaniem się nowej fali „myślenia odwrotnego", jako opozycji do tego wszystkiego, co wiązało się z tradycyjnym rozwojem oprogramowania – rodziły się konkretne praktyki, stosowane przez *ambasadorów zmiany*. Lata 80. i 90. to czas stosowania alternatyw, w pewnym sensie uczenia się na błędach, wynikających z próby zaszczepienia idei linii produkcyjnej do świata tworzenia oprogramowania. Jest to okres, w którym różne grupy praktyków, na swój własny i niczym nie skrępowany sposób, zaczynają tworzyć nową rzeczywistość, która później zostanie nazwana słowem: „agile". Lata 80. i 90. to również czas, kiedy próbuje się nazywać i systematyzować metody agile'owe. To wtedy, w roku 1986 zaczyna być głośno o metodzie Scrum; zostaje opracowana *Dynamic Systems Development Methodology* (1994); Kent Beck nazywa praktyki Extreme Progamming (1996, chodź prawdziwy bum tej metody to tak naprawdę rok 2000); Alistar Cockburn mówi o metodach Crystal, a Jeff DeLuca publikuje *Feature-Driven Development* (1998). Nowa fala praktyk nabiera rozmachu.

W roku 2001, w ośrodku wypoczynkowym Snowbird, w USA (stan Utah) zbiera się grupa zwolenników nowego podejścia, celem nazwania tego, co tak naprawdę charakteryzuje rodzące się metody. W efekcie, zostaje opracowany tzw. *Manifesto for Agile Software Development* (z ang. Manifest Zwinnego Tworzenia Oprogramowania), który stanowi deklarację podstawowych wartości i zasad *agile* (w całości do przeczytania na: *http://agilemanifesto.org/*). Pierwsza część manifestu to cztery krótkie stwierdzenia, które w sposób prosty oddają *filozofię zwinności*. Mówią o tym, jakie wartości zwolennicy „lekkich" metod cenią sobie najbardziej:

- **Ludzie i interakcje ponad procesami i narzędziami**
  Można mieć świetnie zdefiniowane procesy, kupić bardzo drogie narzędzia, ale i tak, koniec końców, największy wpływ na powodzenie naszych projektów mają ludzie zaangażowani w ich rozwój. Czynnik ludzki jest niestety elementem, które-

go bardzo brakuje we wszystkich modelach i standardach zaawansowanych procesów wytwórczych, jakim jest np. model CMMI. Oczywiście można odpierać ten atak, mówiąc, że model ten ma trochę inne zastosowanie – jego zadaniem jest nakreślić mapę drogową, która pomoże zorganizować świat procesów w firmie. Niemniej prawda jest taka, że w praktyce nie zawiera żadnych mechanizmów, które by wpływ tego czynnika uwydatniały – czy to na poziomie życia organizacji, czy porządkując różnego rodzaju działania projektowe. Oczywiście umożliwia wprowadzenie określonych „zwinnych" praktyk, które promują pracę zespołową, wzmacniają komunikację interpersonalną, jednak w żaden sposób nie zapewnia, że zostaną wprowadzone.

- **Działające produkty ponad złożoną dokumentację**
  Czytając to stwierdzenie manifestu, przypominam sobie rozmowy, prowadzone przy okazji różnego rodzaju wdrożeń – rozmowy z programistami, którzy narzekali na prowadzenie i utrzymywanie dokumentacji w ich projektach. Często mieli wrażenie, że poruszają się między jedną a drugą ryzą papieru; że w efekcie produkują stosy dokumentów, które są generowane, bo taka jest polityka firmy i tego wymaga od nich menedżment. A tymczasem klienta i tak interesuje działający produkt, który jest wolny od defektów i dostarczony na czas. Dlatego dokumentacja powinna raczej wspierać rozwój produktów, a nie go utrudniać.
- **Współpraca z klientem ponad negocjacją kontraktu**
  Tutaj, ale i przy wszystkich pozostałych wartościach i zasadach manifestu, należy pamiętać, że jest jeszcze realny świat naszych projektów i wszystko to, co tworzy ich niepowtarzalną rzeczywistość, w której żyjemy. Dlatego, nawet jeżeli wasi klienci kupią idee filozofii zwinności, to jednak zawsze będą chcieli się jakoś zabezpieczyć (a na pewno ich dział finansowy czy prawny), na wypadek, gdyby coś poszło nie tak. Ciągła współpraca z klientem na każdym etapie prac rozwojowych jest jedną z podstawowych charakterystyk projektów agile'owych, która jest odpowiedzią na metody kaskadowych to zmienia my na kaskadowe, gdzie podpisanie kontraktu z klientem stanowiło o tym, czy dany projekt wystartuje, czy nie. Filozofia agile to obecność klienta na każdym etapie prac rozwojowych. Praca programistów osadzona jest bardzo mocno na informacji zwrotnej, którą dostarcza klient.
- **Reagowanie na zmiany ponad trzymaniem się planu**
  Osoby, które kiedykolwiek miały do czynienia z tradycyjnymi metodami tworzenia oprogramowani, bardzo dobrze pamiętają te chwile, kiedy w trakcie implementacji pojawiały się nowe wymagania lub klient nagle coś zmieniał. Zmiana w trakcie kolejnych faz rozwojowych była wtedy najmniej pożądaną rzeczą. Mogliśmy przeżyć brak kawy w firmie, ale nie kolejne „wrzutki" klienta. Zmiana była czymś obcym, niechcianym. Baliśmy się jej jak ognia. Dużą zasługą metod agile jest „oswojenie" zmiennych życzeń klienta w trakcie prac rozwojowych. Zmiana jest dobra, jest niezmiennym elementem naszych prac wytwórczych. Oczywiście to nie oznacza, że nasze projekty nie powinny mieć planu i planowanie jest niepotrzebne. Przeciwnie! – plan jednak, musi być każdorazowo dopasowywany do zmieniających się warunków środowiskowych.

Podstawowe wartości manifestu zostały dodatkowo wzbogacone o 12 zasad, które można potraktować jako swoistą listę kontrolną „zwinnego" projektu. Można się z nimi zapoznać na wspominanej już stronie http://agilemanifesto.org/.

Jedna filozofia, różne metody

Wartości oraz zasady manifestu wyrażają wszystko to, co składa się na *filozofię zwinności*. Tak więc *agile* wyraża bardziej pewną postawę, niż konkretne podejście. Jest to postawa otwartości na: komunikację interpersonalną, pracę zespołową, kreatywność, samoorganizowanie się. W tym horyzoncie napotykamy konkretne praktyki, które w różnych konstelacjach tworzą określone metody. Do najbardziej znanych należą:

- **Scrum** – to innowacyjna metoda zarządzania projektami, niekoniecznie informatycznymi. Bardzo mocno akcentuje ideę samoorganizujących się zespołów. Do jej kluczowych praktyk należą: realizacja projektu w krótkich, maksymalnie 30-sto dniowych iteracjach, zwanych „sprintami"; spotkania planistyczne na początku każdej iteracji; krótkie, codzienne spotkania projektowe (tzw. Daily Scrums); prezentacja wyników iteracji interesariuszom projektu, a także retrospektywa, podsumowująca przebieg prac w iteracji. Scrum definiuje specyficzne role projektowe: ScrumMaster, Product Owner (Właściciel Produktu) oraz Zespół (idealny 5-9 osób).
- **Extreme Programming** – prawdopodobnie jedna z bardziej znanych „zwinnych" metod tworzenia oprogramowania. Opiera się na 4 wartościach: komunikacji, prostocie, informacji zwrotnej oraz odwadze. Można w niej wyróżnić 12 podstawowych praktyk, z których część to praktyki *stricte* programistyczne. Należą do nich: ciągła integracja, test-driven development, programowanie w parach, refaktoryzacja kodu. Extreme Programming bardzo często stanowi naturalne dopełnienie metody Scrum, która jako taka nie zawiera żadnych praktyk inżynieryjnych.
- **Feature-Driven Development (FDD)** – jest to próba przetłumaczenia idei „zwinności" na język stosowany w tradycyjnych projektach IT. Metoda FDD wyróżnia 5 podstawowych procesów wytwórczych: (1) Stworzenie ogólnego modelu, (2) Opracowanie szczegółowej listy funkcjonalności oraz ustalenie ich priorytetów, (3) Planowanie implementacji funkcjonalności, (4) Projektowanie funkcjonalności oraz (5) Implementacja funkcjonalności. Procesy: #4 i #5 są realizowane iteracyjnie. FDD definiuje konkretne role projektowe, które w odróżnieniu od metody Scrum pozostają raczej w konwencji projektów kaskadowych. Są to: Główny Architekt (ang. *Chief Architect*), Właściciel Klasy (ang. *Class Owner*) oraz Zespół Funkcjonalny (ang. *Feature Team*).
- **Dynamic Systems Development Methodology** – metoda ta wyróżnia 3 fazy projektowe: (1) *Faza Przed-Projektowa*, w której następuje identyfikacja i wybór potencjalnych projektów do realizacji, definiowany jest budżet projektu oraz podejmowane określone zobowiązania, (2) *Faza Cyklu Życia Projektu*, w trakcie której są uzgadniane, projektowane i implementowane konkretne funkcjonalności, (3) *Faza Post-Projektowa*, której celem jest dbałość o wydajne funkcjonowanie wdrożonego systemu. W tej fazie realizowane są wszystkie aktywności związane z podtrzymaniem systemu (ang. *maintenance*), wdrażane są określone usprawnienia (ang. *enhancements*) i poprawki. Metoda DSDM definiuje aż 12 ról projektowych, których ze względu na ograniczenia tekstowe nie będę przytaczał.
- **Crystal** – to podejście nie jest metodą w rozumieniu określonych ram postępowania, które wyznaczają cykl życia projektu. Crystal należy traktować raczej jako model – zbiór bardzo konkretnych reguł, które sprawdziły się i są wykorzystywane w różnych metodykach projektowych. Model Crystal opiera się na założeniu, że nie istnieje jedna uniwersalna metoda, która miałaby zastosowanie we wszystkich projektach. Crystal to zbiór dobrych praktyk, które przynależą do różnych metod i które można dowolnie konfigurować, tworząc własne metody dopasowane do określonego kontekstu i specyfiki

własnych projektów. Do najważniejszych należą: (1) częste dostarczanie fragmentów działającego produktu do klienta (ang. *frequent delivery*), (2) osmotyczna komunikacja (ang. *osmotic communication*) oraz (3) refleksyjne doskonalenie (ang. *reflective improvement*).

Warto bo…

Jak się łatwo domyśleć, metody agile'owe mają swoich zwolenników i przeciwników. Ja należę zdecydowanie do tej pierwszej grupy, co wydaje się być trochę dziwne, bo jestem dość mocno związany z modelem CMMI, który zupełnie niesłusznie kojarzony jest z tradycyjnym stylem prac wytwórczych.

Metody agile'owe przede wszystkim cementują pracę zespołową w projekcie. To jest coś, czego bardzo brakuje w projektach prowadzonych metodą kaskadową. Doskonale pamiętam tego typu projekty z czasów, gdy pracowałem jeszcze w jednej ze znanych korporacji. Za wymagania i testy odpowiadały dwa, zupełnie odrębne zespoły. My w zasadzie zajmowaliśmy się analizą, pisaniem dokumentacji i kodowaniem. Przepływ informacji i komunikacja interpersonalna były mocno ograniczone. Duży nacisk kładliśmy na dokumentację projektową, a nie potrafiliśmy wydobyć tzw. „wiedzy ukrytej", którą ma każdy członek zespołu projektowego w swojej głowie, a która ujawnia się właśnie w wyniku rozmów, wspólnego dyskutowania i rozwiązywania problemów. Integracja i scalanie zespołu było możliwe dzięki wewnętrznym katalizatorom – osobom z dużymi pokładami energii i „miękkich" umiejętności, które sprzyjały formowaniu się genialnej pracy zespołowej.

Metody agile'owe zwiększają także zaangażowanie pracowników w wykonywaną pracę oraz dają im większą satysfakcję zawodową. 15 miesięcy po wdrożeniu metody Scrum w firmie Salesforce (Salesforce.com), wśród pracowników została przeprowadzona ankieta, badająca poziom zadowolenia. Jak się okazało, 86% badanych osób uznało, iż czas pracy spędzony w firmie po wdrożeniu uznaje za dobry lub bardzo dobry. Przed wprowadzeniem Scruma zaledwie 40% pracowników udzieliło podobnych odpowiedzi. W tej samej ankiecie 92% badanych powiedziało, że poleciłoby tę metodykę pracownikom innych firm.

Niekwestionowaną zaletą metod agile'owych jest ich podejście do zmienności wymagań w trakcie prac rozwojowych. Zmiana jest czymś naturalnym i klient może zgłaszać nowe pomysły i zmieniać stare, praktycznie w trakcie całego trwania projektu. Znów wracam myślami do projektów kaskadowych, gdzie faza zbierania miała być dopięta na ostatni guzik tak, żebyśmy na spokojnie mogli ruszyć z pracami projektowymi i implementacją. Praktyka jednak, pokazywała zupełnie coś innego Klientowi się nie odmawiało. Akceptacja zmiany przydaje się zwłaszcza w przypadku klientów, którzy często zachowują się jak płaczące dziecko – sami do końca nie wiedzą, czego chcą. Akceptacja zmiany daje niesamowitą elastyczność w prowadzeniu takich projektów.

Agile zwiększa również produktywność zespołów. Należy zwrócić uwagę, że produktywność dla każdego znaczy co innego. Martin Fowler nawet twierdzi, że nie jesteśmy w stanie zmierzyć produktywności całego zespołu. Myślę, że coś w tym jest. Zasadniczy problem polega na tym, że nie ma stałej charakterystyki produktywności. Dla jednych jest to liczba napisanych linii kodu, dla innych suma dostarczonych punktów funkcyjnych lub liczba zaimplementowanych funkcjonalności. Kiedy wdrażałem metody agile'owe po raz pierwszy w projektach, w Motoroli – przez produktywność rozumieliśmy liczbę napisanych linii kodu. Po wprowadzeniu metody Scrum w połączeniu z kilkoma praktykami XP, produktywność naszych zespołów zwiększyła się 3 – krotnie względem wcześniejszych metryk.

Stosowanie „zwinnych" metod zwiększa czas wprowadzenia produktu na rynek (ang. *time-to-market*). Badania, które przeprowadziła amerykańska firma VersionOne (znana z narzędzia do zarządzania „zwinnymi" projektami), pokazują, iż 64% procent respondentów wyznało, że wdrożenie metod agile'owych w ich firmach zwiększyło (41% badanych) lub znacznie zwiększyło (23% badanych) czas wprowadzenia wyrobu na rynek. Skrócenie *time-to-market* może być wytłumaczone dwoma powodami. Po pierwsze, ma na to wpływ wyższa produktywność zespołów agile'owych, a co za tym idzie szybsze tempo prac rozwojowych wymaganych funkcjonalności. Drugim powodem są inkrementalne wydania produktu. Uczestniczyłem kiedyś w projekcie, którego klient nie miał żadnych doświadczeń w stosowaniu metod agile'owych. Jednak po krótkim szkoleniu, wprowadzającym go w tajniki Scrum, bardzo szybko się zafascynował tą metodą. Co więcej, jak zobaczył, że to faktycznie działa i po każdym sprincie dostaje kawałek działającej funkcjonalności, bardzo szybko doszedł do wniosku, że nie będzie czekał z wszystkimi funkcjonalnościami produktu na ich finalną integrację. W efekcie wypuścił zamówiony produkt z okrojoną funkcjonalnością, którą w pełnym zakresie otrzymał za pół roku.

Ta sama firma VersionOne w swoich badaniach wykazała, że projekty, wykorzystujące metodę Scrum, zwiększają jakość prowadzonych prac. 68% procent badanych wykazało, że agile poprawił (44%) lub znacznie poprawił (24%) jakoś wytwarzanego softwaru. Dodatkowo, aż 84% procent stwierdziło, że stosowanie metod agile'owych zmniejszyło o 10% i więcej liczbę wykrywanych błędów w rozwijanych produktach; natomiast 30% uznało, że agile zredukował liczbę defektów o 25% i więcej. Agile poprawia jakość rozwijanych produktów przede wszystkim ze względu na stałe i równomierne tempo pracy, które znajduje swoje dopełnienie w takich praktykach jak: ciągła integracja kodu, refaktoryzacja, wczesne i ciągłe testy automatyczne oraz programowanie w parach.

O tym artykule

Artykuł stanowi wprowadzenie do rodziny „zwinnych" metod tworzenia oprogramowania. Pokazałem, czym różnią się te metody w kontekście tradycyjnych podejść, a także na czym polega ich wywrotowość. Dodatkowo omówiłem podstawowe idee, które składają się na tzw. *filozofię zwinności*. Świadomie nazwałem ją *filozofią*, gdyż zawiera potężny ładunek wartości i zasad (zebranych i udokumentowanych w „Manifeście Agile"), które stanowią oś wszystkich metod. Artykuł zamyka prezentacja możliwych korzyści, płynących ze stosowania idei zwinności w codziennej pracy projektowej.

Odniesienia

- *http://agilemanifesto.org/*
- *http://www.youtube.com/watch?v=Q5k7a9YEoUI*
- *http://www.devagile.com/*
- *http://agilemethodology.org/*
- *http://scrummethodology.com/*
- *http://www.agilealliance.org/*

Mariusz Chrapko

Wieloletni praktykant w zakresie doskonalenia procesów tworzenia oprogramowania w oparciu o model CMMI®. Wspiera firmy informatyczne na terenie całej Europy, prowadząc coaching zespołów projektowych oraz szkolenia na różnych szczeblach organizacyjnych. Dodatkowo jest praktykiem we wdrażaniu i adaptacji metod Agile Software Development (wcześniej Agile Coach/ Centrum Oprogramowania Motoroli w Krakowie), Programu Metryk Organizacyjnych, a także procesu Peer Review. Jest autorem wielu publikacji z zakresu inżynierii oprogramowania oraz prelegentem na konferencjach krajowych i międzynarodowych. Autor pierwszej w Polsce książki na temat modelu CMMI oraz jego praktycznego zastosowania. Na co dzień, współpracuje z firmą LOYCON® business solutions.

Kontakt z Autorem: *mariusz.chrapko@gmail.com*

IBM Rational Requirements Composer

Zmieniające się otoczenie wymusza na organizacjach dokonywanie zmian m.in. w produktach, procesach i strukturach organizacyjnych. W celu wprowadzenia zmian powołuje się projekt albo zbiór projektów, zwany programem projektowym, w którym bierze udział wiele osób odpowiedzialnych za różne obszary/aspekty organizacyjne i projektowe. Osoby te, zwane interesariuszami (ang. stakeholders), często mają różne cele i potrzeby względem realizowanych projektów. Dlatego bardzo istotne jest określenie wspólnego zbioru potrzeb i oczekiwań wyrażonych za pomocą wymagań.

Wiele firm zarządza wymaganiami. Każda na swój sposób, z wykorzystaniem mniej lub bardziej zaawansowanych technik i narzędzi. Z mniejszymi bądź większymi sukcesami. W niniejszym artykule chciałbym przedstawić narzędzie IBM, które jest w stanie przynieść ogromne korzyści poprzez usprawnienie procesów komunikacji i zbierania wymagań. Dalszymi konsekwencjami zastosowania IBM Rational Requiremnts Composer (w skrócie RRC) jest redukcja ryzyk i kosztów projektów oraz zwiększanie jakości produktów. RRC jest produktem z najnowszej rodziny produktów IBM Rational, opartym na otwartej platformie Jazz. W artykule zostaną omówione najważniejsze funkcjonalności produktu.

Jest wiele metodyk i procesów, w których jest opisany między innymi proces zarządzania wymaganiami, dla przykładu (może i dla kontrastu) Rational Unified Process, Volere, ICONIX Process itp. RRC jest w stanie wesprzeć każdą ze wspomnianych metodyk.

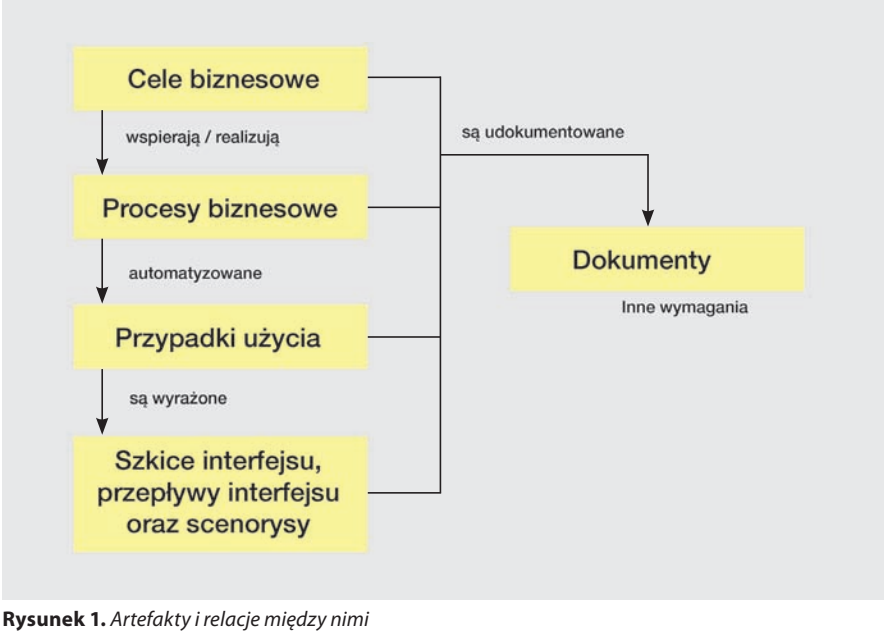
Najważniejszymi obszarami działań w zakresie inżynierii wymagań są:

- wydobycie (ang. *elicitation*),
- analiza (klasyfikacja, konceptualne modelowanie, projektowanie architekturalne i alokacja wymagań, negocjacja),
- specyfikacja (definicja systemu, system requirements specification, software requirements specification),
- weryfikacja (przeglądy, prototypowanie, wizualizacja modelu, testy akceptacyjne).

Z chwilą, jak zostaną określone źródła wymagań, można przystąpić do procesu ich wydobywania. To znaczy zastosowania różnych technik umożliwiających wyartykułowanie wymagań: wywiady, scenariusze, prototypy, scenorysy itp. Wiele ze wspomnianych technik jest wspieranych w IBM Rational Requirements Composer.

Użycie Rational Requiremetns Composer'a

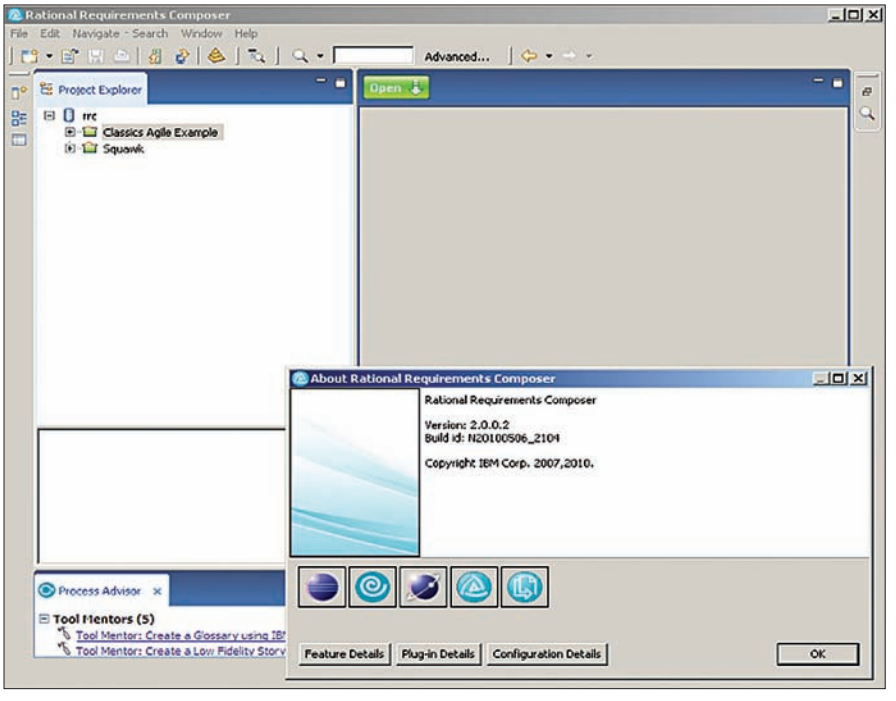
Organizacja, wprowadzając zmiany, ma na celu osiągnięcie określonych celów biznesowych, które są wspierane/realizowane przez procesy biznesowe. Wiele ze wspomnianych procesów jest automatyzowanych przez system informatyczny. Do opisania interakcji między systemem informatycznym a światem go otaczającym często wykorzystuje się przypadki użycia. Niejednokrotnie interakcja ta przebiega między systemami informatycznym, ale nie tylko. Niejednokrotnie mamy do czynienia z bezpośrednią interakcją człowiek-system. Ta interakcja odbywa się przy wykorzystaniu graficznego interfejsu użytkownika, który jest przypisany do przypadku użycia, scenorysie oraz w dokumentach opisujących wymagania.



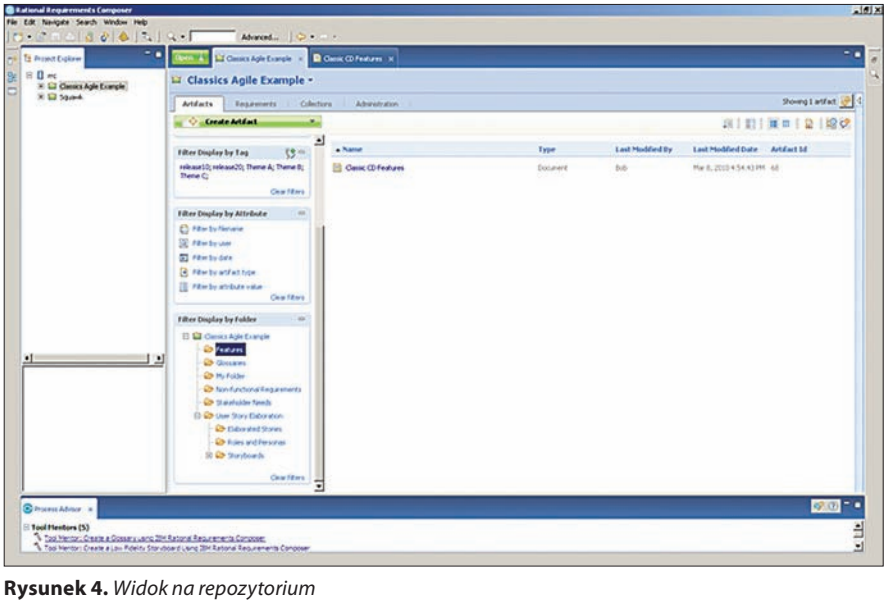
Rysunek 1. *Artefakty i relacje między nimi*

Actor	Aktor,
Business Process Diagram	Diagram Procesu Biznesowego,
Collection	Kolekcja (artefaktów),
Document	Dokument,
Glossary	Słownik,
Requirement	Wymagania,
Screen Flow	Przepływu ekranów,
Storyboard	Scenorys,
Use Case	Przypadki Użycia,
Use-Case Diagram	Diagramy Użycia,
User Interface Part	Reużytkowalny fragmentów interfejsów użytkownika,
User Interface Sketch	Szkic Interfejsu Użytkownika (poniżej rysunek)

Rysunek 2. *Artefakty w RRC*



Rysunek 3. *Gruby klient RCC*



Rysunek 4. *Widok na repozytorium*

Można to zobrazować w następujący sposób (Rys 1).

Powyższy model ma swoje odzwierciedlenie w funkcjonalności RRC.

Przykładowy przepływ prac

Po uruchomieniu serwera RRC, możemy uruchomić klienta, który jest grubym klientem na platformie Eclipse. Część funkcjonalności jest dostępna również poprzez wykorzystanie aplikacji WWW, na przykład przegląd i komentowanie artefaktów.

Po lewej stronie wyświetlone są projekty w repozytorium (por. Rys. 3). Po dwukrotnym kliknięciu na projekt Classic Agile Example, otwiera się okno z podglądem do informacji zgromadzonych w repozytorium (por. Rys. 4), odnośnie projektu, w postaci dokumentów, dołączonych plików, wymagań, diagramów przypadków użycia oraz innych przydatnych artefaktów. Oczywiście można zawężać ilość wyświetlanych informacji, poprzez różnego rodzaju dostępne filtry.

Kolejnym krokiem może być import istniejącego dokumentu opisującego założenia projektowe bądź jego stworzenie w RRC.

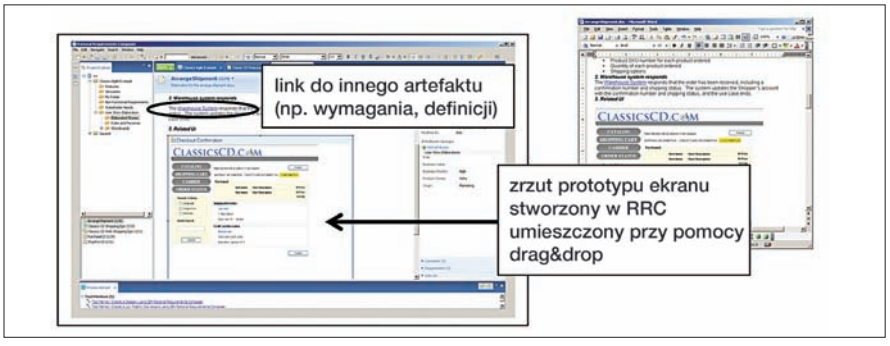
Dokumenty, wymagania oraz procesy biznesowe

Do RRC można zaimportować dokument MS Word, zawierający opis strategii rozwoju produktu. Na podstawie tego dokumentu możemy stworzyć dokument RRC, który przypomina dokument MS Word (można wygenerować z niego dokument MS Word (por. Rys. 5), ale posiada znacznie bardziej rozbudowane możliwości łączenia fragmentów dokumentu z innymi artefaktami w repozytorium.

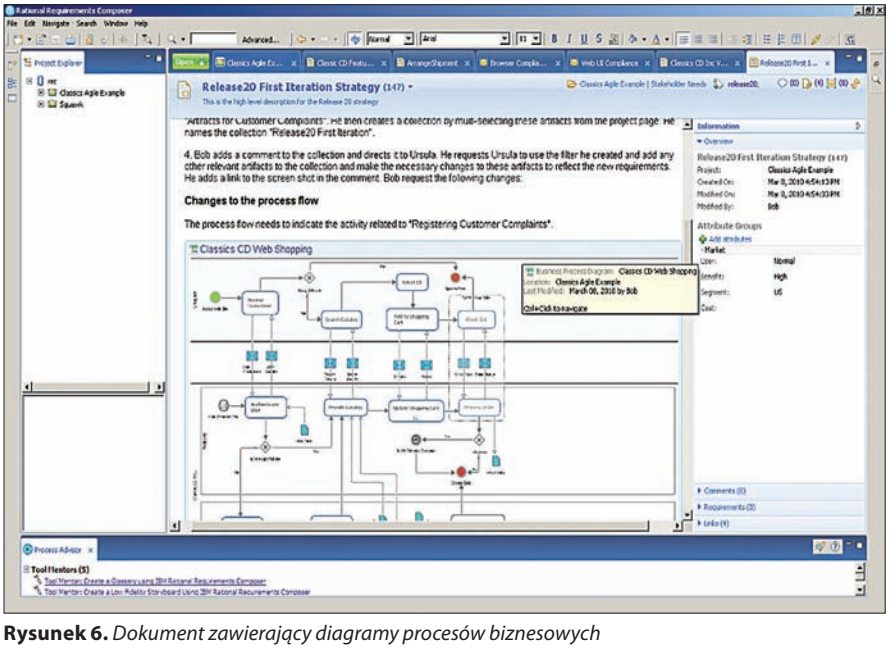
Myślę, że jedną z najważniejszych funkcjonalności jest to, że możemy na zasadzie drag &drop umieszczać inne artefakty (np. ekrany prototypu interfejsu czy modele procesów biznesowych) (por. Rys. 5).

Umieszczenie obiektu na zasadzie drag& drop zapewnia, że teselement umieszczony w dokumencie zawsze przy otwieraniu będzie wyświetlany w aktualnej wersji zawartej w repozytorium. W ten sposób mamy gwarancję, że taki dokument będzie zawierał zawsze aktualną wersję umieszczonych artefaktów, czy to będzie prototyp interfejsu (por. Rys. 5) czy też model procesu biznesowego (por. Rys. 6).

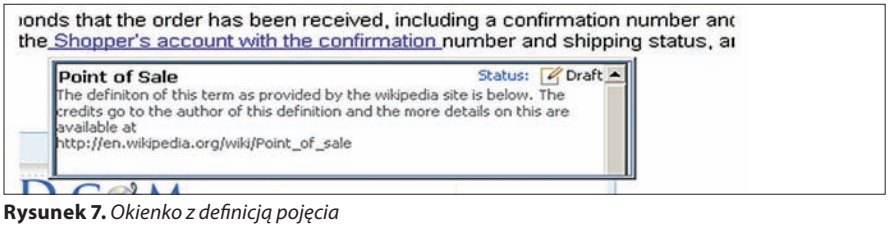
Można łączyć artefakty bez ich bezpośredniego zagnieżdżania, to znaczy stworzyć tylko linki w dokumencie bez ich umieszczania w dokumencie. W takim przypadku w dokumencie umieszczona zostanie jedynie nazwa powiązanego artefaktu. Po tej czynności tekst będzie oznaczony kolorem niebieskim i będzie podkreślony- stworzy się hiperłącze. Od tej pory, po najechaniu kursorem myszki na tekst,



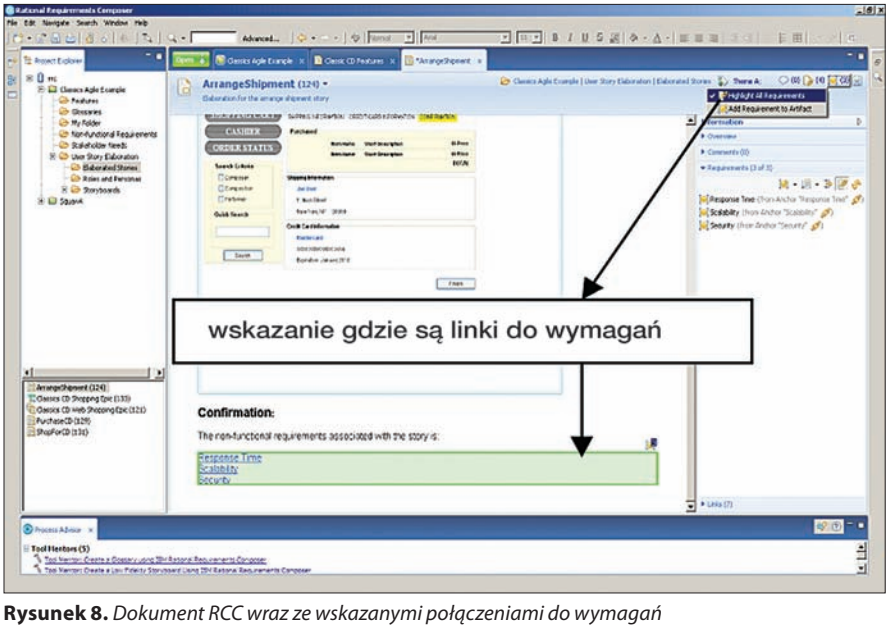
Rysunek 5. *Dokument w RCC*



Rysunek 6. *Dokument zawierający diagramy procesów biznesowych*



Rysunek 7. *Okienko z definicją pojęcia*



Rysunek 8. *Dokument RCC wraz ze wskazanymi połączeniami do wymagań*

będzie się pokazywać skrócony opis elementu dołączonego do tekstu (por Rys 7).

Jeśli natomiast będziemy chcieli się dostać do źródła podłączonego artefaktu, to wystarczy kliknąć na hiperłącze z przyciskiem [*Ctrl*].

Z dokumentów opisujących potrzeby i cele biznesowe mogą wynikać określone wymagania w stosunku do produktu. Dzięki wspomnianemu mechanizmowi łączenia artefaktów możliwe jest połączenie ich z wymaganiami. Dzięki funkcjonalności, wskazującej gdzie w dokumencie znajdują się hiperłącza, do wymagań, łatwiejsza jest nawigacja po wszystkich elementach opisujących realne potrzeby (por. Rys. 8).

Po kliknięciu na wybrane wymagania wyświetla się dokument zawierający treść wymagania (por. Rys. 9).

Oprócz dokumentów wymagań, w RRC możliwe jest zdefiniowanie pojęć dla projektu i ich asocjacja z innymi artefaktami. Możliwe jest określenie synonimów oraz akronimów. Glosariusze mogą zawierać atrybuty oraz komentarze. Pojęcia natomiast zawierają następującą treść (por. Rys. 10):

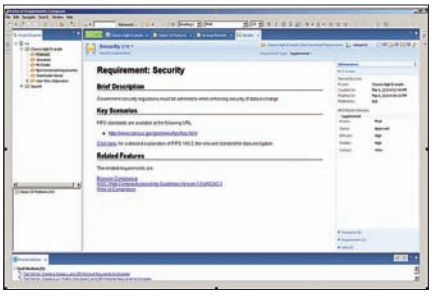
- nazwę,
- definicję,
- status (propozycja (ang. *draft*) bądź opublikowany),
- powiązane elementy,
- synonimy,
- akronimy,
- linki do innych artefaktów, w tym wymagań,
- komentarze.

Po stworzeniu pojęć, są one dostępne podczas pisania dokumentów. Działa to w następujący sposób: pisząc tekst, możemy kliknąć prawym przyciskiem na wybranym słowie i jest dostępna opcja w menu kontekstowym „Lookup Up Term". Po wyborze tej opcji, narzędzie szuka definicji wybranego słowa w glosariuszu i ją wyświetla.

Przypadki użycia i scenorysy

Do opisu interakcji systemu ze środowiskiem zewnętrznym najczęściej stosuje się przypadki użycia. Dokumentuje się je w postaci statycznych diagramów przypadków użycia i ich specyfikacji. Dodatkowo w RRC można tworzyć scenorysy.

Przy wykorzystaniu w projekcie scenorysów w RRC możliwe jest zdefiniowanie kroków np. ze specyfikacji przypadku użycia i przyłączanie do nich szkiców interfejsu (opisane w kolejnym punkcie). Dokładnie mówiąc, do ich instancji, które można dostosowywać tak, by wyświetlały określone wartości i emulowały zachowywanie się systemu. Dzięki temu można wizualnie przedstawić, jak system powinien się zachować w określonej sytuacji (por. Rys. 11).



Rysunek 9. *Dokument wymagań*

Rysunek 10. *Glosariusz*

Szkice interfejsu użytkownika

W RRC możliwe jest tworzenie prototypu interfejsu przy pomocy schematów (ang. *User Interface Sketch*). Przypomina to tworzenie stron WWW przy pomocy edytora graficznego (por. Rys.12).

Do dyspozycji mamy wszelkie kontrolki niezbędne do stworzenia zaawansowanego interfejsu graficznego aplikacji. Poczynając od pola tekstowego, przycisku, menu, paska narzędziowego, a kończąc na obrazkach. Każdy element można opisać, skomentować i nadać priorytet. Komentarz może być przepisany do określonej osoby bądź całego zespołu.

Tworząc interfejs użytkownika, można wydzielić elementy do ponownego wykorzystania. To znaczy, że tworzymy tak zwaną część interfejsu użytkownika (ang. *User Inteterface Part*), która może być wykorzystana na jednym bądź wielu szkicach interfejsu. Dzięki takiemu komponentowi możemy wydzielić elementy wspólne dla wielu ekranów. Oznacza to, że jeśli część interfejsu użytkownika zostanie zmodyfikowana, to modyfikacja będzie widoczna na wszystkich komponentach interfejsu użytkownika, które ją wykorzystują. Po stworzeniu poszczególnych ekranów można zdefiniować przepływ między nimi (ang. *Screen Flow*). Przy pomocy RRC można zdefiniować, jak użytkownik będzie nawigował między stronami.

Na poniższym obrazku (Rysunek 13) przedstawiony jest przykładowy przepływ między szkicami interfejsu użytkownika.

Przedstawione na powyższym rysunku prostokąty reprezentują szkice, natomiast strzałki definiują możliwy przepływ między nimi. Użytkownik może definiować przepływy, ale również ma możliwość podglądu poszczególnych szkiców. Dokonuje tego przez kliknięcie na prostokąt, który reprezentuje szkic interfejsu użytkownika.

Atrybuty, znaczniki i administracja

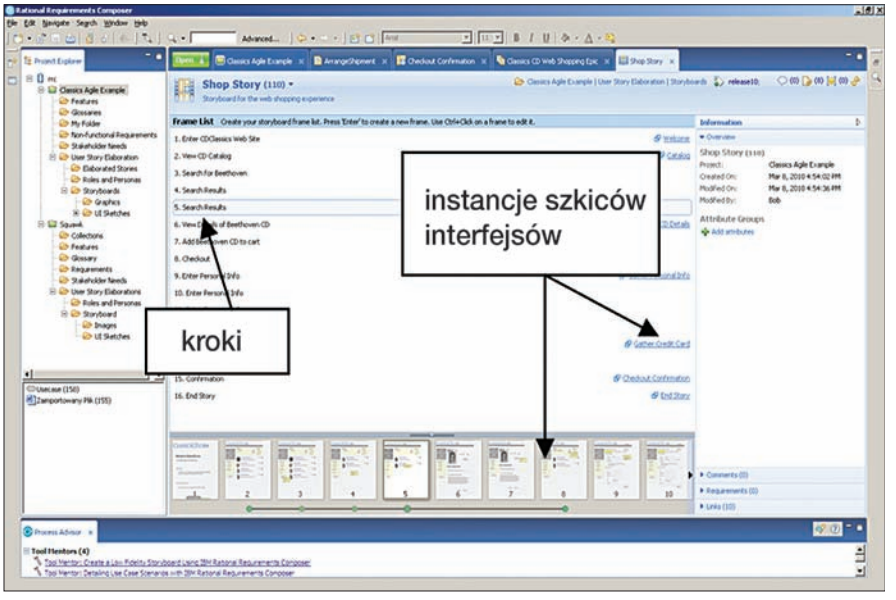
Każdy artefakt w RRC, w tym i wymagania, posiadają atrybuty, które możemy zdefiniować jeśli chodzi o typ, wartości i wymagalność.

Po dwukrotnym kliknięciu na projekt, w naszym przypadku Classics Agile Example, wyświetla się ponownie okno administracyjne projektu (por. Rys. 4), na którym możemy nie tylko przeglądnąć zawartość repozytorium, ale możemy zdefiniować atrybuty (por. Rys. 14) – ich typy, możliwe wartości i obowiązkowość.

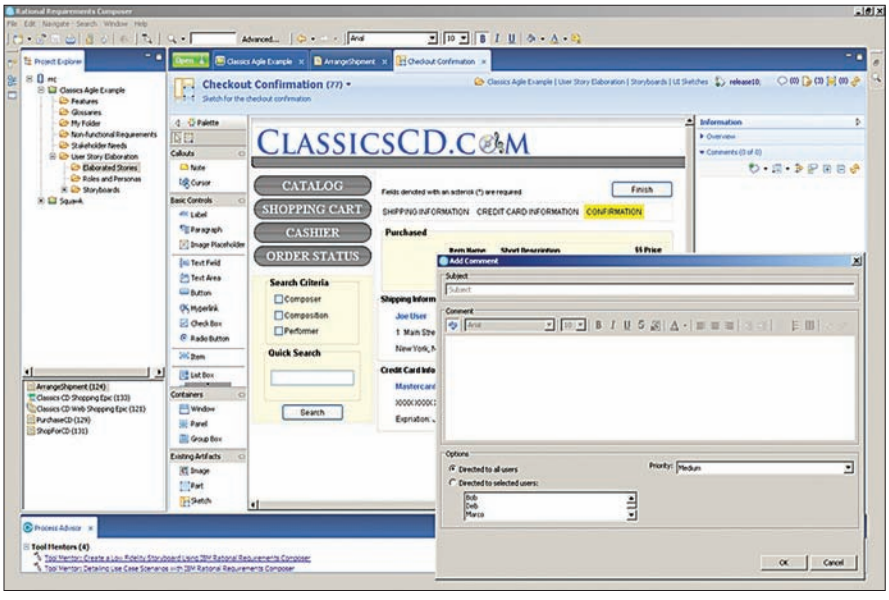
W zakładce administracyjnej mamy możliwość tworzenia użytkowników i nadawania im uprawnienia do zasobów repozytorium.

W RRC są predefiniowane 4 role:

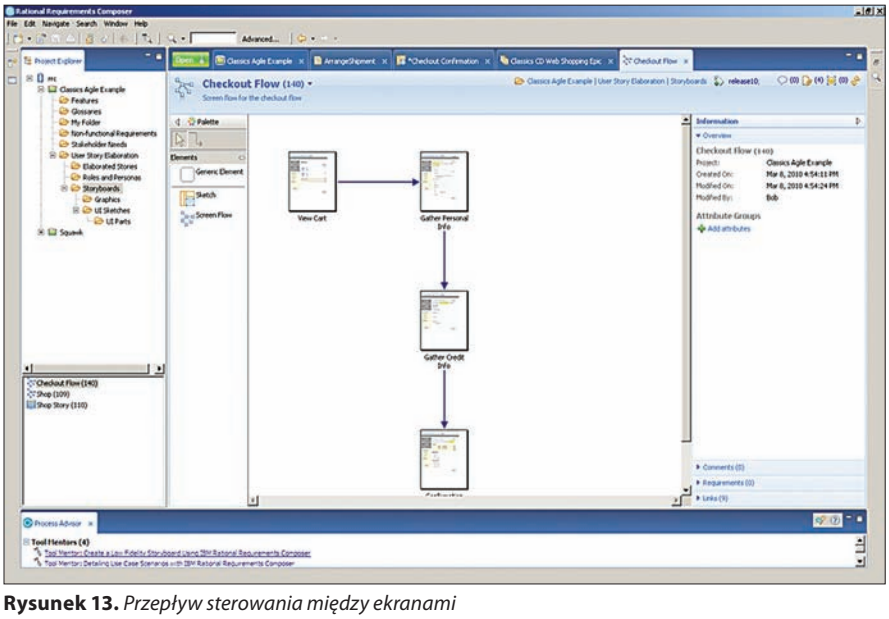
- administrator – użytkownik z tą rolą może dodawać i usuwać użytkowników;



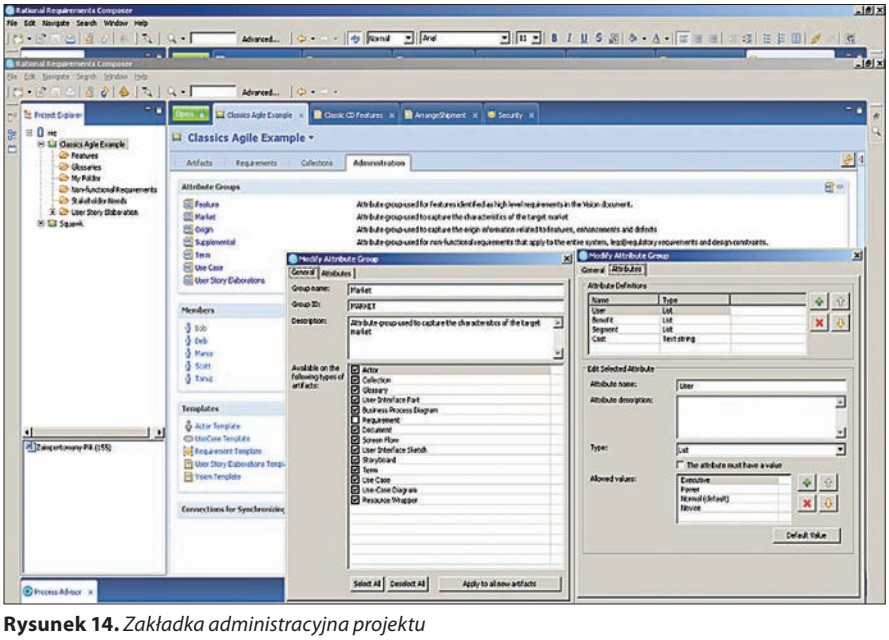
Rysunek 11. *Scenorys przypadku użycia*



Rysunek 12. *Szkice interfejsu użytkownika*



Rysunek 13. *Przepływ sterowania między ekranami*



Rysunek 14. *Zakładka administracyjna projektu*

tworzyć, edytować i usuwać połączenia z Requisite Pro oraz DOORS; zarządzać wzorcami artefaktów; zarządzać grupami atrybutów; ma dostęp do strony administracyjnej projektu.

- autor – użytkownik z tą rolą może zarządzać artefaktami, schematami interfejsu użytkownika oraz procesami biznesowymi; komentować artefakty; tworzyć prywatne i publiczne znaczniki artefaktów; przenosić artefakty między katalogami.
- komentator (ang. *commenter*) – może przeglądać artefakty; przygotowywać przeglądy; komentować artefakty; tworzyć prywatne znaczniki artefaktów.
- administrator migawki – użytkownik ma prawo do zarządzania migawkami (ang. *snapshot*) całego repozytorium.

DOORS

Po zdefiniowaniu artefaktów, które opisują między innymi wymagania, jest możliwość ich eksportu do narzędzi, które służą do zarządzania wymaganiami. Te narzędzia to DOORS czy RequisitePro. W niniejszym artykule skoncentrujemy się na IBM Rational DOORS. Integracja z RequisitePro przebiega w odmienny sposób, co wynika z architektury RquisitePro.

Po stworzeniu dokumentu(ów) w RRC i umieszczeniu w nim uprzednio zdefiniowanych artefaktów (pojęcia, procesy, przypadki użycia, wymagania, szkice interfejsów itp.) można zasocjować dokument z DOORS tak, aby na jego podstawie został stworzony moduł w DOORS. Następnie należy określić relacje między importowanymi wymaganiami a zawartymi w DOORS. Dane są propagowane z RRC do DOORS. Zatem dane powinny być modyfikowane w RRC (por. Rys. 15). Po tym, jak w DOORS zdefiniujemy relacje między innymi wymaganiami zawartymi w modułach, możemy dokonywać złożonych analizy wpływu czy importu z innych źródeł danych wymagań.

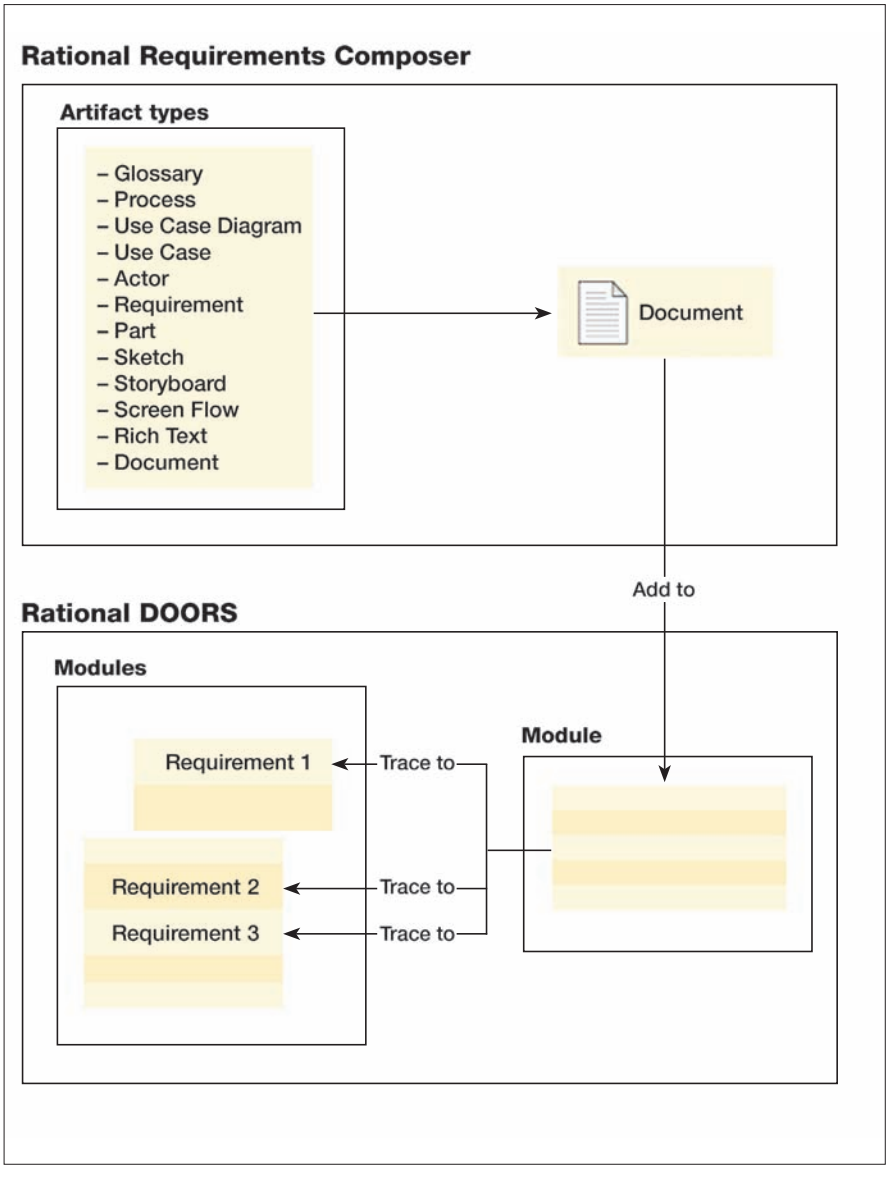
Środowisko

Na końcu chciałbym wspomnieć o wymaganiach systemowych IBM Rational Requirements Composer'a. RRC funkcjonuje na platformie Windows oraz Linux. Jeśli chodzi o przeglądarki, to wspiera go Mozilla Firefox i Internet Explorer. Co do serwerów baz danych, to wspierane są DB2, Oracle oraz SQL Server. W najnowszej wersji 2.0.0.2 RRC integruje się z innymi rozwiązaniami IBM:

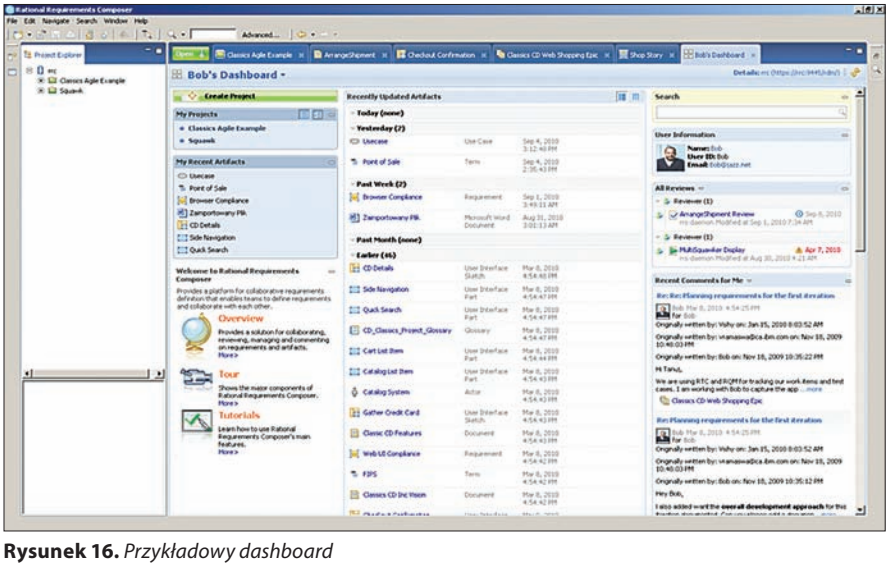
- Rational Team Concert Standard Edition,
- Rational Quality Manager,
- Rational Publishing Engine,
- Rational DOORS,
- Rational Method Composer,
- Rational RequisitePro,

- Rational Software Architect,
- Rational Software Architect RealTime Edition,

Osoby zainteresowane szczegółami zapraszam na stronę: *http://www-01.ibm.com/support/docview.wss?rs=3499&uid=swg27017009*



Rysunek 15 *Artefakty RRC oraz DOORS i relacje między nimi*



Rysunek 16. *Przykładowy dashboard*

Podsumowanie

IBM Rational Requiremntents Composer posiada szereg dodatkowych funkcjonalności, niezbędnych i bardzo przydatnych w procesie inżynierii wymagań. Do takich należy na przykład możliwość dokonywania przeglądów poszczególnych artefaktów bądź ich grup w celu ciągłego zwiększania jakości tworzonych artefaktów oraz samego procesu inżynierii wymagań.

Warto wspomnieć, iż narzędzie ma wbudowanego „doradcę procesowego" (ang *Process Advisor*), w którym są zawarte rekomendacje metodyczne, które sugerują, w jaki sposób można efektywnie wykorzystać narzędzie w procesie inżynierii wymagań. Dodatkowo istnieje możliwość tworzenia migawek projektu i raportów na temat zgromadzonych informacji. Raporty są na przykład w formie dokumentów MS Word, ale nie tylko. Osoby uprawnione mogą tworzyć konfigurowalne dashboardy (por. Rys. 16), które raportują stan postępu prac w projekcie dla całego zespołu, jak i poszczególnych osób w projekcie. Jest to kluczowa funkcjonalność dla leaderów i kierowników projektów.

Dodatkowo RRC posiada interfejs w formie aplikacji webowej, dzięki której możliwe jest wyszukiwanie w przeglądarce internetowej, dokonywanie przeglądów oraz komentowanie artefaktów stworzonych z wykorzystaniem grubego klienta w platformie Eclipse. W aplikacji webowej dostęp jest również konfigurowalny, co do treści i formy, dashboard, który umożliwia monitorowanie stanu prac.

Requirements Compomposer jest nowoczesną platformą do współpracy zespołu, w znaczący sposób usprawniającą proces tworzenia i gromadzenia wymagań. Jest zintegrowanym środowiskiem do komunikacji i wypracowywania artefaktów opisujących wymagania w formie graficznej oraz tekstowej. RRC jest zintegrowany ze środowiskami do zarządzania wymaganiami oraz z Software Architect'em, który umożliwia projektowanie i realizację systemów w określonej technologii realizujących wymagania. Stosując RRC, już na bardzo wczesnych etapach projektu, znacznie redukujemy ryzyko, że budowany system nie będzie odpowiadał potrzebom biznesowym i użytkowników.

Parys Waicis

Odpowiedzialny za wsparcie w projektach w zakresie zarządzania wymaganiami, architekturą korporacyjną oraz usprawnianie innych krytycznych procesów IT z wykorzystaniem platformy IBM Rational w CEE.

Uprzednio, jako konsultant, brał udział w licznych projektach w USA oraz Polsce.

Absolwent Politechniki Wrocławskiej, Wydziału Informatyki i Zarządzania. Ukończył MBA w Carlson University w Minesocie oraz SGH.

Adaptacja zwinnych metodyk z użyciem IBM Rational Team Concert

na przykładzie Scrum

Nie będzie to książkowy artykuł o tym, jak trudno wdrożyć Scrum w firmie stosującej przez ostatnie 20 lat tradycyjne metodologie rozwoju oprogramowania. Nie będzie to też artykuł o tym, co to jest sprint (iteracja), „scrum master" oraz retrospekcja. Będę starał się opisać w jaki sposób zaadoptowaliśmy metodologię Scrum do naszych własnych potrzeb, w jaki sposób nasi klienci milcząco tę metodologię przyjęli, jak pomógł nam w tym produkt IBM o nazwie Rational Team Concert oraz jak musieliśmy pomóc sobie sami.

Poniedziałek rano. Najbardziej nielubiany przeze mnie czas spędzony w pracy. Ale to nie dlatego, że skończył się weekend. To dlatego, że przede mną tak zwane 'raportowanie czasów'. Ja i mój zespół zajmujemy się głównie konsultingiem SAP. W tym świecie przyjętą normą jest przesyłanie wspomnianych raportów z końcem każdego tygodnia. Polega to po prostu na przesłaniu listy czasów spędzonych nad rozwiązywaniem problemów klientów.

Organizując pracę z naszymi klientami, nie mieliśmy nigdy problemów z wdrożeniem zwinnych metodologii takich jak Scrum. A to dlatego, że w świecie SAP normą jest chaos. Niestety pozwolenie sobie na przyjęcie go bez ograniczeń zazwyczaj skutkuje utratą klienta po pierwszym miesiącu współpracy. Klienci nie wymagają niczego oprócz dwóch rzeczy:

- tygodniowego raportu czasu pracy,
- periodycznych telekonferencji przez telefon (czasami jest to raz na tydzień, a czasami codziennie).

Po naszej stronie pozostaje zorganizowanie sobie pracy i zarządzania zadaniami w ten sposób, aby tygodniowy raport czasu pracy był wiarygodny.

Tak naprawdę, kiedy pierwszy raz usłyszałem o Scrum, zacząłem się zastanawiać, czy to nie jest po prostu nazwanie tego procesu, który konsultanci SAP stosowali od lat. Oczywiście jest w tym zdaniu dużo przesady. Chciałem jednak tym dosyć mocnym stwierdzeniem pokazać, że wcześniej opisana przeze mnie specyfika pracy z klientami SAP miała już dwa podstawowe filary metodologii Scrum zaszyte w sobie:

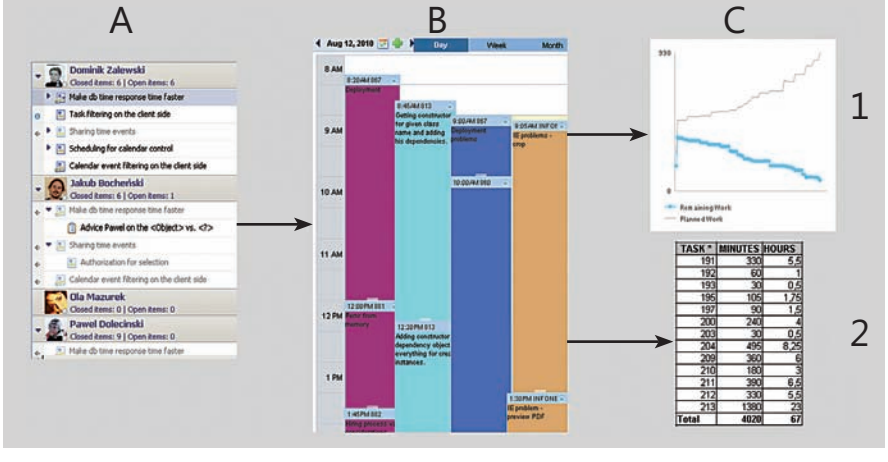
- klienci chcieli się komunikować często i wszelkie swoje wymagania definiować (i zmieniać) werbalnie przez telefon – brak dokumentacji, często zmieniające się wymagania,
- klienci chcieli wiedzieć za co płacą, aby móc kontrolować ten proces (w odróżnieniu od otrzymania na koniec miesiąca faktury z jedną pozycją: „Godziny konsultingu" w wymiarze XX).

Klienci sami z siebie byli świadomi, iż nie są w stanie jasno przed rozpoczęciem całego procesu zdefiniować swoich wymagań i tym samym oszacować kosztów i pracochłonności. Tak jak w zwinnych metodologiach, chcieli oni po prostu uruchomić ten proces i kontrolować go na każdym kroku. Nie mieliśmy więc tego problemu, który pojawia się zazwyczaj przy wdrażaniu Scrum, tzn. że proces ten jest wygodny dla programistów, lecz bardzo niewygodny dla księgowych.

Modelowanie procesu

Kluczowymi dla nas decyzjami związanymi z modelowaniem procesu współpracy z naszymi klientami były odpowiedzi na następujące pytania:

- jakich darmowych narzędzi możemy użyć,
- na jakie płatne narzędzia pozwala nasz budżet,
- jakie narzędzia musimy napisać sami?



Rysunek 1. *Model procesu*

Szybko zorientowaliśmy się, iż proces, bardzo dobrze obrazuje Rysunek 1.

Kolumna (A) przedstawia zaplanowany sprint, czyli w skrócie listę zadań, które należy zrealizować. Kolumna (B) to wykaz czasu pracy w podziale na programistów wraz z opisami, co w danym przedziale czasu zostało zrealizowane. Kolumna (C) pozwala kontrolować proces. Zawiera ona tzw. „burndown chart" (część (C1)), czyli zapożyczony ze Scrum sposób wizualizacji pracy, która została jeszcze do wykonania. Natomiast w części (C2) widzimy raport dla klienta zawierający czasy w podziale na rozplanowane w kolumnie (A) zadania. W dalszej części opiszę każdą z kolumn Rysunku 1 szczegółowo.

Standardowe narzędzia

Na rynku jest dostępnych wiele płatnych i bezpłatnych narzędzi do zarządzania zadaniami. W zasadzie problem zarządzania zadaniami jest tak stary, iż wydawałoby się, że wystarczy użyć jednej z popularnych wyszukiwarek internetowych, aby znaleźć narzędzie odpowiednie dla naszych potrzeb. Jednakże problem pojawia się, jeżeli zaczynamy te narzędzia rozpatrywać przez pryzmat rozszerzalności, integrowalności i elastyczności – słowem względem czegoś, co obecnie etykietuje się modnym akronimem SOA (*Service Oriented Architecture*). Odpowiedzi na takie pytania jak:

- czy będę miał programy klienckie dostępne zarówno na platformie Linux, jak i Windows,
- czy programy klienckie będą dostępne pod dowolną przeglądarkę internetową,
- w jaki sposób mogę nowe narzędzie zintegrować z narzędziami już używanymi w firmie,
- czy strona serwerowa jest uruchamiana na dowolnej platformie,
- jaki jest koszt zakupu takiego rozwiązania,

powodują ograniczenie naszego wyboru do zaledwie kilku narzędzi, a w dalszej części tego tekstu będę starał się przekonać iż w zasadzie do jednego – Rational Team Concert. Wymagania (a)-(c) RTC spełnia bezapelacyjnie. To, na czym chciałbym się skupić, to wymaganie (d). Jednak zanim przejdę do szczegółów, porównam najpierw RTC z innymi narzędziami, które również były brane przez nas pod uwagę, nie dlatego, że były lepsze względem kryteriów (a)-(d), lecz ze względu na, nie da się ukryć, bardzo ważne, finansowe kryterium (e). Systemy zarządzania zadaniami, które rozpatrywaliśmy, to: Mantis, Jira oraz RTC. Wszystkie z nich, oprócz RTC, były polecone przez zaprzyjaźnione firmy, w których były używane z powodzeniem. Naturalne było zweryfikowanie ich przydatności jako sprawdzonych narzędzi względem jeszcze wtedy przez nas nieznanego nowego narzędzia RTC.

O produkcie o nazwie Mantis wiedziałem już w zasadzie od około 7 lat. Jednakże znałem go jako „system śledzenia błędów". Pomyślałem jednak, iż w tak długim okresie czasu mógł on ewoluować do czegoś bardziej ogólnego. Podczas jego testów okazało się jednak, iż w zasadzie nadal jest on systemem śledzenia błędów i tylko tym. Brak w nim wsparcia dla planowania iteracji. Trochę mniej istotnym dla programistów, a jednak zauważalnym mankamentem jest to, iż interfejs WWW pochodzi w zasadzie z poprzedniej epoki.

Jedynym plusem, który widzieliśmy w tym systemie, to fakt, iż miał on wbudowaną funkcjonalność automatycznej rejestracji czasu pracy. Jest ona zrealizowana w następujący sposób. Programista, klikając na zadanie, powoduje pojawienie się jego szczegółów na ekranie. Tam też znajduje się przycisk „Start" oraz „Stop". Wciśnięcie przycisku „Start", a następnie „Stop", powoduje dopisanie zmierzonego interwału czasu jako oddzielny wpis czasu pracy przypięty do zadania. Jest to rzecz, której brakuje zarówno w produkcie Jira, jak i RTC. Co więcej, jest to funkcjonalność, którą musieliśmy sami wyemulować, integrując nasze autorskie narzędzie z RTC (o czym później). Mimo wszystko jednak funkcjonalność ta zaimplementowana w systemie Mantis pozostawia wiele do życzenia. Jest to powiązane z wcześniej wspomnianym faktem, iż interfejs WWW pochodzi z XX, a nie XXI wieku. Ze względu na to, iż nie jest tam używany AJAX, osoba obsługująca zadanie jest zmuszona do pozostania na tej samej stronie w przeglądarce. Jakakolwiek próba sprawdzenia listy zadań lub przeszukania historycznych wpisów powoduje, iż jesteśmy wyrzuceni ze strony rejestracji czasu pracy i tym samym proces rejestracji przestaje działać.

Podsumowując możliwości systemu Mantis, muszę stwierdzić, iż narzędzie to z naszego punktu widzenia nie rokuje szans na wykorzystanie w nowoczesnej infrastrukturze informatycznej. Brak możliwości wykorzystania go w infrastrukturze typu SOA oraz przestarzałość interfejsu graficznego definitywnie skreśla jego szanse na wdrożenie.

Produkt Jira był podobnie jak RTC nieznanym przez nas produktem. Za zweryfikowaniem jego przydatności przemawiały bardzo dobre materiały umieszczone na stronie producenta. Filmy wideo pokazujące możliwości tego narzędzia dowodzą, iż narzędzie jest wprost przystosowane do wsparcia Scrum. Co więcej, porównanie cen licencji Jira oraz RTC wskazuje na system Jira jako na faworyta. Ponadto Jira ma wtyczkę dla Eclipse, pozwala na pisanie własnych rozszerzeń oraz integrację z SVN.

Zanim zdołaliśmy się zabrać za weryfikację i testy tego narzędzia, zostaliśmy włączeni w projekt u klienta, który już systemu Jira używał. Mogliśmy więc jego przydatność sprawdzić w praktyce. Szybko zorientowaliśmy się, że wtyczka do Eclipse jest przereklamowana podobnie jak integracja z SVN. Można odnieść wrażenie, iż produkt jest podzielony w zasadzie na dwie części. Moduł przeglądarkowy służy do zarządzania zadaniami, natomiast wtyczka do Eclipse służy jedynie zarządzaniu kodem źródłowym i rzekomej integracji z SVN.

Rysunek 2. *Przechowywanie czasu pracy w RTC*

Tabela 1. *Porównanie Mantis, Jira oraz RTC*

	Mantis	Jira	IBM RTC
Integracja Eclipse	Brak	Ograniczona	Pełna
Klient przeglądarkowy	Tak	Tak	Tak
Linux	Tak	Tak	Tak
Windows	Tak	Tak	Tak
Integracja	brak	Tak	Tak
Koszt < 11 programistów	Brak	10 USD	Brak
Koszt > 10 programistów	Brak	od 1200 USD	od 1004 EUR
Wsparcie dla platformy System z	Brak	Brak	Tak
Wsparcie dla platformy i Series	Brak	Brak	Tak

Głównym interfejsem użytkownika okazał się być interfejs przeglądarkowy, dość mało intuicyjny zresztą. Ale to, co najbardziej irytuje w tym systemie, to jedynie dwupoziomowa głębokość zależności rodzic-dziecko przy budowaniu zależności między zadaniami. Żadnych innych zależności („duplikuje", „jest powiązane z") nie da się tam zbudować. Natomiast zmiana przypisania rodzic-dziecko to kwestia przejścia przez 4 ekrany „czarodzieja" (ang. *wizard*). Wtyczka Eclipse próbuje śledzić, jakie zadanie jest wykonywane przez programistę i na tej podstawie dołączać wygenerowane automatycznie wpisy do logów SVN tak, aby można było powiązać zadanie z kodem źródłowym. Według mnie jednak nie można stwierdzić, iż ta integracja została pomyślnie zaimplementowana i w prawdziwym projekcie informatycznym jest bezużyteczna.

Biorąc pod uwagę powyższe dwa zasadnicze niedociągnięcia w systemie Jira, stwierdziliśmy, iż nie będziemy poświęcali czasu na zweryfikowanie rozszerzalności tego systemu, gdyż przeprowadzone dotąd testy dyskwalifikowały to narzędzie do naszych potrzeb.

Narzędzie Rational Team Concert było od samego początku faworytem w naszych testach. A to z uwagi na fakt, iż mieliśmy z nim wcześniej do czynienia. Ja, ucząc jeszcze na uniwersytecie i prowadząc zajęcia dotyczące produktów IBM, zaznajomiłem się z jego możliwościami dość dobrze. Ponadto dwóch moich pracowników napisało prace magisterskie na jego temat. Jednakże były to warunki akademickie, w których nie musieliśmy brać pod uwagę czysto komercyjnych aspektów takich jak na przykład cena rozwiązania. Poniżej zacznę od wymienienia wszystkich cech RTC, tak aby porównanie z systemem Mantis oraz Jira było pełne.

Interesujące nas trzy główne aspekty RTC to zarządzanie zadaniami, kontrola kodu źródłowego oraz możliwość integracji. Do zarządzania zadaniami mamy do dyspozycji wersję oprogramowania działającą w przeglądarce lub klienta Eclipse. Odwrotnie niż w systemie Jira, tutaj nacisk położony jest na klienta Eclipse, co oznacza, iż wszystkie funkcjonalności dostępne są z poziomu Eclipse, a tylko niektóre z poziomu przeglądarki internetowej. Jest to zrozumiałe, gdyż trudno sobie wyobrazić wersjonowanie kodu źródłowego działające w przeglądarce (chociaż używając HTML5, nie jest to niemożliwe). Nie chciałbym jednak wprowadzać Czytelnika w błąd. Przeglądarkowa wersja modułu zarządzania zadaniami jest w pełni funkcjonalna i może być używana zamiennie z klientem Eclipse do codziennej pracy. Powodem, dla którego IBM naprawdę postarał się o funkcjonalność obydwu interfejsów, jest licencjonowanie. Licencje z dostępem jedynie przez przeglądarkę (dla testerów, managerów) są po prostu tańsze.

Zarządzanie zadaniami jest bardzo intuicyjne i elastyczne. Mamy do dyspozycji wiele różnych widoków w zależności od roli odgrywanej w projekcie (manager, programista). Co najważniejsze mamy możliwość bardzo dogłębnej konfiguracji. Nie skłamię, jeżeli powiem, że do tej pory nie udało mi się zgłębić nawet połowy możliwości RTC w tym zakresie.

Zarządzanie kodem źródłowym w RTC to bajka dla programisty przyzwyczajonego do SVN. Rational Team Concert posiada również integrację z SVN, ale dla nowych projektów naprawdę gorąco polecam moduł SCM (Source Code Management) wbudowany w RTC. Już po tygodniu używania SCM zauważyliśmy bardzo miłą rzecz. Funkcjonalność rozwiązywania konfliktów przy zatwierdzaniu kodu źródłowego jest zaimplementowana o niebo lepiej niż w SVN. Kiedy moi programiści byli zmuszeni do powrotu do SVNa w innym, nowym projekcie, już na drugi dzień słyszałem narzekanie na czas, jaki należy poświęcić na scalanie w SVN. Jeszcze inną wyróżniającą SCM cechą jest organizowanie zmian w repozytorium kodu źródłowego w zbiory (ang. *change sets*). Pozwala to na powiązanie zmian w kodzie źródłowym z zadaniami oraz na wybiórczą aktualizację kodu źródłowego. Obydwie te rzeczy są niemożliwe lub bardzo trudne do osiągnięcia, stosując integrację z SVN.

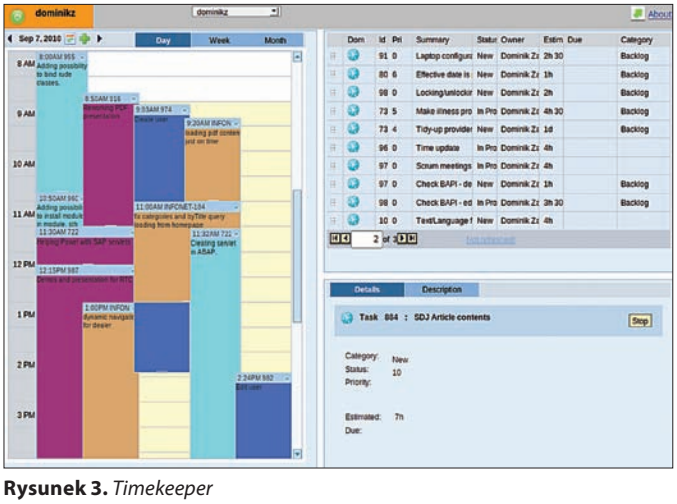
Integracja istniejących narzędzi z RTC jest również bardzo łatwo osiągalna. RTC posiada zbiór bibliotek napisanych w języku Java nazywanych „Plain Java Client Libraries", dzięki którym można z poziomu kodu źródłowego uzyskać dostęp do RTC jako usługi. Przykłady zastosowania dostępu do RTC przez API opiszę w rozdziale „Integracja narzędzi".

Podsumowując porównanie trzech narzędzi, bezapelacyjnie naszym faworytem jest Rational Team Concert. Zwięzłe porównanie cech trzech produktów opisanych w tym paragrafie przedstawiłem w Tabeli 1.

Autorskie Narzędzie

Wybór RTC jako podstawowego narzędzia używanego w naszym procesie obsługi klientów pozostawił nadal otwartą kwestię rejestracji czasu pracy (kolumna (B) z Rysunku 1). Jedyne wsparcie, jakie daje RTC w tym zakresie, to możliwość przechowywania sumy czasów spędzonych nad zadaniem. Jest to przedstawione na Rysunku 2.

W dolnej części rysunku widzimy właściwość „Time Remaining" przechowywaną po prostu jako standardowy atrybut przypięty do zadania. Programista, spędzając powiedzmy 1h nad zadaniem dzisiaj i 2h nad zadaniem jutro, winien sam zaktualizować ten atrybut odejmując od bieżącej wartości jednego dnia 1h, a drugiego dnia 2h. Wszyscy wiemy, jak skorzy są programiści w ogólności do utrzymywania spójnej informacji w ten sposób. Finansowanie naszej organizacji zależy przede wszystkim od dokładności tego właśnie procesu. Nie mogliśmy pozostawić tak dużej luki w naszym procesie. Co więcej, dokładność planowania iteracji oraz tzw. „burndown chart", który jest jądrem procesu Scrum,



Rysunek 3. *Timekeeper*

```
37      TeamPlatform.startup();
38
39      repo = TeamPlatform.getTeamRepositoryService().getTeamRepository(
40              "https://jazz.net:9443/jazz");
41
42      repo.registerLoginHandler(new ITeamRepository.ILoginHandler() {
43
44          public ILoginInfo challenge(ITeamRepository repository) {
45              return new ILoginInfo() {
46
47                  public String getUserId() {
48                      return "dominikz";
49                  }
50
51                  public String getPassword() {
52                      return "secret";
53                  }
54              };
55          }
56      });
57      repo.login(null);
58
59      //code goes here
60
61      repo.logout();
62      TeamPlatform.shutdown();
```

Rysunek 4. *Logowanie z użyciem PJCL*

również zależą bezpośrednio od dokładnej rejestracji czasu pracy. Dlatego też zdecydowaliśmy się na napisanie własnego autorskiego narzędzia służącego temu celowi.

Nasza lista wymagań wyglądała następująco:

- aplikacja winna działać z poziomu przeglądarki internetowej,
- musi być możliwa rejestracja czasu pracy bez podłączenia do sieci,
- musi być możliwa integracja z RTC, ale także z innymi systemami typu „task engine".

Ponieważ nie chcieliśmy wiązać naszego narzędzia bezpośrednio z RTC lub żadną inną platformą, zdecydowaliśmy się na wymaganie (a). W szczególności nasz system nie wymaga w ten sposób instalacji, np. Eclipse, aby działać. Zostawia to pewną niedogodność dla programistów, którzy pracując w Eclipse, chcieliby z poziomu tego środowiska rejestrować czas pracy. Wymaganie (b) wynikało z faktu, iż czasami praca z naszymi klientami wymaga udania się do budynku zewnętrznej organizacji, gdzie z uwagi na wymogi bezpieczeństwa, nie mamy nieograniczonego dostępu do Internetu. Często również pracujemy w tunelu (VPN) lub po prostu w drodze (pociąg). Ze względu na fakt, iż rozwój własnych narzędzi jest założenia kosztowny, chcieliśmy mieć możliwość jak największej elastyczności, tak aby móc to narzędzie w przyszłości integrować z innymi platformami i sprzedawać. Dlatego zdecydowaliśmy się na wymaganie (c).

```
IProcessClientService processClient = (IProcessClientService) repo
        .getClientLibrary(IProcessClientService.class);
IProjectArea projectArea = (IProjectArea) processClient
        .findProcessArea(URI.create("gj-timerec"), null, null);

final List<ContributorImpl> blankContributorList = new ArrayList<ContributorImpl>();
final IQueryCommon queryCommon = (IQueryCommon) repo
        .getClientLibrary(IQueryClient.class);

final List<IQueryDescriptor> listShared = queryCommon
        .findSharedQueries(projectArea, blankContributorList,
                QueryTypes.WORK_ITEM_QUERY,
                IQueryDescriptor.FULL_PROFILE, null);

for (final IQueryDescriptor query : listShared)
    if (query.getName().equals("Open assigned to me")) {

        final IQueryResult<IResolvedResult<IWorkItem>> result = queryCommon
                .getResolvedQueryResults(query, IWorkItem.FULL_PROFILE);

        queryCommon.getQueryResults(query).next(null).getItem();
        while (result.hasNext(null)) {
            final IWorkItem item = result.next(null).getItem();
            System.out.println(item.getHTMLSummary());
        }
    }
```

Rysunek 5. *Uruchomienie zapytania do RTC*

```
        IWorkItemClient workItemClient = (IWorkItemClient) repo
                .getClientLibrary(IWorkItemClient.class);

        IWorkItem wi = workItemClient.findWorkItemById(1002,
                IWorkItem.FULL_PROFILE, null);

        IProcessClientService processClient = (IProcessClientService) repo
                .getClientLibrary(IProcessClientService.class);

        IProjectArea projectArea = (IProjectArea) processClient
                .findProcessArea(URI.create("Demos"), null, null);

        IAttribute attribute = workItemClient.findAttribute(projectArea,
                "timeSpent", null);

        new TimeSpentUpdate(3600000 * 3, attribute).run(wi, null);
```

Rysunek 6. *Aktualizacja zadania w RTC*

```
    private static class TimeSpentUpdate extends WorkItemOperation {

        private long timeSpent;
        protected IAttribute att;

        public TimeSpentUpdate(long duration, IAttribute att) {
            super("Initializing Work Item", IWorkItem.FULL_PROFILE);
            this.timeSpent = duration;
            this.att = att;
        }

        @Override
        protected void execute(WorkItemWorkingCopy workingCopy,
                IProgressMonitor monitor) throws TeamRepositoryException {

            IWorkItem workItem = workingCopy.getWorkItem();
            workItem.setValue(att, new Long(timeSpent));
        }
    }
```

Rysunek 7. *Aktualizacja zdania w RTC, cd.*

Rysunek 3 przedstawia podstawowy widok naszego narzędzia, które nazwaliśmy „Timekeeper". Zasadniczo składa się ono z trzech części. W prawej górnej części widzimy bieżącą listę zadań do wykonania. Dla naszych potrzeb mamy skonfigurowane pobieranie zadań z RTC, jednakże system nie jest ograniczony jedynie do RTC (więcej o tym w rozdziale „możliwości przyszłego rozszerzania"). Zadanie takie można przeciągnąć na pulpit, który znajduje się w dolnej części prawej strony ekranu. Czyniąc tak, powodujemy wyświetlenie się szczegółów zadania na pulpicie oraz, co ważniejsze, rozpoczynamy proces automatycznej rejestracji czasu spędzonego nad zadaniem. Objawia się to pojawieniem się nowego wpisu w lewej części – kalendarzu – który to wpis będzie co 5 minut aktualizowany tak długo, jak zadanie zostaje na pulpicie. Jeżeli skończymy pracę nad zadaniem, należy przycisnąć „Stop" na pulpicie i wpis w kalendarzu przestanie się aktualizować.

Integracja narzędzi

Oprócz interfejsu użytkownika, który opisałem w poprzednim punkcie, w systemie Timekeeper uruchomione są również procesy/usługi działające w tle. Zgodnie z wymaganiem (b) aplikację można używać w trybie offline. Wydawać by się mogło, iż wymagania (a) oraz (b) wykluczają się. Jednakże obydwie funkcjonalności udało nam się zaimplementować z użyciem „Google Gears", który w niedługiej przyszłości ma być zastąpiony implementacjami zgodnymi z HTML5. Oznacza to, że tworzone w kalendarzu wpisy przechowywane są po stronie przeglądarki internetowej w małej bazie danych na dysku w profilu użytkownika. Wpisy te muszą być synchronizowane z serwerem. Co każde 5 minut z poziomu przeglądarki wyzwalana jest synchronizacja wpisów lokalnych z serwerem.

Lista zadań również przechowywana jest w lokalnej bazie przeglądarki, dzięki czemu jest możliwe jej wyświetlenie również gdy nie mamy połączenia z Internetem. Lista ta jest odświeżana na żądanie i powoduje pobranie z RTC listy otwartych zadań do wykonania. Szczegóły tego interfejsu będą opisane w rozdziale „jazz-task".

Kolejnym procesem jest proces synchronizacji wpisów w kalendarzu z RTC (pole „Time Remaining" na Rysunku 2). Po stronie serwera naszej aplikacji uruchomiony jest scheduler, który wyzwala co noc przepisywanie czasów

z kalendarza do RTC. Wykonywane jest wtedy grupowanie względem numeru zadania, tak aby uzyskać sumaryczny czas spędzony nad zadaniem. Proces ten będzie opisany w rozdziale „jazz-feedback".

Do integracji z RTC użyliśmy bibliotek „Plain Java Client" dostępnych standardowo z RTC. Postaram się teraz przybliżyć, jak proste jest użycie tych bibliotek przy integracji niezależnych narzędzi z RTC. Pokażę to na przykładzie użytych przez nas w aplikacji Timekeeper obiektów z tych bibliotek.

Rysunek 4 przedstawia kod potrzebny do uruchomienia komunikacji z RTC. W linii (37) oraz (62) używamy obiektu TeamPlatform, aby zainicjować i zakończyć pracę z obiektami biblioteki PJCL. W linii (57) oraz (61) widzimy logowanie i wylogowywanie się z repozytorium. Dostarczenie hasła i użytkownika to odpowiedzialność zaimplementowanego in-line obiektu ILoginHandler. Natomiast adres samego repozytorium RTC podajemy w linii (40). Mając tak zainicjowane połączenie, możemy przystąpić do pracy z usługami RTC.

jazz-task

Stworzony przez nas moduł „jazz-task" pozwala na pobranie z RTC bieżącej listy otwartych zadań dla konkretnego użytkownika systemu Timekeeper.

Istotny dla artykułu fragment kodu został przedstawiony na Rysunku 5. Uruchomienie tego kodu spowoduje wypisanie na ekranie tytułów wszystkich zadań w obszarze projektowym „gj-timerec", które zwróci zapytanie „Open assigned to me".

W liniach 105-108 pobrana zostaje referencja na obszar projektu. W liniach 114-117 pobieramy wszystkie współdzielone zapytania z obszaru projektu RTC. Następnie iterujemy po nich (linia 119), szukając otwartych zadań przypisanych do programisty (linia 120). W liniach 122-123 uruchamiamy znalezione zapytanie, aby następnie pobrać wszystkie zadania zwrócone z tego zapytania (linie 125-127). W linii 128 drukujemy tytuł pojedynczego znalezionego zadania.

jazz-feedback

Moduł sprzężenia zwrotnego pozwala na aktualizację czasów z systemu Timekeeper to RTC, tak by można było dalej w RTC korzystać z funkcjonalności raportowania oraz monitorowania procesu wytwarzania oprogramowania. Używane przez nas API zostało zamieszczone na Rysunkach 6 oraz 7. Uruchomienie kodu z Rysunku 6 oraz 7 spowoduje zaktualizowanie czasu pracy nad zadaniem o numerze 1002 do 2 roboczogodzin. W liniach (73)-(74) następuje pobranie zadania o numerze 1002. Linie (79)-(80) to prezentowany już wcześniej sposób na pobranie referencji do obszaru projektu. Następnie w linii (82) pobieramy referencję na atrybut „czas spędzony nad zadaniem" w obszarze projektu. Aby zaktualizować zadanie, należy posłużyć się konstrukcją podaną w linii (85), która w rzeczy samej jest bardzo podobna do uruchomienia oddzielnego wątku w Javie. Zadanie aktualizacji musimy uruchomić w ten sposób ze względu na wielodostępowość RTC i możliwość wystąpienia konfliktu przy aktualizacji.

Rysunek 7 przedstawia, w jaki sposób programiści PJCL uwolnili nas od zarządzania wielodostępowością w RTC. Wszystko, co musimy uczynić, chcąc wykonać operację modyfikacji zadania, to rozszerzyć klasę WorkItemOperation. Zadba ona o pobranie najnowszej wersji zadania, wysłanie naszych zmian na serwer oraz potencjalne scalenie zmian z ewentualną modyfikacją, która mogła nastąpić równolegle. Nasz kod aktualizujący jest wolny od tych wszystkich szczegółów i ustawia jedynie pożądaną przez nas wartość atrybutu za pomocą zwykłego settera (linia (149)).

Możliwość przyszłego rozszerzania (SOA)

Sposób, w jaki udało nam się zintegrować RTC z naszą infrastrukturą i autorskim narzędziem (Timekeeper), daje nam bardzo dużą elastyczność. Jest to szczególnie istotne w dzisiejszych czasach, kiedy zmiany w komponentach oprogramowania używanych w firmie często wymuszane są raz na parę lat.

Dla porównania, firma, z którą współpracujemy od wielu lat, używa zarówno do rejestracji czasu pracy, zarządzania zadaniami oraz raportowania czasu do klientów jednego narzędzia. Jest to autorskie narzędzie wytworzone wewnętrznie ponad 10 lat temu. Obecnie firma ta bardzo chętnie wymieniłaby przynajmniej część tego rozwiązania na nowsze. Niestety jednak wokół tego monolitu jest skonfigurowanych tyle zależnych narzędzi, że zadanie to jest praktycznie niewykonalne.

Dla odmiany, nasza infrastruktura zawiera teraz oddzielne komponenty do zarządzania zadaniami i rejestracji czasu pracy. Raportowanie jest jeszcze w fazie rozwoju, ale można je również uważać za oddzielny komponent. Gdybyśmy chcieli w przyszłości wymienić silnik zarządzania zadaniami, możemy to zrobić fazowo:

- najpierw dodając nowy silnik zarządzania zadaniami do narzędzia rejestracji czasu pracy (Timekeeper) – w tym momencie dwa silniki zarządzania zadaniami działają równolegle,
- potem stopniowo doimplementowywać zależne od silnika zadań interfejsy,
- na koniec usunąć powiązanie ze starym silnikiem.

Podobnie możemy postąpić z dowolnym innym komponentem w naszej infrastrukturze. Dzięki temu nasze rozwiązanie jest nie tylko aktualne dzisiaj, ale również będzie aktualne w przyszłości.

Przykładem może być sprzęg z systemem SAP, nad którym obecnie pracujemy. Podobnie jak RTC jest silnikiem zadań, również za taki silnik może służyć system SAP z wdrożonym modułem obsługi HR. Aby nasze narzędzie współpracowało z SAP, wystarczy abyśmy zaimplementowali odpowiedniki serwisów jazz-task oraz jazz-feedback dla SAP. Kiedy skończymy pracę nad tymi modułami, będziemy mieli możliwość rejestracji czasu pracy dla RTC oraz SAP oddzielnie, lub współbieżnie w jednej instancji systemu Timekeeper.

Podsumowanie

Jesteśmy zespołem programistów nie uznającym kompromisów, jeżeli chodzi o jakość i elastyczność wytwarzanego oprogramowania. Problem rejestracji czasu pracy, z którym się zmierzyliśmy, okazał się dużo bardziej skomplikowany niż się spodziewaliśmy. Zachowując otwarty umysł, udało nam się wykorzystać najnowsze technologie (RTC, GWT, Google Gears), zaoszczędzić ile się tylko dało, używając standardowych komponentów (RTC), oraz sporo się nauczyć, modelując proces, w którym uczestniczymy na co dzień.

Dzięki narzędziu Timekeeper otrzymaliśmy możliwość dokładnej rejestracji czasu pracy. Jest to zasadnicze dla zachowania dobrych stosunków z klientami. Narzędzie RTC dostarczyło nam wszystkich funkcjonalności związanych z planowaniem oraz monitorowaniem procesu wytwarzania oprogramowania.

Kończąc niniejszy artykuł, chciałbym tym samym podziękować firmie IBM za udostępnienie tak doskonałego i elastycznego narzędzia jakim jest RTC. Bez niego moje nielubiane poniedziałkowe poranki trwałyby zapewne do wtorku.

Dominik Zalewski

Obecnie prowadzi własną mikro-firmę (genijusz) świadczącą usługi z zakresu integracji systemów informatycznych. Jego czteroosobowy zespół zbudowany jest z młodych wysoko wykwalifikowanych osób, które nie boją się wyzwań takich jak połączenie najnowszych technologii przeglądarkowych z istniejącymi od lat systemami SAP lub AS400. Dominik działa również w stowarzyszeniach typu non-profit: jest dyrektorem ds. edukacji COMMON Polska (common.org.pl) oraz członkiem CEAC (Common Europe Advisory Council). Ma on również solidne podłoże teoretyczne, jako iż posiada tytuł doktora nauk matematycznych w dziedzinie informatyki uzyskany na Poznańskim Uniwersytecie.

IBM Premier Business Partner

PREMIUM TECHNOLOGY
FROM KNOWLEDGE TO PRACTICE

Authorized IBM Training

Z TEORIĄ W PRAKTYKĘ

www.premiumtechnology.pl

Memoria minuitur nisi exercetur

Ostatnio obserwuje się ze strony biznesu coraz większe zainteresowanie pracownikami posiadającymi szeroki i przekrojowy zestaw kompetencji. Otwarte pozostaje pytanie w jaki sposób kształtować rozwój personelu aby to osiągnąć? Rynek sygnalizuje, że coraz trudniej znaleźć osobę łączącą umiejętności analityka, projektanta, testera i jednocześnie posiadającą wiedzę specjalistyczną i biznesową.
Odpowiedzią są szkolenia dedykowane, umożliwiające zdobycie kwalifikacji krzyżowych – pozwalające nauczyć biznes i IT wzajemnego rozumienia, dostrzegania swoich potrzeb i uwarunkowań.
Z zaangażowaniem będziemy wspierać rozwój kadr według modelu hybrydowego profilu, w którym pracownicy wspierają rozwój organizacji swoją wiedzą stając się specjalistami z ważnego pogranicza – biznesu i IT.

Piotr Kowalski, starszy konsultant

Ardua prima via est

Do celu wiedzie zwykle wiele dróg. W szczególności, jeśli celem tym jest stworzenie na czas systemu informatycznego, który realizował będzie oczekiwania użytkowników nie tylko w odniesieniu do tego, co robi, ale również jak to robi. Już wybór odpowiedniej drogi jest często niełatwym do rozwiązania problemem. Czy potrzebujemy sformalizowanego procesu wytwórczego, który jasno będzie precyzował kto, co i kiedy ma wykonywać i jakie produkty dostarczać? Czy może lepiej zrealizować projekt zgodnie z podejściami „lekkimi"? Ale czy nasza organizacja jest na to gotowa? Czy dysponujemy odpowiednimi osobami, dzięki którym odniesiemy sukces? A może własny, dostosowany do specyfiki organizacji proces wytwórczy, który czerpać będzie z wielu różnych podejść i praktyk? Jeśli tak, to jak taki proces opisać, udokumentować i rozpowszechniać? Pytań i wątpliwości jest wiele, zapraszamy do wspólnego z nami szukania odpowiedzi i rozwiązań.

Arkadiusz Drabiński, starszy konsultant

Hannibal ad portas!

Dzisiaj, mówiąc o zapewnieniu jakości oprogramowania, nie można pomijać spraw bezpieczeństwa, które niekwestionowanie stały się integralnym elementem tego procesu. Podobnie – zajmując się bezpieczeństwem IT, nie można ignorować pozostałych aspektów tworzenia i eksploatacji systemów informatycznych. Realizując usługi z zakresu bezpieczeństwa IT zawsze stosujemy podejście pozwalające dostrzec wszystkie istotne aspekty podejmowanego zagadnienia, niezależnie od tego czy jest to kompleksowy audyt czy wsparcie klienta w testach.

Marcin Koprowski, starszy konsultant

Usus magister est optimus

„Nasze oprogramowanie jest bezbłędne". „Zbudowaliśmy system, którego oczekiwał klient".
Czy aby na pewno? Bezbłędne oprogramowanie nie istnieje, a oczekiwania klienta bardzo często zmieniają się podczas trwania projektu.
Testy, nie tylko samego systemu, ale i artefaktów procesu produkcji oprogramowania, pozwalają zwiększyć jakość i zaufanie do produkowanego oprogramowania pod warunkiem, że zostaną przeprowadzone w prawidłowy sposób. Błędy wykryte podczas testów i usunięte przed uruchomieniem są znacznie mniej kosztowne oraz zmniejszają ryzyko utraty naszej wiarygodności u klienta. Zapewnienie jakości oprogramowania jest wieloetapowym procesem, który jest przeprowadzany w zależności od oczekiwań klienta, stopnia złożoności charakteru danego przedsięwzięcia informatycznego.
W oparciu o nasze wieloletnie doświadczenie pomogliśmy wielu organizacjom zorganizować zespoły testowe, zoptymalizować proces testów, przeprowadzić testy wydajnościowe i zautomatyzować testy funkcjonalne.

Krzysztof Zaorski, starszy konsultant

Bene facit, qui ex aliorum erroribus sibi exemplum sumit

Nie ulega wątpliwości, że stosowanie narzędzi wspomagających i automatyzujących wytwarzanie oprogramowania może dać wymierne korzyści, zarówno wytwórcom oprogramowania, jak i firmom, które takie oprogramowanie zamawiają. Zastosowanie w praktyce opisanych w niniejszym numerze technik i narzędzi, może pomóc w lepszym zdefiniowaniu potrzeb użytkowników, sprawniejszym dostarczeniu oczekiwanego produktu oraz zweryfikowaniu, czy spełnia on wymagania jakościowe.
Nie zapominajmy jednak, że zakup narzędzi, nawet tych najbardziej zaawansowanych, może jednak nie przynieść oczekiwanych rezultatów. Nawet wtedy, gdy solidnie przygotowaliśmy się do zakupu, dokonaliśmy przeglądu rozwiązań dostępnych na rynku i wybraliśmy rozwiązanie spełniające nasze oczekiwania. Warto wtedy, korzystając z rad wykwalifikowanych konsultantów, zaplanować proces wdrożenia, obejmujący działania typu proof of concept, szkolenia oraz mentoring, pozwalające osiągnąć większy zwrot z zaplanowanej inwestycji.

Arkadiusz Jemielity, starszy konsultant

Dictum sapienti satt

Działy IT często opierają wiele swoich projektów rozwojowych tylko na wysokich kwalifikacjach swoich pracowników i ich zdolności poznawania nowych technologii. Obecnie coraz więcej przedstawicieli kadry kierowniczej dostrzega konsekwencje takiego podejścia – zespoły uczą się na własnych błędach, poświęcając na próby i eksperymenty czas, który można byłoby wykorzystać bardziej produktywnie.
Wartą uwagi propozycją jest zastosowanie w projekcie nowoczesnego mentoringu, przynoszącego w projekcie większe korzyści niż próby samodzielnego zmierzenia się z problemami. Dobrze prowadzony mentoring pozwala na znacznie szybsze rozwiązanie problemu i uzyskanie wymiernej korzyści biznesowej.

Maciej Skorulski, starszy konsultant

Wytwarzanie oprogramowania

z wykorzystaniem IBM Rational Team Concert

Firma IBM i jej dział Rational od lat jest światowym liderem w zakresie dostarczania rozwiązań wspierających wytwarzanie oprogramowania. Produkty z portfolio Rational zyskały uznanie u wielu klientów, a metodyka Rational Unified Process jest powszechnie stosowanym standardem.

Wraz z wprowadzeniem nowej platformy Jazz IBM udostępnił użytkownikom kilka ciekawych produktów. Jednym z nich jest IBM Rational Team Concert.

Produkt ten można zdefiniować jako platformę wspierającą wytwarzanie oprogramowania. Narzędzie to integruje w sobie wiele funkcjonalności takich jak planowanie, raportowanie, zarządzanie pracą zespołu, wersjonowanie kodu.

Rational Team Concert można zintegrować z innym rozwiązaniami IBM Rational i firm partnerskich, co umożliwia zbudowanie kompletnego środowiska do rozwoju oprogramowania obejmującego swoim zasięgiem m.in. zarządzanie wymaganiami, zarządzanie projektem, testy bezpieczeństwa i zarządzanie jakością czy też narzędzia programistyczne.

Rozwiązanie to wspiera zarówno metodyki zwinne wytwarzania oprogramowania, jak i bardziej "tradycyjne" podejścia. W RTC zaimplementowano szablony dla projektów zarządzanych według powszechnie wykorzystywanych metodyk.

Dodatkową zaletą platformy, która zapewne przypadnie do gustu użytkownikom, jest jej wysoka funkcjonalność i użyteczność interfejsu wykorzystująca wiele sprawdzonych rozwiązań spotykanych w serwisach Web 2.0. Natomiast w przypadku developerów będzie to integracja z narzędziami programistycznymi jak np. Eclipse.

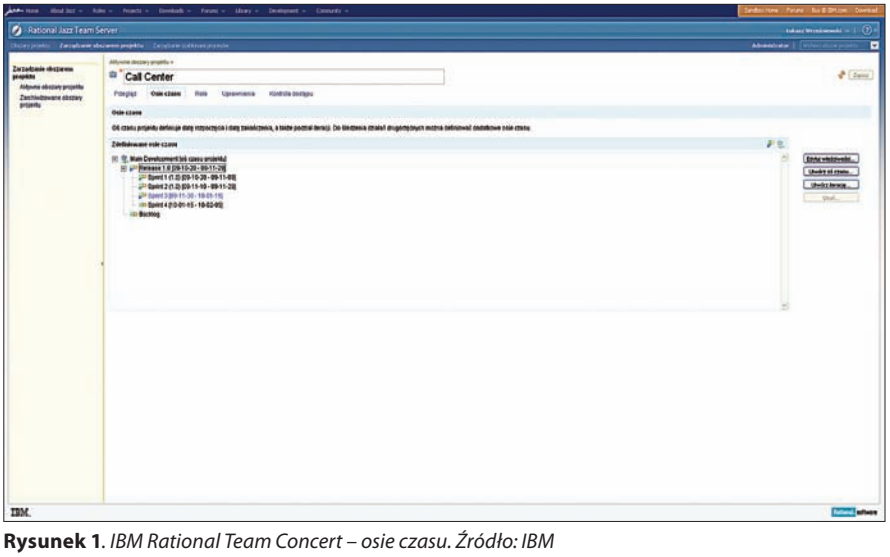
Celem niniejszego artykułu jest przedstawienie podstawowych możliwości, jakie daje tworzącym oprogramowanie IBM Rational Team Concert, oraz przyjrzenie się temu rozwiązaniu z dwóch perspektyw: osoby odpowiedzialnej za sprawną realizację projektu oraz uczestnika projektu – programisty.

Wprowadzenie do Rational Team Concert

Rational Team Concert zbudowane jest w oparciu o platformę Jazz. Team Conert można obsługiwać za pomocą interfejsu webowego uruchomianego w przeglądarce internetowej oraz poprzez narzędzie programistyczne, do którego instalowana jest odpowiednia wtyczka. W artykule posłużymy się przykładem Eclipse, jednak może to być również IBM Rational Application Developer, Microsoft Visual Studio i NetBeans.

Pełniąc rolę menadżera projektu, pracę rozpoczynamy od zalogowania się RTC za pośrednictwem przeglądarki internetowej. Po zalogowaniu dostępnych jest kilka paneli:

- Panele kontrolne (ang. *Dashboard*) - tworzenie paneli, na których możemy umiesz-



Rysunek 1. *IBM Rational Team Concert – osie czasu. Źródło: IBM*

czać istotne dla nas i dla projektu informacje,

- Obszary Projektów (ang. *Project Areas*) - w panelu możemy zdefiniować obszary realizowanych projektów, zespół projektowy oraz skorzystać z szablonu procesów dla danego projektu,
- Plany (ang. *Plans*) - w panelu definiujemy plany projektu na różnym poziomie szczegółowości,
- Elementy Pracy (ang. *Work Items*) – w panelu określamy elementy pracy występujące w projekcie,
- Kontrola kodu źródłowgo - (ang. *Source Control*) wersjonowanie kodu źródłowego,
- Procesy budowania (ang. *Builds*) - w panelu konfigurujemy wydania i możemy pobrać kompilacje kodu z danego dnia,
- Raporty – w panelu definiujemy raporty.

Rozpoczynanie pracy z projektem

Pracę nad nowym projektem rozpoczynamy od zdefiniowania nowego obszaru projektu oraz przydzielenia zespołu. Konieczne są tutaj uprawnienia administratora. Korzystamy z funkcji Obszary projekty (ang. *Project Areas) – Zarządzaj obszarami projektu (ang. Manage Project Areas - Utwórz obszar projektu (ang. Create Project Area).*

W zakładce *Przegląd* definiujemy nazwę i opis projektu, przydzielamy szablon procesu *(Proces),* a także zespół projektowy *(Członkowie)* i administratorów projektu *(Administratorzy).*

Bardzo ważnym elementem przy tworzeniu nowego projektu jest zdefiniowanie osi czasu, gdyż z jej wykorzystaniem będą tworzone plany. W zakładce *Osie czasu* tworzymy główną oś czasu projektu *(Utwórz oś czasu...)* - zaznaczamy opcję *To jest oś czasu projektu*

W kolejnych krokach tworzymy iteracje *(Utwórz iterację ...)* Możemy stworzyć strukturę drzewa z zagnieżdżonymi iteracjami. W przypadku członków zespołu po kliknięciu na dane użytkownika możemy edytować właściwości użytkownika takie jak informacje kontaktowe, role, czas pracy, koszty pracy, umiejętności i kompetencje.

Dostępne są również zakładki *Role, Uprawnienia, i Kontrola dostępu,* gdzie określamy parametry związane z prawami dostępu i rolami w projekcie.

Po zapisaniu zmian nasz projekt jest dostępny w panelu *Obszary projektu.*

Panele kontrolne

Po wybraniu obszaru utworzonego przez nas uzyskujemy dostęp do obszaru roboczego *Panelu kontrolnego* umożliwiającego łatwy dostęp i przegląd istotnych dla nas elementów projektu, jak np. lista członków zespołu, aktualne koszty projektu, aktualne zadania, zdarzenia, linki, raporty, zawartość z aplikacji powiązanych i kompilacje. Wszystkie te elementy w formie *apletów przeglądania* możemy swobodnie konfigurować.

Plany i Elementy pracy

W IBM Rational Team Concert przebieg projektu opiera się o elementy pracy związanych z procesem wytwórczym oprogramowania.

W obszarze *Elementy pracy* wybieramy funkcję *Utwórz element pracy*, a następnie definiujemy jego parametry.

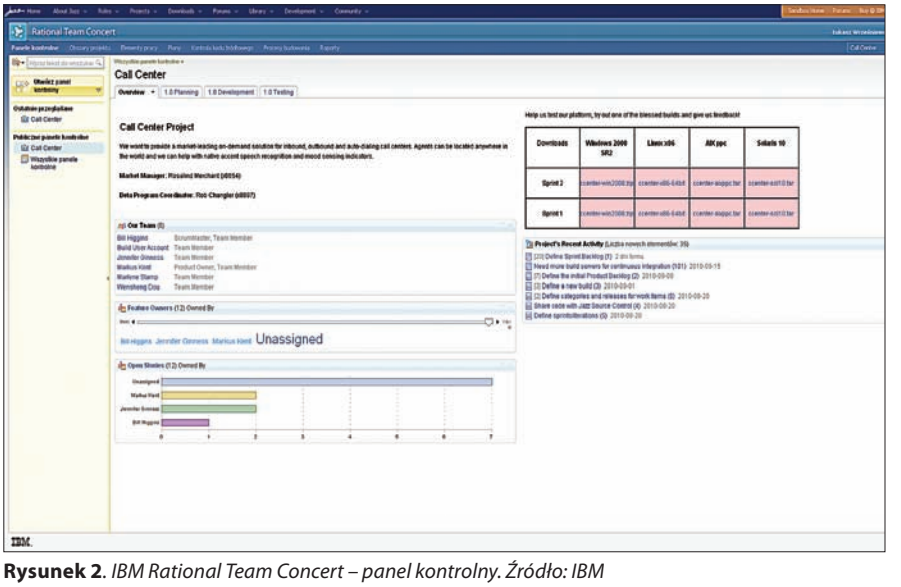
W obszarze *Elementy pracy* możemy tworzyć nowe również kwerendy wyświetlające elementy pracy według określonych parametrów – *Utwórz zapytanie.*

W obszarze *Plany* możemy utworzyć plan dla danego projektu, który bazuje na Osi czasu określonej podczas definiowania obszaru projektu. Plany można wyświetlać w różnych widokach. Mogą to być np. iteracje, gdzie plan jest podzielony na iteracje. Do każdej iteracji możemy dodawać nowe elementy pracy *(ikona +)*, elementy pracy można zagnieżdżać, tworząc np. podzadania *(ikona –> →)*, możemy również przenosić je pomiędzy iteracjami. Co istotne, program obsługuje funkcję drag and drop.

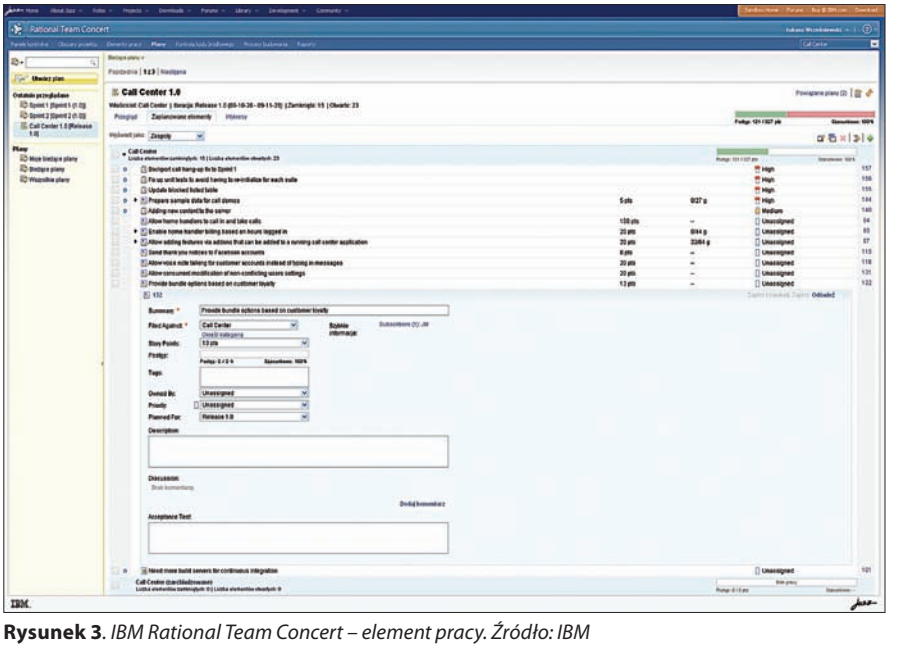
Tworząc element pracy, określamy jego szacowany czas oraz priorytet. Przypisujemy go do określonego użytkownika. Możemy przypisać element do określonej iteracji.

IBM Rational Team Concert oferuje również widok z perspektywy podziału zadań pomiędzy członków zespołu i aktualnie wykonywanej pracy. Pozwala to menadżerowi na łatwe bilansowanie obciążenia pracą poszczególnych członków.

Obszary elementów pracy i planów pozwalają osobie zarządzającej projektem na śledzenie postępów, reagowanie na pojawiające



Rysunek 2. *IBM Rational Team Concert – panel kontrolny. Źródło: IBM*



Rysunek 3. *IBM Rational Team Concert – element pracy. Źródło: IBM*

się problemy oraz na zarządzanie pracami zespołu.

Raporty

Przydatnym elementem RTC jest możliwość generowania raportów. Można tego dokonać w obszarze *Raporty – Utwórz raport*, następnie wybieramy szablon interesującego raportu, który zostaje wygenerowany po kliknięciu opcji *Zapisz*.

Tworzenie oprogramowania

Bardzo wygodną dla developerów funkcjonalnością RTC jest jego integracja z Eclipse – powszechnie znanym i wykorzystywanym narzędziem programistycznym.

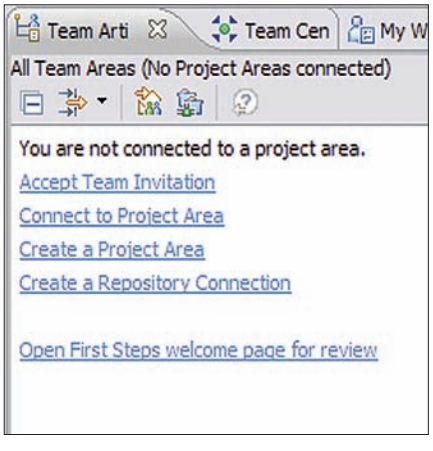
Po uruchomieniu Eclipse i stworzeniu nowego obszaru roboczego otwieramy perspektywę *Team Artifacts*. Możemy zaakceptować zaproszenie do zespołu *Accept Team Invitation* lub też podłączyć się do obszaru projektu *Connect to Project Area*.

Po połączeniu się z repozytorium IBM Rational Team Concert uzyskujemy dostęp do artefaktów związanych z projektem takich jak plany, procesy budowania, kontrola kodu źródłowego, elementy pracy.

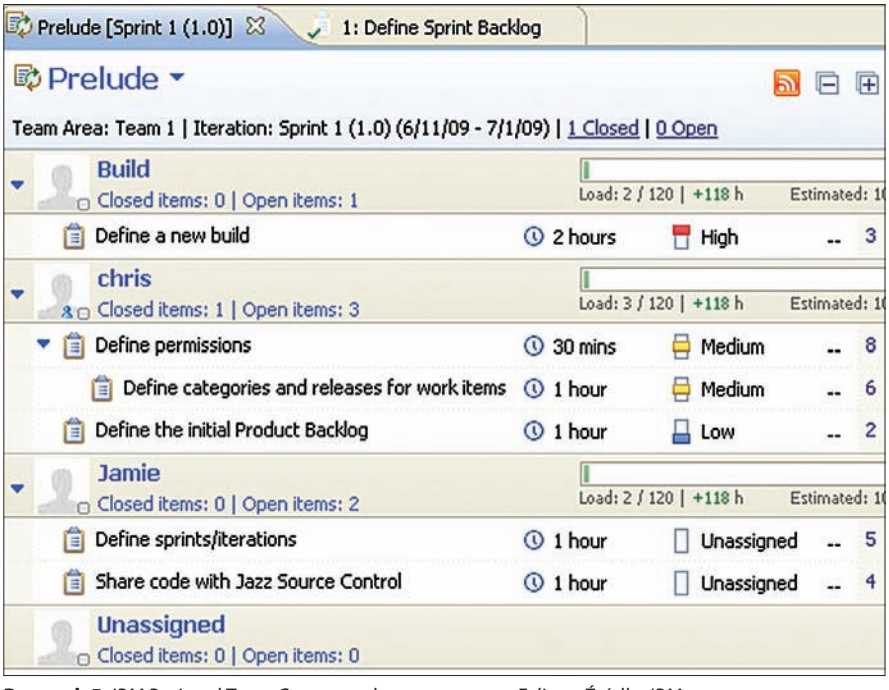
Praca z elementami pracy

W perspektywie *My Work* znajdują się elementy pracy, za które jesteśmy odpowiedzialni. W momencie rozpoczęcia pracy nad danym elementem (klikamy *Start Working*) pozostali członkowie zespołu widzą przydzielone do nas elementy i to, że jesteśmy w trakcie pracy nad nimi. Z poziomu Eclipse możemy również zmieniać parametry elementów oraz komunikować się z innymi członkami zespołu. Funkcjonalności odpowiadają tym wykorzystywanym w interfejsie webowym.

IBM Rational Team Concert mierzy czas poświęcony każdemu zadaniu i aktualizuje plany.



Rysunek 4. *IBM Rational Team Concert – połączenie z obszarem projektu Rational Team Concert za pośrednictwem Eclipse. Źródło: IBM*



Rysunek 5. *IBM Rational Team Concert – elementy pracy w Eclipse. Źródło: IBM*

Wersjonowanie kodu

Platforma Jazz posiada zaimplementowane wsparcie dla wersjonowania kodu, dzięki czemu możliwa jest równoległa praca programistów nad danym zagadnieniem. W perspektywie *Team Artifacts* znajduje się grupa artefaktów o nazwie strumienie. Strumień jest to repozytorium obiektów zawierających komponenty, czyli zbiory artefaktów.

Komponenty możemy zaimportować z repozytorium sieciowego do lokalnego obszaru roboczego.

Po kliknięciu prawym przyciskiem myszy na komponencie i wyborze opcji *Show baselines* uzyskujemy dostęp do repozytorium wersji danego komponentu, który również możemy pobrać do lokalnego obszaru roboczego.

Jeżeli dwóch użytkowników pracuje nad tym samym plikiem, wywoływana jest funkcjonalność rozwiązywania konfliktów, co pozwala na dokonanie najbardziej optymalnej zmiany i porównanie dwóch wariantów pliku.

Wersjonowanie jest również wykorzystywane przez inne komponenty IBM Rational Team Concert, jak procesy budowania i elementy pracy.

Wydania

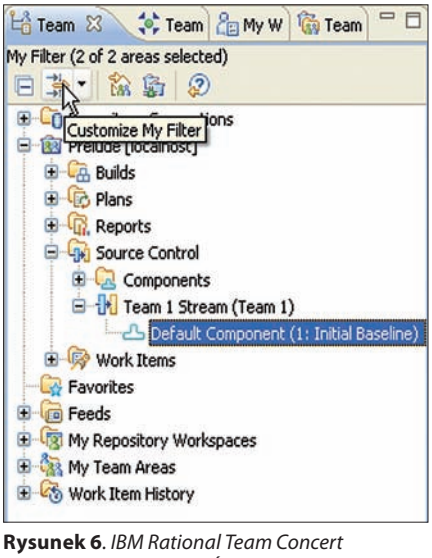
IBM Rational Team Concert posiada mechanizm budowania pozwalający na kompilowanie kolejnych wydań (ang. *Builds*). W przypadku wydań można również korzystać z repozytoriów w podobny sposób jak w przypadku kodu.

W oprogramowaniu Rational Team Concert firma IBM dostarcza wiele wartościowych funkcjonalności wszystkim uczestnikom procesu wytwórczego oprogramowania. Pozwalają one na lepsze śledzenie przebiegu prac projektowych oraz ułatwiają komunikację pomiędzy uczestnikami procesu. Wszystkie te zalety są istotnymi czynnikami sukcesu każdego przedsięwzięcia informatycznego.

W artykule wykorzystano zrzuty ekranu ze stron internetowych *www.ibm.com* i *www.jazz.net*.

Łukasz Wrześniewski

Członek Zarządu firmy Better Poland Sp. z o.o. W swojej pracy zajmuje się rozwojem produktów szkoleniowych w obszarze technologii informatycznych, a także jest odpowiedzialny za dział zajmujący się rozwiązaniami IBM Rational. Interesuje się nowymi trendami w inżynierii oprogramowania i zarządzaniu projektami.



Rysunek 6. *IBM Rational Team Concert – wersjonowanie kodu. Źródło: IBM*

IBM Rational Method Composer – portal procesowy

IBM Rational Method Composer jest narzędziem dla inżynierów procesu, liderów zespołów i osób kierujących projektami lub programami, pozwalającym przygotować definicję procesu wytwórczego oprogramowania na potrzeby danej organizacji lub projektu.

W realizowanych projektach zazwyczaj działamy w oparciu o jakiś proces. Niezależne od tego, czy jest on mniej lub bardziej sformalizowany, często wielu członków zespołu nie potrafi wskazać, gdzie znajdują się dokumenty zawierające definicję tego procesu lub procedury postępowania w projekcie. Dzieje się tak dlatego, że dokumentacja procesu jest zazwyczaj mało przyjazna dla czytelnika, zawiera dużo odwołań do innych dokumentów i często rozmija się z rzeczywistym procesem.

Rozwiązaniem tych problemów może być właśnie narzędzie IBM Rational Method Composer, pozwalające na przygotowanie definicji procesu wytwórczego, a następnie jego publikację w postaci portalu WWW lub dokumentu. Dzięki temu czytelnicy otrzymują wygodną w lekturze postać procesu, gdzie z łatwością odnajdą interesujące ich informacje oraz powiązane z nimi inne elementy, takie jak szablony stosowanych dokumentów.

RMC automatycznie dba o zachowanie spójności wszystkich elementów procesu, jak również pozwala na wykorzystanie w jego opracowaniu sprawdzonych standardów, między innymi takich jak metodyka IBM Rational Unified Process (RUP).

Rational Unified Process (RUP)

RUP jest metodyką wytwarzania oprogramowania niezależnie od typu projektu, jego rozmiaru czy stosowanej technologii. Daje wskazówki poszczególnym osobom zaangażowanym w projekt, jak powinni wykonywać swoje zadania, jakie powinny być wyniki tych prac oraz jakie elementy są niezbędne do wytworzenia danego produktu i kto będzie wykorzystywał ten produkt.

Rational Unified Process został zbudowany na bazie najlepszych praktyk inżynierii oprogramowania. Są to:

- Iteracyjne wytwarzanie oprogramowania (*Iterative Development*),
- Zarządzanie wymaganiami (*Requirement Management*),
- Architektury oparte na komponentach (*Component-based architecture*),
- Modelowanie wizualne oprogramowania za pomocą języka UML,
- Kontrola jakości oprogramowania (*Quality Assurance*),
- Proces kontroli zmian w oprogramowaniu (*Change Management*).

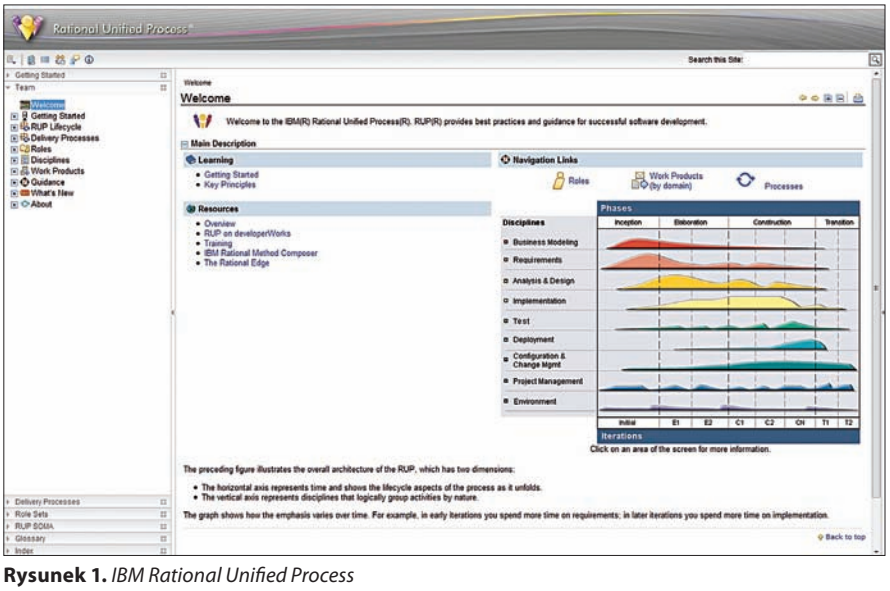
Proces ten został przygotowany w taki sposób, aby efektywnie wspierać uczestników projektu w ich zadaniach. Każda z ról występujących w projekcie znajdzie tam przydatne wskazówki dotyczące wykonywanych czynności i wytwarzanych produktów. Dla poszczególnych produktów znajdziemy nie tylko opis ich celu i zakresu, ale również zestaw szablonów i przykładów, umożliwiających ich przygotowanie i ocenę.

Opis zadań obejmuje informacje i o produktach wytwarzanych i produktach niezbędnych do ich wykonania,a opracowanych w ramach innych czynności. Dodatkowe wsparcie stanowi pomoc narzędziowa opisująca kroki wykonywane w ramach procesu dla technologii dostarczanej przez IBM, np. narzędzia do modelowania, testów itp.

Obecna wersja IBM Rational Unified Process została przygotowana i opublikowana z wykorzystaniem narzędzia IBM Rational Method Composer.

Biblioteki procesowe

Aby nauczyć się stosować metodykę wytwarzania oprogramowania, ludzie sięgają do literatury i wewnętrznych procedur, uczestniczą w szkoleniach lub korzystają z usług zewnętrznych konsultantów. Różne techniki są bogato opisane w dostępnych materiałach i dokumentują konkretne zagadnienia, np. w jaki sposób dokonać transformacji Modelu Przypadków Użycia Systemu do Modelu



Rysunek 1. *IBM Rational Unified Process*

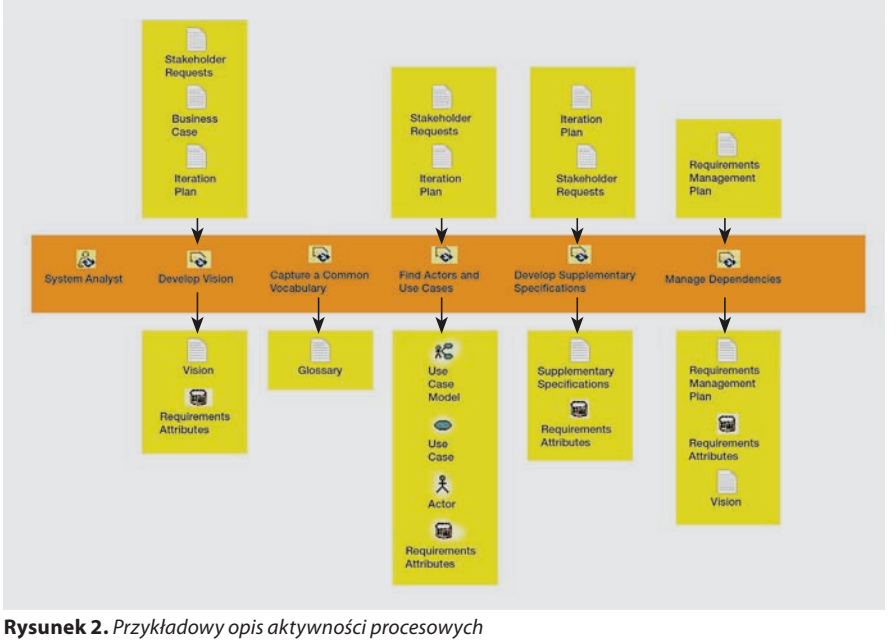
Projektowego, zidentyfikować przypadki testowe itp.

IBM Rational Method Composer (RMC) pozwala na przygotowanie takich informacji dla uczestników projektu według predefiniowanego schematu. Jest nim Unified Method Architecture (UMA), który bazuje na standardzie Software Process Engineering Metamodel (SPEM) opublikowanym przez Object Management Group (OMG). Schemat ten został przygotowany na potrzeby opisu metod i procesu wytwarzania oprogramowania, jednak nie musi się ograniczać tylko i wyłącznie do tego obszaru. Możliwe jest rozszerzenie definicji o procesy wykraczające poza typowe zagadnienia inżynierii oprogramowania, np. procesy związane z utrzymaniem systemów czy też przygotowania oferty handlowej.

W ramach schematu UMA definiuje następujące elementy kluczowe z punktu widzenia procesu:

- Role – definiujące niezbędne umiejętności danej osoby w niej występującej, oraz odpowiedzialność za przygotowanie produktów i wykonanie poszczególnych zadań.
- Produkty – definiujące wyniki wykonywanych zadań.
- Zadania – definiujące sposób wykonania prac przez konkretne role procesowe, każde zadanie posiada zdefiniowane produkty niezbędne do jego wykonania i produkty wynikowe.

Na bazie tych elementów definiowana jest dynamika procesu wytwórczego. Obejmuje ona sekwencję aktywności procesowych, której wykonanie pozwala osiągnąć zamierzone cele. Standardowo opisane są 2 grupy procesów: wzorce procesowe (*Capability Patterns*) i procesy dostarczania (*Delivery Processes*).



Rysunek 2. *Przykładowy opis aktywności procesowych*

Wzorce procesowe definiują sekwencję wykonywaną w określonych obszarach lub etapach prac. W IBM Rational Unified Process podstawowe wzorce obejmują poszczególne dyscypliny inżynierii oprogramowania, występujące zazwyczaj w każdym projekcie. Są to:

- Modelowanie biznesowe (*Business Modeling*),
- Zarządzanie wymaganiami (*Requirements*),
- Analiza i projektowanie (*Analysis & Design*),
- Implementacja (*Implementation*),
- Testy (*Test*),
- Wdrożenie (*Deployment*),
- Zarządzanie zmianą i konfiguracją (*Configuration & Change Management*),
- Zarządzanie projektem (*Project Management*),
- Środowisko (*Environment*).

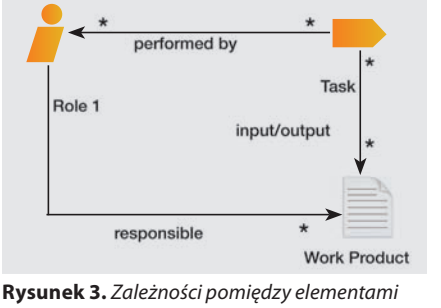
Procesy dostarczania pokazują, jak aktywności, zdefiniowane w poszczególnych wzorcach procesowych, zależą od siebie w trakcie całego cyklu życia projektu, od jego rozpoczęcia do zakończenia. Dla standardowego procesu zdefiniowane są różne typy procesów dostarczania uzależnione od skali projektu (duży, średni, mały) czy też jego charakteru lub stosownej technologii (projekt modelowania procesów biznesowych, projekt realizowany z wykorzystaniem podejścia SOA, itp.).

Przygotowanie procesu na podstawie istniejącej biblioteki procesowej

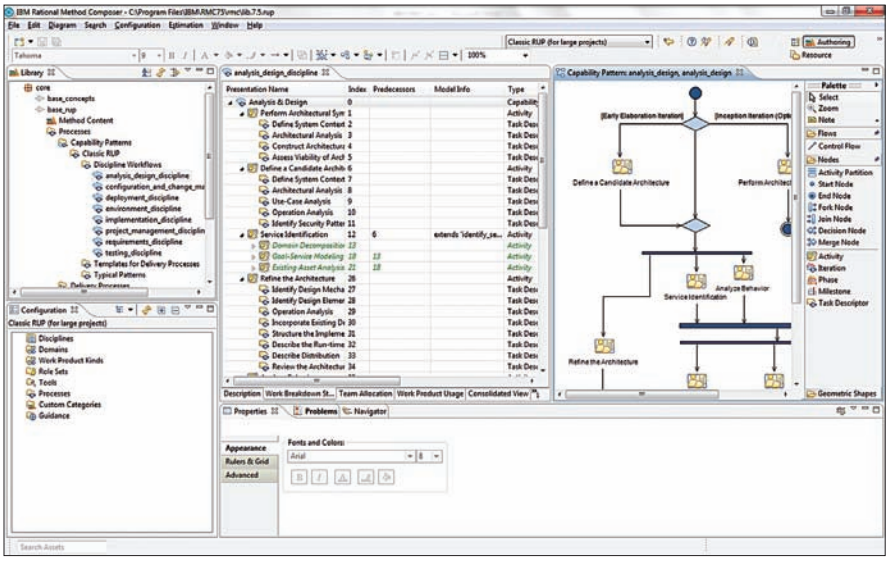
Typową metodą przygotowania własnego procesu jest jego zdefiniowanie na bazie dostarczanych standardowych elementów. Taka metoda nie wymaga od inżyniera procesu definiowania własnych elementów strukturalnych, takich jak role, zadania i produkty. Dzięki temu można bardzo szybko przygotować gotową publikację procesu wytwórczego obejmującą interesujące nas zagadnienia. Odbywa się to za pomocą konfiguracji procesu, gdzie wybieramy poszczególne komponenty procesowe oraz określamy sposób ich przeglądania.

Na bazie tak przygotowanej konfiguracji możemy opublikować własny proces wytwórczy w postaci witryny WWW wyglądającej analogicznie do RUP-a dostarczanego przez firmę IBM.

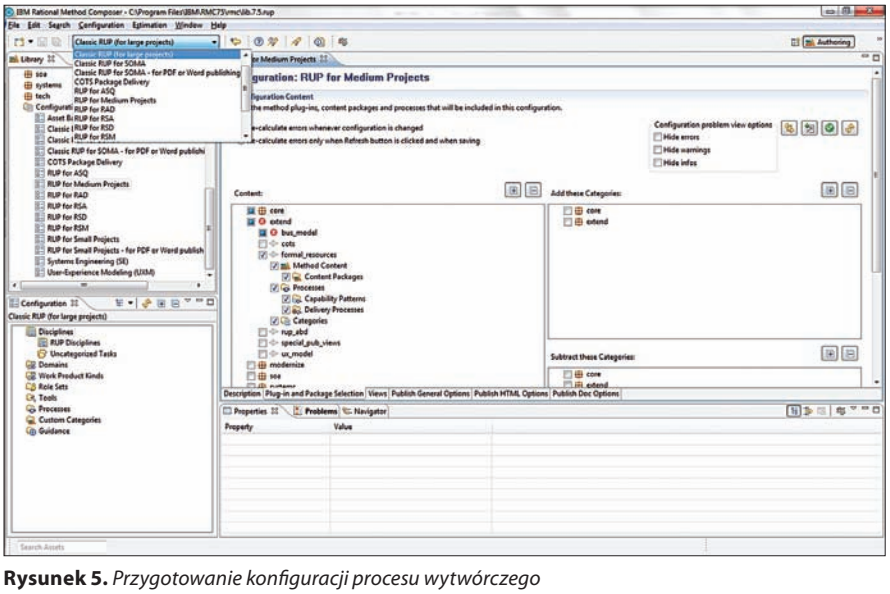
Wraz z produktem dostępne są różne komponenty procesowe, widoczne w narzędziu ja-



Rysunek 3. *Zależności pomiędzy elementami procesu*



Rysunek 4. *Definicja procesu*



Rysunek 5. *Przygotowanie konfiguracji procesu wytwórczego*

ko oddzielne wtyczki (Plug-in). Dostępne są wtyczki dotyczące takich zagadnień jak:

- RUP (*IBM Rational Unified Process*),
- SOA (*Service-Oriented Modeling and Architecture*),
- IT Governance,
- Wsparcie dla CMMI,
- Zarządzanie portfelem projektów,
- ITUP (*IBM Tivoli Unified Process*),
- Projekty utrzymaniowe,
- Rozwój istniejących rozwiązań/systemów,
- MDD (*Model-Driven Development*),
- UEX (*User Experience Modeling*),
- Microsoft .NET,
- J2EE,
- WebSphere Business Modeler,
- oraz inne udostępniane przez producenta i partnerów.

Aktualną listę dostępnych wtyczek można znaleźć na stronie *http://www.ibm.com/developerworks/rational/products/rup/*.

Tworzenie nowej definicji procesu

Alternatywą dla konfiguracji procesu w oparciu o standardowe elementy może być opracowanie własnej definicji procesu wytwórczego, obejmującej specyficzne dla naszej organizacji/projektu role, zadania i produkty. W takim przypadku możemy również wykorzystać istniejące komponenty, modyfikując lub rozszerzając ich definicję np. o własne szablony produktów.

W ramach przygotowania nowego procesu dostępne są wszystkie możliwości dokumentacyjne znane z IBM Rational Unified Process. Dlatego opis aktywności może obejmować nie tylko ich definicje, ale również takie elementy jak: własne przewodniki narzędziowe, poradniki czy też listy kontrolne.

Takie podejście jest bardzo czasochłonne, ale wynik dużo lepiej zaspokaja potrzeby naszych użytkowników procesu. W takim przypadku możliwe jest całkowite zlokalizowanie dokumentacji procesu, co ułatwia jego zrozumienie dla osób nie posługujących się biegle językiem angielskim.

Tworzenie własnych bibliotek procesowych

Jeśli opanujemy już w pełni możliwości narzędzia, możemy pokusić się o przygotowanie własnych wtyczek. Takie własne komponenty możemy następnie udostępnić naszym partnerom/dostawcom lub innym departamentom naszej firmy w celu przygotowania przez nich swoich definicji procesu wytwórczego obejmujących części wspólne z realizowanymi w naszych projektach procesami.

IBM Rational Method Composer

Obecna wersja RMC wydaje się najlepszą propozycją dla organizacji chcących uporządkować swój proces wytwórczy oprogramowania. Poza dedykowaną dla takiego zadania funkcjonalnością, jego niewątpliwą zaletą jest bogata baza wiedzy, którą możemy dowolnie wykorzystywać w swoich procesach.

Budowa portalu procesu wytwarzania oprogramowania z wykorzystaniem IBM Method Composer umożliwia:

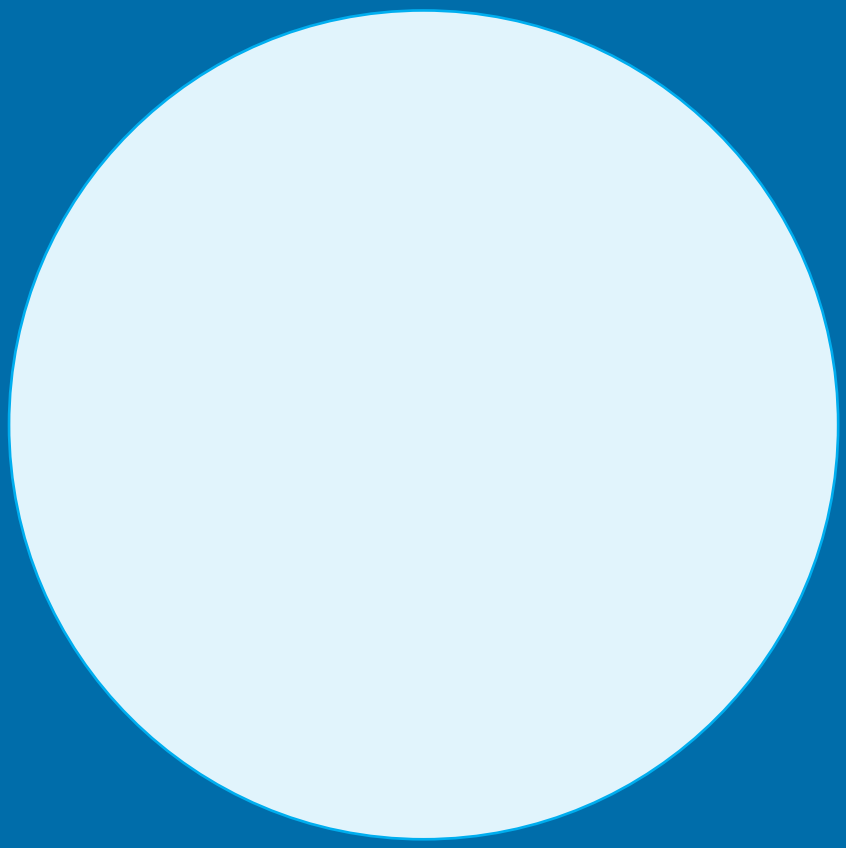
- Lepszą kontrolę procesu wytwórczego.
- Łatwiejsze wdrażanie w proces nowych pracowników i zespołów.
- Standaryzację procesu wewnątrz organizacji.
- Zdefiniowanie oczekiwań względem dostawców i podwykonawców odnośnie przekazywanych produktów i dokumentacji.
- Zwiększenie powtarzalności procesu, a co za tym idzie zwiększenie jakości produktów i obniżenie kosztów ich wytwarzania.
- Łatwiejsze szacowanie i planowanie projektów.
- Zmniejszenie kosztów utrzymania i rozwoju systemów poprzez zwiększenie jakości i dokładności dokumentacji projektowej.

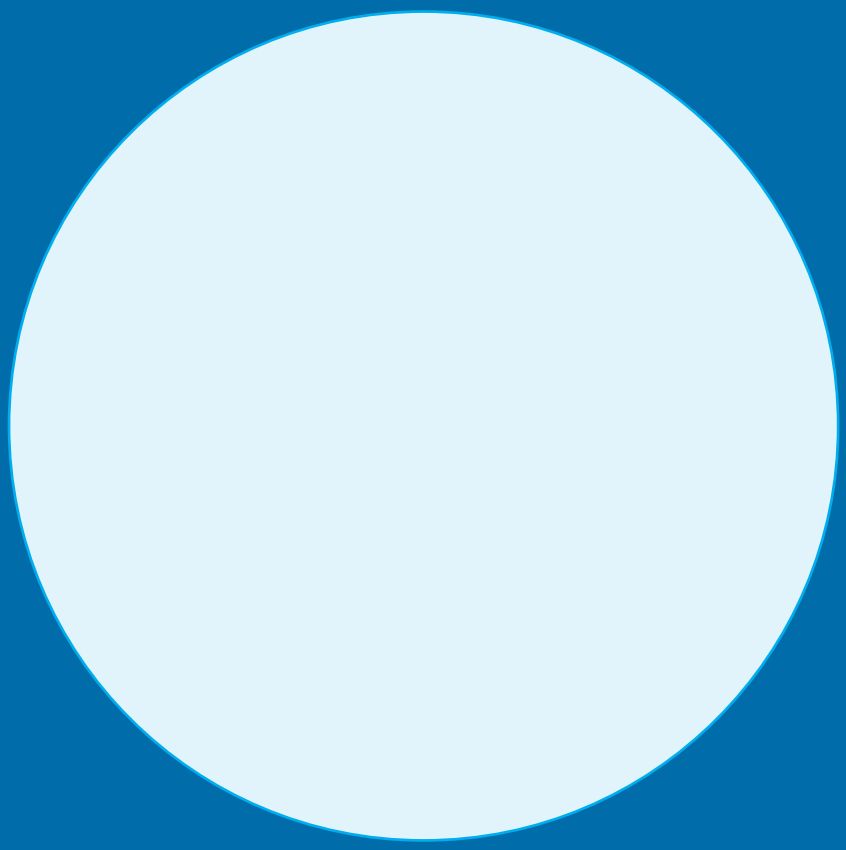
Maciej Skorulski

Starszy Konsultant w firmie Premium Technology. Zajmuje się wsparciem klientów w zakresie wdrażania procesów wytwarzania oprogramowania i zarządzania wymaganiami.

Odniesienia

- *http://www-01.ibm.com/software/awdtools/rmc/*
- *http://www.ibm.com/developerworks/rational/library/nov05/kroll/index.html*
- *http://www.ibm.com/developerworks/downloads/r/rup/*
- *http://www.ploug.org.pl/konf_07/materialy/pdf/12_rup.pdf*
- *http://www.ibm.com/developerworks/rational/products/rup/*
- *ftp://ftp.software.ibm.com/software/rational/web/datasheets/rmc_ds.pdf*






W PEWNEJ KORPORACJI...
BIG.IT CORP.
CAŁKIEM NIEDALEKO STĄD.
OD CIĘŻKIEJ PRACY, AŻ POWIETRZE ROBIŁO SIĘ GĘSTE...
?
INFORMATYCZNE ORŁY MIAŁY JEDNAK TWARDY ORZECH DO ZGRYZIENIA.
PHI!
ECH...
ACH...
SZEF WSZYSTKICH SZEFÓW W TEJ FIRMIE
MOI PANOWIE, CHCIAŁBYM WAM ZAKOMUNIKOWAĆ, ŻE...
WYJEŻDŻAM NA KILKA DNI!
ALE JAK WRÓCĘ, CHCĘ MIEĆ TEN PROJEKT ZAMKNIĘTY!
EHPRO.PL
1


PO CHWILACH KILKU...
WYSZEDŁ?
WYSZEDŁ, MAMY PRZECHLAPANE...
PANOWIE, WIĘCEJ WIARY! CHILLOUT. MAM POMYSŁ!
DAWAJCIE GŁOŚNIKI!
ZDEJMUJCIE KRAWATY!
WYLUZUJEMY SIĘ PRZY...
DOBRYM JAZZIE!
WTEM!
PANOWIE! EUREKA! WIEM JAK ROZWIĄZAĆ NASZ PROBLEM!
PARIRARARARUM, TAJEMNICĄ ZGRANIA NASZEGO ZESPOŁU JEST TA MUZYKA

Zarządzanie procesami zapewnienia jakości z IBM Rational Quality Manager

IBM® Rational® Quality Manager jest narzędziem wspierającym pracę grupy QA, jako części zespołu uczestniczącego w tworzeniu oprogramowania. Narzędzie to zastępuje w portfelu IBM Rational takie produkty jak IBM Rational Manual Tester, IBM Rational ClearQuest® Test Manager i IBM Rational TestManager.

IBM Rational Quality Manager (RQM) bazuje na platformie Jazz (*http://jazz.net*). Interfejs zbudowany jest w oparciu o paradygmat Web 2.0. RQM został zaprojektowany dla zespołów testowych dowolnej wielkości. Dostęp i uprawnienia użytkowników oparte są na rolach. Oprócz ról ściśle związanych ze strukturą zespołów testowych takich jak *test manager*, *test architect*, *test lead*, *tester* i *test lab manager* można zdefiniować własne role. Uprawnienia dla nich można dostosować do konkretnego projektu.

Rational Quality Manager można zastosować do nieomal każdego projektu, czy to opartego o metodykę zwinną, czy o metodykę bardziej sformalizowaną. Część funkcjonalności/elementów narzędzia jest jednak wykorzystywana dla wszystkich projektów.

Plan testów

Plan testów opisuje zakres prac związanych z testowaniem, a także stanowi zapis procesu testowania. Można go skonfigurować pod kątem specyficznych potrzeb zespołu. Zazwyczaj w planie testów identyfikowane są wymagania, ryzyko, przypadki testowe, środowiska testowe, harmonogramy testowania oraz inne elementy. Powiązane plany testów można grupować razem.

Plan testów RQM składa się z kilku wstępnie zdefiniowanych sekcji. Za pomocą opcji Manage Sections użytkownik może dodawać własne sekcje i usuwać te, które nie są w danym projekcie potrzebne.

Tworzenie nowego planu testów rozpoczyna się od wyboru szablonu. Użytkownik może wybrać jeden z domyślnych szablonów lub utworzyć własny. Szablon planu testów jest zbiorem sekcji planu testów. Tworzenie szablonu odbywa się poprzez dodawanie i usuwanie istniejących sekcji lub tworzenie nowych.

Każda organizacja może zaprojektować własny szablon. Elastyczność tworzenia planów testów powoduje, że plan testów jest odpowiedni zarówno dla zespołów testowych bazujących na metodykach zwinnych (ang. *agile*), formalnych, jak i dla zespołów wykonujących inne rodzaje testów.

Z poziomu planu testów można zarządzać pozostałymi artefaktami procesu testowania i można wykonać m.in. następujące zadania:

- Konfiguracja formalnego procesu przeglądu, który jest widoczny cały czas przez wszystkich członków zespołu projektowego.
- Definiowanie harmonogramów dla każdej iteracji testowania.
- Tworzenie instrukcji testu i powiązanie ich z planem testowania.
- Szacowanie ogólnego nakładu pracy związanego z planowaniem testowania i jego wykonaniem.
- Definiowanie celów biznesowych, celów testowania i celów jakościowych oraz kryteriów wejściowych i wyjściowych.
- Import wymagań projektu z zewnętrznych narzędzi do zarządzania wymaganiami i powiązanie ich z przypadkami testowymi.

Work itemy

Work item to sposób na śledzenie zadań i problemów, którymi zespół musi się zająć. Status i liczba work itemów jest jednym ze wskaźników kontroli projektu. Standardowo produkt Rational Quality Manager zawiera następujące typy elementów *Work item*:

- **Task–Quality** – konkretna praca zazwyczaj związana z artefaktami, takimi jak plan testów, przypadek testowy lub skrypt testowy. Jeśli na przykład użytkownik dostanie zadanie uzupełnienia sekcji planu testów, zostaje on przypisany do elementu zadania Task–Quality.
- **Task–Review** – czynność przypisana użytkownikowi, który został poproszony o przegląd lub zatwierdzenie danego artefaktu, na przykład planu testów lub przypadku testowego.
- **Requirement** – wymaganie utworzone w RQM lub zaimportowane z narzędzia do zarządzania wymaganiami.
- **Defect** – element wykorzystywany do śledzenia defektów w testowanym oprogramowaniu.
- **Task** – pozostałe zadania.

Zarządzanie wymaganiami

Zarządzanie wymaganiami zapewnia możliwość śledzenia cyklu życia, pozwalając powiązać przypadki testowe w planie testów z wymaganiami.

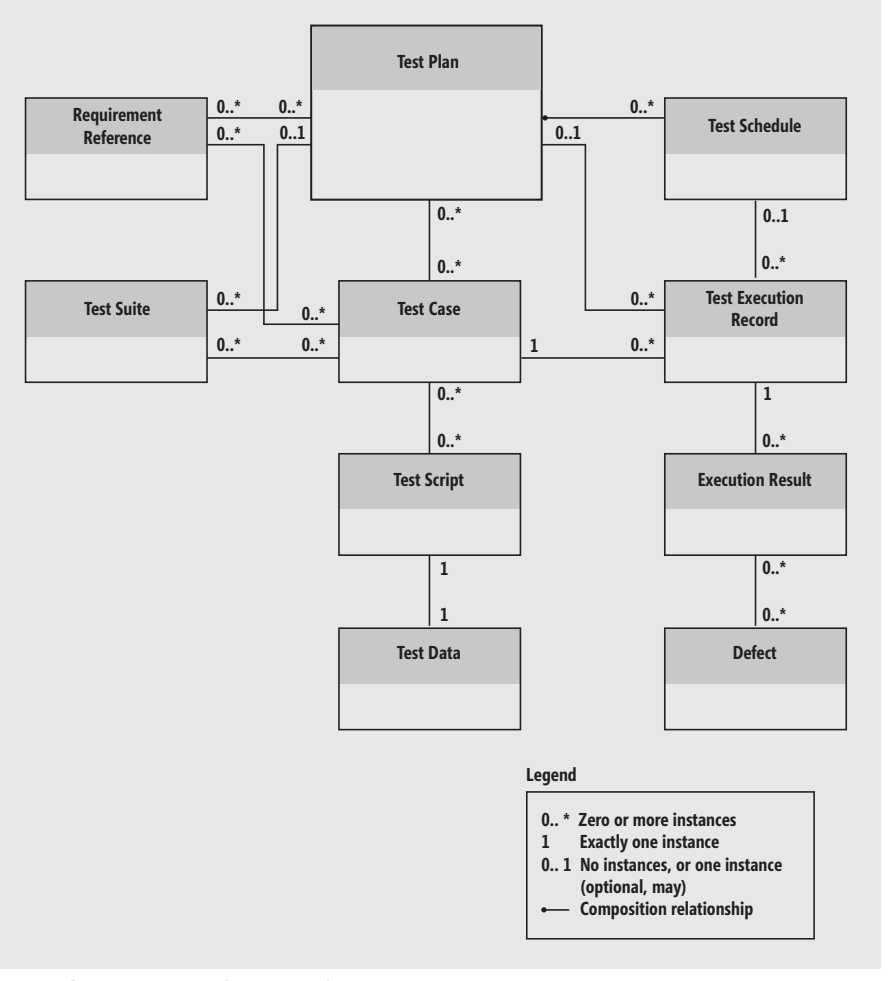
IBM Rational Quality Manager udostępnia kilka możliwości związanych z zarządzaniem wymaganiami:

- Za pomocą produktu IBM Rational Requirements Composer można powiązać przypadki testowe i plany testów z wymaganiami.
- Wymagania można importować z zewnętrznych narzędzi, takich jak IBM Rational RequisitePro i IBM Rational DOORS.
- Wymagania można tworzyć i zarządzać nimi bezpośrednio w RQM.

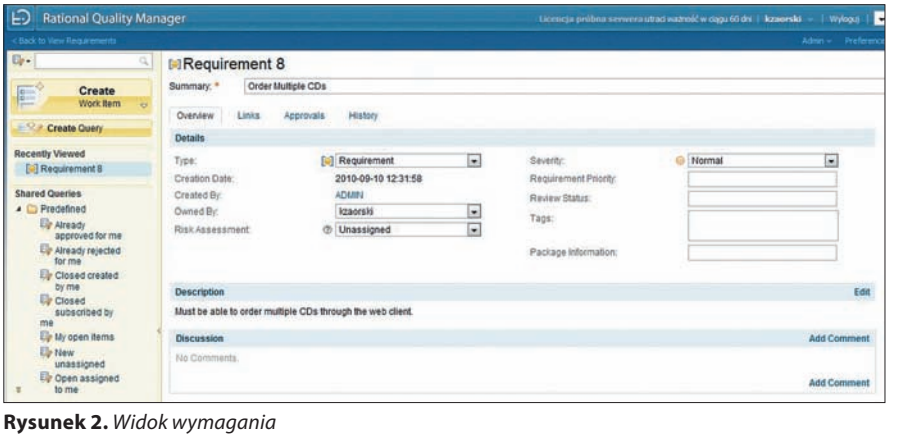
Wymaganie w RQM jest zaimplementowane jako element *Work Item*. W związku z tym z poziomu sekcji dokumentu wymagań Description lub Discussion można w prosty sposób tworzyć powiązania z innymi wymaganiami, defektami lub innymi elementami *Work Item*.

Tworzenie wymagań

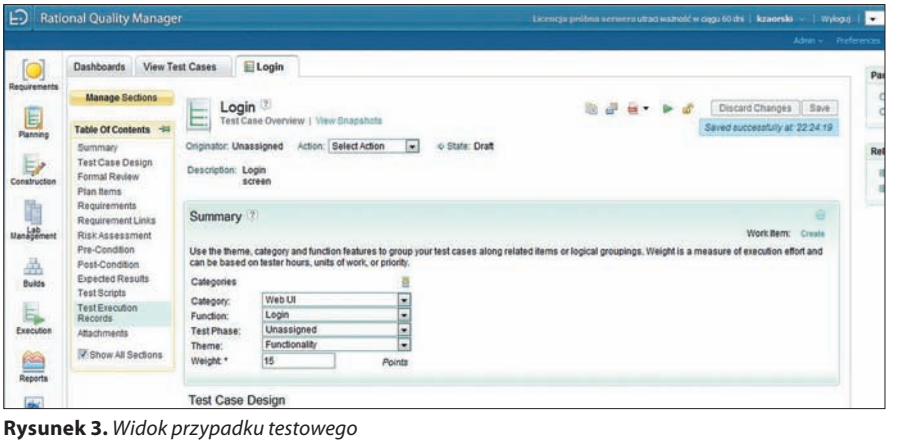
Do tworzenia nowych wymagań można użyć Rational Quality Managera lub stworzyć je za pomocą zewnętrznej aplikacji do zarządzania wymaganiami, takiej jak IBM Rational RequisitePro lub Rational DOORS, i zaimportować je do RQM. W przypadku zmiany



Rysunek 1. *Powiązania elementów planu testów*



Rysunek 2. *Widok wymagania*



Rysunek 3. *Widok przypadku testowego*

lub usunięcia wymagania w aplikacji do zarządzania wymaganiami status danego wymagania jest aktualizowany w RQM.

Istnieje też możliwość tworzenia wymagań i zarządzania nimi za pomocą produktu IBM Rational Requirements Composer oraz tworzenia powiązań do nich z RQM.

Przypadki testowe można generować dla jednego, kilku lub wszystkich dostępnych wymagań. Można wygenerować jeden przypadek testowy dla każdego wymagania, co spowoduje utworzenie wielu przypadków testowych, ale można także wygenerować jeden przypadek testowy pokrywający wiele wymagań.

Przypadki i scenariusze testowe

Predefiniowany przypadek testowy (*Test Case*) w IBM Rational Quality Manager składa się z kilku wstępnie zdefiniowanych sekcji.

Niektóre sekcje, takie jak Test Case Design, Pre–Condition, Post–Condition i Expected Results, posiadają edytory tekstu zawierające standardowe opcje formatowania, takie jak czcionki, tabele czy listy wypunktowane i numerowane. Pozostałe sekcje, takie jak Requirements, Test Scripts i Test Execution Records, zawierają powiązania (linki) do innych artefaktów testowych.

Za pomocą opcji Manage Sections użytkownik może dodawać własne sekcje oraz usuwać te, które nie są w danym projekcie wykorzystywane.

Tworzenie nowego przypadku testowego rozpoczyna się od wyboru szablonu. Użytkownik może wybrać jeden z domyślnych szablonów lub utworzyć własny. Może także wskazać szablony, które mają być traktowane jako domyślne dla nowych przypadków testowych.

Szablon przypadku testowego jest zbiorem sekcji przypadku testowego. Tworzenie szablonu odbywa się poprzez dodawanie i usuwanie sekcji lub tworzenie nowych. Jeśli nazwy sekcji nie są zgodne z przyjętym w organizacji standardem nazewnictwa, można utworzyć nowe sekcje i dodać je do szablonu. Każda organizacja może zaprojektować własne niestandardowe szablony.

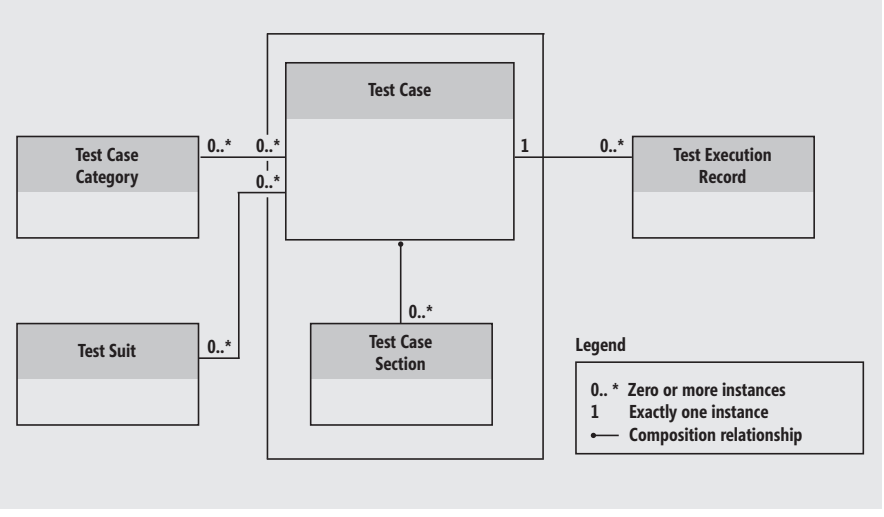
Każdy przypadek testowy zazwyczaj jest powiązany ze skryptem testowym, chociaż możliwe jest uruchamianie testu bez dowiązanego skryptu testowego.

Skrypt testowy (*Test Script*) to ręczny lub zautomatyzowany skrypt, który zawiera instrukcje dotyczące wykonania danego przypadku testowego. Użytkownik może utworzyć skrypty testów ręcznych, które będą wykonywane manualnie, może też dowiązać zautomatyzowane skrypty testów funkcjonalnych, wydajnościowych i bezpieczeństwa.

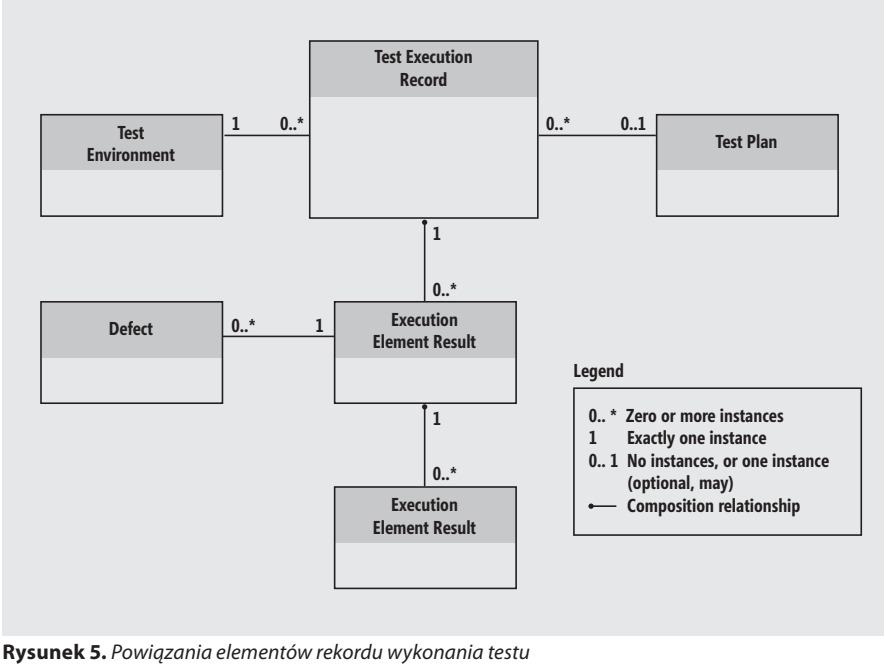
Sekcja warunków wstępnych dla przypadku testowego pozwala wskazać wymagania, które muszą być spełnione przed uruchomieniem przypadku testowego. Warunki końcowe służą przedstawieniu wymagań, które muszą być spełnione po wykonaniu przypadku testowego.

Aby przeprowadzić test, należy wykonać jedno z następujących zadań:

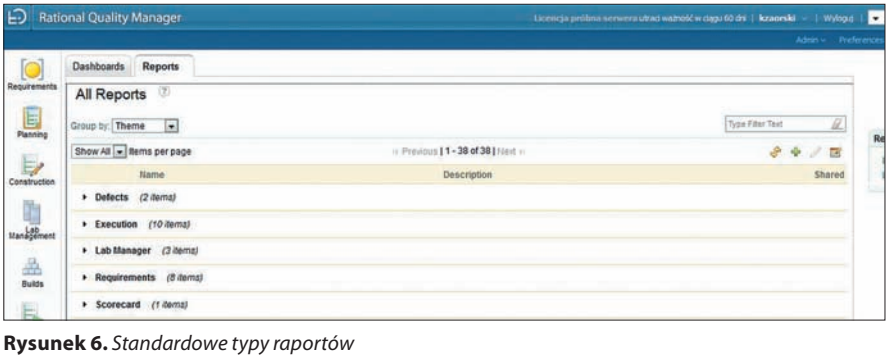
- uruchomić wykonanie przypadku testowego,
- wygenerować zapisy wykonania testu (*Test Execution Records*) dla przypadku testowego i uruchomić wykonanie ich (zapisy wykonania testu opisują konfiguracje dla danego uruchomienia testu),
- połączyć kilka przypadków testowych w scenariusz testowy (Test Suite) i uruchomić go.



Rysunek 4. *Powiązania elementów przypadku testowego*



Rysunek 5. *Powiązania elementów rekordu wykonania testu*



Rysunek 6. *Standardowe typy raportów*

Wykonywanie testów

IBM Rational Quality Manager udostępnia kilka wariantów wykonywania testów, co umożliwia zaadaptowanie tego narzędzia do wielu rodzajów zespołów testowych.

W RQM podstawowym komponentem jest przypadek testowy. Najprostszą realizacją wykonania testu jest utworzenie przypadku testowego i wykonanie go.

Inne artefakty testowe, takie jak plany testów, skrypty testowe, środowiska testowe oraz zapisy wykonania testu, są opcjonalne. Użytkownik może stworzyć takie powiązania i wykorzystać pełny zakres funkcjonalności RQM, jednak nie jest to niezbędne.

Scenariusz testowy (*Test Suite*) jest to zbiór przypadków testowych, które zostały zgrupowane w celu ich wykonania. Scenariusze testowe mogą obejmować testy ręczne i zautomatyzowane. W trakcie uruchamiania scenariusza można określić, czy przypadki testowe mają być wykonywane sekwencyjnie czy równolegle.

Zapisy wykonania testu przypisują daną konfigurację środowiska testowego do przypadku testowego. Określają środowisko sprzętowe i programowe dla danego wykonania. Wykonanie testu jest definiowane w następujący sposób: wykonaj przypadek testowy w tym środowisku, na danej platformie sprzętowej, działającej pod kontrolą tego systemu operacyjnego, wykorzystując daną przeglądarkę. Aby upewnić się, że przypadek testowy uruchamiany w czterech różnych środowiskach testowych zostanie poprawnie wykonany, należy utworzyć zapis wykonania testu dla każdego wymaganego środowiska testowego.

Za pomocą produktu Rational Quality Manager można uruchamiać testy manualne i zautomatyzowane takie jak:

- Testy manualne, które są tworzone za pomocą RQM,
- Testy z obsługą słów kluczowych, które zostały utworzone w RQM,
- Testy manualne zmigrowane z produktu IBM Rational Manual Tester,
- Testy zautomatyzowane utworzone za pomocą takich narzędzi jak:
  - IBM Rational Functional Tester,
  - IBM Rational Performance Tester,
  - IBM Rational AppScan,
  - HP QuickTest Professional,
  - HP LoadRunner.

Podczas tworzenia testów zautomatyzowanych tworzone są połączenia do skryptów testowych, które są stworzone za pomocą narzędzia testów zautomatyzowanych. Skrypty te mogą znajdować się na maszynie testowej lub na współdzielonym zasobie sieciowym. W trakcie uruchamiania testu RQM kopiuje test na maszynę testową i uruchamia go.

Śledzenie defektów

Zarządzanie defektami jest możliwe za pomocą samego RQM, ale można do tego celu wykorzystać także inne narzędzia, takie jak IBM Rational Team Concert lub IBM Rational ClearQuest.

Jeśli testowana aplikacja jest tworzona z wykorzystaniem Rational Team Concert, defektami można zarządzać, korzystając z elementu *Defect* w Rational Team Concert. Reprezentację defektu w produkcie Rational Team Concert można zobaczyć przy każdym zgłoszeniu defektu, co pozwala na lepszą komunikację z programistami odpowiedzialnymi za poprawianie defektów.

Jeśli do śledzenia defektów zespół używa produktu Rational ClearQuest, można wykorzystać work item Defect z Rational Quality Manager'a i użyć narzędzia Rational ClearQuest Connector w celu zsynchronizowania lokalnego repozytorium defektów z Rational ClearQuest.

Raportowanie

W Rational Quality Manager istnieje zbiór predefiniowanych raportów, których można użyć do analizy wykonywanych testów i zidentyfikowania trendów w projekcie.

W każdym raporcie znajduje się sekcja Parameters, w której wymienione są kryteria, których można użyć do wygenerowania raportu. W przypadku wielokrotnego korzystania z takich samych raportów i ich parametrów można zrobić kopie raportu i zapisać parametry, aby mieć do nich łatwy dostęp. Wiele raportów ma charakter typu drill–down, co umożliwia docieranie do szczegółowych informacji wymienianych w raporcie.

Można również zainstalować narzędzie Rational Common Reporting, które umożliwia tworzenie niestandardowych raportów z wykorzystaniem technologii raportowania Cognos. Można tworzyć nowe własne raporty lub też kopiować i modyfikować zestaw wstępnie zdefiniowanych raportów, które są częścią Rational Common Reporting.

Rational Common Reporting jest wyposażony w łatwy w użyciu interfejs użytkownika służący do tworzenia i przeglądania niestandardowych raportów. Następnie raporty należy zaimportować do Rational Quality Manager, gdzie można je uruchamiać i przeglądać tak jak standardowe raporty RQM.

Rational Quality Manager ułatwia udostępnianie informacji do współużytkowania z innymi członkami zespołu. W systemie elementów *Work Item* opartym na platformie Jazz członkowie zespołu mogą wzajemnie przypisywać sobie zadania i defekty oraz wyświetlać status każdego z członków zespołu. Autorzy planów testów i projektanci przypadków testowych mogą przekazywać swoje prace do przeglądu i śledzić status każdego recenzenta. Można zobaczyć, kto jest zalogowany i nad czym pracuje. Członkowie zespołu mogą być automatycznie powiadamiani o zmianach, danych wejściowych i kamieniach milowych, które mają wpływ na ich pracę.

Krzysztof Zaorski

Pracuje w firmie Premium Technology, gdzie odpowiedzialny jest za obszar zapewnienia jakości oprogramowania, testy wydajnościowe i automatyzację testów funkcjonalnych. Jego znajomość narzędzi IBM w tej dziedzinie potwierdzona jest certyfikatami: IBM Certified Solution Designer – IBM Rational Unified Process V7.0, IBM Certified Solution Designer – Rational Functional Tester for Java, IBM Certified Solution Designer – Rational Performance Tester, IBM Certified Specialist – Rational Test Management and Robot v2003.

IBM Rational AppScan Standard Edition

Robot w białym kapeluszu

IBM Rational AppScan Standard Edition jest czołowym narzędziem do automatyzacji oceny bezpieczeństwa aplikacji webowych. Zaawansowane rozwiązanie typu WASS (ang. *Web Application Security Scanner* – skaner bezpieczeństwa aplikacji webowych) pozwala znacznie zredukować koszty związane z manualnymi testami bezpieczeństwa, ułatwiając jednocześnie zapewnienie odpowiedniego poziomu ochrony aplikacji przed ofensywnymi działaniami cyberprzestępców.

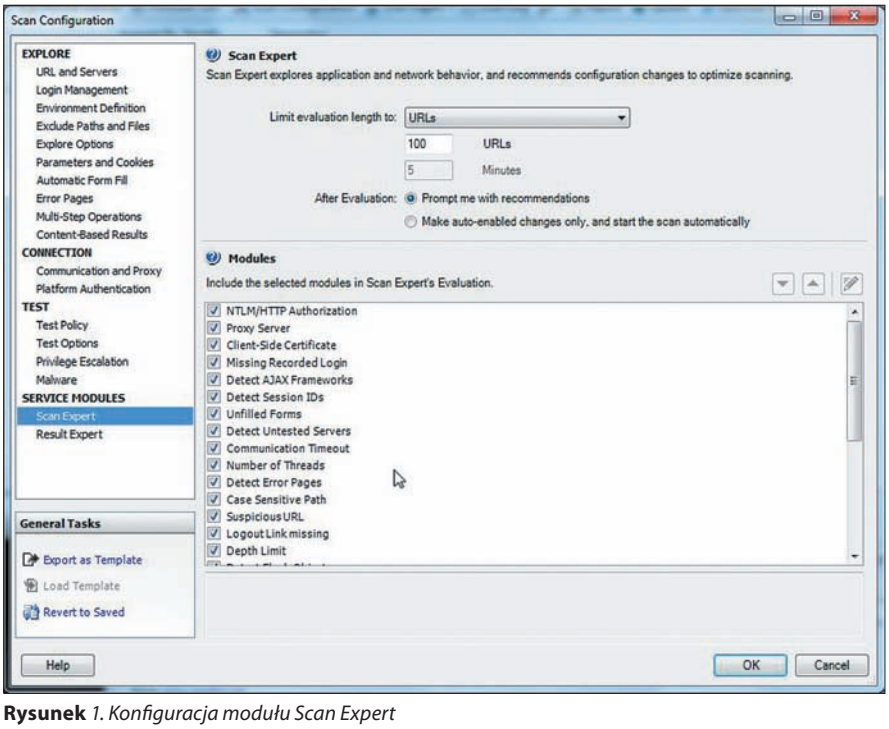
Program umożliwia automatyczną realizację czarnoskrzynkowych testów podatności zabezpieczeń, dostarcza rekomendacji naprawczych i oferuje bogatą funkcjonalność raportową, w tym także raporty zgodnościowe. Artykuł przedstawia podstawowe informacje o tym rozbudowanym oprogramowaniu.

IBM Rational AppScan Standard Edition (AppScan, AppScan Standard, ASSE) jest narzędziem automatyzującym ocenę bezpieczeństwa aplikacji webowych i ich szczególnej formy, jaką są usługi sieciowe (ang. *Web Services*). Poprzez ocenę bezpieczeństwa należy w tym wypadku rozumieć pewien zestaw działań, na który składają się: testy podatności zabezpieczeń, określenie i opisanie ujawnionych podatności, wytworzenie rekomendacji naprawczych i raportowanie. We wszystkich tych zakresach AppScan Standard oferuje bogatą funkcjonalność.

Badanie podatności zabezpieczeń wykonywane przez ASSE ma charakter testów czarnoskrzynkowych (ang. *blackbox*), w przypadku aplikacji webowych polegają one na wysyłaniu odpowiednio zmodyfikowanych żądań HTTP i analizie odpowiedzi. Są to więc typowe testy penetracyjne. Wykonywane w oparciu o stale aktualizowaną, ogromną (tysiące testów) bazę technik wykorzystania luk bezpieczeństwa. To, które testy zostaną wykonane w danym badaniu, wynika z konfiguracji narzędzia i zastosowania zaawansowanych zaimplementowanych w ASSE mechanizmów testowania adaptacyjnego.

AppScan wykonuje testy pod kątem najważniejszych zagrożeń i podatności aplikacji webowych, odnotowywanych w kluczowych rejestrach takich jak WASC Threat Classification, CVE czy zestawieniach takich jak OWASP Top 10. Zawiera także potężną bazę ataków specyficznych dla danych technologii i produktów. Narzędzie w pełni wspiera badanie ośrodków webowych zrealizowanych według założeń i postulatów Web 2.0, silnie wykorzystujących takie technologie jak AJAX. Wspiera także testowanie aplikacji typu „Mega Script" (jedna strona główna, która zmienia swoją strukturę i zawartość na podstawie kombinacji parametrów nawigacyjnych), aplikacje wykorzystujące zakodowane URL-e i portale wykorzystujące w interfejsie widżety.

Cechą wyróżniającą AppScana jest zdolność do testowania aplikacji czy komponentów aplikacji zbudowanych w technologii Adobe



Rysunek *1. Konfiguracja modułu Scan Expert*

Flash, w tym możliwość parsowania i wykonywania Flasha oraz parsowania i analizy komunikacji Flash AMF (komunikaty *Action Message Format*). Umożliwia to wykrywanie takich podatności flaszowych jak Cross-Site Flashing, Cross-Site Scripting czy Insecure Direct Object Reference!

Inną ciekawą cechą skanera jest możliwość budowy mapy badanego ośrodka na podstawie zawartości, a nie adresów URL. W odniesieniu do aplikacji zbudowanych zgodnie z paradygmatem MVC pozwala to prowadzić miarodajne testy i zwiększyć czytelność drzewa ośrodka dla operatora.

Testy wykonywane przez ASSE obejmują swoim zakresem również elementy infrastruktury takie jak serwer webowy i jego konfiguracja czy komponenty dostawców zewnętrznych.

Zapis każdej luki bezpieczeństwa znajdującej się w bazie ASSE zawiera dokładny opis doradczy, wraz z odpowiednimi informacjami klasyfikacyjnymi i referencyjnymi oraz opisem ryzyka. Wykorzystywane są uznane w gospodarce rejestry błędów takie jak CVE, zaimplementowane jest także określanie wag błędów według skoringu CVSS – pozwala to na dokładne określenie, analizę ujawnionej podatności i ocenę stwarzanego przez nią ryzyka. Podobnie – do każdego problemu dołączony jest szczegółowy dokument rekomendacji naprawczej, składający się z dwóch części: ogólnej i zawierającej wskazówki dla konkretnych języków i platform programistycznych – wspierane są ASP.NET, J2EE i PHP.

Aktualizacja aplikacji odbywa się w sposób podobny do tego znanego z oprogramowania antywirusowego, w tym jednak przypadku każdorazowo wymagana jest akceptacja użytkownika. Moduł aktualizacyjny obsługuje zarówno ładowanie nowych reguł bezpieczeństwa, jak i zmiany w oprogramowaniu.

Silnik raportowy AppScana Standard oferuje szeroki zestaw predefiniowanych raportów kierowanych do różnych odbiorców – są to typowe raporty bezpieczeństwa, raporty zgodności z uregulowaniami prawnymi i standardami gospodarczymi, raport porównania wyników badania i raporty użytkownika budowane na podstawie szablonów tworzonych w procesorze tekstu Microsoft Office Word (MS Word).

AppScan Standard Edition jest aplikacją desktopową przeznaczoną dla systemu Microsoft Windows. Podczas typowej pracy operator wykorzystuje interfejs graficzny, który jest podstawowym środowiskiem pracy z aplikacją. ASSE może być jednak używany w trybie wsadowym.

Narzędzie kierowane jest do wszystkich uczestników procesu zapewnienia bezpieczeństwa aplikacji, zarówno tych zlokalizowanych w organizacyjnych jednostkach odpowiedzialnych za bezpieczeństwo teleinformatyczne, jak i bezpośrednich uczestników procesu wytwórczego funkcjonujących po stronie deweloppmentu.

Odbiorcami raportów z AppScana są osoby bardzo różnych ról związanych z IT – od projektantów, programistów, testerów, przez menedżerów projektów, pracowników bezpieczeństwa, osoby zajmujące się architekturą korporacyjną, audytorów wewnętrznych, oficerów zgodności, aż po zarząd.

Praca z aplikacją

Rozpoczęcie pracy z ASSE wiąże się z utworzeniem nowego skanu – jest to podstawowy zbiór danych, na którym operuje narzędzie. Zawiera on wszystkie informacje związane z przeprowadzanym badaniem: pełną konfigurację oraz wyniki. Skan może być zapisany na każdym etapie pracy, niezależnie od zakresu wprowadzonych danych. Może to być zatem zapis samej konfiguracji narzędzia do komunikacji z aplikacją, a może to być kompletny zapis badania włącznie z wynikami i zapisami zawartości badanego ośrodka (z etapu eksploracji). Konfiguracja skanowania obejmuje następujące elementy:

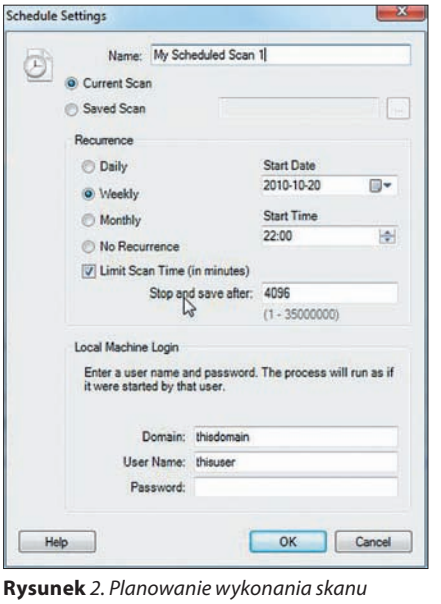
- Konfiguracja etapu eksploracji.
- Konfiguracja połączenia.
- Konfiguracja etapu wykonania testów.
- Konfiguracja modułów wspomagających (ang. *Service Modules*).

Konfiguracja skanu może być zapisana w postaci szablonu, który można później wykorzystywać. Podobnie część elementów składających się na tę konfigurację może zostać zapisana w zewnętrznych plikach do użycia w innych konfiguracjach.

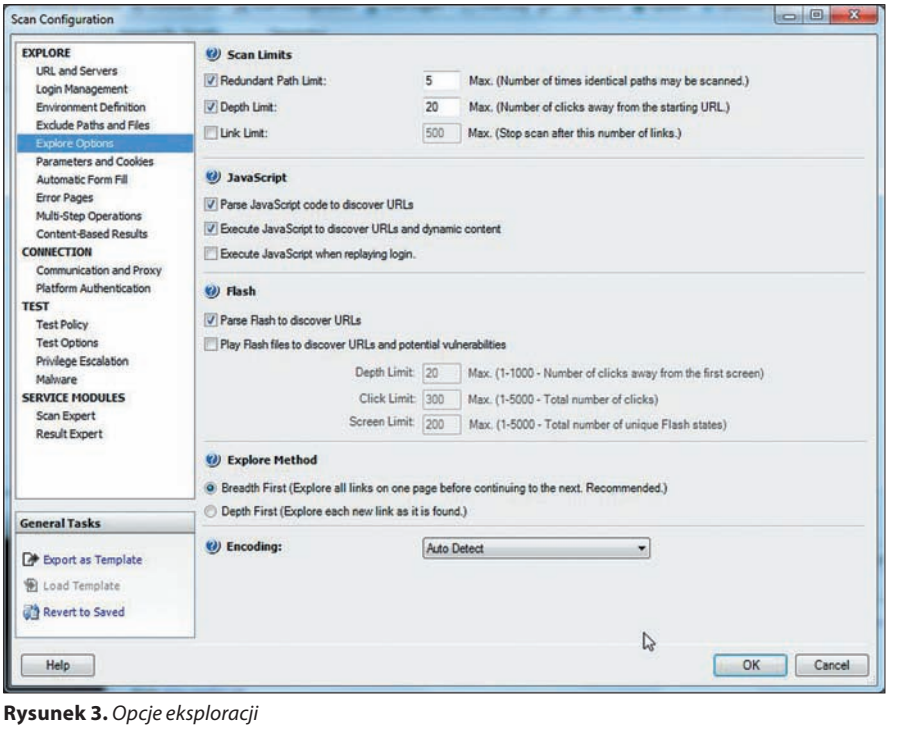
Typowy cykl toku pracy z ASSE przewiduje dalej rozpoczęcie wykonania badania, obejmujące etap eksploracji testowanej aplikacji i wielofazowy etap wykonania testów. Etapy te mogą być wykonywane rozdzielnie, w każdym momencie istnieje też możliwość zatrzymania trwającego skanowania i zapisania jego aktualnego stanu z możliwością późniejszego wznowienia.

Odmiennie wygląda proces skanowania usług sieciowych (*Web Services*) – w takim przypadku proces eksploracji odbywa się z wykorzystaniem wyspecjalizowanego klienta (*Generic Services Client*).

Często proces badania aplikacji można rozpocząć, wykonując minimalną liczbę czynności



Rysunek *2. Planowanie wykonania skanu*



Rysunek 3. *Opcje eksploracji*

konfiguracyjnych i korzystając z wbudowanego kreatora, w praktyce jeszcze częściej wymagana jest konfiguracja przynajmniej pewnej ilości ustawień przez operatora.

Pierwszą rzeczą, którą trzeba skonfigurować, jest zestaw ustawień dotyczących połączenia z aplikacją – obejmuje to ustawienia komunikacyjne (timeouty, liczba wątków i proxy) oraz ustawienia uwierzytelniania na poziomie serwera. W tym zakresie wspierane jest uwierzytelnianie HTTP i NTLM oraz uwierzytelnianie za pomocą certyfikatu.

Na wstępie warto także skonfigurować moduły wspomagające – Scan Expert i Result Expert. Pierwszy z nich na podstawie zachowania sieci i aplikacji wytwarza rekomendacje co do zmian optymalizacyjnych w konfiguracji skanu, drugi zaś oferuje zestaw modułów przetwarzających wyniki skanowania i opracowujących dodatkowe informacje dostępne w panelu szczegółowej informacji o wynikach.

AppScan umożliwia także wykonywanie skanu w formie zadania zaplanowanego, można także wykorzystać do tego celu interfejs linii komend (CLI). W formule zadania zaplanowanego możliwe jest ograniczenie czasu trwania wykonania badania.

Ustawienia etapu eksploracji

W etapie eksploracji budowane jest drzewo aplikacji – hierarchia adresów URL, plików i innych komponentów, a także wykonywana jest analiza danych i na jej podstawie tworzony jest zestaw testów do wykonania. W większości sytuacji eksploracja wykonywana jest w pełni automatycznie zgodnie ze zdefiniowaną konfiguracją. Etap eksploracji może być także wykonywany manualnie przez operatora z użyciem przeglądarki wbudowanej w narzędzie (IE) albo korzystając z przeglądarki zewnętrznej. Użycie przeglądarki zewnętrznej wiąże się z wykorzystaniem dostępnej w ASSE funkcjonalności serwera pośredniczącego HTTP (proxy). Możliwość użycia przeglądarki zewnętrznej jest szczególnie istotna w kontekście istniejących problemów z kompatybilnością pomiędzy różnymi ośrodkami webowymi a przeglądarkami. Dane pozyskane w trakcie fazy eksploracji dostępne są w widoku danych aplikacji (*Application Data*). Można je także eksportować i importować z pliku zewnętrznego.

W przypadku eksploracji automatycznej należy poprawnie skonfigurować szereg ustawień narzędzia, których najważniejsze obszary to:

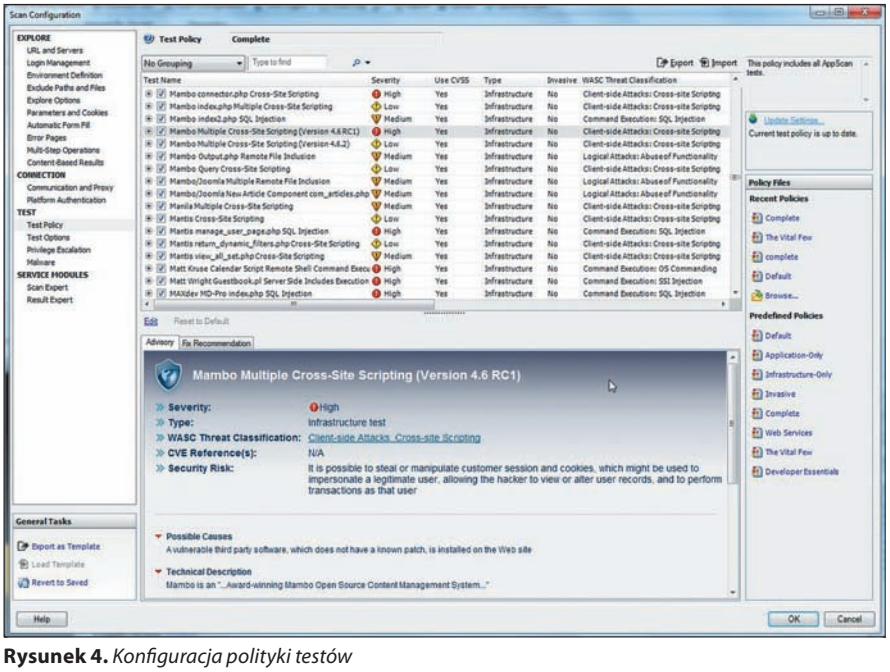
- Adres URL badanej aplikacji i serwery podlegające skanowaniu.
- Ustawienia logowania i sprawdzania utrzymania sesji.
- Wyłączenia ścieżek i plików.
- Opcje eksploracji.
- Parametry i zmienne cookie.
- Automatyczne wypełnianie formularzy.
- Operacje wielokrokowe.

W odniesieniu do ustawień logowania, najczęściej stosowanym rozwiązaniem jest nagranie tego procesu. Można także zastosować logowanie automatyczne (z wykorzystaniem automatycznego wypełniania rozpoznanych pól formularza) albo logowanie manualne. O ile mamy do czynienia z aplikacją wymagającą zalogowania się użytkownika, poprawne ustawienia związane z logowaniem są kluczowe dla całego procesu badania.

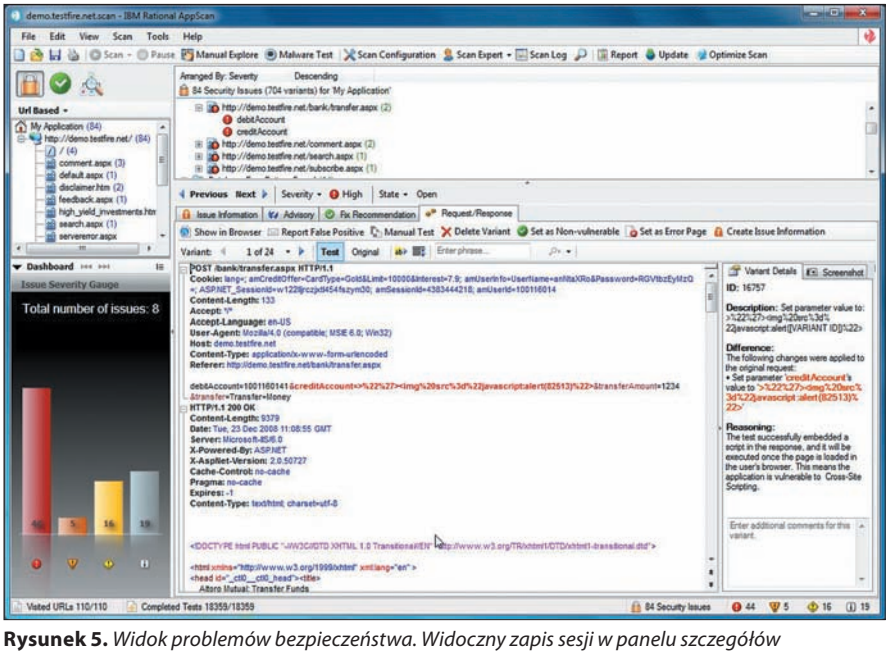
Ustawienia eksploracji decydują m.in. o takich elementach jak głębokość crawlingu, sposób eksploracji kodu JavaScript (parsowanie, wykonywanie), sposób eksploracji elementów Flash (parsowanie, wykonywanie) czy sposób poruszania się po ośrodku (podążanie za linkiem, eksploracja wszystkich odnośników na stronie). Ustawienia te mają walny wpływ na szybkość i uzyskane pokrycie, jak również na powodzenie całego badania (błędny dobór ustawień może np. doprowadzić do niekończącego się skanu).

Nie mniej ważna jest poprawna konfiguracja obsługi parametrów i zmiennych cookie, szczególnie w odniesieniu do utrzymania sesji czy unikania testów nadmiarowych.

Operacje wielokrokowe pozwalają na uzyskanie dostępu do takich części aplikacji, które dostępne są tylko w przypadku zachowania określonego porządku wysyłanych żądań.



Rysunek 4. *Konfiguracja polityki testów*



Rysunek 5. *Widok problemów bezpieczeństwa. Widoczny zapis sesji w panelu szczegółów*

Funkcjonalność ta pozwala na nagrywanie takich sekwencji.

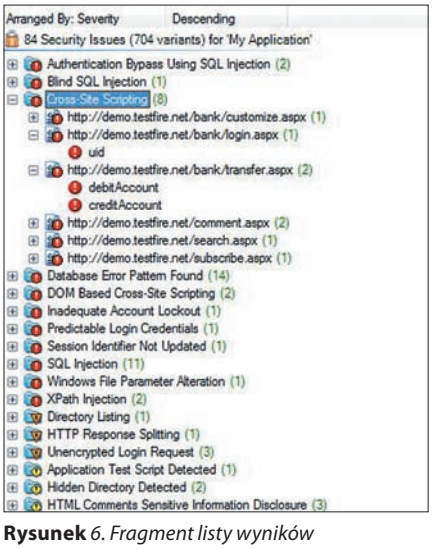
Niektóre ustawienia, takie jak pule danych do automatycznego wypełniania formularzy, mogą być eksportowane i importowane z plików zewnętrznych.

W wielu elementach konfiguracji możliwe jest stosowanie wyrażeń regularnych (np. w ustawieniach parametrów i zmiennych *cookie*).

Warto także poświęcić uwagę ustawieniom związanym z definicją środowiska pracy aplikacji– odpowiednio wypełnione dane o infrastrukturze pozwalają zoptymalizować proces testowania. Ustawienia te obejmują również informacje używane przy określaniu skoringu CVSS, takie jak wymagania odnośnie dostępności czy lokalizacja ośrodka.

Ustawienia etapu wykonania testów

Etap ten polega na wykonywaniu wytworzonych testów. Działanie to może być wielofazowe (zależnie od konfiguracji) – w takim przypadku, jeśli podczas wykonywania testu zostanie wykryta (w odpowiedzi aplikacji) nowa, nie ujawniona w początkowej fazie eksploracji zawartość, to dla tej zawartości zostaną powtórzone etapy eksploracji i testów. Ilość takich faz jest konfigurowalna do maksymalnie dziesięciu. Widać tutaj, że etapy eksploracji i wykonania testów mogą się zazębiać.



Rysunek *6. Fragment listy wyników*

Najważniejszym ustawieniem dla etapu wykonania testów jest polityka testowa, czyli zestaw testów, które mogą być wykonane w danym badaniu. Wykorzystywana polityka może być edytowana przez użytkownika z dokładnością do wariantu testowego dla danej luki bezpieczeństwa. Polityki testowe można zapisywać i importować z plików zewnętrznych.

Kolejnym ustawieniem są opcje testowania, które m.in. określają ilość faz dla procesu wykonywania testów, odpowiadają za włączenie bądź wyłączenie mechanizmu testowania adaptacyjnego, decydują o tym, czy mają być wykonywane testy na stronach logowania i wylogowywania.

Także w ustawieniach testów mamy możliwość skorzystania z funkcjonalności testów eskalacji przywilejów – możemy tutaj wskazać inne skany, wykonywane z użyciem konta użytkownika określonej roli, które zostaną następnie porównane w celu ustalenia stopnia, do jakiego uprzywilejowane zasoby są dostępne dla użytkowników z niewystarczającymi uprawnieniami dostępowymi. Funkcjonalność ta pozwala na testowanie błędów kontroli dostępu.

W tej części konfiguracji mamy także dostęp do ustawień funkcjonalności związanej z wykrywaniem szkodliwej zawartości i odnośnikach do niebezpiecznych ośrodków, zawartych na stronach badanej aplikacji. Ustawienia te dotyczą modułu testów na szkodliwą zawartość, który jest uruchamiany samodzielnie i nie wpływa na przebieg procesu eksploracji i skanowania.

Analiza wyników

Dane wynikowe prezentowane są w ASSE w formule trzech widoków:

- Problemów bezpieczeństwa (*Security Issues*)
- Zadań naprawczych (*Remediation Tasks*)
- Danych aplikacji (*Application Data*)

W każdym z widoków dane (opisowe, aplikacji) przypisane są do adresów URL widocznych w lewej górnej ramce okna głównego (drzewo aplikacji). Drzewo możemy wykorzystać do nawigacji i filtrowania widoku wyników. Lista rezultatów zlokalizowana jest w prawej górnej ramce i przedstawia zależnie od wybranego widoku: problemy bezpieczeństwa, zadania naprawcze albo dane aplikacji. Dalsze informacje zostaną przedstawione na przykładzie widoku problemów bezpieczeństwa, który należy traktować jako widok podstawowy.

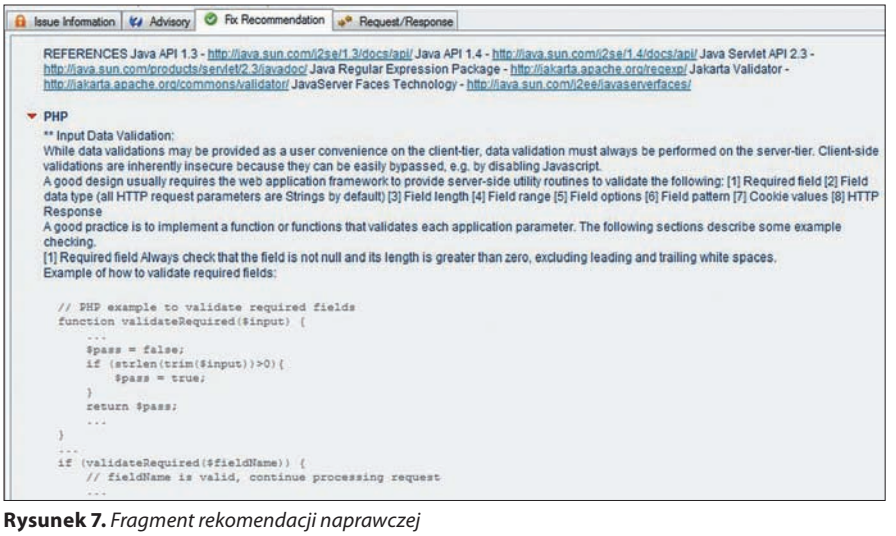
Wewnętrzna klasyfikacja AppScana Standard przewiduje cztery wagi ujawnionych podatności (od najwyższej):

- Wysoka
- Średnia
- Niska
- Informacyjna

Widok problemów bezpieczeństwa służy do prezentacji listy wyników i informacji szczegółowych wyświetlanych w panelu szczegółów (*Detail Panel*) – panel ten zlokalizowany jest w prawej dolnej ramce, pod listą wyników.

Zawiera on następujące zakładki:

- Informacja o problemie – podatności zabezpieczeń (*Issue information*)
- Opis doradczy (*Advisory*)
- Rekomendacja naprawcza (*Fix Recommendation*)
- Przebieg komunikacji testowej (*Request/Response*)

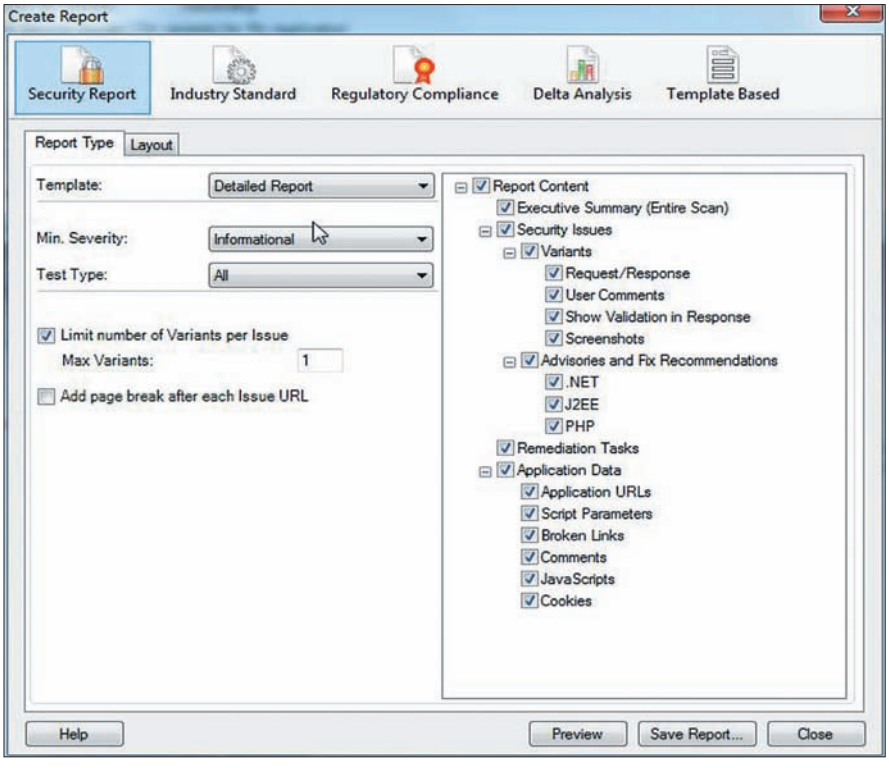


Rysunek 7. *Fragment rekomendacji naprawczej*

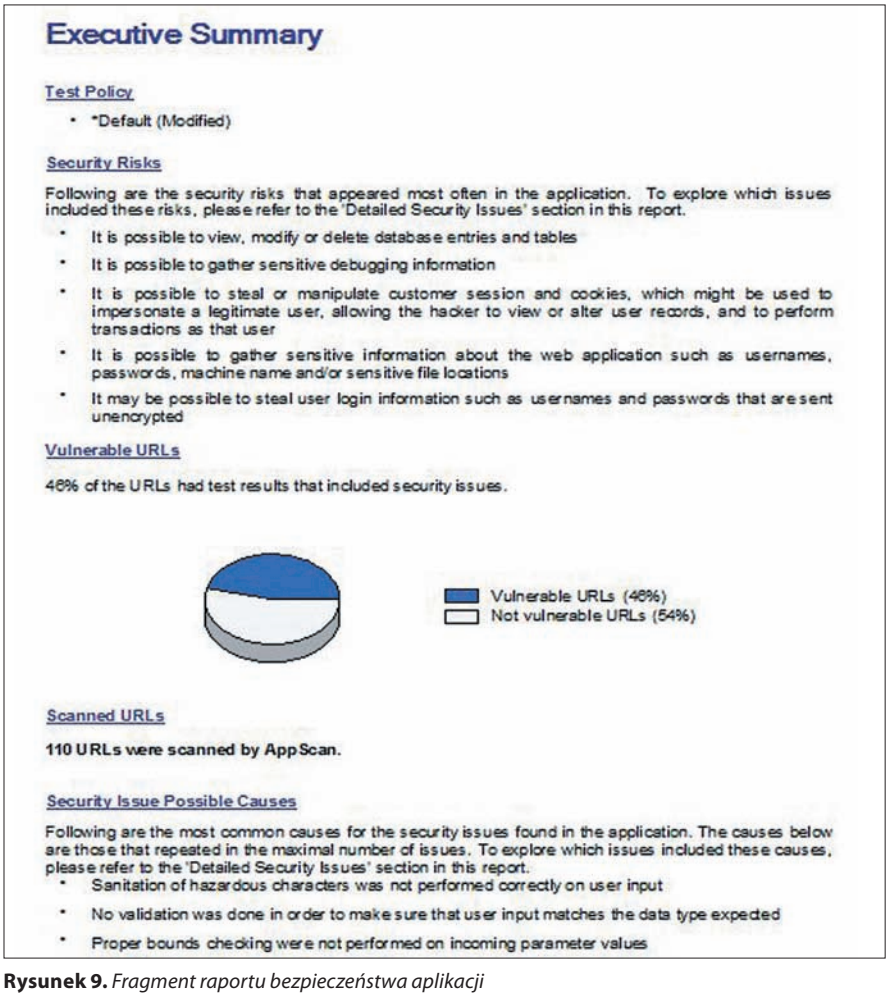
Dokument informacji o problemie zawiera krótką przeglądową informację o wykrytym problemie oraz dodatkowe informacje dostarczone przez moduł Scan Expert. W szczególności przedstawia wagę błędu według skoringu CVSS, może także zawierać odpowiedni zrzut ekranu. Dane te są przechowywane w zapisie skanu i mogą być wyprowadzone do raportu MS Word.

Opis doradczy zawiera szczegółowe określenie i analizę danej podatności, a także m.in. klasyfikację według wag stosowanych w ASSE, numer podatności w katalogu CVE, odnośnik do opisu zagrożenia według klasyfikacji WASC (*Web Application Security Consortium Threat Classification*), informacje o podatnych produktach. Część dokumentów typu opis doradczy zawiera również krótkie prezentacje warsztatowe wykonane w technologii Adobe Flash wyjaśniające i prezentujące dany typ problemu.

Rekomendacja naprawcza przedstawia szczegółowo znane rozwiązania programistyczne danego typu problemu – prezentowane są wskazówki generyczne i specyficzne dla platform ASP.NET, J2EE i PHP. Przebieg komunikacji testowej prezentuje szczegółowy zapis poszczególnych testowych sesji typu żądanie/odpowiedź – sesji oryginalnej (zgodnej z logiką biznesową aplikacji) i zawierających warianty wykonywanego ataku. Dane przedstawiane są w sposób przypominający edytory programistyczne. Dzięki temu łatwo można prześledzić przyczyny, dla których wynik testu został uznany za pozytywny – ta informacja jest także wyświetlana w formie opisowej. Taka formuła pozwala na wygodną analizę przebiegu testu i jego weryfikację. Adekwatne działania, takie jak oznaczenie danego wariantu jako niepodatny, także podejmuje się w tej części interfejsu.



Rysunek 8. *Tworzenie raportu bezpieczeństwa aplikacji*



Rysunek 9. *Fragment raportu bezpieczeństwa aplikacji*

Dokumenty prezentowane w zakładkach panelu szczegółów można zapisywać do plików zewnętrznych, w szczególności plików HTML/MHT.

Raportowanie

Raporty dostępne w aplikacji można podzielić na cztery główne typy: raporty bezpieczeństwa, raporty zgodności, raporty porównania wyników badania i raporty użytkownika budowane na podstawie szablonów tworzonych w MS Word.

Raporty bezpieczeństwa mogą zawierać dane z podstawowych widoków ASSE (podatności zabezpieczeń, zadania naprawcze, dane aplikacji) – zakres danych i ich dokładność jest określana przez użytkownika. Dostępne są m.in. następujące szablony raportów bezpieczeństwa:

- *Executive Summary* – wysokopoziomowe podsumowanie zarządcze, zawierające informacje o zidentyfikowanych ryzykach oraz informacje statystyczne o przeprowadzonym skanowaniu.
- *Detailed* (raport szczegółowy) – obejmujący wszystkie dane z widoków ASSE.
- *Remediation Tasks* – raport zawierający informacje o zadaniach naprawczych.
- *Developer* – raport zawierający szczegółowe informacje techniczne o ujawnionych podatnościach, zapisy zawartości badanego ośrodka, opisy doradcze i rekomendacje naprawcze.

W odniesieniu do ujawnionych podatności, które mają zostać umieszczone w generowanym raporcie, można zawęzić ich wybór ze względu na typ testów (wszystkie, aplikacyjne, infrastrukturalne) i ograniczyć liczbę wariantów dla danej luki bezpieczeństwa (do określonej liczby). Użytkownik ma także możliwość wpływania w pewnym zakresie na wygląd raportów bezpieczeństwa – dotyczy to m.in. takich kwestii jak umieszczenie w raporcie odpowiedniego logo, zmiany części tytułów czy zawartości nagłówka i stopki.

Raporty zgodności przedstawiają wyniki przeprowadzonego badania w odniesieniu do zgodności (bądź niezgodności) aplikacji z wybranymi unormowaniami prawnymi albo standardami gospodarczymi. Szablony dla tego typu raportów mogą być tworzone przez użytkownika w postaci plików XML zawierających odpowiednie elementy. Wraz z ASSE dostarczane są gotowe szablony m.in. dla takich standardów i regulacji prawnych jak:

- OWASP Top 10 2010,
- WASC Threat Classification,
- ISO 27002,
- ISO 27001,
- Dyrektywa Parlamentu Europejskiego i Rady 95/46/WE,
- Dyrektywa Parlamentu Europejskiego i Rady 2002/58/WE,
- Sarbanes-Oxley Act (SOX),
- Basel II,
- PCI DSS 1.2,
- SANS/CWE Top 25 Most Dangerous Programming Errors.

Raporty porównania wyników badania (*Delta Analysis*) pozwalają porównać dwa skany w zakresie zidentyfikowanych adresów URL i ujawnionych luk bezpieczeństwa.

Raporty oparte na szablonach MS Word pozostają całkowicie pod kontrolą użytkownika – szablony buduje się w procesorze tekstu za pomocą wstawiania pól odpowiadających poszczególnym danym przechowywanym w zapisie skanu.

Wszystkie raporty mogą być zapisane do pliku PDF, natomiast raporty oparte na szablonach MS Word zapisuje się do plików .doc.

Rozszerzenia i narzędzia zewnętrzne

AppScan umożliwia definiowanie własnych testów podatności zabezpieczeń wraz z możliwością wprowadzania własnych danych opisowych, takich jak rekomendacje naprawcze. Testy takie mogą mieć następujący charakter:

- Testy infrastruktury,
- Testy modyfikacji parametru,
- Testy z dodaniem parametru,
- Poszukiwanie wzorca (np. numeru PESEL).

Dane z wykonania tak zdefiniowanych testów będą procesowane w narzędziu w sposób identyczny jak testy pochodzące z bazy dostarczanej przez producenta. Bardziej złożone typy testów można realizować, programując je z wykorzystaniem rozszerzeń AXF, o czym poniżej.

Wraz z aplikacją dostarczany jest zestaw narzędzi zewnętrznych PowerTools wspomagających typowe działania podejmowane przez analityka bezpieczeństwa czy pen-testera. Są to między innymi takie narzędzia jak tester poprawności wyrażeń regularnych (*Expression Test*), analizator tokenów sesyjnych (*Token Analyzer*) czy narzędzie (*Encode/Decode*) do dekodowania i kodowania różnych formatów tekstowych (m.in. URL, Base64, MD5, 3DES).

Ponadto narzędzie dostarczane jest wraz z rozbudowanym pakietem SDK (*Software Development Kit*), dzięki któremu można praktycznie dowolnie dostosowywać i rozbudowywać jego funkcjonalność. SDK może być wykorzystywane w każdym języku zgodnym z .NET-CLR oraz w języku Python poprzez rozszerzenie integracyjne PyScan, dostarczane razem z ASSE (obecnie wspierany jest Python 2.5).

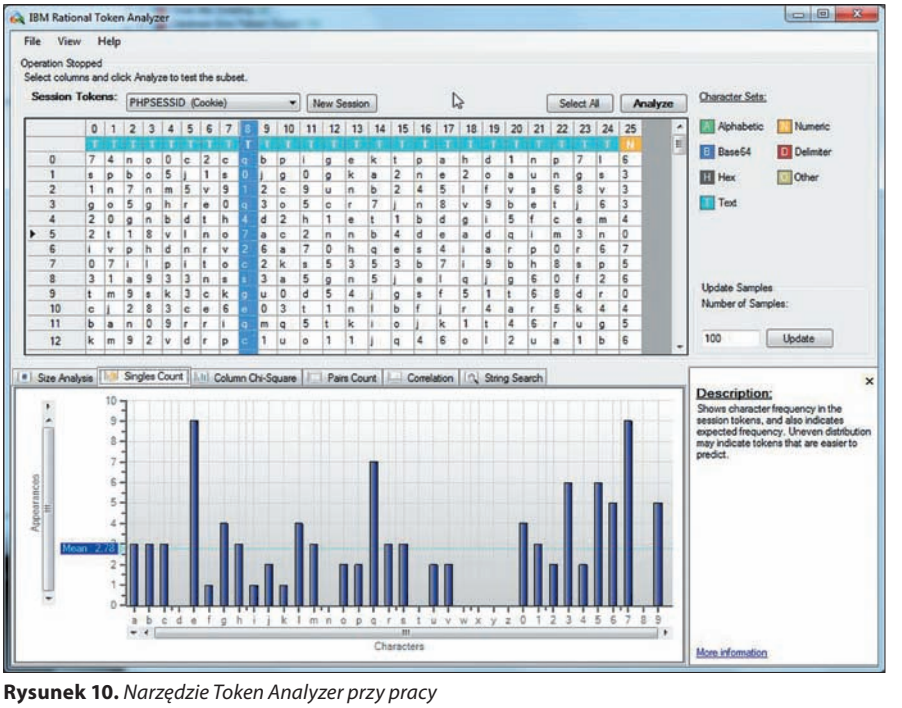
Część z dostępnych rozszerzeń dystrybuowana jest bezpłatnie w ramach AppScan eXtension Framework (adres w ramce), inne są m.in. dostarczane w ramach wsparcia – przykładem jest rozszerzenie Scan Optimizer, wspomagające trudny proces konfiguracji skanowania dużych ośrodków webowych wykorzystujących technikę *URL rewriting*.

Integracja

Poprzez rozszerzenia AXF DefectLoggerCQ, DefectLoggerMQC82, DefectLoggerMQC90 i DefectLoggerTFSASSE umożliwia zgłaszanie błędów do – odpowiednio – IBM Rational ClearQuest, HP/Mercury Quality Center 8.2, HP/Mercury Quality Center 9.0 i Microsoft Team Foundation Server. Ponadto wyniki skanowania można eksportować do postaci pliku bazy danych FireBird i XML.

Marcin Koprowski

Starszy konsultant w firmie Premium Technology. Jego obowiązki koncentrują się głównie w dziedzinie bezpieczeństwa IT. W obszarze produktowym IBM Rational Quality Management posiada certyfikaty: IBM Certified Specialist – Rational AppScan Standard Edition, IBM Certified Solution Designer – Rational Functional Tester for Java.



Rysunek 10. *Narzędzie Token Analyzer przy pracy*

AppScan Standard w Internecie

- Strona informacyjna o produkcie:
  http://www-01.ibm.com/software/awdtools/appscan/standard/
- Polskojęzyczna strona informacyjna o produkcie:
  http://www-142.ibm.com/software/products/pl/pl/appscanse/
- Strona wsparcia techniczego dla produktu:
  http://www-947.ibm.com/support/entry/portal/Overview/Software/Rational/Rational_AppScan_Standard_Edition
- Strona dla deweloperów z materiałami dotyczącymi ASSE:
  http://www.ibm.com/developerworks/rational/products/appscan/
- Strona poświęcona rozszerzeniom AppScan eXtensions Framework:
  http://www.ibm.com/developerworks/rational/downloads/08/appscan_ext_framework/

Architektura Korporacyjna w pigułce

Dla nowoczesnego przedsiębiorstwa ważne jest spełnianie potrzeb swoich klientów w najbardziej efektywny sposób, pozwalający szybko reagować na zmiany zachodzące w bliższym i dalszym otoczeniu. Głównym wyzwaniem wszystkich przedsiębiorstw jest dostarczanie wartości biznesowej (*business value*).

Należy zauważyć, że w organizacjach o wielopoziomowych strukturach, a co za tym idzie o złożonych procesach wewnętrznych i zewnętrznych; w szczególności takich, w których systemy informatyczne są nierozerwalnie związane z dostarczaniem wartości biznesowej , szczególnie trudne staje się zachowanie właściwej proporcji pomiędzy zapewnieniem odpowiednich standardów a zachowaniem koniecznej elastyczności. Złożoność procesów może powodować bowiem brak elastyczności w dostarczaniu wartości biznesowej.

Najlepiej obrazują to firmy mniejsze, konkurujące na poszczególnych wycinkach portfela produktowego dużych firm. Małe firmy mogą być w stanie dostarczyć wartość biznesową szybciej, ponieważ ogół ich procesów i systemów informatycznych na poszczególnych obszarach wspierających dostarczanie wartości jest zasadniczo mniej złożony i posiadający o wiele mniej zależności. Firmy te zwinniej operują w swojej niszy na rynku i niejednokrotnie skutecznie konkurują z większymi dostawcami. Konieczne zatem jest stworzenie modelu działania pozwalającego na zwinne konkurowanie z mniejszymi dostawcami i jednocześnie na wykorzystanie korzyści skali istniejących w większych organizacjach.

Zbyt wysoki stopień złożoności implikuje:

- Niewystarczający przepływ wiedzy,
- Bardzo złożone portfele projektowe,
- Złożone, niezintegrowane i zróżnicowane środowiska informatyczne,
- Duża niepewność i poziom ryzyka związanego z realizacją wartości biznesowej,
- Niewystarczająca dostępność zasobów ludzkich,
- Za niski poziom wiedzy organizacyjnej i technicznej.

Wpływa na to:

- Złożoność produktów i usług, a co za tym idzie procesów oraz wspierających je systemów,
- Konkurencja, złożoność procesów i coraz szybsze tempo biznesu oznacza, że coraz trudniejsze jest dostarczanie wartości biznesowej.

Celem artykułu jest przedstawienie implementacji architektury korporacyjnej w organizacji jako metody adresowania powyższych problemów. W artykule zaprezentowany zostanie zarys elementów architektury korporacyjnej, jak również krótki wstęp do popularnej metody wdrażania architektury korporacyjnej TOGAF. Głównym punktem wyjścia jest wspomniana wcześniej zależność pomiędzy systemami informatycznymi a dostarczaniem wartości oraz przekształcenie roli działów informatycznych z wspierających na działy nierozerwalnie związane z realizowaniem i formułowaniem strategii przedsiębiorstwa. Tekst stanowi podwalinę teoretyczną pod przyszłe artykuły, w których chcemy opisać praktyczne zagadnienia i problemy związane z wdrożeniem architektury korporacyjnej w oparciu o model TOGAF.

Architektura Korporacyjna (*Enterprise Architecture*)

Czym jest Architektura Korporacyjna? W literaturze definiuje się ją jako opis struktur i funkcji komponentów jednostki (komponentami są np. ludzie, procesy biznesowe, struktury organizacyjne, jak również systemy informatyczne), wzajemnych powiązań pomiędzy tymi komponentami oraz pryncypiów i wytycznych zarządzających ich tworzeniem i rozwojem w czasie. Mówimy zatem nie tylko o modelach, ale także o sposobie działania. Wdrożenie takiego sposobu myślenia o organizacji w przypadku systemów informatycznych ma na celu osiągnięcie większej spójności między celami strategicznymi organizacji a ich wspieraniem i realizacją poprzez IT (*Business IT Alignment*). Zainteresowanie architekturą korporacyjną jest skutkiem biznesowych i technologicznych czynników, które zachodzą każdego dnia. Biznesowe wyzwania, takie jak elastyczność i dostosowywanie się do zachodzących zmian (wewnętrznych i zewnętrznych) – są głównym problemem, z którym boryka się większość firm. Dane wskazują, że od 10 do 40% decyzji podejmowanych w biznesie opartych jest na błędnych informacjach (pochodzących np. z systemów BI), które w rezultacie wywołują niewłaściwe działania. Architektura korporacyjna jako dyscyplina ma na celu umożliwienie zwiększenia efektywności przedsiębiorstwa i gotowości do zmian. Każda zmiana, której źródłem jest architektura korporacyjna lub która podlega weryfikacji przez architekturę korporacyjną, ma na celu wsparcie misji przedsiębiorstwa i zmniejszenia czasu potrzebnego do gotowości w sytuacji nadejścia zmiany.

Wdrożenie architektury korporacyjnej jest procesem wieloetapowym. Główne jego elementy to:

- Strategia (informatyczna),
- Model operacyjny,
- Model zdolności biznesowych (*capability model*),
- Wdrożenie modelu architektury korporacyjnej.

Strategia

Pierwszym ważnym etapem przy tworzeniu architektury korporacyjnej jest planowanie strategiczne.

- Strategia jest zbiorem **celów (zadań)**, ujętych w **programy i plany**. Stanowi wzorzec decyzji, które dotyczą **pozycji i tożsamości organizacji**, jego zdolności do wykorzystywania swych mocnych stron oraz prawdopodobieństwa odniesienia sukcesu (K.R. Andrews, 1971).
- Strategia wyraża **cele długoterminowe** organizacji, odpowiadające generalnym kierunkom działania, a także przedstawia alokację zasobów, jakie są niezbędne do realizacji przyjętych celów (A.D. Chandler; 1962).
- Strategia - szeroki program **wytyczania i osiągania celów** organizacji; **reakcja organizacji w czasie** na oddziaływanie jej otoczenia (J.Stoner, R. Freeman, D. Gilbert 1997).

Planowanie strategiczne objąć powinno również strategię informatyzacji:

- Strategia informatyzacji wytycza **cele** w zakresie technologii IT i systemów informatycznych organizacji oraz określa **sposób zarządzania IT**. Uwzględnia również **alokację niezbędnych zasobów** i określa **harmonogram** realizacji celów.

Na strategię informatyzacji składa się wiele elementów:

- analiza aktualnego stanu informatyzacji,
- analiza aktualnego sposobu funkcjonowania organizacji,
- analiza koniecznych nakładów finansowych,
- określenie sposobu realizacji,
- określenie celów strategicznych dla wykorzystania IT.

Jednak najważniejszym elementem jest określenie miejsca IT w procesie realizacji celów organizacji, czyli w uproszczeniu w procesie dostarczania wartości biznesowej. Miejsce to powinno plasować IT jako kluczową zdolność (capability) w dostarczaniu tej wartości. W tym ujęciu traktujemy zdolność biznesową jako jeden z elementów w łańcuchu wartości, czyli CO organizacja robi, aby dostarczyć wartość biznesową. Innymi słowy, należy traktować funkcję IT jako znaczący element w łańcuchu wartości, a nie tylko jako funkcję wspierającą. Dotyczy to szczególnie firm, których model biznesowy jest nierozerwalnie związany z funkcją IT (banki, linie lotnicze, firmy ubezpieczeniowe etc.), ale również takich, gdzie lepsze wykorzystanie zasobów IT może pomóc w tworzeniu przewagi konkurencyjnej lub wręcz nowych usług (firmy handlowe, produkcyjne, logistyczne etc.).

Model Operacyjny

Strategia organizacji przedstawia jej główne cele i jest dobrym narzędziem do wspierania decyzji dotyczących pozycjonowania firmy, jak również ukazuje, na jakich rynkach firma powinna konkurować. Priorytety często się zmieniają i są przeważnie funkcją sytuacji rynkowej oraz nowopowstałych możliwości. Wiele organizacji stara się zmaksymalizować wartość biznesową inwestycji w IT poprzez zrównoleglenie IT i procesów biznesowych wspieranych przez IT ze strategią biznesową (Business-IT alignment).

Jednak osiągnięcie tego zbliżenia strategii i IT jest trudne z przyczyn wymienionych powyżej, strategia koncentruje się na możliwościach rynkowych i celach organizacji. Organizacje nie mogą opierać się tylko na strategii przy rozwijaniu stabilnych i efektywnych systemów IT oraz zdolności (capabilities) wspieranych przez IT. Opieranie się tylko na strategii, co jest częstą praktyką, powoduje, że IT często jest bierne, rozwija dane zdolności dopiero po zaanonsowaniu (nowej) strategii i w efekcie staje się wąskim gardłem organizacji.

W ten sposób powstaje wyzwanie: *W jaki sposób przekształcić IT w element organizacji, który nie tylko wspiera realizację strategii, ale również pomaga w jej tworzeniu?*

Wyzwanie to może być adresowane przez wybór modelu operacyjnego IT. Model operacyjny definiowany jest jako: *wymagany poziom integracji procesów biznesowych oraz standaryzacji w celu dostarczania usług i produktów klientom.*

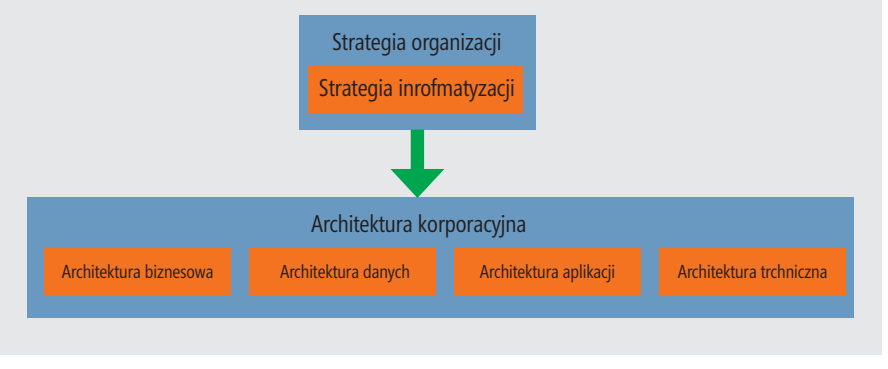
Podstawą modelu operacyjnego jest wybór osiach:

- w jakim stopniu powinny być ustandaryzowane procesy biznesowe,
- w jakim stopniu procesy powinny być ze sobą zintegrowane.

Odpowiedź na powyższe pytania pozwoli nam oprzeć się na jednym z czterech możliwych modeli operacyjnych:

- **Dywersyfikacja** - Charakteryzuje się niską standaryzacją procesów i niską integracją systemów. Jest to model zdecentralizowany. Jednostki biznesowe operują na różnych rynkach z różnymi produktami i usługami i potrzebują autonomii lokalnej, by decydować, jak najlepiej obsługiwać klienta. Model ten często występuje w firmach o bardzo szerokim zakresie usług i produktów, które rzadko podlegają możliwościom interakcji pomiędzy jednostkami i synergii.
- **Unifikacja** - Charakteryzuje się wysoką standaryzacją i wysoką integracją. Tutaj mamy do czynienia z modelem scentralizowanym. Organizacji zależy na pewności, przewidywalności i ustandaryzowanych procesach, przez co powstaje „jedna twarz" dla klientów, niezależnie od lokalizacji. Wiele linii lotniczych operuje w ten sposób.
- **Koordynacja** - Cechuje się niską standaryzacją i wysoką integracją. Model integracyjny. Organizacja tworzy jednolity wizerunek zewnętrzny bez narzucania standardów procesowych dla lokalnych jednostek. Przykładem takiego modelu są sieci dilerów samochodowych, w każdym kraju funkcjonują ograniczone przez lokalne wymagania, ale są w stanie łatwo i sprawnie zamawiać na przykład części w innych krajach poprzez zintegrowane systemy magazynowe i produktowe.
- **Replikacja** – Cechuje się wysoką standaryzacją i niską integracją. Model standaryzacji procesowej. Jednostki korzystają z tych samych systemów i stosują te same procesy, przez co wspierają budowanie jednolitego wizerunku firmy, jednocześnie jednostki te operują autonomicznie i nie polegają na interakcji z innymi jednostkami. Dobrym przykładem są sieci sklepów lub hotele.

Wybór modelu operacyjnego jest ważny z kilku powodów:



Rysunek 1. *Strategia informatyzacji nie jest kompletna bez zdefiniowania modelu operacyjnego IT.*

- Wytycza długofalowe podejście do rynku oraz definiuje jakie procesy i w jaki sposób odróżniają firmę od jej konkurentów.
- Ogranicza zakres zdolności biznesowych (business capabilities), które powinny zostać zdefiniowane w organizacji.
- Umożliwia i ułatwia wielokrotne wykorzystywanie raz zdefiniowanych zdolności w wielu jednostkach i procesach.
- Umożliwia standaryzację technologii i systemów, która w połączeniu z modelem zdolności paradoksalnie zwiększa elastyczność firmy.

Model operacyjny tworzy powtarzalny fundament do wykonywania strategii firmy, przez co IT przeistacza się z wąskiego gardła w środek umożliwiający realizację strategii (strategy enabler). W tym sensie wybór modelu jest ważnym krokiem przy tworzeniu architektury korporacyjnej, gdyż ma on znaczący wpływ na modelowanie organizacji, jej zdolności oraz zależności pomiędzy nimi.

Model Zdolności Biznesowych (*Capability Model*)

Jak już wspomnieliśmy wcześniej, architektura korporacyjna opiera się na tzw. Business Capabilities, czyli zdolnościach biznesowych. Oznacza to, że zamiast tradycyjnego modelowania organizacji w zakresie oderwanych od siebie procesów i funkcji powinniśmy skoncentrować się na zdolnościach biznesowych (capabilities). Zdolności pozwalają dostarczyć pewien element wartości biznesowej organizacji, a cały łańcuch zdolności składa się na łańcuch wartości organizacji. Tworzenie zdolności istnieje w stałym ścisłym związku pomiędzy zasobami ludzkimi, technologią i procesami (wspieranymi przez technologię), pozwala tworzyć bardziej wydajne i efektywne procesy biznesowe oraz wspiera innowację poprzez lepszą dostępność informacji i zwiększoną produktywność.

Model zdolności powstał w efekcie ewolucji wcześniejszych modeli organizacyjnych, takich jak silosy funkcjonalne zawierające wyspecjalizowane procesy i procedury oraz Business Process Reengineering. Model zdolności daje pełnowymiarowy wgląd w łańcuch wartości poprzez uwzględnienie wszystkich jego części składowych w relacji do siebie.

Wyższość podejścia opartego na capabilities nad podejściem czysto procesowym polega między innymi na ułatwieniu przyszłych zmian w modelu biznesowym poprzez biznesowych możliwość organizowania funkcji biznesowych w formie komponentów. Ponadto model zdolności pozwala na stworzenie wspólnego języka wewnątrz organizacji w celu efektywnego dzielenia się informacją i umożliwia lepszy wybór rozwiązań technologicznych w celu rozwikłania problemów/wyzwań biznesu.

Zdolności mogą być bezpośrednio powiązane ze strategią biznesową, tworząc naturalne łącze pomiędzy architekturą korporacyjną a strategią biznesową. Wielu CIO i CEO zdefiniowało potrzebę takiego powiązania; trudną jednak do zrealizowania, kiedy nacisk kładziony był na technologię w odizolowaniu od innych elementów organizacji (Architektura IT) lub na procesy w odizolowaniu od innych elementów organizacji (BPR). Kolejną korzyścią określania zdolności jest możliwość przekazywania celów organizacyjnych z poziomu strategii na poziom taktyczny.

Wdrożenie modelu architektury korporacyjnej

Posiadając jasny obraz strategii, modelu operacyjnego oraz pożądanych zdolności biznesowych, możemy rozpocząć tworzenie organizacyjnego modelu architektury korporacyjnej. Modelowanie architektury korporacyjnej powinno mieć charakter ciągły, oparty na stałym uwzględnianiu tych trzech elementów przy jednoczesnym uwzględnieniu wszystkich aspektów organizacji. Przez lata powstało wiele metod i szkieletów dotyczących wdrożenia architektury korporacyjnej. Najbardziej znanymi z nich są:

- TOGAF - The Open Group Architecture Framework,
- Zachmann,
- DoDAF / MoDAF.

W ostatnich latach dużą popularność zdobył TOGAF z powodu jego kompleksowego, cyklicznego podejścia i jasnego powiązania pomiędzy procesami biznesowymi i IT. W dalszej części przedstawimy ogólny zarys szkieletu TOGAF oraz etapów związanych z jego wdrożeniem.

TOGAF jest w pewnym sensie kontynuacją modelu Zachmanna. Zachmann definiuje taksonomię architektoniczną i mówi nam, jak kategoryzować poszczególne komponenty organizacyjne , natomiast TOGAF definiuje proces, wedle którego te komponenty powstają. TOGAF dzieli architekturę korporacyjną na cztery główne obszary:

- Architektura biznesowa - opisuje procesy i zdolności biznesowe, wykorzystywane przez organizację w osiąganiu celów strategicznych.
- Architektura aplikacyjna – opisuje, jak poszczególne aplikacje powinny być projektowane i w jaki sposób odbywa się ich interakcja.
- Architektura danych – definiuje, w jaki sposób zbiory danych i informacji w organizacji są udostępniane.
- Architektura techniczna - opisuje infrastrukturę sprzętową i systemową, która wspiera ich aplikacje oraz ich interakcje.

Jednocześnie TOGAF opisuje architekturę korporacyjną jako swoiste kontinuum architektur (Enterprise Continuum), od bardzo generycznych po wysoce wyspecjalizowane.

- Foundation Architectures,
- Common Systems Architectures,
- Industry Architectures,
- Organizational Architectures.

Ważnym elementem TOGAF'a jest tzw. Architecture Development Method (ADM). Ta metodyka opisuje proces tworzenia wyspecjalizowanej architektury organizacyjnej przy uwzględnieniu czterech obszarów wymienionych wcześniej. Zgodnie z tą metodyką wdrażanie architektury korporacyjnej opiera się na cyklu ośmiu etapów poprzedzonych etapem wstępnym.

- Etap wstępny (*Framework and Principles*) – W tym etapie tworzy się warunki do wdrożenia architektury korporacyjnej. Na podstawie zdefiniowanego modelu operacyjnego i pożądanych zdolności biznesowych ustalana jest organizacja odpowiedzialna za wdrożenie architektury, tworzony jest system zarządzania procesem tworzenia architektury oraz definiowani są interesariusze. Głównym celem jest zapoznanie się wszystkich graczy z procesem oraz dostosowanie procesu do kultury i specyfiki organizacji.
- *Architecture Vision* – w tym etapie definiowane są zakres, ograniczenia i wymagania biznesowe co do bieżącego cyklu rozwoju architektury. Ponadto walidowane są cele i zasady biznesowe oraz nośniki biznesowe organizacji. Etap tworzenia wizji angażuje również wszystkich wymaganych interesariuszy. Końcowym celem jest przedstawienie wizji architektury odpowiadającej na wszystkie wymienione zagadnienia.
- *Business Architecture* – opisuje bazową (istniejącą) oraz docelową architekturę biznesową, na którą składają się elementy takie jak strategia produktu oraz aspekty związane z organizacją, procesami, informacją i zasobami danego elementu biznesu w oparciu o wcześniej zdefiniowane cele, nośniki i ograniczenia. W tym etapie analizowana jest również luka (*Gap Analysis*) pomiędzy architekturą bazową i docelową.
- *Information Systems Architecture* – celem tego etapu jest rozwój architektury docelowej pokrywającej domeny danych i aplikacji. W przypadku danych chodzi tu przede wszystkim o definicję źródeł i typów informacji wymaganych przez interesariuszy. Chodzi tu przede wszystkim o zdefiniowanie encji i obiektów wymaganych przez organizację, a NIE o projektowanie baz

danych. Podobnie w przypadku domeny aplikacji chodzi o opisanie ich jako logicznych zestawów zdolności (*capabilities*), które operują na obiektach wcześniej zdefiniowanych i wspierających funkcje biznesowe opisane w poprzednim etapie.

- *Technology Architecture* – na tym etapie staramy się zmapować komponenty aplikacji na zestaw komponentów technologicznych, składających się z oprogramowania i sprzętu.
- *Opportunities and Solutions* – celem tego etapu jest opisanie, JAK dostarczyć architekturę i jej komponenty powstałe we wcześniejszych etapach. Utworzenie architektury tranzycji (*Transition Architecture*).
- *Migration Planning* – porządkowanie projektów wedle ich priorytetów, ich korzyści biznesowych i związanego z nimi ryzyka w organizacji.
- *Implementation Governance* – tworzenie specyfikacji dla projektów implementacyjnych ze szczególnym uwzględnieniem procesu kontroli projektów, kryteriów akceptacji oraz ryzyka.
- *Architecture Change Management* – wzbogacenie procesu o nowe artefakty i informacje powstałe podczas bieżącego cyklu.
- *Requirements Management* – ten etap cyklu wspiera wszystkie pozostałe i jest ich nośnikiem. Zapewnia on identyfikację i przechowywanie wymagań architektury korporacyjnej i ich dostarczanie do wszystkich pozostałych etapów ADM.

Powyższy opis jest tylko przybliżeniem cyklu ADM. W rzeczywistości każdy z etapów jest dosyć złożony i wymaga solidnego przygotowania. Pokazuje on jednak, że efektywne wdrożenie architektury korporacyjnej wymaga zgrania elementów wymagań biznesowych i technicznych. Jednocześnie ukazuje on wartość dodaną wcześniejszego zdefiniowania modelu operacyjnego i zdolności biznesowych. Stanowią one bardzo ważny element tworzenia wizji architektury biznesowej i architektury informacyjnej. To właśnie dzięki wcześniejszemu przygotowaniu tych komponentów zapewniamy sobie spójność architektury korporacyjnej z celami i możliwościami organizacji. W dodatku operowanie pojęciami zdolności biznesowych pozwala na łatwiejsze i bardziej przejrzyste komunikowanie architektury korporacyjnej językiem zrozumiałym dla wszystkich członków organizacji.

Podsumowanie

Architektura korporacyjna stara się niwelować luki pomiędzy strategią a realizacją zmian wynikających ze strategii przedsiębiorstwa poprzez:

- Uzyskaniu wglądu w obecne, planowane, jak i pośrednie stany przedsiębiorstwa w spójny i usystematyzowany sposób (śledzenie w dwie strony: strategia-rozwiązania).
- Kontrola alokacji zasobów w każdym z obszarów przedsiębiorstwa.
- Kontroli zakresów projektów i programów.
- Kontroli przebiegu transformacji i zmian w organizacji.
- Wypracowanie spójnej formy komunikowania zmian.

Wszystko to składa się na możliwość przekształcenia działu IT z centrum kosztowego w znaczący nośnik strategii i wartości biznesowej, co zwiększa konkurencyjność firmy. Architektura korporacyjna, oprócz przedsiębiorstw budowanych głównie w oparciu o usługi informatyczne, ma zastosowanie wszędzie tam, gdzie do działania przedsiębiorstwa i realizacji jego misji niezbędne jest wsparcie infrastruktury informatycznej i usług świadczonych przez aplikacje budowane w oparciu o tę infrastrukturę – czyli w zasadzie w każdym większym przedsiębiorstwie. Staje się ona centralną funkcją biznesu i IT. Jesteśmy świadkami znaczącej zmiany w mentalności ludzi – specjaliści IT wiedzą, że to, co robią, stanowi wartość dla biznesu. Natomiast ludzie biznesu, myśląc o swoim przedsiębiorstwie, stają się świadomi, jakiej technologii potrzebują. W znaczący sposób rośnie znaczenie architektów IT, którzy w korporacjach stają się osobami spajającymi biznes i IT.

W przyszłych artykułach przedstawimy praktyczne zagadnienia, wyzwania i problemy związane z wdrożeniem architektury korporacyjnej.

Michał Chełmowski

Dyrektor Operacyjny QNH Polska sp z o.o. Michał przez osiem lat pracował w firmie QNH w Holandii. Zdobył tam doświadczenie przy wielu projektach informatycznych, optymalizacyjnych i doradczych, przede wszystkim w sektorze finansowym w instytucjach takich jak ING Bank, Rabobank, Fortis czy Van Lanschot Bankiers. Od dwóch lat współodpowiedzialny za tworzenie polskiej firmy córki QNH we Wrocławiu i Warszawie.

KENNETH HANSEN

Rozmowa z Kennethem Hansenem

14-krotnym Mistrzem Europy w Rallycross

Jeśli spytamy losowo napotkaną na polskiej ulicy osobę, z czym jej się kojarzy Szwecja, zapewne odpowie, że z zespołem Abba, samochodem Volvo oraz księżniczką Victorią. Gdy o to samo spytamy kogoś ze środowiska sportów samochodowych – wymieni cała plejadę kierowców rajdowych, odnoszących sukcesy w imprezach o randze Mistrzostw Świata. Zapytany o to samo zwykły Szwed odpowie, że dla niego najbardziej znaną i rozpoznawaną osobą jest 14-krotny Mistrz Europy Rallycross 50-letni Kenneth Hansen, mieszkaniec miasta Götene. W jego karierze kibice mogli go podziwiać w kartingu, folkrace, rajdach (również jako pilota), ale głównie upodobał sobie rallycross, rodzaj wyścigu na zamkniętym torze o długości około 1 km, o zmiennej nawierzchni asfaltowo – szutrowej, bywa też, że o zmiennych poziomach.

Gdy przyjrzymy się danym statystycznym o tym zawodniku, są one wprost imponujące. 14 tytułów mistrzowskich, 6 wicemistrzowskich oraz 10 „koron" mistrza Szwecji w rallycrossie, Statystyki prowadzone przez jego zespół logistyczny wykazały poza tym, że startował przez 23 lata w 15 różnych krajach, zaliczył 236 wyścigów, ogólnie zebrał 3676 punktów za 73 zwycięstwa, 58 razy drugie miejsce oraz 30 razy trzecie, co oznacza, że odkąd zaczął starty w Mistrzostwach Europy – w 71% kończyły się dla niego one staniem na podium. Sukcesy Kennetha nie były możliwe do osiągnięcia bez współpracy całego zespołu, jaki stworzył; trzeba pamiętać, że to nie tylko kierowca sportowy, ale człowiek – instytucja, będący głównym filarem teamu Kenneth Hansen Motorsport. Kenneth jest żonaty z Susan, ma synów Tima – 17 lat, oraz Kevina, lat 11, obaj z powodzeniem startują w międzynarodowych oraz krajowych mistrzostwach w kartingu.

W tym roku Szwed gościł w Polsce już po raz jedenasty na przedostatniej rundzie Mistrzostw Europy, która odbyła się we wrześniu na torze w Słomczynie.

Korzystając z okazji spotkania Kennetha, warto było spytać go o jego wizję idealnego teamu, która, jak widać w jego przypadku, doskonale się sprawdziła.

Jak długo trwa zbudowanie profesjonalnego teamu wyścigowego, który ma ściśle wyznaczone role i zadania do wykonania, przy całkowitej orientacji na sukcesy swojego frontmana, czyli zawodnika?

Zespół, z którym współpracuję obecnie, został bazowo utworzony już około 10 lat temu, tymczasowo dochodziły do niego osoby z zawodników, którzy wchodzili do mojego teamu. W zeszłym roku był to Fin Pinomaeki, w tym roku jest to Doran, który zostanie u mnie dłużej. Widzę w nim ogromny potencjał, wolę walki oraz niesłychany samoistny talent, który odziedziczył po swoim ojcu, kilkakrotnym mistrzu Wielkiej Brytanii. Najdłużej mam w swoim zespole mechaników specjalizujących się w moim Citroenie, na stałe też współpracuję z działem sportu oraz inżynierami tej firmy, chociaż niektóre modyfikacje wprowadzamy sami. Jak wiadomo, auto wymaga ustawień indywidualnych na każdy tor,

dobór zawieszenia, przełożeń w skrzyni biegów, opon oraz inne nie mniej istotne rzeczy wymagają ścisłej współpracy kilku osób. Ja to wszystko koryguję i „spinam", przekazując, jak to wygląda z mojej strony jako kierowcy.

Jakie są kryteria doboru członków teamu? Czy wszyscy muszą mieć jednakowy, nadzwyczajny poziom wszelkich umiejętności, czy jest to na zasadzie dopełnienia, czyli każdy jest perfekcyjny w swojej dziedzinie i wszyscy razem, wspierając się, kompletują wykonanie zadania?

Faktem jest, że zanim kogoś przyjąłem do teamu, obserwowałem jego poczynania, a przede wszystkim stosunek do pracy, w moim przypadku bardzo wymagającej. Nie każdy jest w stanie znieść większość roku poza domem... Miałem jednak szczęście trafić na ludzi całkowicie oddanych, dla których mój sukces jest najważniejszy i którzy cieszą się z każdego zwycięstwa tak samo jak ja. Każdy ma swoją specjalizację, dodatkowo ogromną rolę w moim zespole pełni moja żona Susan, która zajmuje się stroną medialno – marketingową oraz kontaktami z działami sportu w krajach, w których startuję. Pomagali mi ogromnie moi rodzice, mimo podeszłego wieku jeździli na każde zawody, zajmując się pomiarami czasów, które zawsze prowadzę niezależnie od pomiarów organizatora. Po każdym wyścigu z mechanikami analizujemy jego przebieg, niemal metr po metrze, aby ustalić, co jeszcze można poprawić dla uzyskania lepszego wyniku.

Co jest najważniejszym elementem działania teamu, bez którego niemożliwe jest jego funkcjonowanie?

Współpraca i wzajemna komunikacja, oraz świadomość tego, po co to wszystko robimy.

W jaki sposób team oraz jego frontman przygotowuje się do startów lub testów w nieznanym kraju na nieznanym torze (np. w Tatarstanie)?

Różnie, ale głównie jest to dokładny research, zbieranie wszelkich możliwych informacji, zapisów video, w końcu też wizyta bezpośrednia. Kiedy pierwszy raz przyjechałem do Polski w roku 2000, nic nie wiedziałem o Słomczynie ani o was, Polakach. Miałem pewne obawy, ale po pierwszym wyjeździe na tor wiedziałem, że będzie dobrze. Lubię tu przyjeżdżać, nie tylko dlatego, że wymaga tego strategia moich startów.

Kto bezpośrednio uczestniczy w budowaniu pozytywnego wizerunku kierowcy - mistrza Europy, oprócz jego samego oczywiście - i na czym takie budowanie wizerunku w tym przypadku się opiera?

Robimy to wszyscy, budowanie wizerunku to nie tylko wypowiedzi dla mediów i oprawa informacyjna, ale też kontakty z kibicami, dla których zawsze mam czas i jestem dostępny, lub też nawet tak jak tutaj w Słomczynie udział w wyścigu „szayowozów". Jechałem tym pojazdem pierwszy raz w życiu, jakoś mi to szło, ale dla mnie istotne było też to, że chłopak, który dojechał na metę pierwszy, będzie do końca życia wspominał, że wygrał z Hansenem. Pozytywny wizerunek to uszczęśliwianie innych po prostu.

Czy członkowie Twojego teamu spotykają się tylko na zawodach i przygotowaniu do nich, czy też jakoś spędzacie czas razem pomiędzy imprezami? Czy w jakiś sposób premiujesz ich wysiłki w osiąganiu sukcesów przez Ciebie?

Spędzamy ze sobą większość roku, gdyż zawsze jesteśmy albo przed zawodami, albo po nich, albo też w trakcie przygotowań. Mechanicy pracują również w mojej firmie Kenneth Hansen Motorsport, gdzie zajmujemy się zaopatrywaniem w części do aut wyczynowych innych zawodników lub ludzi, którzy chcą mieć szybkie samochody.

Rallycross to wyścigi samochodowe, które w swojej formie przypominają nieco odcinek specjalny rajdu. Na zamkniętym, szutrowo – asfaltowym torze, startuje jednocześnie kilku (4-8) zawodników. Zwycięzcą zostaje ten, który jako pierwszy przejedzie linię mety w finale.

Układ toru rallycrossowego pozwala kibicom obserwować odbywającą się na nim walkę przez cały wyścig. O widowiskowości tej dyscypliny stanowią bliski kontakt zawodników, liczne stłuczki, otarcia, oderwane fragmenty karoserii oraz to, co kibice lubią najbardziej – efektowne dachowania. W rallycrossie mniej istotna jest prędkość niż przyspieszenie, pozwalające na najbardziej efektywne pokonywanie odcinków pomiędzy zakrętami. Najmocniejsze auta rozpędzają się do „setki" w 2,5 sekundy.

W Polsce, na torze w Słomczynie koło Grójca, odbywa się jedna z dziesięciu rund Mistrzostw Europy – najwyższych rangą zawodów w tej dyscyplinie *http://www.rallycross.pl*

Co do premii – to musisz ich spytać sama, czy jestem wdzięcznym szefem!

Czy budując swój team, Kenneth Hansen opierał się na jakimś pierwowzorze, czy do wszystkiego doszedł sam metodą prób i błędów?

Specyfika każdego zespołu jest inna, więc o ile można stosować jakieś podstawowe zasady ogólnie przyjęte, to już ukierunkowanie na takie czy inne aspekty wymaga metody prób i błędów – czasem idzie to w parze z zyskami, ale czasem i stratami.

W jaki sposób w pracy zespołu wykorzystywana jest elektronika?

Powiem tak: nie byłoby moich sukcesów, gdyby nie wszechobecne wykorzystywanie elektroniki. Mój samochód (obecnie Citroen C4) jest całkowicie centralnie sterowany elektroniką w oparciu o system Peltor. Oznacza to, że możemy przy pomocy tego systemu w każdej chwili zawodów podpiąć się pod laptopa i zmieniać ustawienia lub też odczytywać dane przekazywane przez każdy układ w aucie, stąd np. wiemy, co się stało – jeśli właśnie coś pracuje niewłaściwie. Do odczytów i zmiany ustawień trzeba być wielkiej klasy specjalistą, każda najmniejsza nieodpowiednia ingerencja może zawodnika drogo kosztować. Elektronika jest też wykorzystywana w komunikacji pomiędzy członkami ekipy, czasami jesteśmy w różnych miejscach i musimy natychmiast się spotkać, więc po prostu wołamy się przez radio. Na moim zespole transportowym jest satelita, nie tylko do TV, ale i do szybkiego internetu, jest też cały zestaw multimedialny do analizy obrazów wideo. Jak widzisz, mam 3 komputery i każdy służy do czego innego. Jeden jest tylko i wyłącznie na użytek mediów – to tutaj obrabiane są fotografie moich występów, tworzone komunikaty prasowe i wysyłane do dziennikarzy. Mam też swojego współpracownika, który prowadzi moją grupę na Facebooku. Dzisiaj jak Cię nie ma na Facebooku, to nie istniejesz – śmieje się Kenneth.

A jaki jest jako człowiek? Nasuwa się tylko jedno słowo – charyzmatyczny. Uważny słuchacz stara się zawsze usatysfakcjonować rozmówcę nie tylko słowami, ale i miłym uśmiechem oraz żartobliwym sposobem bycia. Promieniuje z niego tak wiele pozytywnej i spokojnej energii, że chciałoby się być w jego obecności jak najdłużej.

Rozmawiała: Joanna Kalinowska

Jazz i OSLC

Jazz to platforma i linia produktów z kategorii Application Lifecycle Management rozwijana przez IBM od 2008 roku. Jazz powstał w odpowiedzi na rosnące problemy z użytkowaniem tego typu aplikacji poprzedniej generacji. Głównym problemem była fragmentaryzacja i duplikacja danych pomiędzy różnymi narzędziami (tzw. silo approach). Te same artefakty (np. wymagania) przechowywanie i zmieniane były niezależnie w wielu miejscach. Utrudniało to utrzymanie spójnego stanu danych oraz tworzenie raportów. Problem pogłębiało zjawisko vendor lock-in utrudniające współpracę narzędzi pochodzących od różnych producentów. Integracja o ile istniała, to w postaci mostów (bridge) łączących pojedyncze systemy.

Jazz obiecuje rozwiązać ten problem przez zmianę podejścia. Wcześniejsze produkty skupiały się na narzędziach, Jazz w centrum uwagi stawia dane. Dokumenty stanowiące opis projektu definiowane i zapisywane są za pomocą otwartych standardów.

Podstawą dla produktów z linii Jazz jest Jazz Foundation, który udostępnia API realizujące funkcje wspólne dla większości aplikacji, jak np. zarządzanie użytkownikami i przechowywanie danych oraz definiuje podstawowe pojęcia, np. projekt i użytkownik. Narzędzia bazujące na Jazz Foundation komunikują się pomiędzy sobą, oraz z Foundation Serverem za pomocą HTTP, co umożliwia integrację pomiędzy programami stworzonymi w dowolnych technologiach.

Istniejące otwarte standardy pozwalały zrealizować komunikację (HTTP), wymianę danych (XML i RDF) oraz zapewnić bezpieczeństwo (SSL i OAuth). Aby zredukować zjawisko *vendor lock-in*, potrzebne były jednak standardy dotyczące wyższego poziomu abstrakcji. W tym celu równolegle z Jazz IBM rozpoczął inicjatywę Open Services For Lifecycle Collaboration. OSLC jest zbiorem ogólnodostępnych definicji podstawowych zasobów (zmiany, defekty, wymagania, user stories) oraz specyfikacji podstawowych serwisów.

Obecnie dostępnych jest siedem produktów korzystających z Jazz oraz OSLC. Przeznaczony głównie dla programistów Rational Team Concert dostarcza m.in. kontrolę wersji, wsparcie dla procesów, planowanie iteracji, bugtracker i build system. RTC pozwala w prosty sposób zarządzać projektem, bardziej zaawansowane funkcje pozwala realizować Rational Project Conductor. Rational Requirements Composer pozwala na pracę nad wymaganiami. Wszystkie te produkty są zintegrowane, co pozwala np. odszukać, z jakiego wymagania wynikała wybrana zmiana w systemie kontroli wersji.

Sercem produktów opartych na platformie Jazz jest Jazz Foundation Server. Niniejszy artykuł ma na celu zaznajomienie Czytelnika ze sposobem współpracy z nim oraz podstawowymi dostępnymi serwisami. Druga część zawiera opis konfiguracji środowiska pracy koniecznego do stworzenia nowej aplikacji korzystającej z Foundation Servera, oraz opis protokołu OAuth, używanego do uwierzytelnienia użytkowników oraz aplikacji. W części trzeciej stworzona zostanie prosta aplikacja, która uwierzytelni użytkownika, a następnie powita go, wyświetlając jego login. Część czwarta opisuje sposób składowania i przechowywania danych w Foundation Serverze oraz Resource Description Framework – otwarty standard służący do rozproszonego przetwarzania i wymiany danych.

Artykuł wymaga znajomości podstaw programowania aplikacji J2EE, protokołu HTTP oraz formatu XML. Do współpracy z Foundation Serverem posłuży Jazz Foundation Application SDK dostarczone jako paczka OSGi, pomocna będzie więc znajomość środowiska Eclipse.

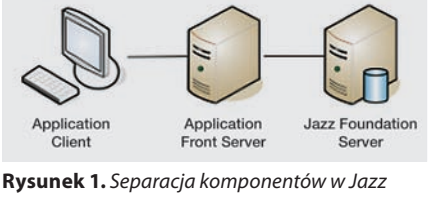
Podstawy

Instalacja

Jazz Foundation Server wraz z licencją testową ściągnąć można z jazz.net. Dostępny jest zarówno instalator, jak i wersja w archiwum ZIP. Oprócz serwera dla właściwej platformy należy ściągnąć Jazz Foundation Application SDK z tej samej strony (niezależne od platformy).

W zależności od wybranego sposobu instalacji należy rozpakować archiwum z serwerem do wybranego katalogu lub uruchomić instalator. Następnym krokiem jest uruchomienie serwera i dodanie nowego użytkownika. Panel administracyjny dostępny jest pod adresem *https://localhost:9443/jazz/admin*, domyślny użytkownik(nie należy używać go do normalnej pracy) i hasło to: *ADMIN/ADMIN*

Do pracy potrzebne będzie również środowisko Eclipse. Konfiguracja ogranicza się do importu zawartości folderu `osgi/lib/eclipse` z archiwum SDK jako docelowej platformy OSGI. Odpowiednie funkcje dostępne są poprzez `Window | Preferences...` w węźle `Plugin development`. Wśród pluginów znajduje się specjalnie przygotowany serwer Jetty, który posłuży do testowania aplika-



Rysunek 1. *Separacja komponentów w Jazz Integration Architecture*

cji bez konieczności umieszczania ich na serwerze J2EE.

Dokładny opis procedury instalacyjnej, razem z zrzutami ekranu znajduje się na Jazz Team Wiki Na problemy z działaniem serwera (np. komunikaty 405 Gone i 404 Not found w odpowiedzi na zapytania HTTP do znanych adresów lub permanentny błąd „OAuth token invalid or expired") często najprostszym rozwiązaniem jest usunięcie i ponowna instalacja.

Jazz Foundation Server domyślnie wymusza komunikację szyfrowaną protokołem SSL. Instrukcję, jak umożliwić nieszyfrowany dostęp do serwera testowego, również znaleźć można na Jazz Team Wiki.

Application Front Server

Jazz Foundation Application SDK pozwala stworzyć tzw. Application Front Server. Jak widać na rysunku, Front Server zajmuje miejsce pomiędzy końcowym klientem (np. przeglądarką) a Jazz Foundation Serverem. Foundation Server jest niewidoczny dla klienta, co pozwala na zmianę wewnętrznych serwisów bez konieczności modyfikacji klientów.

Każdy Front Server stworzony za pomocą JFS-SDK jest niezależną aplikacją J2EE, którą można wdrożyć na serwer w standardowy sposób, jako archiwum WAR. W ramach SDK do wyboru są dwa sposoby integracji: tzw. POJO, który wymaga ręcznego zarządzania bibliotekami SDK, oraz OSGi, który używa mechanizmów zarządzania pluginami. W dalszej części artykuł dotyczył będzie wersji OSGi, nie ma jednak między nimi znaczących różnic.

Ponieważ serwery komunikują się za pomocą protokołu HTTP, istnieje oczywiście możliwość wykonania Front Servera w dowolnej technologii pozwalającej wysyłać zapytania HTTP.

OAuth i uwierzytelnianie użytkowników

Poufność komunikacji zapewnia protokół SSL, natomiast do uwierzytelnienia zarówno klienta, jak i Front Servera służy mechanizm OAuth. OAuth to otwarty protokół autentykacji, który pozwala użytkownikom udostępniać prywatne zasoby (*protected resources*) składowane w jednym serwisie (np. ze zdjęciami) innemu serwisowi (np. zajmującemu się drukiem zdjęć).

Głównym celem OAuth jest umożliwienie konsumentowi serwisu (*service consumer*) wykonywanie u dostawcy serwisu (*service provider*) operacji w imieniu użytkownika (*user*), bez ujawniania konsumentowi hasła ani tożsamości użytkownika. OAuth pozwala określić (zarządzane na poziomie dostawcy serwisu) dla każdego z konsumentów indywidualny zakres uprawnień oraz modyfikację uprawnień w dowolnej chwili.

Listing 1. *Element pliku plugin.xml*

```
<extension point="com.ibm.team.jfs.app.restServices">
      <restService
            alias="library"
            class="com.example.libsample.MainService"
            displayName="Library Sample Service">
      </restService>
   </extension>
```

Listing 2. *MainService.java*

```
 public class MainService extends RestService {
 public void doGet(HttpRequest request, HttpResponse response)
   throws IOException {
  response.setStatusCode(HttpStatus.SC_OK);
  response.setEntity(new StringEntity("Hello World"));
 }
}
```

Listing 3. *Element pliku plugin.xml*

```
   <extension point="com.ibm.team.server.embedded.jetty.webAppContext">
      <context name="/libsample" path="WebContent">
      </context>
   </extension>
```

Listing 4. *Plik web.xml*

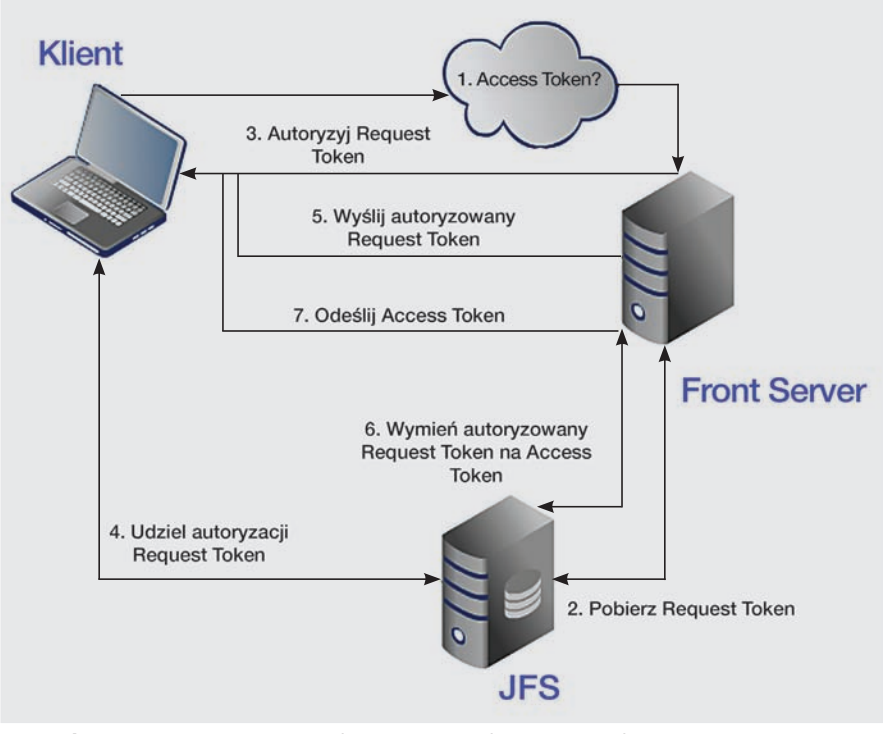
```
    <?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns=
                          "http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
 <servlet id="bridge">
  <description>Equinox Bridge Servlet</description>
  <display-name>Equinox Bridge Servlet</display-name>
  <servlet-name>equinoxbridgeservlet</servlet-name>  <servlet-class>org.eclipse.eq
                    uinox.http.servlet.HttpServiceServlet</servlet-class>
  <init-param>
   <param-name>enableFrameworkControls</param-name>
   <param-value>false</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
  <servlet-name>equinoxbridgeservlet</servlet-name>
  <url-pattern>/*</url-pattern>
 </servlet-mapping>
</web-app>
```

Listing 5. *dw_lib_sample.properties*

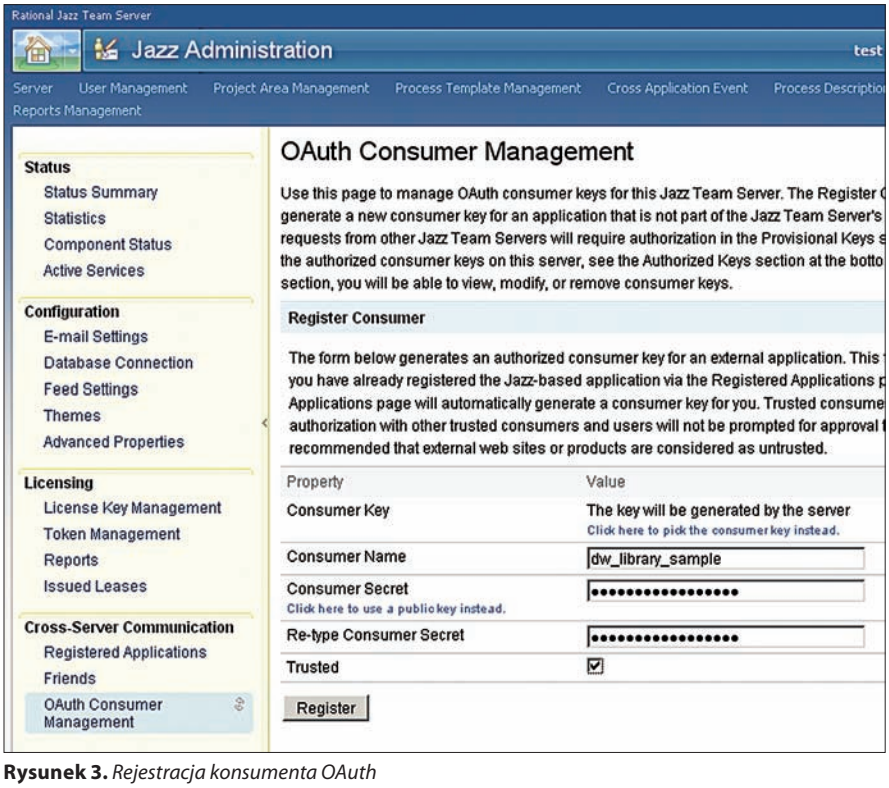
```
com.example.libsample.consumerSecret=dw_library_secret
com.example.libsample.consumerKey=0657a3a4bc71444faf76f216a16269e0
com.example.libsample.jfsRootServicesUrl=
   https://localhost:9443/jazz/   rootservices
```

OAuth (w wersji 1.0a) to w pełni bezpieczny, otwarty standard, sformalizowany (po trzech latach rozwoju) w kwietniu tego roku w postaci RFC 5849. Używany jest między innymi przez Google, Facebook, Twitter oraz oczywiście Jazz Foundation. Obecnie trwają prace nad wersją 2.0 protokołu, szczegóły poznać można na *http://oauth.net*.

Przed przetworzeniem każdego żądania konsument usługi sprawdza, czy dołączono do niego *access token*. Jeśli tak, oznacza to, że zarówno użytkownik, jak i konsument zostali uwierzytelnieni. Jeśli nie, konsument generuje tzw. *request token* i powiadamia użytkownika o konieczności jego autoryzacji. Najczęściej odbywa się to przez przekierowanie użytkownika do serwera - dostawcy usług. Użytkownik autoryzuje *request token* i ponawia żądanie do konsumenta. Konsument wymienia *request token* na *access token* i najczęściej odsyła go razem z odpowiedzią na zgłoszone przez użytkownika żądanie.



Rysunek 2. *Proces wymiany request tokena na access token - tzw. OAuth Dance*



Rysunek 3. *Rejestracja konsumenta OAuth*

W Jazz Foundation rolę dostawcy usług pełni Jazz Foundation Server. Konsumentami są Application Front Servery, a za zarządzanie użytkownikami odpowiada `Users API`. Dzięki użyciu OAuth JFS upraszcza autoryzację użytkowników, wymagając od aplikacji jedynie sprawdzenia obecności odpowiednich tokenów. Ponieważ wszystkie czynności aplikacja wykonuje "w imieniu" użytkownika, nie istnieje możliwość pośredniego dostępu do zasobów, które są dla danego użytkownika zabronione. Jest to korzystne nie tylko ze względu na wygodę programistów, zwiększa również bezpieczeństwo. Jeśli aplikacja - konsument padnie ofiarą włamania, napastnik nie będzie miał możliwości uzyskania z niej danych pozwalających na uwierzytelnienie się jako użytkownik.

Foundation Security Services daje możliwość zdefiniowania abstrakcyjnych akcji, w stosunku do których administrator może określać uprawnienia za pomocą centralnego interfejsu.

Aplikacja Hello World

Ilustracją powyższych wiadomości będzie prosta aplikacja typu "Hello World", która powita użytkownika.

Ponieważ aplikacja korzystać będzie z Jazz Foundation Application SDK zapakowanego w pluginy OSGi, należy utworzyć w Eclipse nowy projekt typu *Plug-in project*, o nazwie *com.example.libsample*. Należy odznaczyć opcje *This plug-in will make contributions to the UI* oraz *Generate an Activator* i wybrać J2SE-1.5 jako środowisko.

Po utworzeniu projektu należy otworzyć plik plugin.xml i wybrać zakładkę *Dependencies*. Naciskając przycisk *Add*, należy dodać elementy składające się na bibliotekę *REST Framework*:

- com.ibm.team.jfs.app
- com.ibm.team.jfs.app.http
- org.apache.http

Następnym krokiem jest zarejestrowanie nowego serwisu REST jako *extension point* pluginu *com.ibm.team.jfs.app*. W tym celu w zakładce *plugin.xml* należy dodać następujący element XML(Listing 1)

Atrybut *alias* określa, na jakie zapytania reagował będzie nasz serwis. Jeśli adres kontekstu (w sensie J2EE) aplikacji będzie *https://localhost:9444/com.example.libsample/* to serwis będzie reagował na zapytania do adresu *https://localhost:9444/com.example.libsample/library*. Istnieje też możliwość dodania pojedynczego znaku *, który zastępuje 0 lub więcej znaków. Atrybut *class* określa nazwę klasy, w której zaprogramowany jest serwis. Jest to serwlet, jednak korzystający z klasy bazowej dostarczonej przez *Rest Framework*. Atrybut *display na-*

Listing 6. *Fragment MainService.java*

```
private static String JFS_ROOT_SERVICES_URL = null;
private static String CONSUMER_KEY =null;
private static String CONSUMER_SECRET = null;

private static final String propFileName = "dw_lib_sample.properties";
private static final String consumerSecretProp =
    "com.example.libsample.consumerSecret";
private static final String consumerKeyProp =
    "com.example.libsample.consumerKey";
private static final String rootServicesProp=
    "com.example.libsample.jfsRootServicesUrl";

private void loadProperties(HttpRequest request) throws Exception {
  if (CONSUMER_KEY != null && CONSUMER_SECRET != null) return;
  IAppStaticContext context = RequestParams.getAppStaticContext
    (request.getParams());
    Properties props = ConfigUtil.getAppConfigProps(context, propFileName);
  JFS_ROOT_SERVICES_URL = props.getProperty(rootServicesProp);
  CONSUMER_SECRET = props.getProperty(consumerSecretProp);
  CONSUMER_KEY = props.getProperty(consumerKeyProp);
}
```

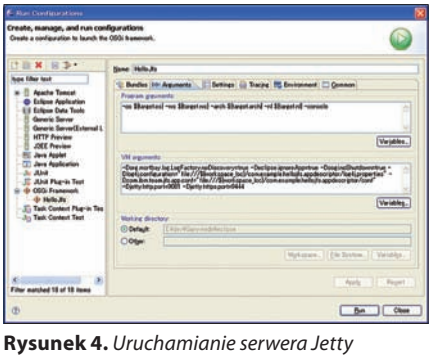
Listing 7. *Fragment MainService.java*

```
import com.ibm.team.jfs.app.discovery.utils.RootServicesUtil;
import com.ibm.team.jfs.app.oauth.OAuthConsumerProperties;
import com.ibm.team.jfs.app.oauth.OAuthProviderProperties;
[..]
private static OAuthProviderProperties provider = null;
private static OAuthConsumerProperties consumer = null;

private void loadProperties(HttpRequest request) throws Exception {
[..]
jfsRootServices = RootServicesUtil.instance(JFS_ROOT_SERVICES_URL);
provider = new OAuthProviderProperties( Config.jfsRootServices.
    getOauthRequestTokenUrl(), jfsRootServices.getOauthUserAuthorizationUrl(),
    jfsRootServices.getOauthAccessTokenUrl());consumer =
    new OAuthConsumerProperties(CONSUMER_KEY, CONSUMER_SECRET );
}
```

me definiuje opisową nazwę serwisu.

Należy dodać nową klasę MainService w paczce *com.example.libsample*. W dialogu dodawania klasy należy wybrać *com.ibm.team.jfs.app.RestService* jako klasę bazową. W utworzonej klasie należy nadpisać metodę *doGet* w następujący sposób.



Rysunek 4. *Uruchamianie serwera Jetty*

Klasa *RestService* zamienia obiekty znane ze standardowych serwletów na klasy dostarczone w bibliotece *Apache HTTP*. Serwlet ustawia kod odpowiedzi HTTP *200 - OK*, oraz odpowiada tekstem *Hello World*. Klasa *StringEntity* również pochodzi z biblioteki *Apache HTTP* i jest jedną z dostępnych metod konstruowania ciała odpowiedzi.

Test deployment descriptor i definicja kontekstu

Aplikacja stworzona za pomocą SDK archiwa typu WAR. Do działania wymagają serwera J2EE.

SDK udostępnia w tym celu zagnieżdżony serwer *jetty*. Aby z niego skorzystać, należy stworzyć *deployment descriptor* - plugin, który zawierał będzie konieczne ustawienia. Podczas dodawania nowego projektu *com.example.libsample.appdescriptor* należy zaznaczyć takie same opcje, jak podczas tworzenia pierwszego projektu.

Po utworzeniu należy otworzyć plik plugin.xml i wybrać zakładkę *Dependencies*. Naciskając przycisk *Add*, należy dodać:

- plugin z serwerem Jetty - *com.ibm.team.server.embedded.jetty*
- Equinox in a Servlet Container - *org.eclipse.equinox.http.servlet*

Następnie należy w zakładce *plugin.xml* zdefiniować kontekst, pod jakim opublikowana będzie aplikacja, dodając następujący *extension point* (Listing 3).

Atrybut *name* określa nazwę kontekstu J2EE. Atrybut *path* określa ścieżkę, z której załadowane będzie archiwum WAR.

Należy dodać w projekcie podany wyżej katalog WebContent. W nim katalog WEB-INF i zgodnie ze specyfikacją J2EE w pliku WebContent/WEB-INF/web.xml należy umieścić deskryptor aplikacji. Deskryptor powinien zdefiniować jeden serwlet: org.eclipse.equinox.http.servlet.HttpServiceServlet - Equinox Bridge Servlet, który jest pomostem, pozwalającym pluginom OSGi przetwarzać zapytania HTTP. Serwlet należy zmapować na całą przestrzeń URL (url-pattern: „/*”) .

Po skonfigurowaniu projektu należy przejść do menu *Run|Run Configuration...* i w węźle *OSGi Framework* dodać nową konfigurację (opcja *New).* Należy wyczyścić zaznaczone pluginy i wybrać jedynie:

- com.example.libsample
- com.example.libsample.appdescriptor
- com.ibm.team.jfs.app
- com.ibm.team.jfs.app.servlet

Następnie należy nacisnąć przycisk *Add Required Bundles*, aby automatycznie dodać niezbędne pluginy.

Naciśnięcie *Run* spowoduje uruchomienie serwera *jetty*. Po wpisaniu *https://localhost:9444/com.example.libsample/library* przeglądarka powinna wyświetlić napis *Hello World*.

Korzystanie z serwisów Jazz Foundation

Rejestracja aplikacji i *Application Static Context*

Opisana wyżej aplikacja korzysta z Jazz Foundation Application SDK, jednak nie komunikuje się z Foundation Serwerem. Pierwszym krokiem jest rejestracja aplikacji jako dostawcy usług. Służy do tego panel administracyjny (*https://localhost:9443/jazz/admin*, zakładka *OAuth Consumer Management).* Rejestracji podlega nazwa aplikacji: *dw_library_sample* i wartości *consumer secret* (tu dowolny ciąg

Listing 8. *Fragment MainService.java*

```
static final String ACCESS_TOKEN_PARAM_NAME = "accessToken";
static final String ACCESS_TOKEN_COOKIE_NAME=dw_libsample__access_token";
static final String ACCESS_TOKEN_SECRET_COOKIE_NAME=dw_libsample__access_token_secret";
 public boolean service(String method, HttpRequest request,
   HttpResponse response) throws IOException {
   OAuthAccessToken accessToken = null;
   if(OAuthHelper.hasAccessToken(request, ACCESS_TOKEN_COOKIE_NAME, ACCESS_TOKEN_SECRET_COOKIE_NAME)){
    accessToken = OAuthHelper.getAccessTokenFromRequest(request,
    ACCESS_TOKEN_COOKIE_NAME, ACCESS_TOKEN_SECRET_COOKIE_NAME);
   }else if(OAuthHelper.hasAuthorizedRequestToken(request)){
    OAuthRequestToken requestToken = OauthHelper.getAuthorizedRequestTokenFromRequest(request);
    accessToken = OAuthHelper.exchangeRequestTokenForAccessToken(getOAuthProviderProperties(),
        getOAuthConsumerProperties(),requestToken);
    OAuthHelper.setAccessTokenAsCookies(response, accessToken, ACCESS_TOKEN_COOKIE_NAME,
        ACCESS_TOKEN_SECRET_COOKIE_NAME, RequestParams.getContextPath(request.getParams()));
   } else{
    authorizeUser(request, response, getOAuthProviderProperties(),
        getOAuthConsumerProperties());
                        return true;
   }   request.getParams().setParameter(ServiceBase.ACCESS_TOKEN_PARAM_NAME, accessToken);
 return false; }
 protected void authorizeUser(HttpRequest request, HttpResponse response,
       OAuthProviderProperties provider, OAuthConsumerProperties consumer) throws IOException{
  OAuthRequestToken newToken = OAuthHelper.getNewRequestTokenFromProvider(provider, consumer);
   if(newToken != null){
    String authorizationURL = OAuthHelper.buildAuthorizationRedirectURL(provider,
        request.getRequestLine().getUri(), newToken);
    response.setHeader(HttpConstants.LOCATION, authorizationURL);
    response.setStatusCode(HttpStatus.SC_MOVED_TEMPORARILY);
   }
 }
```

Listing 9. *Fragment MainService.java*

```
import com.ibm.team.jfs.app.xml.binding.BindingFactory;
[..]
   private static final String RDF_XML_CONTENT_TYPE = "application/rdf+xml";
[..]
protected UserEntry getUser() throws Exception {
      UserEntry user = null;
      HttpGet request = new HttpGet(getWhoamiServiceURL());
      request.addHeader(HttpConstants.ACCEPT, RDF_XML_CONTENT_TYPE);
      URI uri = request.getURI();
      HttpResponse response = httpClient.send(new HttpHost(uri.getHost(),
       uri.getPort(), uri.getScheme()), request);

      if(response.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
         HttpEntity entity = response.getEntity();
         if(entity != null) {
            String location = EntityUtils.toString(entity).trim();
            request = new HttpGet(location);
            request.addHeader(HttpConstants.ACCEPT, RDF_XML_CONTENT_TYPE);
            uri = request.getURI();
            response = httpClient.send(new HttpHost(uri.getHost(), uri.getPort(), uri.getScheme()), request);

            if(response.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
               entity = response.getEntity();
               InputStream in = entity.getContent();
               BindingFactory bindingFactory = new BindingFactory();
               user = bindingFactory.createBinding(UserEntry.class, in);
               in.close();
            }
         }
      }
      return user;
   }
```

znaków, np. *dw_library_secret*). Wygenerowany klucz i podany sekret posłużą do uwierzytelnienia zarejestrowanej aplikacji.

Klucz i sekret konsumenta oraz adres skorowidzu usług należy zapisać w pliku *.properties*. Potrzebne będą przez cały czas, w którym aplikacja korzysta z usług JFS. Są to dane poufne, jednak dla uproszczenia, w przykładzie będą przechowywane w pliku tekstowym.

Serwlet obsługujący żądanie uzyskuje dostęp do konfiguracji za pomocą interfejsu *com.ibm.team.jfs.app.IAppStaticContext*. Podczas produkcyjnego działania serwer określa ścieżki, w których poszukiwane będą pliki konfiguracyjne. W celach testowych należy zdefiniować w zmiennej środowiskowej com.ibm.team.jfs.app.conf ścieżkę dostępu.

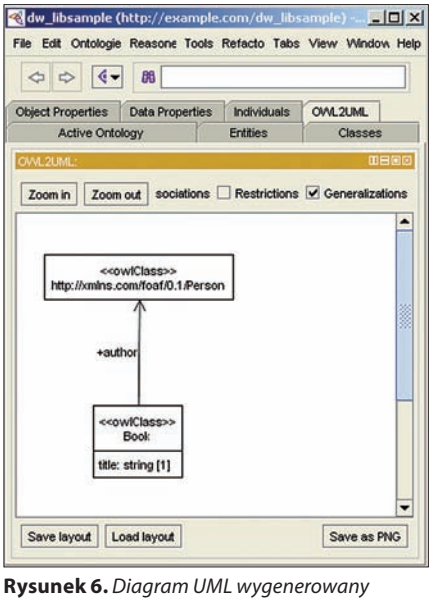
Zmienną środowiskową definiuje się w menu *Run Configurations*. Należy otworzyć utworzoną wcześniej konfigurację i w zakładce *VM Arguments* (parametry maszyny wirtualnej) dopisać:

```
-Dcom.ibm.team.jfs.app.conf=
"file:///${workspace_loc}/com.example.
 libsample.appdescriptor/conf"
```

Za pomocą klasy com.ibm.team.jfs.app.ConfigUtil możemy w prosty sposób otworzyć stworzony plik *dw_lib_sample.properties* w postaci standardowego obiektu java.util. Properties i odczytać potrzebne informacje.

Service Discovery i Jazz Authorization Service

Oprócz klucza i sekretu aplikacja musi znać adres, pod który należy skierować użytkownika w celu autoryzacji request tokena oraz adres, pod którym wymieni autoryzowany request token na access token. Obie te funkcjonalności realizowane są przez niezależne serwisy działające w modelu REST.



Rysunek 6. *Diagram UML wygenerowany z schematu RDFS*

Jazz Foundation Server w ramach Root Services Specfication publikuje dokument XML z adresami serwisów Aplikacja musi znać jedynie adres tego dokumentu, z którego odczytywane są lokalizacje potrzebnych usług. Znajduje się on w pliku dw_lib_sample.properties, który został odczytany w poprzedniej części.

Z konieczności ręcznego parsowania zwalnia klasa RootServicesUtil, będąca częścią SDK. Dostarcza ona metody odczytujące z udostępnionego przez Foundation Serwer dokumentu adresy standardowych usług. Za jej pomocą należy stworzyć instancję OAuthProviderProperties. Informacje o kluczu i sekrecie konsumenta należy zachować w klasie OauthConsumerProperties (Listing 7).

Front Server działa na serwisach *Jazz Foundation* jedynie w imieniu zalogowanego użytkownika. Konieczna jest wiec autoryzacja obsługiwanych zapytań za pomocą OAuth.

SDK zawiera klasę com.ibm.team.jfs.app.oauth.OAuthHelper, służącą do obsługi tego protokołu. Od programisty wymagane jest zdefiniowanie nazw parametrów i cookie.

Do wykonania akcji niezależnych od metody (GET, POST), takich jak autentykacja użytkowników), służy metoda service().

Jak widać, metoda *service()* wykonuje algorytm opisany wyżej w części dotyczącej protokołu OAuth. Najpierw sprawdza, czy do zapytania dołączono *cookie* z *access tokenem*. Jeśli nie, poszukiwany jest uwierzytelniony *request token*. W tym przypadku serwlet kontaktuje się z serwerem Jazz Foundation, aby wymienić *request token* na *access token*, który ustawia jako cookie. Jeśli nie znaleziono żadnego z tokenów, serwlet generuje *request token* i przekierowuje użytkownika do Foundation Serwera w celu jego uwierzytelnienia.

Jeśli uzyskano *access token*, jest on przekazywany dalej za pomocą mechanizmu *request params*. Zwrócona wartość decyduje, czy należy zakończyć przetwarzanie zapytania (wartość *true*).

Whoami Service

Ponieważ OAuth nie udostępnia informacji o uwierzytelnionym użytkowniku, do odczytania jego danych konieczny jest *Whoami Service*.

Zwraca on, w odpowiedzi na zapytanie metodą GET, URL, z którego odczytać można dane użytkownika. Uzyskanie nazwy użytkownika wymaga więc dwóch zapytań: jednego do *Whoami Service* i jednego pod wskazany *URL*. Do wykonania zapytań HTTP posłuży klasa *HttpClient* z biblioteki *Apache HTTP* (Listing 9).

Aby odczytać dane z dokumentu, należy posłużyć się biblioteką XML binding dostarczoną z SDK. Korzystanie z niej wymaga dodania nowego interfejsu: *UserEntity*.

Instancje klasy zostaną utworzone przez fabrykę *BindingFactory*, za pomocą mechanizmu generowania dynamicznych implementacji *java.lang.reflect.Proxy*. Zdefiniowane metody zwrócą wartości zgodne z wyrażeniami *XPath* zawartymi w anotacjach.

Aby powitać użytkownika, wyświetlając jego login, wystarczy w metodzie *doGet()* klasy *MainService* skorzystać z zapisanej wyżej metody *getUser()*.

Przykładowa aplikacja – lista książek

Dokumenty zwracane przez usługi Jazz Foundation nie są zwykłymi plikami XML. Jak można wnioskować po typie MIME (Listing *MainServlet.java*), są to zasoby typu RDF, co jest skrótem od Resource Description Framework.

Resource Description Framework

Dokumenty HTML są jednym z najpowszechniej używanych w Internecie mediów. Twórca strony posiada kontrolę nad warstwą wizualną dokumentu. Może np. umieścić tekst pod obrazkiem, nie ma jednak możliwości określenia, że jest to podpis. Interpretacji znaczenia

Listing 10. *Plik UserEntry.java*

```
import com.ibm.team.jfs.app.xml.binding.Namespace;
import com.ibm.team.jfs.app.xml.binding.Namespaces;
import com.ibm.team.jfs.app.xml.binding.XPathLocation;

@Namespaces({
   @Namespace(prefix="rdf", uri="http://www.w3.org/1999/02/22-rdf-syntax-ns#"),
   @Namespace(prefix="foaf", uri="http://xmlns.com/foaf/0.1/")
})
public interface UserEntry {
   @XpathLocation(path="/rdf:RDF/rdf:Description/foaf:nick")
   String getUserId();

   @XpathLocation(path="/rdf:RDF/rdf:Description/foaf:name")
   String getUserName();
}
```

dokonuje człowiek, automatyzacja tego procesu jest dość trudna. Celem projektu Sieci Semantycznej (*Semantic Web*) rozpoczętego przez W3C w 2001 roku jest dostarczenie narzędzi (głównie protokołów i standardów) do integracji wiedzy w Sieci i umożliwienie przetwarzania jej znaczenia maszynom.

Rysunek przedstawia elementy składające się na stos technologii sieci semantycznej. XML zapewnia podstawową składnię,standardowy sposób zapisu i przetwarzania dokumentów oraz elementarne typy danych (XML Schema). RDF pozwala modelować obiekty - zasoby (resources), oraz relacje pomiędzy nimi za pomocą prostych asercji i definiuje znaczenie podstawowych operatorów. RDF Schema pozwala grupować zasoby w taksonomie - klasy podobne do tych znanych z obiektowych języków programowania, oraz określać dodatkowe własności relacji pomiędzy obiektami - dziedzinę (dozwolone klasy zasobów) i krotność.

SPARQL Protocol and RDF Query Language jest odpowiednikiem SQL i służy do przeszukiwania baz danych (*triplestore*) zawierających dokumenty RDF. Więcej informacji znajduje się w części poświęconej indeksowaniu.

Elementy znajdujące się wyżej na rysunku rozszerzają możliwości automatycznego przetwarzania danych o wnioskowanie regułowe (RIF) oraz logikę wyższych rzędów (OWL). Nie będą one przedmiotem tego artykułu, więcej dowiedzieć się o nich można ze stron W3C.

Dokumenty RDF pełnią w Sieci Semantycznej podobną rolę co dokumenty HTML w sieci WWW, jednak jak napisano wcześniej ich treść stanowi wiedza, a nie dane o prezentacji.

Wiedza w RDF tworzy graf, którego węzłami są zasoby (np. strony WWW, osoby, komputery), a krawędziami właściwości, definiowane w ontologiach. Podstawową jednostką wiedzy jest trójka: podmiot, właściwość, wartość.

Model RDF posiada cechy wspólne zarówno z systemami obiektowymi, jak i relacyjnymi. Różni się on od nich przede wszystkim dwoma cechami Po pierwsze, RDF stosuje zasadę otwartego świata (*open world assumption*): brak wiedzy o istnieniu jakiejś relacji nie pozwala wnioskować o tym, że nie istnieje ona. Jest to założenie naturalne dla programów przetwarzających dane w sposób inkrementalny.

Po drugie, RDF kładzie przede wszystkim nacisk na tożsamość (identity) obiektu. Schemat (w sensie bazodanowym) danych jest drugorzędny i może się zmieniać dynamicznie. Ten sam zasób może być opisany z różnych "perspektyw"; np. osoba w jednym wypadku jako klient biblioteki, a w innym jako lokator. Ta cecha pozwala o wiele łatwiej łączyć wiedzę z niezależnych źródeł danych (np. bazy danych spółdzielni mieszkaniowej i biblioteki), w coś, co w przypadku relacyjnych baz danych wymagałoby dedykowanego skryptu, lub programu, uzależnionego od schematu obu tych baz.

Listing 11. *Zapytanie HTTP tworzące storage area*

```
    <?xml encoding="UTF-8" ?>
<StorageArea
                        xmlns="http://jazz.net/xmlns/foundation/1.0/"
                        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                        rdf:about="">
                        <rdfs:label>Sample Library Catalogue</rdfs:label>
                      <frontsideURL>https://localhost:9444/jazz/library</
                      frontsideURL>
                      </StorageArea>
```

Listing 12. *Fragment MainService.java*

```
private HttpResponse createResource(String uri, String name, String contentType,
                      String content) throws HttpException, IOException,
                      URISyntaxException, JfsHttpException {
      HttpPost request = new HttpPost(uri);

      request.addHeader("Name", name);

      StringEntity entity = new StringEntity(content, "UTF-8");
      entity.setContentType(contentType + " ;charset = UTF-8");
      request.setEntity(entity);

      return httpClient.send(request);
   }
```

Listing 13. *Przykładowy zasób typu Book w formacie XML*

```
<?xml encoding="UTF-8" ?>
<Book
  xmlns="http://example.com/dw_lib_sample"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  rdf:about="">
  <title>A Book</title>
  <author rdf:resource="https://localhost:9443/jazz/users/Test" />
</StorageArea>
```

RDF Schema i Protege

Aby zilustrować podstawowe cechy RDF, zdefiniowany zostanie model danych dla prostej listy książek. Program Protege umożliwia wykonanie tego w sposób znany z graficznych edytorów diagramów encji lub UML. Z gotowego modelu RDF Schema wygenerowany zostanie diagram UML.

Program Protege w wersji 4 można pobrać za darmo ze strony protege.stanford.edu. Aby umożliwić wygenerowanie diagramu UML, konieczny jest plugin OWL2UML.

Po uruchomieniu programu należy wybrać opcję stworzenia nowego projektu. Następnie należy podać *namespace,* w jakim znajdą się stworzone obiekty. W polu O*ntology URI należy wpisać http://example.com/dw_lib_sample.*

Aby zdefiniować nową klasę *Book*, należy przejść do zakładki *Entities.* W panelu *Asseted Hierarchy* należy wybrać element *owl:Thing* i z menu kontekstowego dodać nową podklasę. Klasa *owl:Thing* pełni rolę analogiczną do *java.lang.Object* w Javie - jest nad-klasą wszystkich klas. W dialogu należy podać nazwę klasy: *Book*, która zostanie rozwinięta do *http://example.com/dw_lib_sample#Book.*

Następnym krokiem będzie zdefiniowanie właściwości (*property*) – tytułu. W panelu *Data Property Hierarchy* należy dodać nową *datatype property*. W dialogu, który się pojawi, należy podać nazwę: *title*. W panelu po prawej można określić dziedzinę (*Domains*) - zdefiniowaną wcześniej klasę *Book*, przeciwdziedzinę (*Ranges*) - ciągi znaków, czyli typ *string*, oraz zaznaczyć, że właściwość

jest typu funkcyjnego (*functional)*, tzn. jej kardynalność to N:1.

Oprócz własności, pozwalających przypisać obiektom literały, istnieją też własności pozwalające powiązać dwa obiekty. Tego typu relacja potrzebna będzie, aby przypisać książce autora. Dziedziną nowej własności (*object property*) będzie w tym wypadku również klasa *Book*.

Zamiast definiować od podstaw klasę określającą osobę – autora, posłużymy się istniejącą ontologią *Friend of a Friend*, definiującą pojęcia potrzebne do opisu osób i relacji między nimi. Aby dokonać importu ontologii, należy przejść do zakładki *Active Ontology* i w panleu *Ontology Imports* wybrać *Direct imports.* W kreatorze, który się pojawi, należy wybrać źródło (plik lub URL, ontologie w) i załadować plik *.owl* z ontologią *FOAF.* Następnie należy powrócić na zakładkę Następnie należy powrócić na zakładkę *Properties* i określić przeciwdziedzinę właściwości *author* jako *foaf:Person*.

Po wykonaniu tych kroków na zakładce OWL2UML powinien pojawić się schemat UML podobny do widocznego na poprzednim rysunku.

Jazz Storage API

Storage service daje klientom możliwość składowania danych w formacie RDF. Nie jest to obowiązkowy format danych, jednak jego użycie jest zalecane. Pierwszym krokiem koniecznym, aby skorzystać z tej funkcjonalności, jest stworzenie *storage area* dla aplikacji.

Storage area to fragment przestrzeni nazw (URI), przydzielony do prywatnego użytku aplikacji. Należy zwrócić uwagę, że przestrzeń nazw, mimo że jest hierarchiczna, nie posiada cech pojemników znanych np. z systemów plików – istnienie */foo/bar* nie oznacza, że zasób */foo* również istnieje.

Aby stworzyć *storage area,* należy wysłać metodą POST na adres *Storage service* następujący dokument RDF (Listing 11).

StorageArea, a właściwie *http://jazz.net/xmlns/foundation/1.0/StorageArea* (co wynika z deklaracji xmlns dokumentu) określa typ zasobu, który opisuje dokument. Standardowy element rdfs:Label określa opisową nazwę, pod jaką *storage area* będzie widoczny. *Storage area* dostępny jest jedynie dla aplikacji, która go stworzyła. Klienci aplikacji widzą jej zasoby w zewnętrznej przestrzeni nazw (np. *localhost:9444/library/book-123* a nie *localhost:9444/jazz/storage/library/book-123*). Podanie elementu *frontsideURL* uwalnia aplikację od konieczności przepisywania wewnętrznych URI na zewnętrzne w dokumentach zwracanych przez Storage serivce.

Na taki dokument *Storage service* powinien zwrócić odpowiedź ze statusem HTTP 201 (Created). W nagłówku Location znajduje się adres stworzonego zasobu (Listing 12). Na listingu widać, w jaki sposób wysłać zapytanie typu POST. Adres *Storage API* pozyskać można z klasy *RootServicesUtil* opisanej w części 3. Wartość *contentType* powinna określać typ RDF, tzn: *application/rdf+xml,* nazwa może być dowolna, a treść (*content*) stanowi plik XML widoczny na listingu powyżej.

Do stworzonego *storage area* można dodawać zasoby. Będą to instancje zdefiniowanej wyżej klasy *Book* (Listing 13).

Wartości właściwości klasy zapisane są jako zagnieżdżone elementy. Wartość *title*, która jest typu datatype property zapisana jest jako literał wewnątrz elementu o takiej samej nazwie. Wartość *author* określona jest za pomocą specjalnego atrybutu *rdf:resource*, który wskazuje na URI obiektu - osoby, która jest autorem. Użytkownicy zarejestrowani w Jazz Foundation opisani są za pomocą ontologii FOAF, jak można sprawdzić na listingu klasy *User Entry.java*. Zalogowany użytkownik jest obiektem typu *foaf:Person*, jest wiec poprawną wartością własności *author*. Oprócz wskazywania URI obiekt będący wartością właściwości można zapisać wewnątrz elementu.

Dodanie nowego zasobu ilustruje poniższy listing. Metoda *createBook()* korzysta z metody *createBookContent()*, która powinna zwrócić dokument (jako ciąg znaków - *String)* widoczny na listingu powyżej. Metoda *getUserURI()* zwraca URI użytkownika, pobrany z *Whoami Service* (por. cz. 3).

Aby umożliwić użytkownikom dodawanie książek, konieczna jest jeszcze implementacja metody *doPost()* w serwlecie *Main Service,* która wywoła *createBook().*

Listing 14. *Fragment MainService.java*

```
private void createBook() throws HttpException, IOException, URISyntaxException,
                    JfsHttpException{
      String storageAreaURL = getStorageServiceURL()+ "/"+ Constants.STORAGE_
                    AREA_NAME;
      createResource(storageAreaURL,
            "Book",
            Constants.RDF_XML_CONTENT_TYPE,
            createBookContent("A Book", getUserURI()));
   }
```

Listing 15. *Zapytanie SPARQL*

```
PREFIX ls: <http://example.com/dw_lib_sample>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title ?authorName ?creator
WHERE {
 ?book ls:title ?title .
 ?book ls:author ?author .
 ?book dc:creator ?creator .
 ?author foaf:name ?authorName .
 FILTER (?author = <https://localhost:9443/jazz/users/Test>)
}
```

Listing 16. *Fragment MainService.java*

```
public String query(String user) throws Exception{
      String query =  URLEncoder.encode( getQueryString(user),  "UTF-8");
      HttpGet request = new HttpGet(getQueryServiceURL() + "?" + query);
      request.addHeader(HttpConstants.ACCEPT, "applicatioStorage Servicen/
                    atom+xml");
      HttpResponse response = httpClient.send(request);
      int status = response.getStatusLine().getStatusCode();
      if (status == HttpStatus.SC_OK) {
HttpEntity entity = response.getEntity();
return EntityUtils.toString(entity)
}
      return null;
   }
```

Jazz Query API

Jazz Query API umożliwia przeszukiwanie zasobów zapisanych do *Storage API*. Zapytania formułowane są w języku *SPARQL*. Poniższy listing przedstawia zapytanie o książki, których autorem jest użytkownik Test. Zapytanie zwraca tytuły książek, imię i nazwisko autora (zdefiniowane w ontologii *FoaF*) oraz identyfikator użytkownika, który utworzył zasób. Ta ostatnia informacja dodawana jest przez *Storage API* automatycznie do każdego zasobu. Właściwość *creator* zdefiniowana jest w ontologii *Dublin Core Metadata* (Listing 15).

Zapytanie SPARQL składa się z kilku klauzul. Klauzula *PREFIX* służy do określenia skrótów przestrzeni nazw. W zapytaniach SPARQL adresy URL zapisuje się pomiędzy znakami "<" i ">".

Klauzula *SELECT* służy do określenia wartości, jakie zapytanie ma zwrócić. Znak "?" przed nazwą identyfikuje zmienną, która musi zostać związana (*bound*) w klauzuli *WHERE*. Zamiast *SELECT* użyć można klauzuli *DESCRIBE*, co spowoduje zwrócenie opisu (wybranych przez serwer właściwości) wymienionych zasobów. Trzeci wariant - *ASK* służy sprawdzeniu, czy istnieją obiekty opisane w klauzuli *WHERE*, bez zwracania wartości.

Klauzula *WHERE* pozwala na określenie warunków zapytania. Każdy warunek składa się z trzech elementów. Pierwszy po lewej stronie jest obiekt, podany jako zmienna lub URL, np. ?book. Po obiekcie następuje właściwość oraz wartość, jaką ma ona przyjąć. W miejsce wartości podać można zmienną, obiekt lub, dla właściwości typu *datatype property* literał. Operator *FILTER* pozwala ograniczyć wyniki zapytania na różne sposoby, np. przez relacje równości i porządki oraz wyrażenia regularne.

Aby umożliwić użytkownikom przeszukiwanie zapisanych książek, należy zmienić metodę *doGet()* tak, aby wywoływała widoczną na listingu metodę *query()* z parametrem odczytanym z URLa. Wyniki zwrócone zostaną w formacie *ATOM*. Można je odczytać za pomocą *BindingFactory* użytej w części poświęconej *Whoami Service*, lub przekazać użytkownikowi w takiej formie.

Domyślnie wszystkie właściwości składowanych dokumentów RDF są indeksowane. W praktyce, aby nie przeciążać bazy danych, należy ograniczyć indeksowanie jedynie do istotnych dla wyszukiwania właściwości. Można to zrobić, wysyłając odpowiedni dokument RDF do Query API. Korzysta się tu z faktu, że definicje właściwości są pełnoprawnymi obiektami RDF. SDK zawiera bibliotekę Query Lib, która pozwala tworzyć proste zapytania, w rodzaju *wlasciwosc1 = wartość AND wlasciwosc2 < wartość*, bez znajomości SPAQL. Dodatkową zaletą jest to, że zapytania w takiej formie można zapisać jako link i np. zachować jako zakładkę.

Podsumowanie

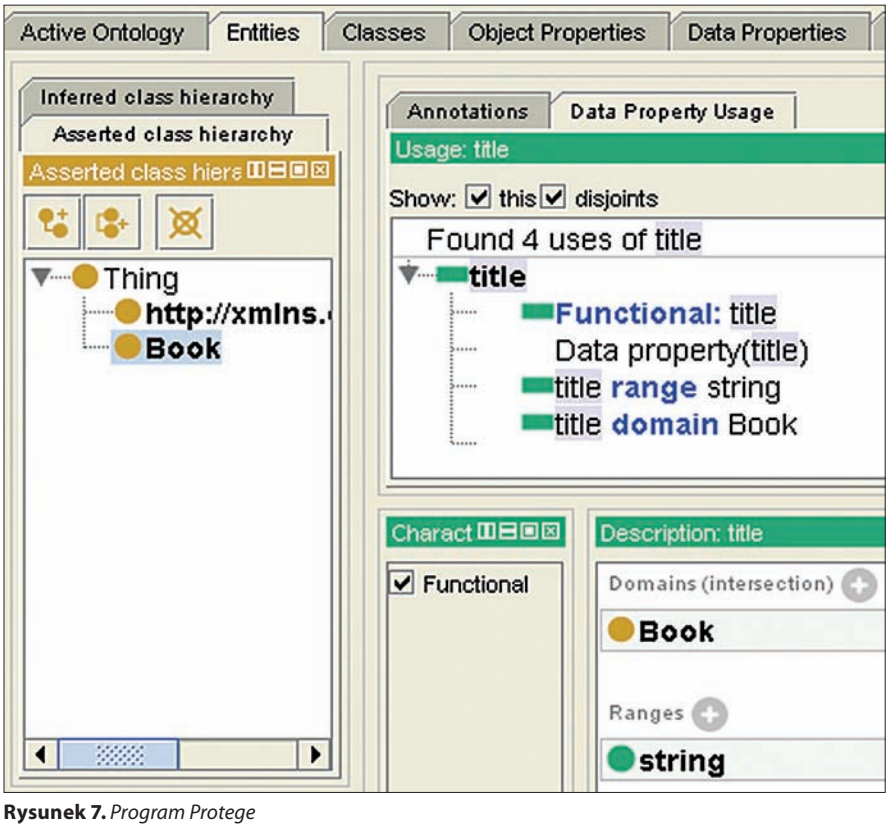
Niestety objętość artykułu nie pozwala opisać dokładnie wszystkich funkcjonalności dostępnych dla stworzonej aplikacji. Storage API dostarcza mechanizm zapobiegający równoległym zmianom oparty o HTTP. Każde żądanie HTTP modyfikujące istniejący dokument musi zawierać nagłówek *ETag* lub *Last-Modified*. Jeśli podana wartość wskazuje na nieaktualną wersję bazową, modyfikacja zostanie odrzucona.

Change API pozwala agregować tworzone i zmieniane dokumenty w transakcje, które następnie aplikowane są w postaci atomowej operacji. History API przechowuje informacje o zmianach dokonanych na składowanych dokumentach i pozwala odtworzyć ich stan w wybranym czasie.

Centralne przechowywanie danych ułatwia backup (służy do tego narzędzie repotools) oraz eksport do hurtowni danych.

Znakomitym wprowadzeniem do RDF jest strona www.rdfabout.com. Oprócz przystępnego wprowadzenia udostępnia też przykładowe zbiory danych (np. spis powszechny w USA, kilka milionów krotek) dostępnych do ściągnięcia oraz przeszukania za pomocą SPARQL. Naturalnie wszystkie specyfikacje znaleźć można na stronach W3 Consortium.

Więcej informacji oraz przykładowe aplikacje korzystające z Jazz Foundation Application SDK można znaleźć na wiki jazz.net. Warto dowiedzieć się więcej, zwłaszcza że Jazz Foundation Server razem z Rational Team Concert dostępny jest z licencją dla 10 użytkowników za darmo.



Rysunek 7. *Program Protege*

W sieci:

- Jazz.net wiki – Jazz Foundation Application SDK: *https://jazz.net/wiki/bin/view/Main/ServerSDKMain*
- Jazz.net wiki – Storage AP: *https://jazz.net/wiki/bin/view/Main/JFSStorageAPI*
- Jazz.net wiki – Index Query API: *https://jazz.net/wiki/bin/view/Main/JFSIndexStoreQueryAPI*
- Jazz.net wiki – biblioteka Query Lib pozwalająca na formułowanie prostych zapytań bez znajomości SPARQL: *https://jazz.net/wiki/bin/view/Main/JFSApplicationQueryLib*
- Jazz.net wiki – określenie indeksowanych właściwości: *https://jazz.net/wiki/bin/view/Main/JFSRdfIndexingRules*
- Dokument OWL z ontologią FOAF: *http://attempto.ifi.uzh.ch/site/docs/OWL_to_ACE/foaf.owl*

Jakub Bocheński

Obecnie pracuje jako programista w firmie genijusz, świadczącej usługi z zakresu integracji systemów informatycznych z wykorzystaniem nowoczesnych rozwiązań internetowych. Ukończył specjalizację Inteligentne Systemy Wspomagania Decyzji na Politechnice Poznańskiej, pisząc pracę magisterską n.t. implementacji systemów wspomagania decyzji na platformie Jazz.

Publikacje IBM Redbooks
źródło wiedzy i doświadczenia

IBM

Kolaboracyjne zarządzanie cyklem życia aplikacji przy pomocy produktów z rodziny IBM Rational

ibm.com/redbooks

Redbooks

W niniejszej publikacji, będącej częścią serii IBM® Redbooks®, prezentujemy schemat Kolaboracyjnego Zarządzania Cyklem Życia Aplikacji (ang. Collaborative Application Lifecycle Management, CALM) i pokazujemy jak produkty z rodziny IBM Rational® wspierają ten dynamicznie rozwijający się rynek.
Kierując się biznesowymi wymaganiami związanymi z dostarczaniem oprogramowania w skali globalnej, wiele dużych organizacji szuka porady i pomocy przy wdrażaniu zwinnych metod zarządzania projektami.

W niniejszej książce przedstawiamy referencyjny scenariusz oraz architektury narzędzi, które pozwalają wdrażać produkty z rodziny IBM Rational w istniejących środowiskach biznesowych.

ibm.com/redbook

Referencja Klienta IBM

Turkiye Is Bankasi A.S.

Czołowy, komercyjny bank turecki wprowadza innowacyjne rozwiązania związane z rozwojem oprogramowania, uzyskując znaczną redukcję czasu potrzebnego na realizację projektów oraz usprawnienie zarządzania, wliczając w to konsolidację 100 istniejących skryptów w jeden. Wszystko to dzięki wsparciu ze strony zespołu IBM Software Services dla Rational i zastosowaniu oprogramowania IBM Rational Build Forge Enterprise Edition, IBM Rational ClearCase oraz IBM Rational ClearQuest.

Lokalizacja: Istambuł, Turcja
Branża: Bankowość
URL: *http://www.isbank.com.tr/English/*

Informacje o kliencie:
Turkiye Is Bankasi A.S. jest tureckim bankiem komercyjnym, który oferuje swoim klientom detaliczne i komercyjne produkty oraz usługi bankowe.

Turkiye Is Bankasi A.S. jest siódmym w kolejności bankiem w Turcji, w kontekście ilości posiadanych zasobów i plasuje się na czwartej pozycji w odniesieniu do wielkości sieci oddziałów, której rozmiar liczy 586. Turkiye Is Bankasi A.S posiada co najmniej jeden oddział w każdym tureckim mieście.

Potrzeba biznesowa:
W celu ugruntowania swojej liderskiej pozycji na globalnym rynku bankowym, Turkiye Is Bankasi A.S. stosuje zaawansowane oprogramowanie oraz architekturę sprzętową, umożliwiające oferowanie wsparcia dla zdalnych operacji bankowych realizowanych w czasie rzeczywistym, a także szereg opcji bankowej samoobsługi. Architektura tego rozwiązania opiera się na zrębie aplikacji zbudowanym na bazie technologii Java 2 Enterprise Edition, obsługującym ponad sto produkcyjnych aplikacji. Ze względu na bardzo dużą złożoność tego systemu, bank wykorzystywał ponad stu programistów, którzy aktualizowali, dostosowywali i utrzymywali wspomniane wyżej oprogramowanie.

Niestety, przy stosowaniu takiego podejścia bank miał poważne problemy związane z konfiguracją i wdrażaniem swojego oprogramowania. Między innymi zespół projektowy borykał się z koordynacją ponad dwustu zakodowanych na stałe skryptów budujących, co przy braku automatyzacji owocowało bardzo dużym narzutem związanym z zarządzaniem budowanymi pakietami oprogramowania. W związku z powyższym bank cierpiał z powodu następujących niedogodności:

- Zbyt dużo czasu było tracone na przygotowywanie kolejnego pakietu oprogramowania (ze względu na manualny charakter tego procesu trwało to średnio około jednego dnia),
- Pojawiła się potrzeba zdefiniowania procesów budowania i wdrażania nowych wersji oprogramowania ze względu na wymogi związane z zarządzaniem jakością,
- System był mocno zależny od manualnej synchronizacji procesu tworzenia oprogramowania,
- Tworzenie nowych widoków stało się bardzo trudne,
- Tworzone oprogramowanie bardzo słabo nadawało się do ponownego użycia, niemożliwym stało się odtwarzanie historii zmian w nim zachodzących.

Rozwiązanie:
Turkiye Is Bankasi A.S. koordynowany przez IBM Software Services dla Rational, rozpoczął badanie, uaktualnianie oraz ulepszanie swoich procesów projektowych. Na początek zespół IBM ocenił procesy budowania, wersjonowania, wdrażania i konfiguracji tworzonego oprogramowania, co pozwoliło wykryć luki w tych procesach i usunąć zbędne, tudzież niewłaściwie skonstruowane procedury. Następnie zespół usług softwarowych IBM zbudował i wdrożył pilotażowy projekt wykorzystujący oprogramowanie IBM Rational.

Widząc sukces osiągnięty w tych pierwszych krokach, klient zdecydował się wdrożyć oprogramowanie IBM w swoim środowisku produkcyjnym. Zespół IBM przeniósł sto skryptów narzędzia Ant na platformę Rational Build Forge Enterprise Edition, jednocześnie je konsolidując. W ten sposób, w przeciągu miesiąca powstał pojedynczy, parametryzowany skrypt służący do wdrażania budowanego oprogramowania. W dalszej kolejności IBM rozszerzył architekturę systemu o powiadomienia, synchronizację oraz obsługę wielowątkowości, dzięki czemu uzyskano w pełni zautomatyzowany przepływ pracy zintegrowany z architekturą Rational Build Forge.

Aby uzupełnić poczynione zmiany, bank dodatkowo wdrożył oprogramowanie IBM Rational ClearCase i IBM Rational ClearQuest, przeprowadzając w ciągu dwóch tygodni restrukturyzację procesów zarządzania zmianami w oprogramowaniu w celu integracji nowych aplikacji. Nowe części systemu wniosły takie możliwości jak raportowanie bilansu materiałów oraz możliwość podglądu całego procesu i przeglądania zunifikowanej dokumentacji dotyczącej budowanych pakietów oprogramowania.

Zalety rozwiązania:
Dzięki zastosowaniu oprogramowania IBM Rational, Turkiye Is Bankasi A.S. zabezpieczył i w dużej mierze zautomatyzował przepływy pracy związanej z procesami wytwarzania oprogramowania oraz z zarządzaniem zmianami zachodzącymi w tym oprogramowaniu. Dzięki temu organizacja zaoszczędziła na czasie przygotowywania paczek oprogramowania, redukując go z okresu jednego dnia do kilku minut. Dzięki temu bank mógł zwiększyć liczbę rozwijanych w danym czasie projektów. Oprócz tego wprowadzenie automatyzacji pomogło zredukować liczbę nadużyć operacyjnych oraz usprawnić częstotliwość oraz dokładność raportów wdrożeniowych.

Ponadto, oprogramowanie Rational pomaga:

- Konsolidować operacje, dzięki konwersji stu lub więcej zakodowanych na stałe skryptów w jeden, parametryzowany, skrypt, nadający się do wielokrotnego użycia.
- Automatyzować procesy wdrożenia.
- Synchronizować powiadomienia dla programistów oraz poprawiać komunikację.
- Łatwo rozprzestrzeniać informacje wśród personelu dzięki dostępności uproszczonych mechanizmów tworzenia widoków.
- Usprawniać przejrzystość oraz możliwość śledzenia zmian w projekcie dzięki gromadzeniu historycznych danych opisujących procesy wytwarzania oprogramowania.
- Redukować liczbę potencjalnych nadużyć.
- Upraszczać reprodukcję i powtórne użycie istniejących pakietów oprogramowania.
- Szybciej i pewniej rozwiązywać problemy związane z tworzeniem pakietów oprogramowania.

Referencja Klienta IBM

Agencja ds Systemów Informacyjnych

oraz Rozwoju Technologii Informacyjnych (APIS IT)

Rządowa agencja rozwoju z Europy Wschodniej jest w stanie stworzyć i obsłużyć system do głosowania dla obywateli mieszkających poza granicami państwa przy użyciu aplikacji Web 2.0, opracowanej za pomocą oprogramowania IBM Rational Team Concert Standard, IBM Rational Developer oraz IBM Rational AppScan

Lokalizacja: Zagrzeb, Chorwacja (Hrvatska)
Branża: Administracja
URL: *http://www.apis-it.hr/en/index.html*

Client Background

Agencja ds Systemów Informacyjnych oraz Rozwoju Technologii Informacyjnych (APIS IT) jest agencją rozwojową założoną w 2005 roku na mocy umowy pomiędzy rządem Republiki Chorwackiej, a władzami miasta Zagrzeb. APIS IT wspiera rząd w rozwoju strategii eBiznesowej, buduje i rozwija ogólnodostępną infrastrukturę technologii informacyjnych i komunikacyjnych (ICT) oraz promuje najlepsze praktyki tworzenia systemów informatycznych, ochronę danych osobowych oraz rozwój ogólnodostępnych, elektronicznych serwisów oferujących dostęp do rządowych zasobów informacyjnych.

Zapotrzebowanie biznesowe

Agencja ds Systemów Informacyjnych oraz Rozwoju Technologii Informacyjnych potrzebuje aplikacji bazującej na standardach Web 2.0, stanowiącej rozszerzenie systemu obsługi wyborów, która pozwoliłaby przetwarzać głosy obywateli Chorwacji mieszkających poza granicami kraju. Rozwiązanie to powinno być:

- Bezpieczne, jako że ma być przeznaczone do obsługi procedur wyborczych, obłożonych rygorystycznymi wymaganiami dotyczącymi bezpieczeństwa.
- Przysposobione do funkcjonowania w istniejącej infrastrukturze IT, wykorzystywanej wcześniej do utrzymywania aplikacji Web 1.0, pisanych w języku Java i bazujących na silniku Spring (Spring to napisane w języku Java narzędzie służące do tworzenia i uruchamiania aplikacji biznesowych).

Rozwiązanie

Zespół Enterprise Generation Language (EGL) z Agencji ds Systemów Informacyjnych oraz Rozwoju Technologii Informacyjnych rozpoczął pracę nad aplikacją do głosowania po kilku miesiącach eksperymentowania z innymi aplikacjami. Programiści łatwo przerzucili się na technologię webową, dzięki temu iż od początku pracowali ze znajomym językiem EGL.

Proces tworzenia oprogramowania był realizowany przy użyciu oprogramowania Rational Business Developer V7.5.1.3 oraz platformy IBM Jazz. Oprogramowanie IBM Rational Team Concert Standard V2.0 zostało wykorzystane do wspomagania pracy zespołowej. Oprogramowanie Rational Team Concert było głównym rozwiązaniem wspomagającym zarządzanie konfiguracją oraz pracę zespołową. Oprogramowanie Rational Business Developer V7.5.1.3 wykorzystano w celu zbudowania cechującej się bogatym interfejsem aplikacji Web 2.0. Klient użył tego oprogramowania w celu budowania serwisów webowych wykorzystujących w nowym środowisku predefiniowane serwisy IBM Customer Information Control System (CICS). Ponieważ bezpieczeństwo było jednym z głównych wymagań narzuconych tworzonej aplikacji, zastosowano oprogramowanie IBM Rational AppScan Standard Edition V7.9. Pozwoliło to przeprowadzić wstępne testy bezpieczeństwa w celu identyfikacji potencjalnych zagrożeń przez zmienić na przed upublicznieniem aplikacji i w rezultacie – zapobiec wystąpieniu takich zagrożeń w przyszłości, co mogłoby negatywnie wpłynąć na wiarygodność wyników wyborów.

Architektura IT rozwiązania została niemalże w całości skompletowana z produktów IBM. Można tu wymienić takie oprogramowanie jak IBM Tivoli Access Manage, IBM HTTP Server oraz IBM WebSphere Application Server. Jeśli chodzi o rozwiązanie bazodanowe to wybór padł na IBM DB2 dla z/OS. Aplikacja została wdrożona na serwerze aplikacji WebSphere Application Server 6.1, zainstalowanym pod Microsoft Windows. Dane aplikacji obsługiwane były przez bazę danych DB2 dla z/OS Version 8.1 na System z10 EC.

Zalety rozwiązania

Reakcje APIS IT odnośnie oprogramowania firmy IBM były bardzo pozytywne:

- Za jego pomocą udało się dostarczyć na czas bezpieczny, oparty na standardach Web 2.0 system zdalnego głosowania, bez wykroczeń poza założony budżet.
- Pomogło programistom pracować bardziej efektywnie, pozwoliło zredukować czas i koszt wytworzenia oprogramowania.
- W porównaniu do poprzedniego rozwiązania, wprowadziło nową jakość w dziedzinie graficznego interfejsu aplikacji.
- Zaowocowało stworzeniem oprogramowania o bardzo wysokim stopniu niezawodności: w trakcie jego użytkowania nie pojawiły się żadne skargi użytkowników, zaś serwerowa część aplikacji generowała bardzo małe obciążenie, co wyeliminowało problemy związane z wydajnością.

Co czyni to rozwiązanie lepszym:

- Pozwala zidentyfikować potencjalne zagrożenia przed upublicznieniem aplikacji, co pozwala zespołowi IT wyłapać i usunąć potencjalne luki w zabezpieczeniach, które mogłyby zaszkodzić bezpieczeństwu wyborów.
- Pozwala zamienić zestaw niezależnych aplikacji i szereg biurokratycznych procedur na jedno, zintegrowane rozwiązanie oparte na standardach Web 2.0.
- Posiada zintegrowany mechanizm rejestracji głosujących co gwarantuje to, że w głosowaniu wezmą udział tylko osoby do niego uprawnione.

Dodatkowe infomracje od smarter planet:

- Inteligentny: Ponieważ bezpieczeństwo jest w tym przypadku kluczowym aspektem,

zastosowane rozwiązanie identyfikuje potencjalne zagrożenia przed upublicznieniem aplikacji, co pozwala zespołowi IT wyłapać i usunąć potencjalne luki w zabezpieczeniach, które mogłyby wpłynąć na wiarygodność wyników wyborów.

- Połączony: Dzięki integracji procesu rejestracji głosujących z procesem głosowania, zbudowane rozwiązanie zapewnia że jedynie uprawnieni, zarejestrowani użytkownicy mogą oddawać głosy, co w efekcie wpływa bardzo pozytywnie na wiarygodność wyników wyborów. Wszystkie informacje dotyczące procesu są zintegrowane, poczynając od daty rejestracji, poprzez oddanie głosu, kończąc na obliczeniu ostatecznego wyniku głosowania.
- Zinstrumentalizowany: rozwiązanie zlicza każdy głos oddany przez obywatela Chorwacji mieszkającego poza granicami kraju i przekazuj te informacje do narodowego systemu wyborczego w celu finalnego zliczenia głosów i obliczenia ostatecznego wyniku wyborów. Użytkownicy ocenili graficzny interfejs zastosowany w aplikacji jako najbardziej intuicyjny i najprostszy w użyciu – w porównaniu ze wszystkimi poprzednim rozwiązaniami. Reakcje ze strony zespołu APIS odnośnie oprogramowania IBM były bardzo pozytywne; użytkownicy tego oprogramowania IBM były bardzo pozytywne; użytkownicy tego oprogramowania raportowali, iż z jego pomocą mogli pracować bardziej efektywnie.

Cytat klienta:
Dzięki zdecydowaniu się na używanie EGL RichUI byliśmy w stanie szybko reagować na żądania użytkowników, nawet te pojawiające się w ostatniej chwili. EGL RichUI podniosło poprzeczkę jakości dla wszystkich aplikacji budowanych w naszej firmie, w których liczy się wygoda obsługi przez użytkownika końcowego oraz możliwość szybkiego ich wytworzenia.

Dzięki EGL RichUI udało nam się uzyskać olbrzymi postęp w dziedzinie wygody obsługi oraz produktywność użytkownika, bez potrzeby rezygnowania z bezpieczeństwa i niezawodności aplikacji. Co ważne, uzyskanie tak dobrego efektu wymagało jedynie pobieżnego przeszkolenia programistów CICS.

EGL RichUI pozwoliło naszemu zespołowi szybko budować bezpieczne i nowoczesne aplikacje Web 2.0 bez potrzeby szkolenia naszych programistów w dziedzinie technologii webowych. W efekcie pozwoliło nam to o wiele sprawniej wykonywać naszą pracę.

EGL RichUI pozwoliło nam zbudować bezpieczną, wizualnie atrakcyjną i – co najważniejsze, nowoczesną aplikację webową, którą bez problemu udało się zintegrować z istniejącą architekturą używaną przez inne nasze aplikacje webowe, oparte na technologiach JEE/Spring. Dzięki temu uniknęliśmy potrzeby tworzenia i konfigurownia nowego środowiska produkcyjnego.

STWORZYMY DLA CIEBIE BRAKUJĄCY ELEMENT

- integracja z systemami SAP
- aplikacje przeglądarkowe pod SAP i AS/400
- programowanie Java, ABAP
- technologie sieciowe: WebServices, GWT

INTEGRACJA SYSTEMÓW INFORMATYCZNYCH

Genijusz Dominik Zalewski
dominikz@genijusz.com +48 692 48 48 01 genijusz.com

genijusz

Naszą pasją jest podnoszenie efektywności zespołów projektowych

Wspieramy Dostawców systemów informatycznych w realizacji projektów informatycznych od etapu negocjacji do wdrożenia systemu informatycznego:

- dostosowujemy metodykę projektowania Rational Unified Process do potrzeb projektu informatycznego,
- dobieramy wsparcie narzędziowe do potrzeb Dostawcy i ze względu na jego uwarunkowania,
- szkolimy zespół Dostawcy z dostosowanej metodyki,
- wspieramy Dostawcę w trakcie realizacji projektu.

Przygotowujemy Zamawiających systemy informatyczne do efektywnego zarządzania wymaganiami, przeprowadzenia testów oraz odbioru zrealizowanego systemu.

RightSolution™ Autoryzowane Centrum Edukacyjne IBM Rational oferuje szkolenia Rational Unified Process (RUP) w zakresie:

- procesu wytwórczego RUP,
- modelowania biznesowego,
- zarządzania wymaganiami,
- analizy i projektowania obiektowego z UML,
- projektowania i implementacji systemów informatycznych w architekturze SOA,
- testów,
- zarządzania zmianą,
- zarządzania projektami.

Authorized IBM Training

www.rightsolution.pl