Symulacja złożonego wdrożenia systemu IT

Łukasz LASZKO, Andrzej STASIAK

Instytut Teleinformatyki i Automatyki WAT,
ul. Gen. S. Kaliskiego 2, 00-908 Warszawa
l.laszko@ita.wat.edu.pl, a.stasiak@ita.wat.edu.pl

STRESZCZENIE: W artykule przedstawiono opis zaproponowanej przez autorów symulacyjnej metody weryfikacji i walidacji planów wdrożenia, która na etapie projektowania systemu ułatwia zrozumienie jego dynamicznych aspektów oraz pozwala na wczesne wykrycie błędów projektowych i architektonicznych bez konieczności istnienia rzeczywistych produktów wdrożenia i docelowego środowiska działania. Metoda bazuje na wykorzystaniu topologii (jako planu wdrożenia) oraz języków: UML (opisu środowiska wdrożenia), OCL (opisu ograniczeń) i UAL (do opisu imperatywu semantyk akcji). Środki te pozwoliły na zbudowanie precyzyjnych modeli, zarówno planu wdrożenia, jak i symulacyjnego środowiska do jego badania. Symulację w tej metodzie wykorzystano do badania planów wdrożenia jako środek ich ewaluacji.

SŁOWA KLUCZOWE: symulacja, wdrożenie, model topologii, UML, UAL, OCL.

1. Wprowadzenie

Adekwatny opis planowania wdrożenia przedstawiony w [1], opierający się na koncepcji MDD [2], jest przewodnikiem dla architektów i wdrożeniowców służącym do skutecznej realizacji architektury złożonego systemu IT.

Wniosek płynący z tej pracy, mówiący, że nie warto czekać na implementację wdrożenia, aby zweryfikować decyzje architektoniczne jego dotyczące, to motywacja do planowania wdrożenia w sposób zaproponowany przez autorów głównie dlatego, że opisany sposób budowania planu jest niezależny od stosowanej metodyki wytwórczej i może być stosowany zarówno w popularnych metodykach ciężkich [3], jak i lekkich (np. zwinnych) [4][5].

W pracy[1] zakładamy, że opracowane adekwatne modele systemu mogą być wykorzystane zarówno do sprawdzenia poprawności produktu, jak i samych

[1] Artykuł powstał w ramach realizacji grantu nr RMN/699/2012 (Wydział Cybernetyki WAT).

modeli. Dlatego w niniejszym artykule przedstawimy opisy procesu odbioru zarówno: systemu (jego wdrożenia), jak i modeli wdrożenia – stawiając tezę, że aby zmniejszyć liczbę błędów w procesach projektowania i realizacji (przeprowadzania) wdrożenia, skuteczne jest jego planowanie oraz weryfikacja i walidacja topologii (planu wdrożenia), która poprzedza proces testowania wdrożenia.

Niniejszy artykuł jest kontynuacją treści przedstawionych w [1] oraz rozwinięciem idei symulacji modeli opracowanej w [6]. Przedstawiona zostanie w nim metoda symulacyjnego sprawdzania poprawności opracowanych planów wdrożenia oraz ich realizacji (w procesie testowania).

Określenie poprawności modeli, tradycyjnie szacowane przez aksjomaty i reguły zależności w logice temporalnej [7] lub w procesach inspekcji [8], w tej pracy zostanie przeprowadzone:

a) w procesach weryfikacji i walidacji[2]: jako opis eksperymentów symulacyjnych potwierdzających zgodność modeli architektury systemu i jej wdrożenia (opisanych przez modele UML) ze specyfikacją przedstawioną w planie wdrożenia (model topologii wdrożenia);
b) w procesie testowania: jako raport z procesów testowania wdrożenia systemu „Jazz" (szerzej przedstawionego w [1]).

Jak z tego wynika, zarówno planowanie wdrożenia – prowadzące do opracowania modeli topologii i samo wdrożenie, którego produktem jest działający system, podlegają sprawdzeniu w celu oszacowania ich jakości. Modele weryfikuje się pod kątem zasad budowania modeli i spełnienia przez te modele ograniczeń oraz się je waliduje, tzn. potwierdza się ich zgodność z wymaganiami [9]. Produkt natomiast, jako wdrożenie systemu, jest testowany, tzn. sprawdzany pod kątem jego zgodności z wymaganiami.

W kolejnych rozdziałach przedstawiono założenia, realizację oraz środowisko zaproponowanego podejścia (specyfikując wykorzystane narzędzia). Następnie zaprezentowano procesy weryfikacji, walidacji i testowania, rozdziały 3 i 4 pokazują zastosowanie metody, uzyskane wyniki. W ostatnim rozdziale przedstawione zostały wybrane aspekty narzędzi: IBM Rational Software Architect oraz IBM Rational Quality Management, jako przykład środowiska do symulacyjnej weryfikacji modeli UML i topologii [6].

[2] Autorzy zwracają szczególną uwagę na fakt, że użyte sformułowania dotyczące jakości mają w tym opracowaniu znaczenie zgodne z dokumentami standaryzującymi, np. [9]. Często w literaturze występują one zamiennie lub znaczą co innego. Przy czym zazwyczaj terminem weryfikacja określa się najogólniej sprawdzenie poprawności, jej uszczegółowienie w zakresie syntaktyki modelu – to walidacja, natomiast jej uszczegółowienie w zakresie pożądanego działania produktu – testowaniem. W tej pracy weryfikacja i walidacja mają inne definicje.

2. Metoda przeprowadzania złożonego wdrożenia

Zakładamy, że zasadnicze znaczenie w metodzie odegra symulacja modeli UML, antycypując kierunek rozwoju języków modelowania. Budowany symulacyjny model wdrożenia jest ponadto UML-ową reprezentacją procesu (złożonego) wdrożenia systemu. Na model ten, obok modeli wdrożenia w języku UML, składają się: stany, zdarzenia i czas.

Stany – opisują wartości cech systemu i relacji między jego elementami w ustalonych przedziałach czasu. Zdarzenia – określają chwile, w których następuje zmiana stanu systemu, lub jego elementów, a czas – a ściślej jego upływ reprezentowany jest za pomocą sekwencji zdarzeń, zgodnie z algorytmem „as soon as possible". W naszym projekcie symulacja komputerowa została wykorzystana do badania opracowanych modeli systemu (UML i topologii, jako: modeli konceptualnych, logicznych i fizycznych w procesach definiowania jego architektury), dla których określono model i środowisko symulacji, odwołując się do opracowanych w tym zakresie standardów OMG (fUML, Alf, UAL), które wspiera wykorzystany produkt, tj. IBM Rational Software Architect z rozszerzeniem Simulation Toolkit.

Badanie wdrożenia systemu polega w tym środowisku na przeprowadzaniu eksperymentów symulacyjnych na opracowanych modelach, a następnie analizie uzyskanych wyników i ich dokumentowaniu, co przedstawiono w kolejnych podrozdziałach.

W proponowanym podejściu wyróżniliśmy następujące (klasyczne) etapy budowy modelu symulacyjnego:

1) określenie systemu;
2) sformułowanie modelu;
3) ustalenie danych dla modelu;
4) planowanie eksperymentów symulacyjnych;
5) ocenę uzyskanych wyników;
6) ocenę adekwatności modelu;
7) dokumentowanie.

Świadomie pominięto etap zaprogramowania modelu, ponieważ środowisko symulacji bezpośrednio potrafi uruchamiać opracowane modele zapisane w językach UML, OCL, UAL oraz topologie.

W artykule skoncentrujemy się na przedstawieniu trzech zasadniczych faz procesu symulacyjnego:

- budowa modelu symulacyjnego – etapy 1-4, została przedstawiona w poprzednim artykule i obejmują konstruowanie modeli topologii wdrożenia oraz modeli behawioralnych opisujących procesy wdrożenia w języku UML, ich implementację komputerową w środowisku symulacyjnym (UML, ALF, OCL) oraz opracowanie planu eksperymentu;

- eksperymentowanie – czyli faza realizowania planu badań symulacyjnych, która zostanie przedstawiona w kolejnych rozdziałach;
- analiza wyników eksperymentów – (etapy 5-7) polega na przeprowadzeniu weryfikacji postawionej tezy (że: symulacja planu wdrożenia na etapie projektowania systemu ułatwia zrozumienie dynamicznych aspektów systemu oraz pozwala na wczesne wykrycie błędów projektowych i architektonicznych bez konieczności istnienia rzeczywistych produktów wdrożenia i środowiska, w którym te produkty będą w przyszłości działały) na podstawie udokumentowanych wyników symulacji.

Ważne jest przy tym określenie warunków początkowych i końcowych eksperymentu symulacyjnego, w którym sam model symulacyjny traktujemy jak „czarną skrzynkę" (identycznie jak w procesie testowania traktujemy system), a następnie: określamy zestawy danych wejściowych, ustalamy zestawy charakterystyk wyjściowych; ustalamy kryteria jakości badań symulacyjnych.

Dzięki takiemu podejściu wielokrotnie wykorzystane zostaną przygotowane testy i scenariusze testowe. Raz w procesach weryfikacji i walidacji, a następnie testowania. Oczywiście inne będzie jedynie środowisko badania.

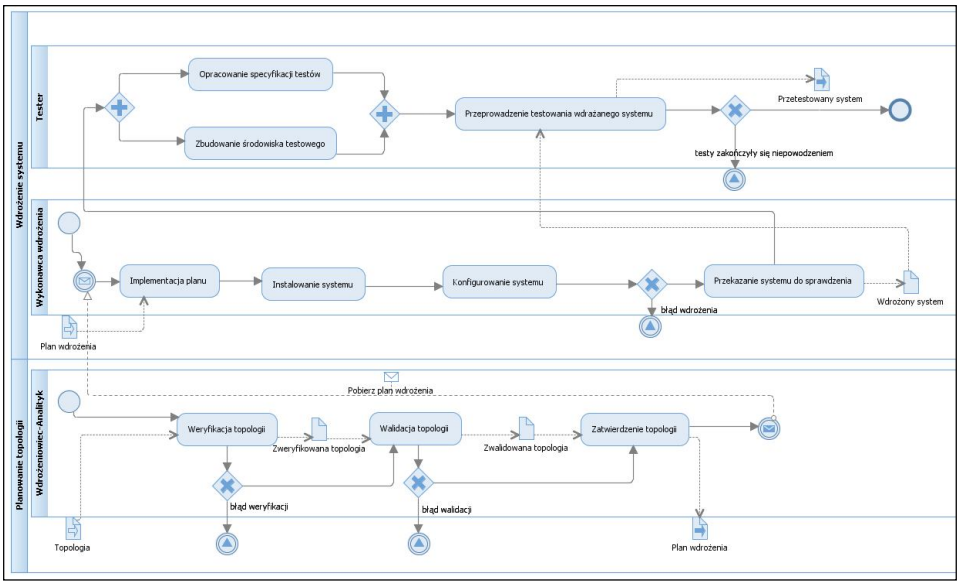
2.1. Założenia metody

Celem metody jest obniżenie kosztów złożonego wdrożenia poprzez wcześniejszą weryfikację i walidację planu w warunkach symulacyjnych, a nie rzeczywistych (docelowych), w których ma miejsce wdrożenie.

Osobliwość metody polega na przeciwstawieniu procesom weryfikacji i walidacji procesu testowania. Proces weryfikacji i walidacji odnosi się do planu wdrożenia i będzie realizowany metodą symulacyjną (szczegółowo przedstawiony zostanie w pkt. 2.5). Natomiast przedmiotem testowania jest wdrożenie (które dotyczy rzeczywistych obiektów wdrożenia).

Zakładamy, że eksperyment symulacyjny adekwatnie odwzorowuje rzeczywiste procesy wdrożenia, pozwalając na eksperymentowanie (zmianę topologii). Ponadto przyjmujemy, że symulacja planu jest procesem powtarzalnym, podczas gdy wdrożenie wykonuje się zazwyczaj raz.

Przepływ prac w metodzie przedstawiony jest na diagramie BPMN na rys. 1. Skupiono się na pokazaniu głównego przepływu, wyróżniając zdarzenia, które generują postęp metody (przejścia), pomijając obsługę zdarzeń wyjątkowych. Zaznaczono jedynie możliwe miejsca wystąpienia tych zdarzeń, przyjmując, że generują one powrót na początek przepływu. Wynikiem przedstawionego przepływu prac jest skuteczne wdrożenie systemu (produkt).



Rys. 1. Przepływ prac w opracowanej metodzie w notacji BPMN

2.2. Organizacja wdrożenia

Wyróżniono dwa współbieżne procesy: planowanie topologii i wdrożenie systemu oraz trzy role, tj. Wdrożeniowca-analityka, Wykonawcę wdrożenia i Testera, wraz z ich odpowiedzialnościami w postaci zadań. Zadania reprezentują czynności, które należy wykonać, by na podstawie planu przeprowadzić skuteczne wdrożenie.

Wyżej wymienione role posiadają następujące odpowiedzialności:

- Wdrożeniowiec-analityk – sprawdzić poprawność i adekwatność topologii,
- Wykonawca wdrożenia – przeprowadzić wdrożenie na podstawie planu wdrożenia,
- Tester – sprawdzić poprawność wdrożenia.

2.3. Procesy metody

W metodzie skupiono się na dostarczeniu funkcjonalności określonej przez poniższe historyjki użytkownika:

1) Jako Wdrożeniowiec-analityk chcę zweryfikować i zwalidować opracowane modele (w procesach ich symulacji), aby moje decyzje umożliwiły skuteczną pracę wykonawcy wdrożenia.
2) Jako Wykonawca wdrożenia chcę osadzić „złożony system” – Jazz Team Server w infrastrukturze WCY, aby wspomagać prace projektowe realizowane na wydziale (przez studentów i pracowników).

3) Jako Tester chcę dokonać sprawdzenia wdrożonego systemu (w procesach testowania), aby potwierdzić jakość wdrożenia (poprawna instalacja i konfiguracja).

2.4. Środowisko metody

2.4.1. Narzędzia symulacyjnej weryfikacji i walidacji topologii

Opisywane podejście do wdrożenia określa, że weryfikacja i walidacja dotyczą tylko modeli. Przeprowadzono je metodą symulacyjną, do której wykorzystano zintegrowane narzędzie do modelowania i programowania IBM Rational Software Architect 8.5 (RSA) razem z dodatkiem: Extension for Deployment Planning (do planowania wdrożenia) oraz zestawem narzędzi Simulation Toolkit (do uruchamiania modeli topologii i UML).

Środowisko RSA wraz z wymienionymi dodatkami zapewnia możliwość planowania topologii, tj. opisu jednostek topologii i łączy między nimi, oraz uruchamiania topologii (symulacji). Tworzenie ww. łączy wykonać można ręcznie, bądź wykorzystując do tego celu diagram opisujący zachowanie w systemie (metodą „przeciągnij i upuść"). W tej sytuacji zbudowane zostanie powiązanie między topologią a opisem zachowania, dzięki któremu możliwe jest uruchomienie topologii. Podczas symulacji środowisko odwzorowuje powiązane z topologią zachowanie (scenariusz), kolejno przechodząc przez jednostki topologii wg kolejności ustalonej przez scenariusz. Sterowanie symulacją odbywa się za pomocą zdarzeń z poziomu kodu UAL bądź ręcznie (np. przy nieprecyzyjnie opisanych blokach decyzyjnych).

Symulacja może być przeprowadzana w trybie wykonawczym (ang. *execute*) bądź w trybie debugowania (ang. *debug*). W pierwszym trybie Wdrożeniowiec-analityk jest tylko obserwatorem symulacji, natomiast w drugim trybie ma możliwość ingerowania w wykonanie (np. poprzez zmianę w modelach), podgląd i zmianę zmiennych wykonania, wybór ścieżek decyzyjnych itp.

Opisywane środowisko umożliwia zebranie następujących wyników symulacji: historii komunikatów przesyłanych między obiektami, śladów przepływu komunikatów, historii zapisów wprowadzonych i wyprowadzonych na konsoli. Możliwości środowiska można znacznie rozszerzyć, stosując język UAL do specyfikacji zachowania konkretnych elementów, w celu np. symulacji interakcyjnej pracy, wykorzystania plików czy stosowania złożonych struktur danych[3].

[3] Autorzy zwracają uwagę na fakt, że w opisywanej w tej pracy wersji RSA możliwość stosowania języka UAL jest znacznie ograniczona ze względu na niepełną (lub błędną) składnię tego języka (niezgodną z dokumentami normatywnymi [9]).

2.4.2. Narzędzia testowania

W zaproponowanym podejściu sprawdzenie poprawności wdrożenia, wykonanego na podstawie planu wdrożenia, następuje w drodze testowania. Testowanie przeprowadzono metodą porównawczą, porównując wynik uruchomienia skryptu testowego z oczekiwanym wynikiem utworzonym na etapie specyfikacji testów[4].

Do testowania wykorzystano narzędzie IBM Rational Quality Management (QM) w wersji 4.0. Jest to aplikacja WWW osadzona na platformie IBM Jazz przeznaczona do zarządzania jakością produktów. Dostarcza się ją wraz z predefiniowanymi szablonami testowania, umożliwiającymi profilowanie procesów testowania (np. w celu dopasowania do metodyki wytwarzania oprogramowania). Narzędzie to umożliwia między innymi planowanie testowania, projektowanie testów, definiowanie i uruchamianie procesów i zasobów testowania, raportowanie i analizę wyników oraz tworzenie zadań jakości[5], a także określanie innych istotnych w procesie testowania elementów[6]. Nie wymaga przy tym żadnych dodatkowych rozszerzeń.

Ogólny sposób pracy z QM jest następujący. Na wstępie tworzony jest plan testowania. Zależnie od wybranego szablonu plan ten zawiera definicje przypadków testowych[7] (określanych przez producenta jako *instrukcje testowania*), może również zawierać definicję procesu przeglądu, celów jakościowych i łagodzenia ryzyka. Utworzenie tego planu to zazwyczaj odpowiedzialność głównego testera, który tworzy go na podstawie wymagań pochodzących z produktu do zarządzania wymaganiami[8].

Następnie główny tester wiąże wymagania z przypadkami testowymi i przypisuje testerom. Tester tworzy pakiety testów[9] zawierające wybrane przypadki testowe. Potem buduje samodzielnie skrypty testowe (ręczne skrypty testowe) bądź wykorzystuje narzędzie do automatycznego budowania skryptów. Narzędzia do automatycznego budowania skryptów to rejestratory działań

[4] W przypadku produktów „z półki" specyfikacja testów powstaje na podstawie wymagań dotyczących wdrożenia.

[5] Zadanie jakości w tym narzędziu to element pracy zespołowej i jest rozumiane jako sprzężenie zwrotne skierowane do grupy projektowej.

[6] https://jazz.wcy.wat.edu.pl:9443/clmhelp/topic/com.ibm.rational.test.qm.doc/topics/c_qm_overview.html

[7] Przypadek testowy (ang. *test case*) oznacza zbiór warunków określających, czy produkt został zbudowany poprawnie (jako odpowiedź na przypadek użycia).

[8] Według producenta aplikacji QM, wymagania mogą pochodzić z aplikacji IBM Requirements Management, IBM Rational RequisitePro lub IBM Rational DOORS.

[9] Pakiet testów (ang. *test suite*) jest również określany jako pakiet walidujący. Pakiet ten zawiera przypadki testowe pogrupowane w celu sprawdzenia poprawności konkretnej funkcji lub cechy programu komputerowego. Por. http://en.wikipedia.org/wiki/Test_suite.

wykonywanych przez operatora (testera) na komputerze, które zapisują zdarzenia powstające podczas wykonywania czynności przez testera (np. uruchomienie aplikacji, wyświetlenie komunikatu itp.). Zbudowany skrypt testowy podlega zatwierdzeniu, co leży zazwyczaj w gestii głównego testera.

W przypadku testowania ręcznego tester uruchamia pakiety testów, wykonuje kolejne kroki skryptu testowego, a następnie określa status ich wykonania (*Sukces, Trwale zakończone niepowodzeniem, Niedokończone, Nierozstrzygnięte, Częściowo zablokowane, Odroczone, Zakończone niepowodzeniem, Błąd, Zablokowane*) oraz ogólny wynik testowania.

W przypadku testowania automatycznego pakiety testów (bądź pojedyncze przypadki testowe) uruchamiane są w środowisku testowym opisanym przez tzw. laboratorium. Laboratorium jest specyfikacją zasobów (sprzętu, oprogramowania, czasu itp.), które tworzą środowisko testowe. Na etapie testowania (np. gdy dostępna jest nowo zbudowana wersja oprogramowania) tester rezerwuje zasoby laboratorium, generuje rekordy wykonania dla przypadków testowych oraz pakietu testów, po czym uruchamia te rekordy i śledzi ich wykonanie w produkcie do śledzenia defektów. Za testowanie w tym przypadku odpowiada agent, który wykonuje kolejne kroki skryptu testowego i porównuje uzyskane wyniki z oczekiwanymi, po czym na tej podstawie określa status ich wykonania. Następnie na podstawie uzyskanych wyników porównania określa ogólny wynik testowania[10] na zasadzie: *Sukces*, w przypadku gdy wszystkie testy zakończyły się sukcesem; *Zakończone niepowodzeniem* w innym przypadku.

Na potrzeby testowania wdrożenia zdefiniowano następujące elementy: plan testów, pakiet testów, przypadki testowe, skrypt testowy oraz ustanowiono wzajemne zależności pomiędzy przypadkami testowymi a wymaganiami[11] por. [1] rys. 8. Następnie przygotowano maszynę wirtualną z zainstalowanym oprogramowaniem do automatyzacji testów IBM Rational Functional Tester w wersji 8.3.0 (RFT) oraz przeglądarką internetową, po czym uruchomiono adapter RFT i skonfigurowano aplikację QM do zarządzania zasobami maszyny wirtualnej przez ten adapter (opisują zasoby w laboratorium).

W takim środowisku uruchomiono pakiet testów, który zawierał wybrany podzbiór przypadków testowych wraz ze wskazanymi skryptami testowymi.

W przypadku testu manualnego aplikacja QM podpowiadała operatorowi -testerowi kroki skryptu testowego i oczekiwała wprowadzenia wyniku rzeczywistego. Po zakończeniu testu wynik testu ustalał tester.

[10] Jest to podpowiedź dla testera, który może dowolnie ustalić wynik.

[11] Warto zwrócić uwagę na fakt, że wymagania mogą pochodzić z różnych źródeł, np. w opisywanym narzędziu mogą być one rozproszone pomiędzy aplikacje: Requirements Management, IBM Rational RequisitePro i IBM Rational DOORS. Bez ustanowionych zależności zarówno analiza pokrycia, jak i śledzenie zmiany wymagań byłyby znacznie utrudnione.

W przypadku testu zautomatyzowanego aplikacja QM przekazuje skrypt testowy do środowiska testowania (RFT uruchomionego na zdalnej maszynie), a RFT wykonuje kroki skryptu testowego i raportuje wyniki do QM. Po zakończeniu testu, wynik ustala się według zasady opisanej powyżej.

2.5. Proces weryfikacji i walidacji planu topologii

W zaproponowanym podejściu wynikiem procesu planowania topologii jest plan wdrożenia. Plan wdrożenia jest zweryfikowanym i zwalidowanym modelem topologii, na który składają się jednostki oraz powiązania (odwzorowane modelem zachowania UML, por. pkt 2.4.1).

Kryterium wejściowym procesu symulacji jest istnienie modelu topologii. Model ten może powstać na jednym z etapów projektowania systemu (w celu analizy docelowego wdrożenia) bądź na etapie planowania wdrożenia (dla produktów gotowych lub „z półki").

W pierwszym przypadku, gdy końcowa struktura produktu nie jest jeszcze znana (np. brak specyfikacji wszystkich komponentów), symulacja już na tym etapie może pomóc ocenić to rozwiązanie (np. przez wykrycie błędów syntaktycznych) i odnaleźć istnienie „wąskich gardeł" (np. w dostępie do antycypowanej usługi). Przez to, na długo przed wdrożeniem, może pomóc w podjęciu decyzji, które będą miały wpływ na wdrożenie systemu, a potem na pracę z nim.

W drugim przypadku, kiedy model topologii powstaje, gdy produkt jest gotowy, symulacja pozwala bardzo dokładnie przebadać nadchodzące wdrożenie, gdyż model topologii można już dokładnie określić. Wtedy taki model wskazuje nie tylko komplet cech opisu konfiguracji (dla wdrożenia), lecz także specyfikuje realizacje jednostek topologii (sprzętowe lub programowe). Poza tym stosując odpowiednio dobrane ograniczenia (OCL), Wdrożeniowiec-Analityk może sprawdzić, czy produkt spełnia wymagania klienta w jego infrastrukturze.

Bez względu na czas planowania topologii i stopień abstrakcji jej analizy, opisywane podejście wymaga sprawdzenia, czy topologia została zaplanowana poprawnie składniowo (weryfikacja) oraz jej zgodności z wymaganiami klienta (walidacja).

Weryfikacja jest pierwszym etapem sprawdzania poprawności. Plan wdrożenia stanowi jeden z elementów wektora danych wejściowych do symulatora. Pozostałe elementy tego wektora to: stany, zdarzenia i akcje (UAL). Eksperyment symulacyjny zaczyna się uruchomieniem planu wdrożenia w środowisku symulacyjnym. W czasie symulacji, zgodnie z upływem zdarzeń, symulator wykonuje scenariusz (opisany na diagramie zachowania, por. pkt 2.4.1) powiązany z topologią i odwzorowuje działanie (np. akcje) w modelu topologii. Zdarzenia wysyłane są do konkretnych instancji i to w tych instancjach musi być zdefiniowany wyzwalacz (ang. *trigger*) oraz obsługa tych zdarzeń.

Uruchomienie (wyzwolenie) zdarzeń może następować dzięki blokom oczekiwania na zdarzenie (ang. *Accept Event Action*), przejściom stanów (tranzycjom), bądź jako akcja UAL w dowolnej specyfikacji zachowania (ang. *Opaque behavior*).

Obsługa zdarzeń może polegać na realizacji aktywności (ang. *Activity*), wykonaniu akcji przejścia do stanu (ang. *Effect*), bądź wykonaniu dowolnej specyfikacji zachowania, która opisana została językiem akcji. W ostatnim przypadku, dzięki językom specyfikującym semantyki akcji (wg deklaracji [9]), istnieje bardzo szeroka możliwość opisu zachowania (od podstawowych działań w systemie plików po interakcję z operatorem np. przez konsolę CLI).

W symulacji, zależnie od sposobu wysyłania, odbierania zdarzeń i reakcji na nie, może brać udział operator (Wdrożeniowiec-analityk). W przypadku symulacji z udziałem operatora, ma on możliwość podglądu na bieżąco stanu symulacji (gdyż zachowanie jest animowane) oraz ingerowania w wykonanie (np. przez zmianę danych) lub wybór ścieżki decyzyjnej. Kiedy w modelu symulacyjnym określono wymaganie wprowadzania danych dynamicznie (na bieżąco), wtedy obecność operatora jest nieodzowna.

W przypadku symulacji bez udziału operatora, wybór ścieżek decyzyjnych jest losowy (zależnie od środowiska symulacji), możliwości interakcji muszą być sprowadzone do innej formy współdziałania (np. w oparciu o pliki), a sam model powinien być bardzo zwięzły (szczegółowy) i z kompletnie opisanymi akcjami, by plan mógł faktycznie zostać zweryfikowany.

Eksperyment symulacyjny (pojedynczy przebieg symulacji) kończy się, gdy:

- sterowanie osiągnęło jawnie określony w specyfikacji zachowania punkt końcowy;
- wyczerpała się pula zdarzeń do wysłania;
- wykryto błąd w modelu;
- zażądał tego operator.

W wyniku symulacji, środowisko powinno dostarczyć kompletną historię zdarzeń (w postaci śladów w modelu topologii, historii komunikatów oraz zapisów w konsoli CLI lub w innej postaci). Model topologii można uznać za zweryfikowany, gdy w wyniku symulacji osiągnięto punkt końcowy (jak wyżej), a historia zdarzeń potwierdza deterministyczność modelu i działanie całkowicie zgodne ze scenariuszem.

Walidacja jest drugim etapem sprawdzania poprawności modelu topologii. W wyniku walidacji oczekuje się potwierdzenia adekwatności modelu, to znaczy, czy dany model opisuje dany produkt.

Potwierdzenie czy model jest adekwatny w opisywanym podejściu, następuje w drodze sprawdzenia (spełnienia) ograniczeń modelu. Ograniczenia są częścią modelu i specyfikowane są w języku OCL (ang. *Object Constraint*

Language) [11]. Specyfikacja ograniczeń w tym języku umożliwia uwzględnienie zarówno formalnych wymogów produktu (np. wymóg istnienia komponentu, ścisłej kolejności czy niezmienności cechy, itp.), jak i wymogów pozafunkcjonalnych, które można opisać w sposób mierzalny (np. liczba możliwych otwartych sesji użytkownika na serwerze, przepustowość łączy komunikacyjnych w kbps, dostępność wyrażona w procentach itp.).

Ograniczenia, które stosują się do topologii, pochodzić powinny z każdego etapu projektowania systemu, jednak dopiero na etapie planowania wdrożenia można zastosować je w modelu. Dzieje się tak dlatego, że ograniczenia silnie zależą od aktualnej struktury systemu, przez co budowanie ograniczeń jest procesem trudniejszym niż specyfikacja akcji (na potrzeby weryfikacji).

Proces walidacji w opisywanym podejściu powinien następować zarówno „na żądanie" (dosłownie w trybie wsadowym, ang. *batch*), jak i „na żywo" (ang. *live*). Pierwszy tryb będzie stosowany podczas planowania topologii (kiedy ograniczenia nie mogą jeszcze zostać spełnione) oraz w procesie symulacji (np. podczas zmiany stanu). Drugi natomiast będzie stosowany na etapie zmian w topologii (np. po symulacji), by na bieżąco wykluczyć możliwość budowania modeli nie adekwatnych (nie spełniających wymagań).

Po pozytywnej walidacji, tzn. gdy model spełnia wszystkie ograniczenia, model topologii podlega zatwierdzeniu (por. rys. 1), po czym jako plan wdrożenia zostaje przekazany wykonawcy wdrożenia.

W przypadku kiedy walidacja wykazała niespełnienie co najmniej jednego ograniczenia, model topologii (i inne modele) należy poprawić, gdyż wg założenia opisywanej metody plan wdrożenia nie będzie adekwatny.

2.6. Proces testowania wdrożenia

W zaproponowanym podejściu po pozytywnej weryfikacji i walidacji topologia jest planem, na podstawie którego wykonywane jest wdrożenie. Wdrożony system stanowi następnie daną wejściową do procesu testowania, by zgodnie z opisywanym podejściem potwierdzić słuszność decyzji architektonicznych podjętych na etapie planowania.

Główną ideą w opisywanej metodzie jest mapowanie scenariuszy z procesu symulacji na proces testowania. Specyfikacje zdarzeń, wykorzystanych wcześniej do sterowania symulacji, w procesie testowania posłużą do zbudowania skryptów testowych (jako zdarzenia rzeczywiste). Natomiast ograniczenia walidujące zgodność modelu z wymaganiami w procesie testowania będą oczekiwanymi wynikami dla przypadków testowych.

Testowanie wdrożenia rozpoczynają dwa współbieżne zadania: opracowanie specyfikacji testów, na którą składają się: plany testów, pakiety

testowania, przypadki testowe oraz skrypty testowe (opisane w pkt. 2.4.2) oraz zbudowanie środowiska testowego (na podstawie specyfikacji jak w pkt. 2.4.2). Oba zadania muszą zakończyć się równocześnie, by móc przejść do kolejnego etapu, czyli do przeprowadzenia testowania wdrażanego systemu.

W procesie testowania specyfikacje zdarzeń, które sterowały symulacją modelu topologii, są podstawą dla rzeczywistych zdarzeń, zachodzących w działającym produkcie (np. specyfikacja zdarzenia „Wyświetlenie komunikatu na ekranie" będzie realnym zdarzeniem pojawienia się komunikatu na ekranie monitora itp.).

Testowanie odbywa się przez wykonanie kolejno kroków skryptu testowego (przez testera bądź automatycznie) oraz porównanie oczekiwanego wyniku z wynikiem uzyskanym (por. pkt 2.4.2). Test zakończony jest sukcesem, jeśli uzyskano wynik zgodny. Proces testowania wdrożenia zakończony będzie sukcesem, jeśli wszystkie skrypty testowe (testy wszystkich ścieżek w systemie) zakończą się sukcesem. Oznacza to, zgodnie z zaproponowaną metodą, potwierdzenie słuszności decyzji dotyczących wdrożenia, podjętych na etapie, który bezpośrednio poprzedza symulacja modeli topologii.

W przypadku innego wyniku proces testowania zakończony będzie porażką, co należy uznać (zakładając brak sytuacji wyjątkowych, pomyłek, awarii itp.) za niepotwierdzenie się decyzji podjętych na etapie planowania wdrożenia.

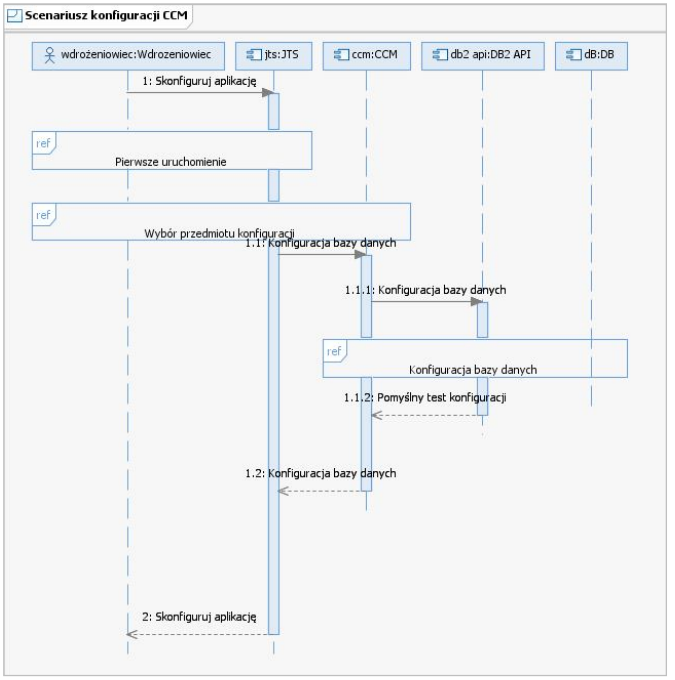
3. Planowanie wdrożenia

W rozdziale przedstawione zostanie praktyczne użycie przedstawionego w pkt. 2 autorskiego podejścia do realizacji złożonego wdrożenia systemu IT, na przykładzie wdrożenia systemu pracy grupowej IBM Jazz na Wydziale Cybernetyki WAT. W tym rozdziale szczegółowo opisany zostanie przykład związany z planowaniem wdrożenia, w kolejnym natomiast przykład wdrożenia systemu.

Kryterium wejściowym do rozpoczęcia procesu planowania wdrożenia jest istnienie modelu topologii. Procesy związane z planowaniem topologii opisano w [1]. Do niniejszego przykładu wybrano logiczny model topologii przedstawiony w [1] na rys. 2. Ograniczenia w modelu topologii (na potrzeby walidacji) są reprezentowane przez wiązania topologiczne (utworzone na podstawie modelu zachowania, por. 2.4.1), do utworzenia których wybrano skonkretyzowany scenariusz[12], według szablonu opisanego w [1] na rys. 17, oraz

[12] Przywołany diagram sekwencji, wykorzystany do opisu zachowania związanego z konfiguracją aplikacji, jest szablonem scenariusza, którego użycie musi wiązać się ze skonkretyzowaniem

konkretny typ aplikacji – CCM. Instancja scenariusza przedstawiona jest na rys. 2.



Rys. 2. Scenariusz konfiguracji aplikacji CCM

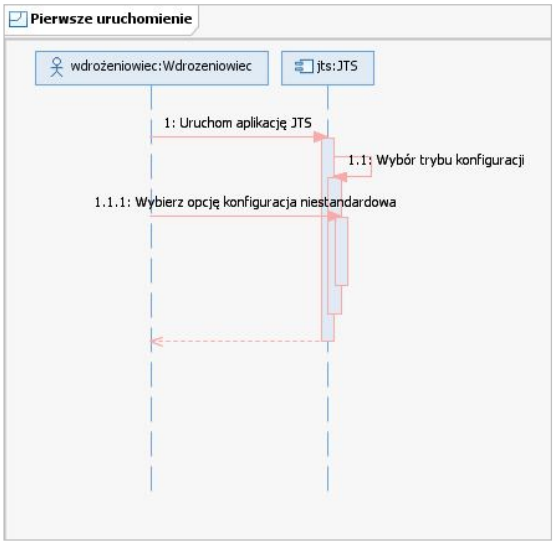
Fragmenty referencji (*ref*) mają na celu zwiększenie widoczności oraz jawne wyspecyfikowanie zakresu interakcji, do którego odwołuje się wybrany na potrzeby symulacji podzbiór stanów wdrożenia, zobrazowany na rys. 5 (stany reprezentowane są przez obiekty «datastore», przejścia natomiast przez aktywności). Fragmenty te przedstawiono na kolejnych rysunkach.

Model symulacyjny tworzą także sygnały, których specyfikacja przedstawiona jest na rys. 6, typy danych (enumeratory) wykorzystane do specyfikacji sygnałów przedstawia rys. 7.

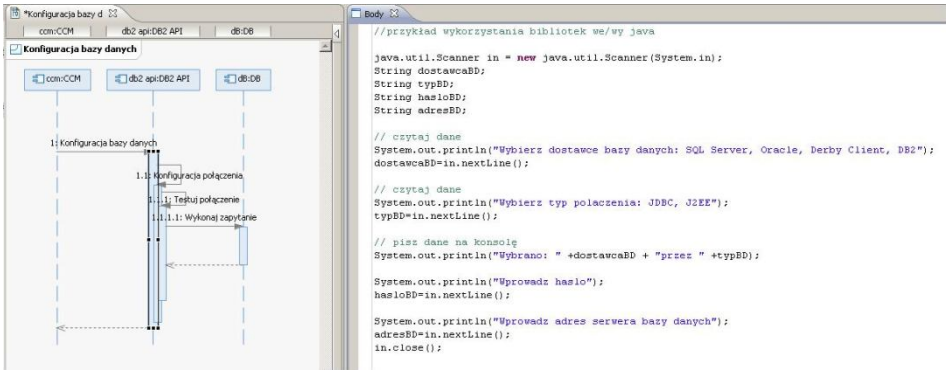
Wybrany model topologii przebadano w środowisku opisywanym w pkt. 2.4.1. Dzięki dodatkowi RSA Simulation Toolkit możliwe było uruchomienie modeli topologii i UML wymienionych w tym rozdziale. Uruchomienie modelu spowodowało utworzenie zmiennych sesji wykonania symulacji. Jedną z tych zmiennych był *stan*, dzięki któremu środowisko symulacji mogło sterować blokami decyzyjnymi podczas symulacji, por. rys. 5.

aplikacji (zgodnie z zasadą abstrakcji i dziedziczenia). Por. [1] pkt 3.2.2.2.

Niestety monitorowanie tej zmiennej było bardzo ograniczone ze względu na brak historii (pamięci) wartości tej zmiennej, co świadczy o niedopracowaniu środowiska symulacji. Historia zmian została zatem wykonana w sposób autorski, co przedstawia rys. 5.



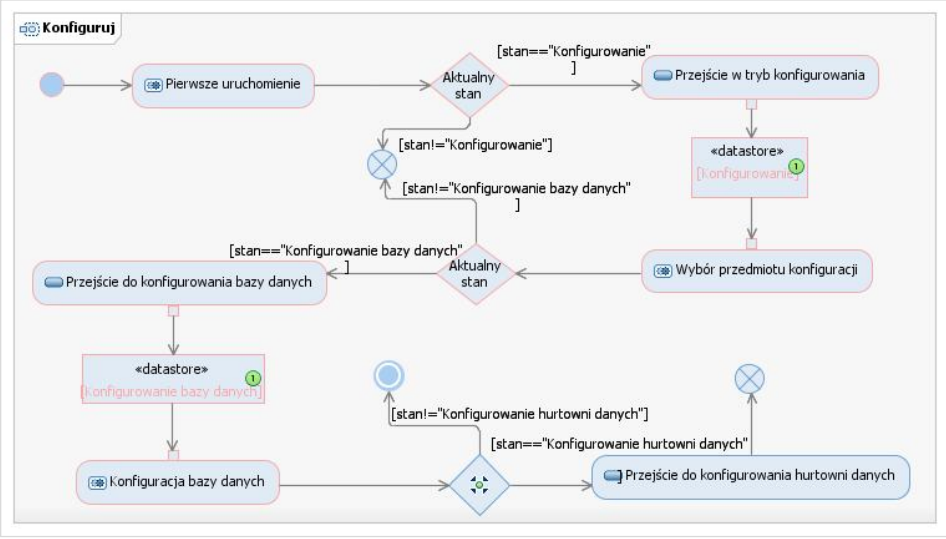
Rys. 3. Fragment interakcji *Pierwsze uruchomienie*



Rys. 4. Fragment interakcji *Konfiguracja bazy danych* wraz z specyfikacją akcji w kodzie UAL

W trakcie symulacji aktualne miejsce przetwarzania reprezentowane jest przez znacznik (ang. *token*, por. rys. 5). Ślad przetwarzania (historia przebywania znacznika) jest pamiętany na zasadzie historii komunikatów przesyłanych między obiektami oraz pośrednio jako znakowanie ścieżek przebytych (zostawiania śladu) por. rys. 8.

Zbudowany symulator w tym badaniu używał konsoli CLI do interakcji z operatorem. Wykorzystanie konsoli możliwe jest dzięki specyfikacji akcji UAL w wybranych elementach reprezentujących zachowanie (np. aktywności, operacji itp.). W tym przypadku symulator prosił operatora o podanie danych konfiguracyjnych połączenia z bazą danych. Przykładowe dyrektywy w języku UAL, dzięki którym przeprowadzono symulację, opisano na rys. 4.

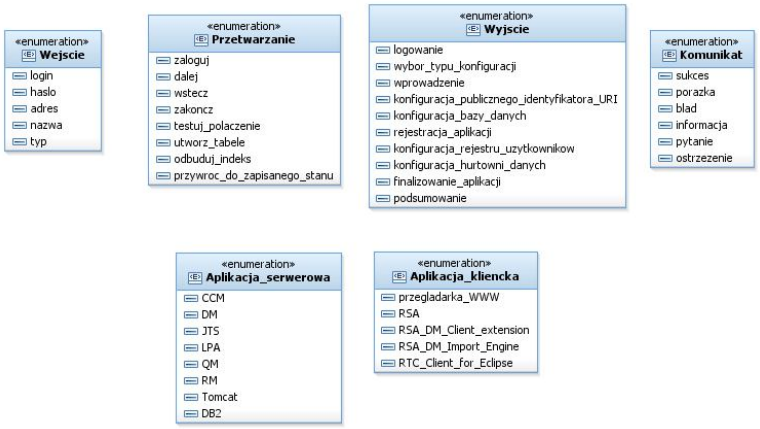


Rys. 5. Zobrazowanie wybranych stanów i przejść obiektu wdrożenia w trakcie symulacji

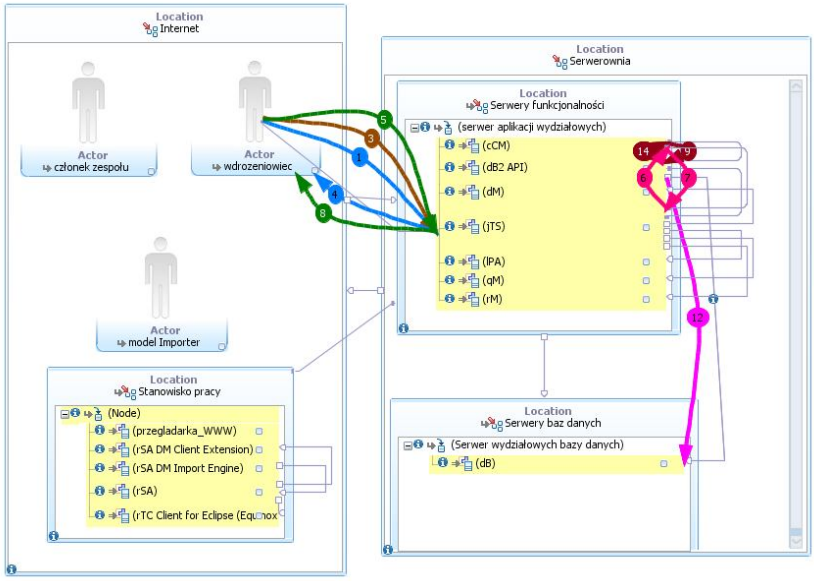
Rys. 6. Specyfikacja sygnałów

Historię zapisów na konsoli przedstawiono na rys. 9. Te zapisy wraz z modelami z rys. 3, rys. 4, rys. 5 i rys. 8 dokumentują wybrany zakres weryfikacji i walidacji podjęty przez autora tej pracy. Wynikiem tego etapu jest plan wdrożenia, czyli zweryfikowany i zwalidowany model topologii dla konkretnego wdrożenia. Kolejnym etapem w opisywanej metodzie (por. rys. 1) jest wdrożenie, które zostało wykonane i udostępnione jest pod adresem:

`https://jazz.wcy.wat.edu.pl:9443/jts`

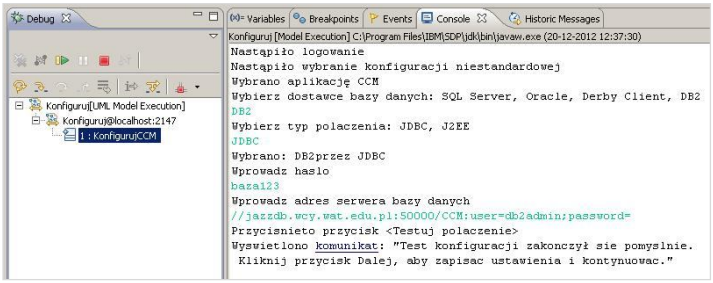


Rys. 7. Typy enumeratywne wykorzystane w specyfikacji sygnałów



Rys. 8. Model topologii w trakcie symulacji. Ślady pomiędzy jednostkami topologii reprezentują historię przesyłanych komunikatów

Pozostałe wdrożone aplikacje oraz uwagi do przeprowadzonego wdrożenia opisano w [1]. W kolejnym etapie, zgodnie z opisywaną metodą, zostanie sprawdzona poprawność wdrożenia, czyli zgodność jego realizacji na podstawie planu wdrożenia opisywanego w tym rozdziale z wymaganiami. Sprawdzenie to opisane zostanie w rozdziale kolejnym.



Rys. 9. Historia zapisów na konsoli po procesie symulacji

4. Realizacja wdrożenia

Niniejszy rozdział przedstawia raport z procesów testowania modeli topologii opisujących złożone wdrożenie systemu Jazz na Wydziale Cybernetyki WAT. Celem testowania było sprawdzenie, czy zbudowane modele odpowiadają wymaganiom postawionym przed procesem konfiguracji systemu (por. [1] rys. 7).

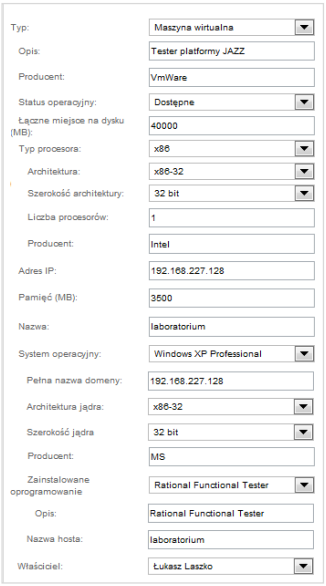
Proces testowania przeprowadzono w następujący sposób. Utworzono plan testowania. W planie tym między innymi znalazły się informacje o kolekcji powiązanych wymagań (por. [1] rys. 8), o przypadkach testowych i pakietach testów. Następnie utworzono powiązanie utrzymujące relacje śledzenia zmian wymagań, umożliwiające także analizę pokrycia wymagań testami.

Rys. 10. Fragment listy utworzonych skryptów testowych

Na podstawie analizy wymagań utworzono przypadki testowe, czyli deklaracje aspektów wdrożenia przewidzianych do sprawdzenia. Potem dokonano implementacji przypadków testowych przez utworzenie skryptów testowych (por. rys. 10), będących kompletnym scenariuszem dla testerów i automatów testujących. Przykładowy skrypt przedstawia rys. 11.

Krok	Opis:
1	enter "https://jazz.wcy.wat.edu.pl:9443/ccm/setup{ENTER}" into the browser
2	select the "Logowanie - Jazz Team Server" window
3	click the "ID użytkownika:" textbox
4	enter "tester" into the "ID użytkownika:" textbox
5	click the "Hasło:" textbox
6	enter "test123" into the "Hasło:" textbox
7	click the "Zaloguj" button
8	select the "Konfiguracja serwera Jazz Team Server i aplikacji" window
9	click the "Konfiguracja niestandardowa Ten kreator umożliwia przeprowadzenie pełnej..." radiobutton
10	click the "Dalej >" button
11	select the "Konfigurowanie produktu Jazz Team Server" window
12	click the "Dostawca bazy danych:" list
13	click the "Dostawca bazy danych:" list at text "DB2"
14	click the "Typ połączenia:" list
15	click the "Typ połączenia:" list at text "JDBC"
16	enter "baza123" into the "Wartość domyślna" textbox
17	enter "//jazzdb.wcy.wat.edu.pl:50000/CCM:user=db2admin;password=" into the "Przykład: //localhost:50000/JTS:user=db2inst1;password={password};" textbox
18	click the "Testuj połączenie" button
19	click the "Dalej >" button

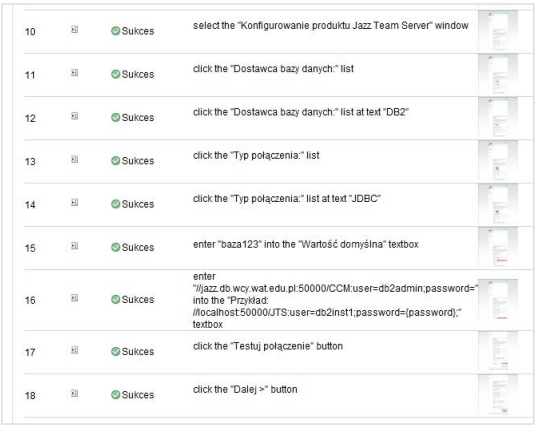
Rys. 11. Przykładowy skrypt testowy



Rys. 12. Opis zasobów laboratorium utworzonego na potrzeby testowania wdrożenia

Następnie opracowano specyfikację laboratorium (por. pkt 2.4.2) o parametrach pokazanych na rys. 12. Na jej podstawie zbudowano środowisko wirtualne (maszynę wirtualną) w technologii VmWare® oraz zainstalowano oprogramowanie wymagane do sprawdzenia, zgodnie z opisaną procedurą. Następnie uruchomiono agenta RFT (por. 2.4.2), który oczekiwał na żądania od aplikacji QM.

W kolejnym kroku dla każdego kroku skryptu testowego określono oczekiwany wynik jego wykonania por. tab. 1. Następnie przeprowadzono testy zgodnie z procedurą opisaną w pkt. 2.6, żądając zasobów laboratorium z aplikacji QM. Wynik wykonania każdego kroku skryptu testowego porównywano z wynikiem oczekiwanym. W wyniku testowania uzyskano wynik zgodny z oczekiwanym por. rys. 13.



Rys. 13. Fragment raportu procesu testowania aplikacji CCM

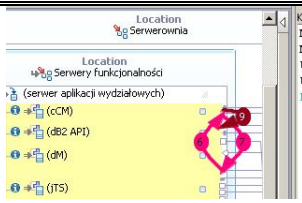
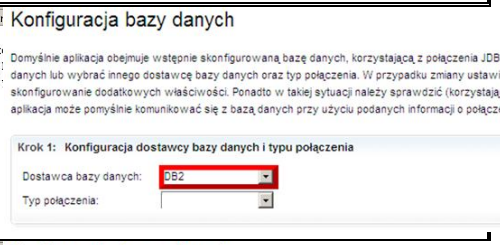
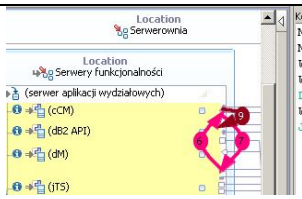
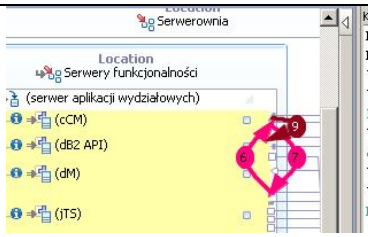
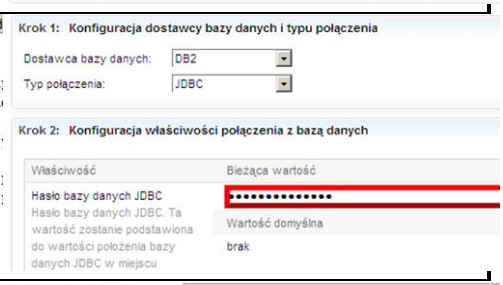
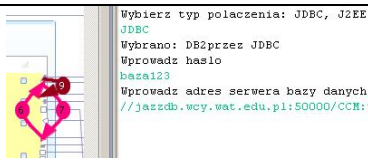
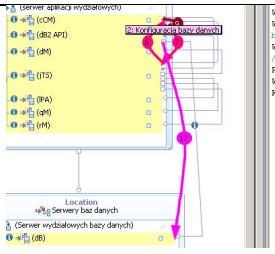
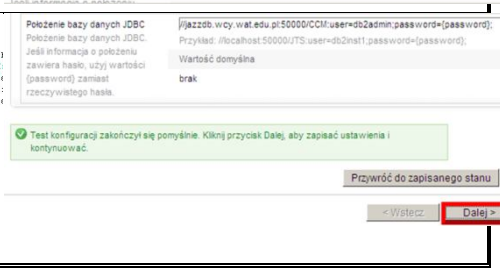
5. Wyniki

Odzwierciedleniem wyników jest tabela 1. Lewa kolumna reprezentuje wyniki uzyskane przed wdrożeniem (na etapie symulacji topologii), prawa kolumna natomiast wyniki rzeczywiste, uzyskane po wdrożeniu. Kolumna ***k*** odzwierciedla kroki skryptu testowego (por. rys. 11).

Wyniki przedstawione w kolumnie lewej pochodzą ze środowiska symulacyjnego w trakcie przeprowadzania badania. Składają się na nie: ślady komunikacji w modelu topologii oraz zapis danych na konsoli operatorskiej w danym kroku wykonywania skryptu testowego.

Wyniki przedstawione w prawej kolumnie pochodzą ze środowiska do testowania. Wykonanie kolejnych kroków skryptu testowego udokumentowane zostało rysunkami z rzeczywistego (wdrożonego) produktu. Kolejne wiersze opisują zgodne z zaproponowanym podejściem rezultaty.

Tab. 1. Porównanie wyników badań wdrożenia *a priori* i *a posteriori* (wybrane kroki skryptu testowego)

k	Weryfikacja i walidacja planu wdrożenia	Testowanie wdrożenia
13		
15		
16		
17		
18		

6. Wnioski

W pracy opisano opracowaną metodę symulacyjnej weryfikacji i walidacji modeli topologii i UML, która na etapie projektowania systemu ułatwia zrozumienie dynamicznych jego aspektów oraz pozwala na wczesne wykrycie błędów projektowych i architektonicznych bez konieczności istnienia rzeczywistych produktów wdrożenia i docelowego środowiska działania. Metodę sprawdzono przez zastosowanie jej do analizy i projektowania złożonego wdrożenia systemu IT.

Wyniki zastosowania metody potwierdziły tezę pracy, co oznacza, że przy przyjętych założeniach (por. pkt 2.1) możliwy jest adekwatny opis (plan) złożonego wdrożenia systemu IT oraz dzięki weryfikacji i walidacji tego planu, przeprowadzenie wdrożenia systemu może być bardziej skuteczne i tańsze, ze względu na możliwość eksperymentowania na modelu, bez konieczności istnienia gotowych produktów i środowisk wdrożenia.

Podstawą metody jest wykorzystanie modeli topologii (języka opracowanego przez IBM), języka UML, języka OCL wraz z opisem semantyk akcji, za pomocą języka UAL. Wykorzystanie tych języków pozwoliło uzyskać bardzo precyzyjne modele, które stanowiły podstawowy element symulatora.

Dzięki symulacji możliwe było przebadanie różnych wersji wdrożeń i wybranie tej, które spełnia ograniczenia postawione przed docelowym środowiskiem.

Zaproponowane podejście do wdrożenia zostało sprawdzone praktycznie na przykładzie realnego wdrożenia systemu do pracy grupowej IBM Jazz na Wydziale Cybernetyki WAT oraz przebadane w drodze testowania. Testowanie potwierdziło słuszność decyzji architektonicznych przedsięwziętych dzięki wcześniejszej symulacji modelu topologii (planu wdrożenia).

Uzyskane wyniki mogą być motywacją dla architektów oprogramowania i wykonawców wdrożeń do budowania planów wdrożeń w sposób opisany w pracy oraz ich symulacyjnej weryfikacji i walidacji, w celu potwierdzenia (lub zaprzeczenia) słuszności swoich decyzji architektonicznych.

Opracowana metoda jest ukierunkowana na wdrażanie systemów rozproszonych, złożonych oraz systemów mniej złożonych, ale takich, których wdrożenie może odbyć się tylko raz (np. systemów pokładowych statków kosmicznych itp.), gdyż ciężar pracy przy wdrożeniu systemu w tej metodzie przeniesiono z procesów wykonawczych na procesy analityczne.

W pracy opisano również niektóre cechy języków wykorzystanych do zbudowania symulatora, środowisko symulacyjne i środowisko testowania, wraz z cechami pożądanymi dla tych środowisk.

Literatura

[1] LASZKO Ł., STASIAK A., *Planowanie złożonego wdrożenia systemu IT*, „Biuletyn IAiR", 31/2011, str. 55 – 78.

[2] SNOECK M., DEDENE G., *Experiences with object oriented model-driven development*, Software Technology and Engineering Practice, Proceedings., Eighth IEEE International Workshop on incorporating Computer Aided Software Engineering, pp. 143-153, 14-18, Jul 1997.

[3] KRUCHTEN P., *The Rational Unified Process: An Introduction (3th ed.)*. Boston, Addison-Wesley, 2003.

[4] SCHWABER K., BEEDLE M., *Agile Software Development with Scrum (1st ed.)*, Prentice Hall PTR, 2001.

[5] BECK K., ANDRES C., *Extreme Programming Explained: Embrace Change (2nd Edition)*, Addison-Wesley, 2004.

[6] ZIELIŃSKI Z., STASIAK A., DĄBROWSKI W., *UML Simulation of a Topology Configuration Model*, „Journal of Telecommunications and Information Technology", No. 4, 2012, pp. 1–8.

[7] BEN-ARI M., PNUELI A., MANNA Z., *The temporal logic of branching time*, „Acta Informatica 20", 1983, pp. 207-226.

[8] FAGAN M. E., *Design and Code inspections to reduce errors in program development*, „IBM Systems Journal", Vol. 15, No 3, 1976, pp. 182–211.

[9] IEEE Std 1490™-2011. *IEEE Guide – Adoption of the Project Management Institute (PMI®) Standard. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Fourth Edition*, Project Management Institute, 2008.

[10] *Action Language for Foundational UML (Alf), Concrete Syntax for a UML Action Language, FTF – Beta 2*, http://www.omg.org/spec/ALF/1.0/Beta2/PDF.

[11] Object Constraint Language, OMG Available Specification, Version 2.0, formal/06-05-01, http://www.omg.org/spec/OCL/2.0/PDF/.

Simulation of IT system complex deployment

ABSTRACT: Simulation method for verification and validation of deployment plans was presented. Using this approach at the design stage of a system, makes it easier to understand dynamics of the system and allows for early detection of design and architecture errors without the need for an actual product deployment. The method is based on the use of topology model (as a deployment plan) and the following languages: UML (description of deployment environment), OCL (constraints) and UAL (imperative semantic of actions). The use of these languages yielded highly accurate models which are the key element of the simulator. The simulation enabled examination of different deployment plans and selection of the one that met the constraints brought to the target environment.

KEYWORDS: simulation, deployment, topology model, UML, UAL, OCL.

Praca wpłynęła do redakcji: 25.04.2013 r.