BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 31, 2011

Planowanie złożonego wdrożenia systemu IT

Łukasz LASZKO, Andrzej STASIAK

Instytut Teleinformatyki i Automatyki WAT,
ul. Gen. S. Kaliskiego 2, 00-908 Warszawa
llaszko@wat.edu.pl, astasiak@wat.edu.pl

STRESZCZENIE: W artykule przedstawiono zestaw technik i modeli wspierających procesy planowania wdrożeń systemów IT. Złożoność tych procesów zilustrowano przykładem opisu wdrożenia produktów platformy Jazz wykorzystywanych w badaniach i dydaktyce na Wydziale Cybernetyki WAT. Dodatkowo wskazano kierunek rozwoju tych procesów: od modeli konfiguracji w języku UML, do kompletnych opisów wdrożeń w postaci modeli topologii. Modele topologii mogą być uruchamiane, co pozwala na prowadzenie eksperymentów w zakresie architektury oprogramowania.

SŁOWA KLUCZOWE: model topologii, model wdrożenia, planowanie wdrożenia, języki dziedzinowe, UML.

1. Wprowadzenie

W artykule przedstawiony został opis procesu planowania złożonego wdrożenia systemu IT na przykładzie wykorzystywanego w pracach badawczych oraz udostępnionego studentom Wojskowej Akademii Technicznej rozwiązania serwerowego opartego na produktach platformy IBM Jazz®[1]. Studenci i pracownicy mają również możliwość wykorzystania innej platformy, np. Microsoft® Team Foundation Server (TFS). Proces planowania wdrożenia, ze względu na swoją złożoność [7][2], [3], został udokumentowany z wykorzystaniem języka UML [1] i opracowanego przez IBM języka dziedzinowego (DSL) do opisu wdrożeń systemów IT [3], [4]. Podyktowane zostało to tym, że

[1] W artykule przyjęto, że system Jazz tworzy platforma i aplikacje działające w jej środowisku.
[2] Firma IBM w swoim przewodniku do instalacji wskazuje na trzy typy organizacji wdrożenia, od prostej, typu „evaluation", przez „department" (wybraną do wdrożenia na Wydziale Cybernetyki WAT) do korporacyjnej, typu „enterprise".

środki dostępne w UML, to jest diagramy: wdrożenia (ang. *deployment diagram*), komponentów (ang. *component diagram*) i klas (ang. *class diagram*), nie pozwoliły na pełną specyfikację (z wymaganą ekspresywnością) kompletu cech opisu konfiguracji (dla planowanego wdrożenia) [1]. Szczegółowość przedstawionych w artykule modeli została dobrana tak, aby można było zastosować techniki symulacyjnej weryfikacji modeli (UML i topologii), które pozwalają potwierdzić słuszność podjętych decyzji architektonicznych przed przeprowadzeniem wdrożenia. W artykule wykazujemy, że możliwe jest opracowanie modeli topologii, które adekwatnie opisują wdrożenie, a zaproponowana metoda jego planowania jest skuteczna.

Artykuł rozpoczynamy od krótkiej charakterystyki dostępnych technik prowadzących do opracowania modelu wdrożenia – koncentrując się na opisie wdrożenia rozwiązania serwerowego i określając jedynie podstawowe własności stacji klienckich. Zasadniczą część artykułu stanowią modele UML i opis ich powiązań z modelami topologii. W pracy przedstawiamy również praktyczne uwagi dotyczące przeprowadzonego wdrożenia, które naszym zdaniem są istotne dla planowania złożonych wdrożeń systemów IT.

2. Ewaluacja modeli wdrożenia

Obecnie najpopularniejszym sposobem na utrwalanie decyzji dotyczących wdrożeń systemów IT (krótko przedstawionym w kolejnym podrozdziale 2.1) jest dostarczenie specyfikacji wdrożenia, która zasadniczo bazuje na diagramach wdrożenia języka UML [6] i jest uzupełniana diagramami opisu struktury i zachowania.

Jednak środki te nie są wystarczające. Nie pozwalają w precyzyjny sposób opisać procesów konfigurowania wdrażanych systemów tak, aby opis ten stanowił szczegółowy przewodnik (plan wdrożenia). Dla zwiększenia precyzji modeli wdrożenia skuteczne są dwie drogi:

- pierwsza sprowadza się do dostarczenia dedykowanego profilu języka UML;
- druga zakłada zdefiniowanie własnego języka dziedzinowego (który nie bazuje na UML).

Rozwój własnego języka dziedzinowego wybrała firma IBM [2], [3]. Jest on dostarczany wraz platformą IBM Rational Software Architect [4], [5] jako narzędzie wsparcia procesów modelowania topologii i opisu prowadzonych wdrożeń systemów IT i zostanie szerzej opisany w podrozdziale 2.2.

2.1. Modele konfiguracji w języku UML

Diagramy wdrożenia UML pozwalają na utrwalenie decyzji dotyczących architektury systemu. Przedstawiają one:

- opis fizycznych elementów wdrożenia;
- związki między oprogramowaniem a sprzętem wchodzącym w skład systemu;
- procesy dystrybucji systemu.

Diagramy wdrożenia są zwykle opracowywane podczas definiowania architektury budowanego systemu oraz wdrażania jego oprogramowania i przedstawiają rozmieszczenie węzłów w systemie rozproszonym, artefakty, które są przechowywane w węzłach, oraz ich realizacje.

Węzły reprezentują urządzenia, takie jak komputery, czujniki, drukarki, jak również inne urządzenia tworzące środowisko wykonawcze (ang. *runtime*). Ścieżki komunikacyjne i linki (wiązania) na diagramach wdrożenia określają fizyczne połączenia w systemie.

2.2. Modele topologii

Planowanie wdrożenia i jego automatyzacji jest pomostem pomiędzy projektowaniem aplikacji i ich wdrażaniem, który nazywamy modelem topologii [2].

Modele topologii stosujemy, aby uprościć cykl życia aplikacji na trzy sposoby:

1. Twórcy aplikacji mogą a priori przyjąć, że ich aplikacje będą działać w środowisku wdrożenia (zamiast czekać na ich zbudowanie i weryfikację swoich decyzji).
2. Wdrożeniowiec może przewidzieć potrzeby aplikacji i zapewnić, że środowisko wdrażania pozwoli na ich poprawne działanie (nie tylko ze względu na wymagania technologiczne, ale i dodatkowe, dotyczące np. użyteczności, niezawodności, wydajności i skalowalności – zgodnie z modelem FURPS).
3. Architekci mogą wymusić stosowanie najlepszych praktyk i standardów korporacyjnych w zakresie tworzenia planów rozmieszczenia oraz badać historię scenariuszy wdrożeń, które zakończyły się sukcesem (jak i porażką).

W dużym uproszczeniu możemy przyjąć, że modele topologii definiują architekturę rozlokowania elementów systemu, odwołując się do dwóch pojęć: jednostka i link [4], [5], [7], gdzie:

- jednostkami są fragmenty aplikacji lub elementy infrastruktury wdrażania (w tym sprzęt serwera, serwer aplikacji, bazy danych i systemy operacyjne);
- linki określają relacje (związki) między jednostkami.

Na przykład aplikacja Web, aby mogła poprawnie działać, musi być uruchomiona na serwerze WWW. W topologii, która opisuje tę relację, jednostka reprezentująca aplikację internetową jest hostowana, czyli powinna być połączona związkiem typu hosting z serwerem WWW.

Rozszerzając wcześniejsze określenie modelu topologii (jako pomostu pomiędzy projektowaniem aplikacji a ich wdrażaniem), pod pojęciem model topologii systemu IT rozumieć będziemy celowy opis systemu, który obrazuje jego kształt, określony przez zasoby systemu i zależności między nimi. Ważną cechą podejścia topologicznego jest możliwość planowania, a następnie weryfikacji modelowanych scenariuszy rozlokowania.

Planowanie wdrożenia wiąże się z określeniem architektury rozlokowania elementów systemu (zarówno jako pojęć konceptualnych, jak i fizycznych) i jest trudnym procesem, który może doprowadzić do pojawienia się dużej liczby błędów. Ryzyko popełnienia błędu można ograniczyć przez zastosowanie się projektanta do następujących reguł:

- wprowadzenie procesu planowania wdrożenia we wczesnej fazie wytwarzania aplikacji (już podczas projektowania);
- ponowne użycie szablonów topologii, zdefiniowanych i zweryfikowanych we wcześniejszych wdrożeniach;
- zastosowanie wczesnej walidacji scenariusza architektury rozlokowania, a tym samym identyfikacja niekompatybilności artefaktów modelu.

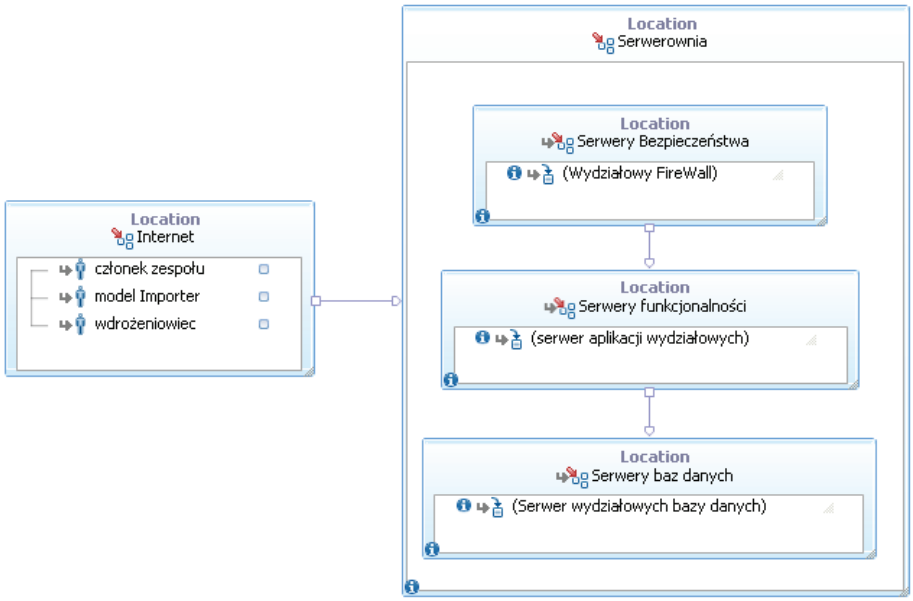
2.2.1. Konceptualne modele topologii

Do realizacji procesów planowania topologii aplikacji użytecznym narzędziem będzie edytor, który umożliwi włączanie elementów sprzętu komputerowego i oprogramowania do planu opisów, przed planowaniem szczegółów technologicznych. Za jego pomocą będzie możliwe tworzenie modeli wysokiego poziomu [2], [4].

Model logiczny poddawany jest badaniom mającym na celu zastosowanie modelowania operacyjnego, które w zakresie topologii jest podejściem do podejmowania decyzji o sprawnej architekturze systemu komputerowego. Podczas tworzenia modelu operacyjnego powstaje opis systemu na wysokim poziomie abstrakcji, w którym są wyszczególnione komponenty aplikacji oraz

ich powiązania, lokalizacja oraz rozwiązania dla postawionych wymagań biznesowych określonych w kontrakcie (rys. 1, rys. 2). Nie specyfikuje on natomiast szczegółów technologicznych aplikacji czy infrastruktury, takich jak system operacyjny czy serwer aplikacji, który będzie użyty. Topologia na tym poziomie abstrakcji odwołuje się do modeli logicznych (koncentrując się na prezentacji aspektu struktury systemu).

Na rys. 1 przedstawiono model konceptualny dla planowanego wdrożenia. W dalszej części artykułu przedstawimy jedynie opis serwerów funkcjonalności i baz danych, pomijając w nim aspekt bezpieczeństwa (komponenty tworzące „Wydziałowy FireWall").

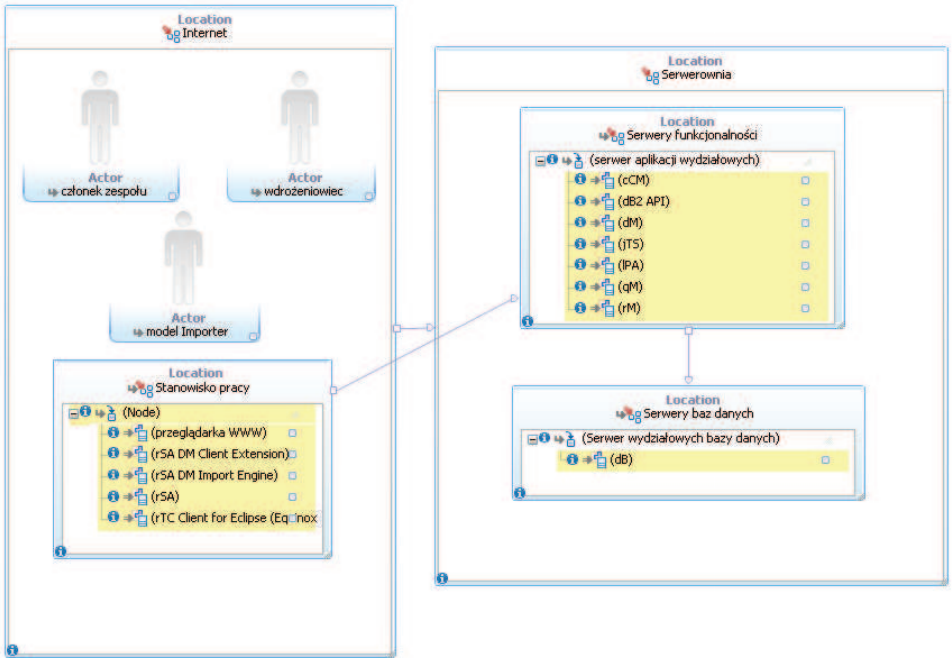


Rys. 1. Logiczny model wdrożenia serwera platformy Jazz

Przedstawiony na rys. 1 model specyfikuje jedynie zasoby serwera, natomiast na rys. 2 uzupełniono go o element „Stanowisko pracy" (jako aplikację kliencką). W modelu logicznym mogą być wykorzystane następujące elementy:

- Rozlokowanie – fizyczne, geograficzne miejsce albo strefa bezpieczeństwa (np. rys. 1: Serwerownia, Internet, Serwery Bezpieczeństwa, …);
- Aktor – osoba albo zewnętrzny system komputerowy, który jest w interakcji z modelowanym systemem (np. klient, administrator, czy, jak na rys. 1: wdrożeniowiec, członek zespołu, model importer – jako zewnętrzny system);

– Komponent – relokowalna część oprogramowania (np. na rys. 2: przeglądarka WWW, CCM, …);
– Węzeł – wymagana do działania, instalacji czy konfiguracji komponentu lub innego elementu programowego infrastruktura (np. na rys. 2: stanowisko pracy, serwer aplikacji wydziałowych, serwer wydziałowych baz danych);
– Jednostka konfiguracji – aspekt komponentu aplikacji, który czyni go niezależnym od środowiska działania aplikacji.



Rys. 2. Logiczny model wdrożenia typu „departament"

2.2.2. Fizyczne modele topologii

Topologia opisuje elementy i łącza między elementami. Elementy reprezentują części systemu komputerowego, takie jak np. sprzęt komputerowy, serwery, ich oprogramowanie, oraz obsługę systemów i komponentów aplikacji [3], [5]. Łącza reprezentują tu związki między tymi elementami, takie jak na przykład komunikacja.

Tworzenie samego diagramu topologii typowo realizowane jest w następujących dwóch krokach:

– budowa perspektywy konfiguracji;
– budowa topologii.

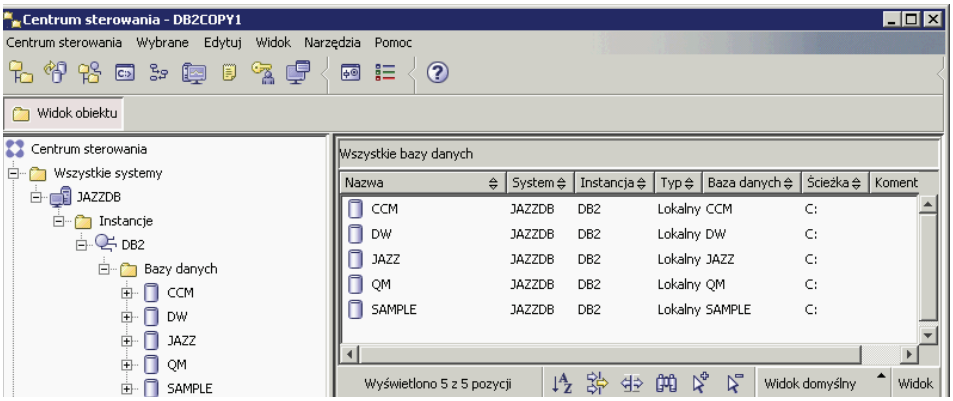
Na diagramie umieszczane są elementy, które reprezentują koncepcję budowanej aplikacji w postaci komponentów (np. komponentów WWW).

Elementy mogą być połączone między sobą za pomocą jednego z pięciu typów łączy [1]:

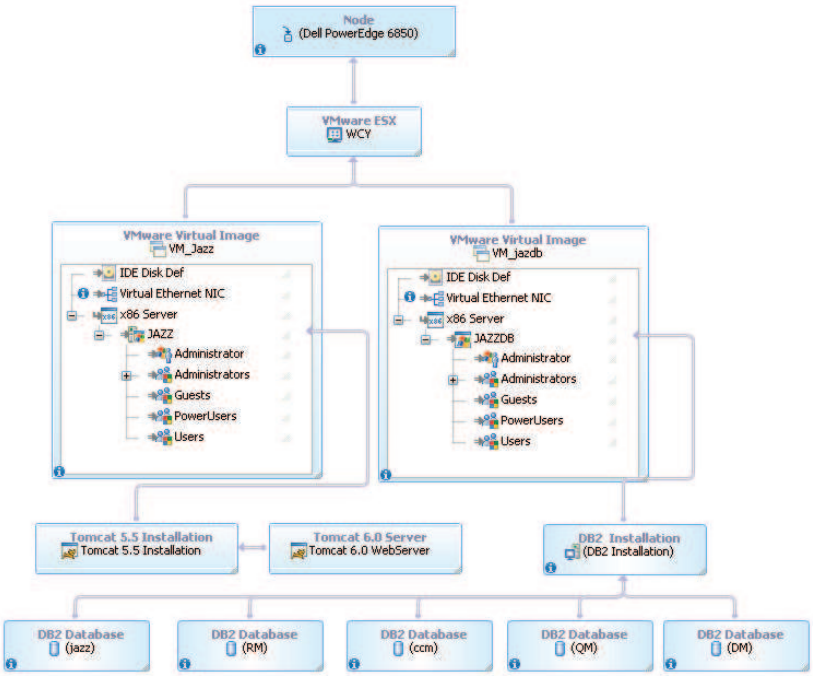
- Hostowanie (ang. *hosting link*) – łącze z elementami hostującymi wskazuje, że jeden element topologii („gość") jest goszczony (hostowany) przez inny element – „gospodarz" (tzn. „gospodarz" spełnia wszystkie wymagania „gościa");
- Realizacja (ang. *realization link*) – łącze realizacji wskazuje, że element wyjściowy jest specyficznym odwzorowaniem cech opisanych przez element źródłowy;
- Zależność (ang. *dependency link*) – łącze zależności wskazuje, że jeden element wymaga innego albo określa, czy usługa jest dostarczona przez ten element;
- Ograniczenia (ang. *constraint link*) – łącze ograniczeń występuje między dwoma elementami, wskazując na ograniczenia relacji między nimi;
- Należy do (ang. *membership link*) – łącze zawierania wskazuje, że jeden element zawarty jest w innym elemencie (należy do niego) albo jest członem innego elementu.

Narzędzia konfiguracji architektury skoncentrowane są wokół modelu topologii. Topologia, również i w tym zakresie, zawiera elementy, które mogą reprezentować różne byty (na różnych poziomach abstrakcji), włączając w to sprzęt komputerowy, oprogramowanie i ich zasoby konfiguracyjne, w tym np. użytkowników, grupy użytkowników itd. Na rys. 4 przedstawiono fragment modelu wdrożenia, określany jako fizyczny model topologii, tzn. model, który określa rzeczywiste elementy, które zostaną użyte podczas wdrożenia. Te rzeczywiste elementy są obrazami maszyn wirtualnych pracujących w środowisku serwera Dell PowerEdge 6850. Ciekawa jest możliwość definiowania kompletu szczegółów konfiguracyjnych nie tylko dla zasobów fizycznych, ale i wirtualizowanych (np. liczba procesorów i ich rdzeni, wielkość pamięci RAM). Warto tu również zauważyć, że model logiczny określony w poprzednim kroku może być zaimportowany do modelu fizycznego, a jego elementy mogą być ponownie wykorzystane do zbudowania modelu wdrożenia. Przykładem może tu być ponowne użycie programowych fizycznych elementów modelu stanowiących komponenty bazy danych, jako bazy danych odpowiadających im produktów platformy jazz, przedstawionych na (rys. 5).

Bazy danych, w rozumieniu modelu topologii wdrożenia, są hostowane na serwerze JAZZDB. W środowisku produkcyjnym, zgodnie z przedstawionym na rys. 4 modelem, pracują one pod kontrolą systemu operacyjnego Windows XP (lub jego emulacji, w środowisku wirtualizacji VMware).



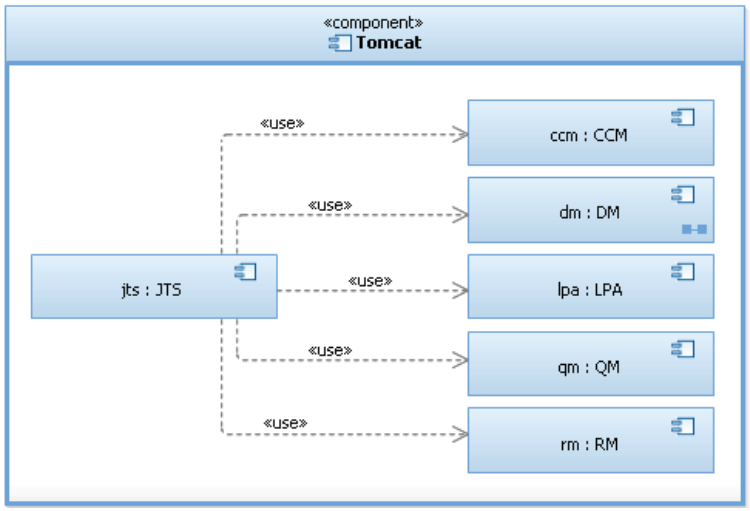
Rys. 3. Lista baz danych utworzonych na potrzeby systemu Jazz



Rys. 4. Fragment modelu realnego (fizycznego) wdrożenia

3. Planowanie wdrożenia systemu Jazz (JTS)

Jako przykład złożonego wdrożenia zostanie przedstawiona instalacja i konfiguracja systemu pracy grupowej IBM Jazz Team Server (JTS) [7][3] wraz z wybranymi aplikacjami rozszerzającymi jego funkcjonalność o administrowanie projektami cyklu życia (LPA), zarządzanie wymaganiami (RM), zarządzanie zmianą (CCM), testowanie (zarządzanie jakością, QM) oraz zarządzanie modelami graficznymi (publikowanie i komentowanie diagramów, DM) (rys. 5).



Rys. 5. Komponenty programowe utrzymywanego środowiska produkcyjnego

Na omawianej platformie JTS możliwe jest osadzenie również innych aplikacji udostępnianych w ramach inicjatywy Jazz[4]. Jednak te, które zostały wybrane, zdaniem autorów stanowią niezbędne minimum do budowania skutecznego wsparcia dla zespołów projektowych. Warto nadmienić, że wspomniane rozwiązanie alternatywne (TFS, które funkcjonalnie pokrywa obszary CCM, QM, RM, pomijając istotne dla dydaktyki obszary DM i LPA [8]), dostępne dla pracowników i studentów, nie jest obecnie powszechnie stosowane w środowisku akademickim, w naszej ocenie wynika to głównie z tak szerokiego zakresu możliwości użycia platformy Jazz (rys. 5).

[3] W artykule opisano wersję 3.0.1.
[4] Lista aplikacji udostępnionych w ramach inicjatywy Jazz dostępna jest pod następującym adresem: https://jazz.net/projects. JTS jest platformą przygotowaną do osadzania aplikacji webowych, zatem, z uwagi na luźne powiązania między aplikacjami, możliwe są różne konfiguracje aplikacji na tej platformie.

3.1. Instalacja

Wynikiem instalacji będzie zbudowana infrastruktura, która po konfiguracji komponentów programowych (rys. 5) będzie udostępniała zakładaną funkcjonalność systemu pracy grupowej[5]. Instalacja polega na scaleniu części rodziny aplikacji Jazz Team Server z aplikacją specjalizowaną, służącą do zarządzania modelami graficznymi (Design Management (DM) w wersji 3.0.1) tworzonymi w środowisku narzędzia CASE (IBM Rational Software Architect lub IBM Rational Rhapsody) [4], [5]. Firma IBM, właściciel inicjatywy Jazz, mimo dojrzałości środowiska rozwijanego w ramach tej inicjatywy (obecnie wydawana jest wersja 4) nie udostępniła kompletnego podręcznika instalacji dla architektury opisywanej w tej pracy [7], co stanowiło dla autorów dodatkową zachętę do udokumentowania tego procesu.

Według założeń opublikowanych na stronie inicjatywy Jazz, udostępniane są dwa sposoby instalacji[6]: instalacja z kompletnego archiwum oraz instalacja z sieci (Web)[7]. Bez względu na sposób instalacji omawiana wersja aplikacji zawiera, oprócz docelowych aplikacji, podstawowe licencje (w tym licencje czasowe) oraz serwer aplikacji[8] Apache Tomcat w wersji 5.5.30 (rys. 4, rys. 5). Alternatywnym serwerem aplikacji, zalecanym przez Jazz, jest IBM Websphere Application Server w wersji 6.1 lub nowszej[9].

Zaproponowany przez autorów proces instalacji przedstawiony jest na rys. 6. Wyróżnione w tym procesie aktywności pokazują działania, jakie należy wykonać, by zbudować środowisko projektu o założonej funkcjonalności. Opisany proces instalacji nie uwzględnia postępowania związanego z instalacją bazy danych, gdyż jest on opisany w materiałach udostępnianych przez IBM[10], z uwzględnieniem różnych systemów baz danych, takich jak: Apache Derby w wersji 10.5, IBM DB2 w wersji 9.5 i 9.7 Express, Workgroup i Enterprise, Microsoft SQL Server w wersji od Express 2005 SP3 do 2008 R2 oraz Oracle Database w wersji 10g i 11g Standard i Enterprise Release 2. W opisywanym rozwiązaniu wykorzystano bazę IBM DB2 Express w wersji 9.7. Zaznaczony na omawianym diagramie przepływ obiektów dodatkowo ukazuje zmianę w katalogu docelowym aplikacji DM, która spowodowana jest scaleniem zawartości tego katalogu z katalogiem JTS.

[5] https://jazz.net/projects/jazz-foundation oraz https://jazz.net/projects/design-management

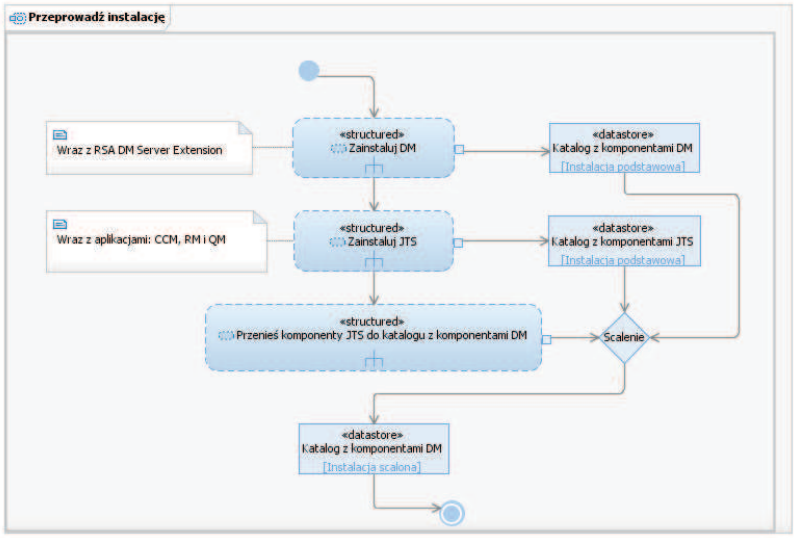
[6] Dostępne mechanizmy instalacji zależne są od wyboru docelowej platformy, por. https://jazz.net/downloads/jazz-foundation/releases/3.0.1?p=allDownloads
Design Management w omawianej wersji dostępny jest tylko na platformę MS Windows i Linux.

[7] Omawiane oprogramowanie udostępnia także możliwość pobrania zestawu SDK na potrzeby wykorzystania w projektach typu opensource.

[8] Precyzując, jest to serwer webowy i kontener serwletów.

[9] Wymagania wstępne dla JTS opisuje: https://jazz.net/library/article/632

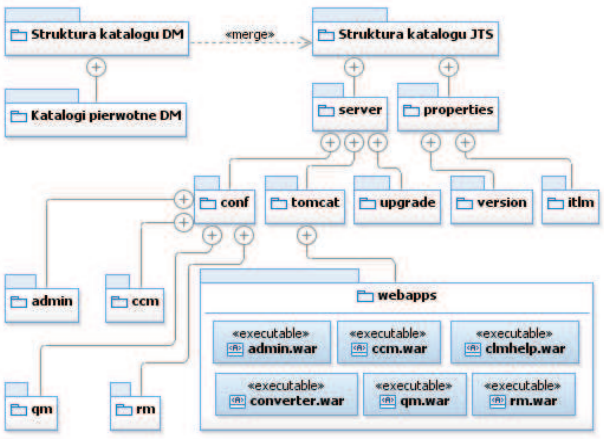
[10] Por. np. https://jazz.net/help-dev/clm/topic/com.ibm.jazz.install.doc/topics/t_s_server_installation_setup_db2.html



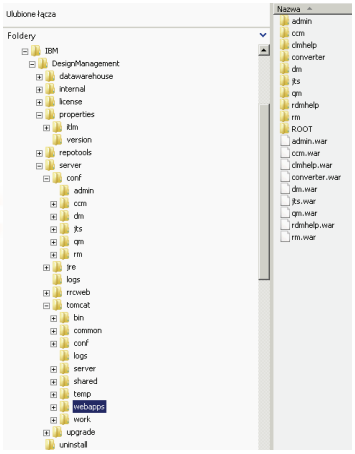
Rys. 6. Proces autorskiej instalacji systemu JTS

Wspomniane wyżej scalenie polega na przeniesieniu zawartości niektórych podkatalogów katalogu JTS do katalogu DM. Podkatalogi te przedstawiono na rys. 7a. Dla ukazania związku między pakietami wykorzystano notację zagnieżdżenia pakietu oraz zależność ze stereotypem «merge» oznaczającą scalenie zawartości dwóch pakietów. Zastosowanie tej zależności skutkuje utworzeniem spójnej przestrzeni nazewniczej (w tym przypadku katalogowej).

a)



b)



Rys. 7. a) Struktura elementów JTS wymaganych do scalenia w DM
b) Wynikowa struktura katalogu Design Management

Na rys. 7a. jawnie wskazano miejsce osadzenia artefaktów wykonywalnych (tutaj archiwów aplikacji webowych) w ramach jednego węzła wdrożenia. Zawartość pozostałych istotnych pakietów składa się z dokumentów konfiguracyjnych, licencji oraz skryptów. Te elementy nie zostały ukazane ze względu na zachowanie czytelności diagramu. Wynikową strukturę katalogów po instalacji przedstawiono na rys. 7b.

Opisany proces instalacji (organizacja wdrożenia typu „departament" [7]), z uwagi na charakter aplikacji udostępnianych w ramach inicjatywy Jazz[11], można stosować do opisu instalacji innych aplikacji rozszerzających funkcjonalność JTS. Ponadto autorzy zwracają uwagę na fakt, że komponenty aplikacji nie muszą być rozlokowane w ramach jednego węzła.

3.2. Konfiguracja

Zbudowana po instalacji infrastruktura projektu nie udostępnia jeszcze zakładanej funkcjonalności, to jest w pełni funkcjonalnego systemu pracy grupowej. Dzieje się tak z powodu braku powiązań między aplikacjami składającymi się na ten system. Powiązania te zostaną utworzone podczas procesu konfiguracji. Wymagania procesu konfiguracji przedstawiono w postaci przypadków użycia na diagramie na rys. 8. Wymagania pogrupowano w podsystemy[12] w celu:

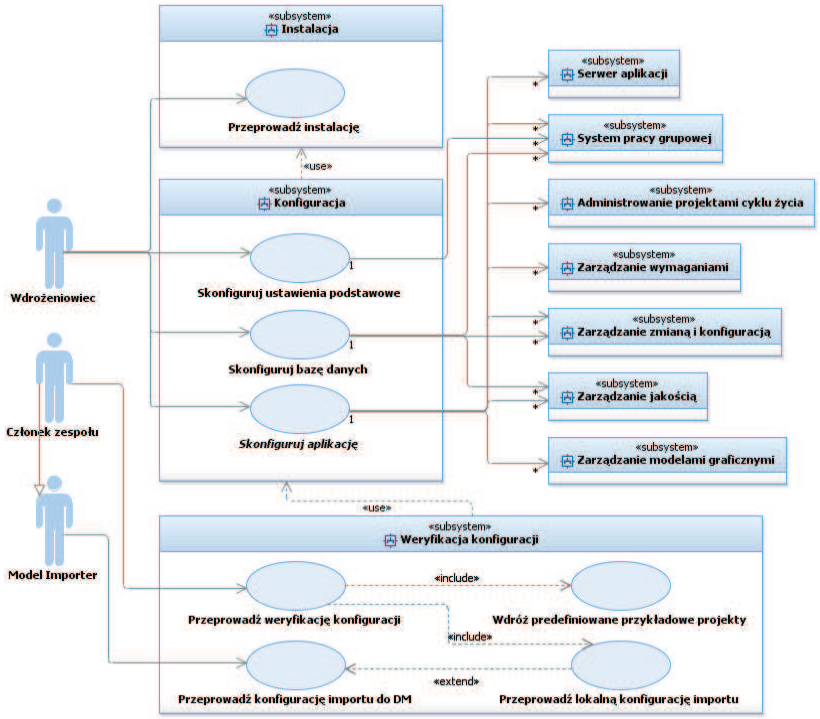
- jawnego wskazania zakresu działań opisywanych przez przypadki użycia;
- zaznaczenia aplikacji, których dane wymaganie dotyczy;
- wyróżnienia tych wymagań, które formułowane są wielokrotnie w stosunku do różnych aplikacji (ang. *reuse*).

Na potrzeby konfiguracji autorzy wyróżnili następujące role:

- Wdrożeniowiec;
- Uczestnik projektu;
- Model Importer.

Reprezentantami tych ról są aktorzy, których specyfikacji dokonano w tab. 1. Do zakresu zagadnień podejmowanych w niniejszym artykule mają zastosowanie jedynie te przypadki użycia, które wiążą się bezpośrednio z rolą Wdrożeniowca. Realizacje tych przypadków użycia omówione zostaną w dalszej części artykułu.

[11] Zakładający luźne powiązania między aplikacjami w ramach wspólnej platformy.
[12] Podsystem stanowi wyróżnioną funkcjonalnie logiczną część systemu nadrzędnego.



Rys. 8. Wymagania wobec procesu konfiguracji systemu JTS

Tab. 1. Specyfikacja aktorów

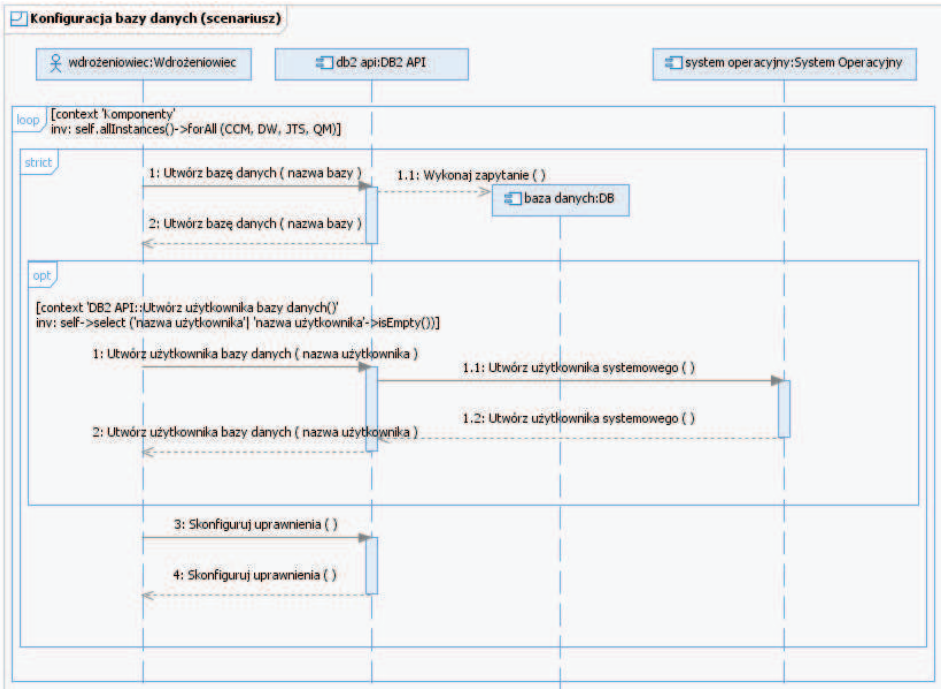
Lp.	Nazwa	Opis	Cel	Odpowiedzialność
1.	Wdrożeniowiec	Rola, której zadaniem jest dostarczenie użytkownikowi końcowemu kompletnej infrastruktury systemu pracy grupowej JTS	Zbudować i skonfigurować infrastrukturę systemu pracy grupowej JTS	– Zainstalowanie systemu JTS; – Konfiguracja systemu JTS; – Konfiguracja bazy danych.
2.	Członek zespołu	Rola, której zadaniem jest weryfikacja skonfigurowanej infrastruktury pod kątem poprawności współdziałania tworzących ją aplikacji	Zweryfikować poprawność konfiguracji infrastruktury systemu pracy grupowej JTS	– Wdrożenie predefiniowanych, przykładowych projektów
3.	Model Importer	Rola, której zadaniem jest konfiguracja importu modeli graficznych do aplikacji zarządzającej tymi modelami (DM)	Skonfigurować import modeli graficznych do DM	– Skonfigurowanie lokalnego importu do DM – Skonfigurowanie importu do DM z systemu kontroli wersji

3.2.1. Utworzenie baz danych

Konfiguracja systemu poprzedzona zostanie przygotowaniem baz wykorzystywanych przez aplikacje systemu Jazz (rys. 5). Przy tworzeniu bazy danych można wykorzystać wbudowany w system zarządzania bazą danych kreator lub wykonać proste polecenie SQL, np. polecenie tworzące bazę danych JAZZ w katalogu głównym dysku C: ma postać:

```
create database JAZZ on c: using codeset UTF-8 territory en PAGESIZE 16384
```

Na potrzeby omawianego rozwiązania należy w analogiczny sposób utworzyć następujące bazy danych: CCM, DW, QM. Scenariusz realizacji tego zadania opisany jest na rys. 9. Lista utworzonych w ten sposób baz widoczna jest na rys. 3.



Rys. 9. Scenariusz konfiguracji bazy danych

Do każdej z utworzonych baz należy przypisać użytkownika, który jest jednocześnie użytkownikiem systemu operacyjnego[13], po czym wypromować go

[13] W celu zwiększenia bezpieczeństwa należy ograniczyć możliwości logowania użytkownika do systemu, wyłączając dla niego logowanie w trybie interakcyjnym, a pozostawiając dopuszczalne jedynie logowanie w trybie usługi.

na administratora bazy danych (DBADM), za pomocą kreatora bądź wykonując proste polecenie SQL. Tak skonfigurowane bazy danych stanową główny magazyn danych przetwarzanych przez system pracy grupowej.

3.2.2. Konfiguracja systemu Jazz

3.2.2.1. Konfiguracja ustawień podstawowych

Dla opisywanego wdrożenia opracowano scenariusze konfiguracji, które są opisane jako dialog między komponentami systemu. Scenariusz opisujący konfigurację ustawień podstawowych systemu Jazz Team Server, w oparciu o wbudowany kreator, przedstawia rys. 10. Dostęp do kreatora odbywa się z wykorzystaniem przeglądarki internetowej, po wprowadzeniu adresu serwera JTS, w następującej postaci:

```
https://{adres serwera}:9443/jts/setup
```

przy wykorzystaniu domyślnych poświadczeń administratora [7]: (login: `ADMIN`; hasło: `ADMIN`). Konfiguracja rozpoczyna się ustanowieniem publicznego adresu URI (ang. *Uniform Resource Identifier*). Adres ten jest niezbędny na potrzeby późniejszej komunikacji z aplikacją Jazz oraz z pozostałymi aplikacjami zależnymi. Zgodnie z ostrzeżeniem wypisywanym w oknie kreatora raz ustanowionego adresu URI nie można modyfikować[14] (por. rys. 11).

Kolejnym krokiem jest konfiguracja połączenia z bazą danych (rys. 12) oraz utworzenie w jej obszarze wymaganych tabel. Tworzenia tabel można dokonać w kreatorze, bądź za pomocą dostarczonego skryptu[15], znajdującego się w katalogu instalacji:

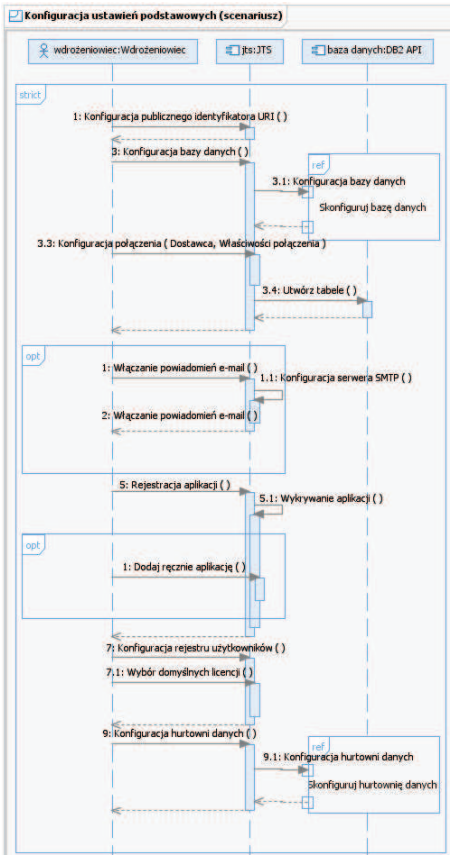
```
repotools-jts.bat -createTables
```

Pozostałe czynności konfiguracyjne, według scenariusza, dotyczą powiadomień e-mail, rejestracji aplikacji, rejestru użytkowników oraz hurtowni danych. Powiadomienia spełniają bardzo istotną rolę w omawianym systemie. Umożliwiają między innymi: rozsyłanie powiadomień grupowych, informowanie o zdarzeniach oraz zapraszanie nowych członków zespołów. W kreatorze można wyspecyfikować tzw. białą listę odbiorów wiadomości, dzięki której możliwe jest ograniczenie liczby adresatów. Brakuje jednak możliwości określenia zakresu powiadomień (filtra zdarzeń istotnych), co przy dużej liczbie zdarzeń, stanowi potencjalne zagrożenie zalewania skrzynek pocztowych nieistotnymi wiadomościami oraz, przypuszczalnie, blokowania adresu wybranego serwera SMTP.

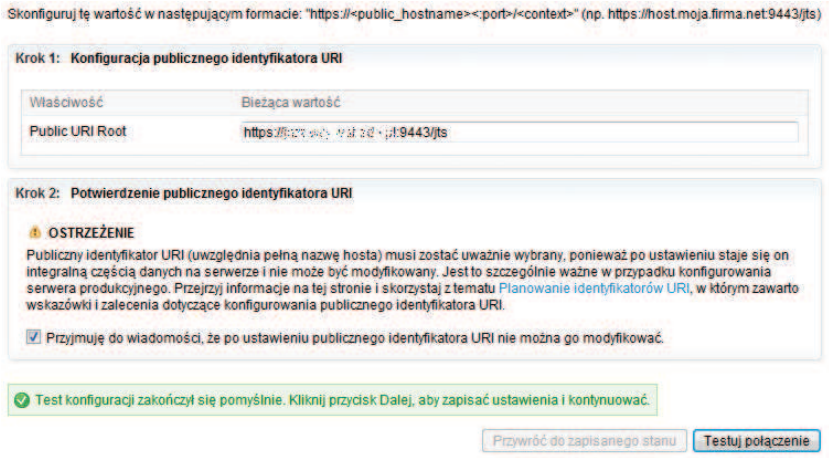
[14] Por. pkt 3.2.3.
[15] jw.

W następnym kroku dokonywana jest rejestracja aplikacji. Jest to kluczowy element wiązania aplikacji zainstalowanych na platformie Jazz. W tym punkcie dochodzi do ustanowienia instancji relacji między komponentami poprzez utworzenie wiązań topologicznych. Powołanie wiązań między komponentami umożliwia komunikację między nimi.

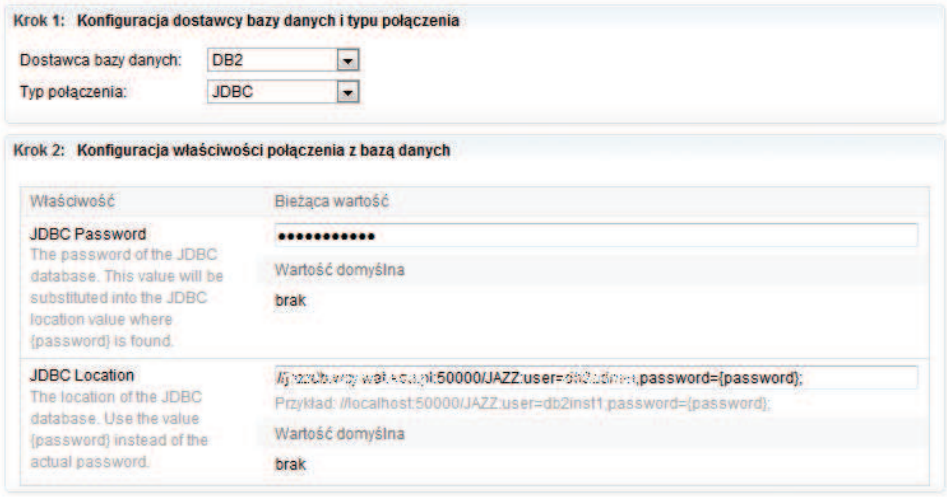
Z uwagi na luźne powiązania między aplikacjami, topologia systemu może podlegać zmianom (rozszerzeniom lub zawężeniom, poprzez dodawanie lub usuwanie komponentów, co może odbywać się również w czasie działania w środowisku produkcyjnym), podczas rejestracji nowym aplikacjom przypisywane są unikalne klucze konsumenta, które służą ich uwierzytelnianiu względem platformy Jazz. Aplikacje nieuwierzytelnione nie są dostępne (widoczne) z poziomu serwera JTS. Wynik rejestracji aplikacji pokazano na rys. 13.



Rys. 10. Scenariusz konfiguracji systemu JTS



Rys. 11. Konfiguracja URI



Rys. 12. Konfiguracja połączenia z bazą danych

Następna czynność wykonywana podczas konfiguracji dotyczy wyboru rejestru użytkowników systemu Jazz. Domyślnie informacje o użytkownikach magazynowane są w plikach serwera Tomcat, możliwe jest także wykorzystanie usługi LDAP lub innego zewnętrznego rejestru. W przypadku wyboru usługi LDAP należy podać dodatkowe wartości wymaganych parametrów[16], natomiast w sytuacji wyboru ustawień domyślnych taka potrzeba nie występuje.

Przykład wykorzystania lokalnego magazynu użytkowników serwera Tomcat przedstawia rys. 14.

Rys. 13. Wynik rejestracji aplikacji

[16] Pomocny przy określaniu tych wartości jest następujący artykuł dostępny pod adresem: http://www-01.ibm.com/support/docview.wss?uid=swg21445366

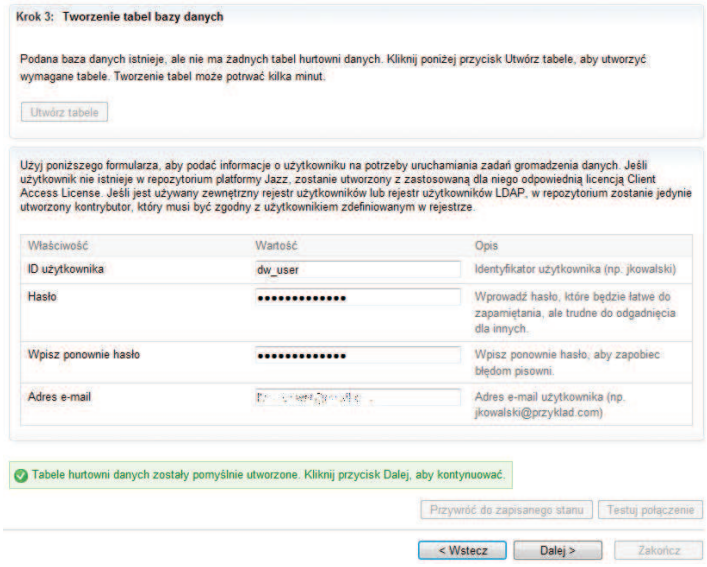
Oprócz wskazania rejestru użytkowników można także wybrać licencje domyślne dla nowo tworzonych użytkowników[17]. Warto zauważyć, że aplikacje osadzone na platformie Jazz posiadają oddzielne licencjonowane. Jazz w tym przypadku odpowiada jedynie za zarządzanie tymi licencjami.

```
<?xml version="1.0" encoding="UTF-8"?>
-<tomcat-users>
    <role rolename="JazzDWAdmins"/>
    <role rolename="JazzGuests"/>
    <role rolename="JazzUsers"/>
    <role rolename="JazzProjectAdmins"/>
    <role rolename="JazzAdmins"/>
    <user roles="JazzUsers" fullName="Sally" password="fc80b22ff203a1a88470b19ab19228044d066d66" username="sally"/>
    <user roles="JazzUsers" fullName="Tammy" password="3f242793018adad59211fb958f8dc084cd9f5a14" username="tammy"/>
```

Rys. 14. Fragment zawartości magazynu użytkowników serwera Tomcat

Ostatnim krokiem konfiguracji ustawień podstawowych jest konfiguracja hurtowni danych, która wykorzystywania jest do raportowania i analiz. Proces konfiguracji niewiele różni się od konfiguracji bazy danych. Różnica polega na utworzeniu nowego użytkownika, który będzie odpowiedzialny za gromadzenie danych. Po wprowadzeniu wymaganych parametrów konfiguracji hurtowni danych, należy dokonać utworzenia tabel. Czynność tę można wykonać przy pomocy kreatora (rys. 15) bądź z wykorzystaniem skryptu:

```
repotools-jts.bat -createWarehouse
```



Rys. 15. Konfiguracja hurtowni danych

[17] Istnieje możliwość zainstalowania licencji przed konfiguracją systemu. Można to zrobić w kreatorze: https://{adres_serwera}:9443/jts/admin#action= com.ibm.team.repository.admin.manageLicenses.

Choć część konfiguracji JTS dotycząca ustawień podstawowych została zakończona, to (zawartość pliku konfiguracyjnego przedstawia rys. 16), z uwagi na zależność aplikacji osadzonych na omawianej platformie od samej platformy (rys. 2, rys. 5), proces konfiguracji systemu pracy grupowej nie został zakończony.

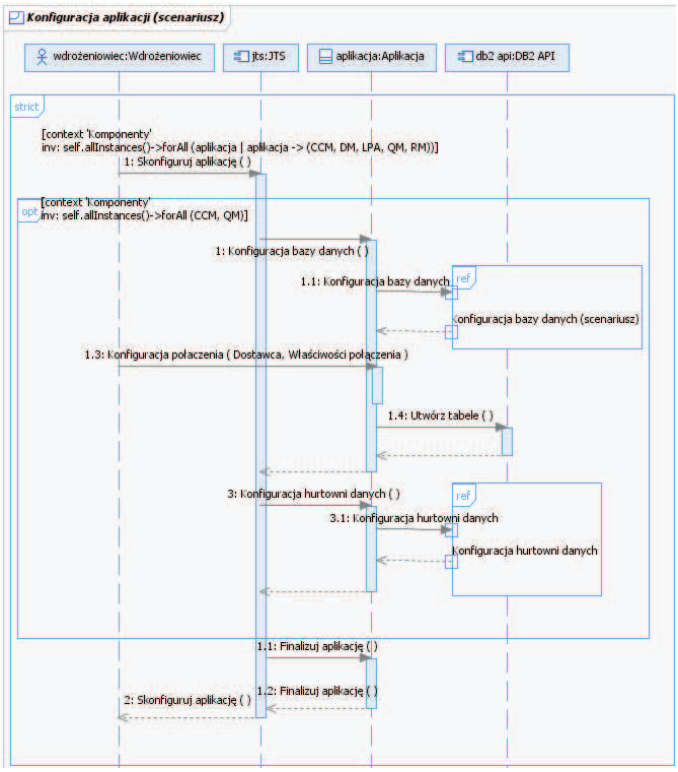
```
com.ibm.team.repository.user.registry.type=DETECT
com.ibm.team.repository.db.jdbc.location=//{adres serwera bd}:50000/JAZZ\:user\={nazwa użytkowanika};password\={password};
com.ibm.team.repository.db.jdbc.password={hasło w postacji zaszyfrowanej}
com.ibm.team.repository.db.repoLockId={klucz blokady bd}
com.ibm.team.repository.db.vendor=db2
com.ibm.team.repository.changeEvent.expirationDefault=1209600
com.ibm.team.repository.server.webapp.url=https\://{adres serwera}\:9443/jts
com.ibm.team.repository.oauth.accessToken.timeout=7200
com.ibm.team.repository.notification.mail.smtp.user={nazwa użytkownika konta e-mail}
com.ibm.team.repository.notification.mail.from={adres e-mail}
com.ibm.team.repository.notification.mail.smtp.starttls=true
com.ibm.team.repository.notification.mail.smtp.server.port=587
com.ibm.team.repository.notification.mail.smtp.server={serwer smtp}
com.ibm.team.repository.notification.mail.enabled=true
com.ibm.team.repository.notification.mail.smtp.password={hasło w postacji zaszyfrowanej}
com.ibm.team.repository.licenseService.floatingLicenseServerURI=https\://{adres serwera}\:9443/jts/
com.ibm.team.repository.web.suppressedPages={"com.ibm.team.repository.web.admin"\: ["com.ibm.team.repository.provision"]}
com.ibm.team.repository.web.helpuri=/rdmhelp/index.jsp
com.ibm.team.datawarehouse.db.net.port=1527
com.ibm.team.datawarehouse.db.vendor=db2
com.ibm.team.datawarehouse.db.jdbc.password={hasło w postacji zaszyfrowanej}
com.ibm.team.datawarehouse.datawarehouse.provider=Remote
com.ibm.team.datawarehouse.db.automatic.setup=true
com.ibm.team.datawarehouse.db.jdbc.location=//{adres serwera bd}\:50000/DW\:user\={nazwa użytkownika};password\={password};
com.ibm.team.datawarehouse.auth.userId=dw_user
com.ibm.team.datawarehouse.auth.password={hasło w postacji zaszyfrowanej}
com.ibm.team.jfs.index.root.directory=indices
com.ibm.team.fulltext.indexLocation=conf/jts/indices/workitemindex
```

Rys. 16. Zawartość pliku konfiguracyjnego JTS (*teamserver.properties*) po zakończonym procesie konfiguracji ustawień podstawowych

3.2.2.2. Konfiguracja aplikacji

Aplikacje, które zostały osadzone i zarejestrowane na platformie Jazz, nie udostępniają jeszcze zakładanej funkcjonalności, por. pkt 3.2, gdyż same aplikacje nie zostały skonfigurowane. W stosunku do wybranych aplikacji proces konfiguracji będzie podobny, choć w niewielkim stopniu różny, przedstawia go diagram z rys. 17, drobne różnice w konfiguracji wypływają ze specyfiki aplikacji, co zostało oznaczone na rysunku przez fragment typu **opt**.

Wymaganie: „*Skonfiguruj aplikację”* wskazane na rys. 8, według wyżej zauważonych rozbieżności w konfiguracji, opisane zostało jako abstrakcyjne, co oznacza, że wymagania pochodne w stosunku do niego będą konkretyzowane, w zależności od danego podsystemu pozostającego w relacji z wymaganiem abstrakcyjnym. Na tym kończą się procesy instalacji i konfiguracji sytemu Jazz.



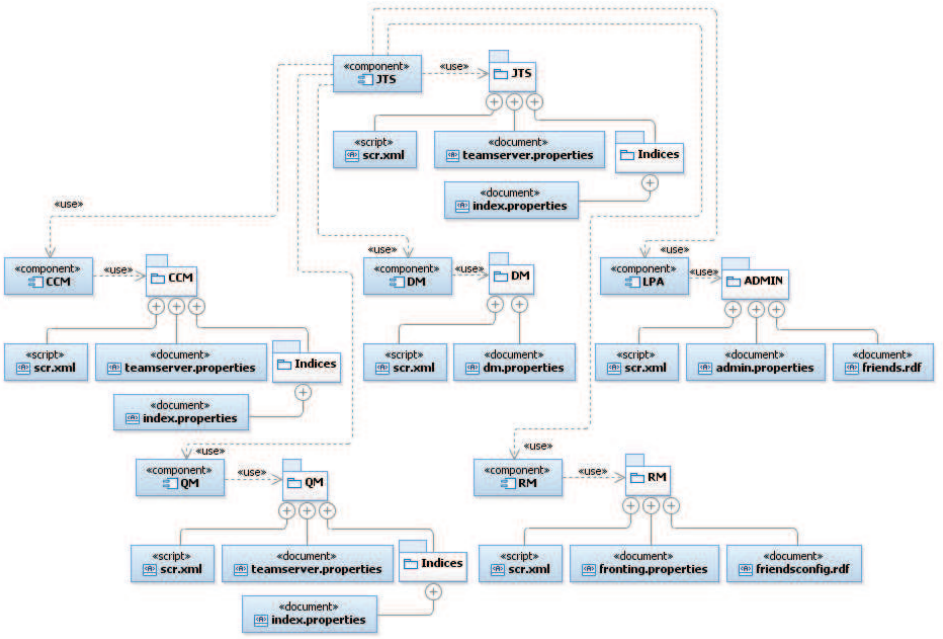
Rys. 17. Konfiguracja aplikacji

3.2.3. Praktyczne uwagi do przeprowadzonego wdrożenia

W rozdziale tym przedstawiono praktyczne uwagi (najlepsze praktyki) dotyczące przeprowadzonego wdrożenia, które naszym zdaniem mogą mieć istotne zastosowanie przy planowaniu i przeprowadzaniu wdrożenia oraz modyfikacjach, takich jak: zmiana rozmieszczenia aplikacji, zmiana lokalizacji serwera (w tym adresu URI), itp.

Opisany w pkt. 3.1 proces instalacji autorzy przeprowadzili kilkakrotnie, częściowo również bez użycia kreatora, tzn. modyfikując zapisy w plikach konfiguracyjnych JTS oraz innych aplikacji (rys. 18, zmiany wymagają artefakty ze stereotypem *document*). Możliwość samodzielnej modyfikacji tych plików dała podstawę do eksperymentu przeniesienia infrastruktury do innej lokalizacji i zmiany adresu URI, co nie jest możliwe do wykonania przy pomocy standardowego kreatora[18] (por. rys. 11).

[18] Podręcznik instalacji tej wersji zaznacza wprost, że nie jest możliwa taka zmiana, jedyna wskazywana możliwość obejścia tego ograniczenia istnieje w związku z modyfikacją zapisów w DNS (pozostawiając niezmienioną nazwę hosta), por.



Rys. 18. Komponenty konfiguracyjne systemu JTS

Przeprowadzono kilka prób, w których:

- udało się:
  - przenieść lokalizację serwera;
  - zmodyfikować adres URI;
  - przywrócić poprawność wiązań w obszarze topologii (w tym między aplikacjami oraz aplikacjami a bazą danych);
  - podmienić komponent zarejestrowany z niezarejestrowanym tego samego typu (łącznie z podmianą klucza konsumenta);
  - zachować poprawność i ciągłość licencjonowania po zmianie lokalizacji serwera;
  - zachować konta użytkowników i ich licencje po zmianie lokalizacji serwera;
- nie udało się:
  - przywrócić współdziałania obszarów projektów w ramach cyklu życia projektu w nowej lokalizacji serwera.

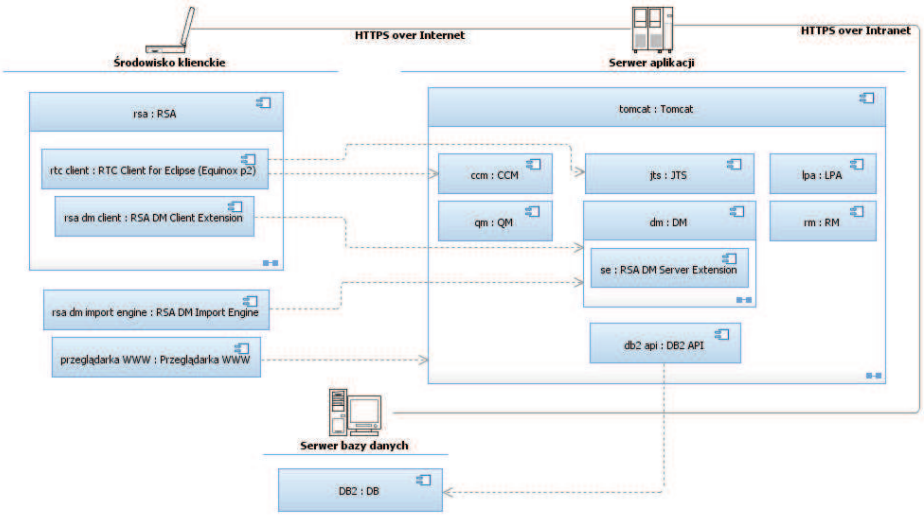
http://publib.boulder.ibm.com/infocenter/clmhelp/v3r0/topic/com.ibm.jazz.install.doc/topics/c_planning_URLs.html

Wszystkie eksperymenty przeprowadzone zostały w środowisku wirtualizacji (rys. 4). Uzyskane wyniki mogą stanowić zalecenia do planowania wdrożeń systemu Jazz w oparciu o rozwiązania szablonowe, które mogą posłużyć budowaniu infrastruktury:

– o wysokiej dostępności (ang. *high availability systems*) – przez utworzenie węzłów redundantnych;
– o szybkiej naprawialności (ang. *reparable systems*) – przez wymianę komponentu lub całego węzła.

Należy zwrócić uwagę, że wybór odpowiedniej wersji JTS oraz innych aplikacji jest kluczowy dla przeprowadzenia omawianego wdrożenia. Autorzy zalecają wykorzystywanie wyłącznie wersji oznaczonych jako **wydanie** (ang. *Release*). Do pozostałych udostępnionych wersji, tzn. *Beta*, *Milestone* i *Release Candidate*, brakuje dokumentacji, a archiwa instalacji mogą zawierać różne wersje aplikacji, co może stwarzać dodatkowe problemy przy wdrażaniu tego rozwiązania.

Modelowy opis przeprowadzonego wdrożenia przedstawiono na diagramie wdrożenia UML, na rys. 19.



Rys. 19. Architektura systemu pracy grupowej Jazz przedstawiona na diagramie wdrożenia UML

4. Podsumowanie

W artykule przedstawiono ogólną metodę realizacji procesu planowania wdrożenia bazującą na koncepcji MDD (ang. *Model Driven Development*), w której proces wdrożenia opisany jest za pomocą modeli: UML i topologii systemu. Modele topologii są dialektem języka UML przeznaczonym dla architektów i wdrożeniowców oprogramowania, i to one pozwoliły nam, z wymaganą dla automatyzacji procesów wdrożenia szczegółowością, opisać ten proces w specjalizowanym języku dziedzinowym i zweryfikować go na przykładzie konkretnego wdrożenia, zrealizowanego na Wydziale Cybernetyki WAT.

Dodatkową korzyścią płynącą z opracowania modeli topologii jest możliwość ich symulacji, bez konieczności oczekiwania wdrożeniowców na weryfikację założeń architektonicznych na „fizycznych zasobach systemu". Weryfikację tę możemy przeprowadzić, zanim te zasoby zostaną zbudowane i, prowadząc eksperymenty, określić, które z opracowanych wzorców topologii będą akceptowalne, a które należy odrzucić.

Literatura

[1] DĄBROWSKI W., STASIAK A., MARKOWSKI K., *Modeling using configuration topology*, „Przegląd Elektrotechniczny", 9/2010, pp. 239–242.

[2] *Planning deployment with the topology editor*, IBM Tutorial, IBM, 2008.

[3] *Modeling deployment topologies*, IBM Tutorial, IBM, 2008.

[4] NARINDER M., *Anatomy of a topology model used in IBM Rational Software Architect Version 7.5: Part 1: Deployment modeling*, IBM, 2008.

[5] NARINDER M., *Anatomy of a topology model used in IBM Rational Software Architect Version 7.5: Part 2: Advanced concepts*, IBM, 2008.

[6] DĄBROWSKI W., STASIAK A., WOLSKI M., *Modelowanie systemów informatycznych w języku UML 2.1*, PWN, Warszawa, 2007.

[7] *Jazz Team Server 3.0.1, Interactive Installation Guide*, IBM, http://publib.boulder.ibm.com/infocenter/clmhelp/v3r0m1/topic/com.ibm.jazz.install.doc/topics/roadmap_form.html.

[8] *Team Foundation Server, Overview*, Microsoft, http://www.microsoft.com/visualstudio/en-us/products/2010-editions/team-foundation-server/overview.

Planning of IT system complex deployment

ABSTRACT: This paper presents a set of techniques as well as models supporting planning of IT systems deployment. The complexity of such processes was depicted using an example of the Jazz platform deployment which is widely use in research and didactics on Cybernetics Faculty at the MUT. Additionally, development directions for such processes were indicated, starting from UML's deployment models and ending in comprehensive topology models which provide the basement for architectural experiments.

KEYWORDS: topology model, deployment model, deployment planning, domain languages, UML.

Praca wpłynęła do redakcji: 16.12.2011