

VIII Konferencja PLOUG
Kościelisko
Październik 2002

Modelowanie systemów w architekturze J2EE z wykorzystaniem notacji UML

Piotr Wilk

Premium Technology Sp. z o.o.

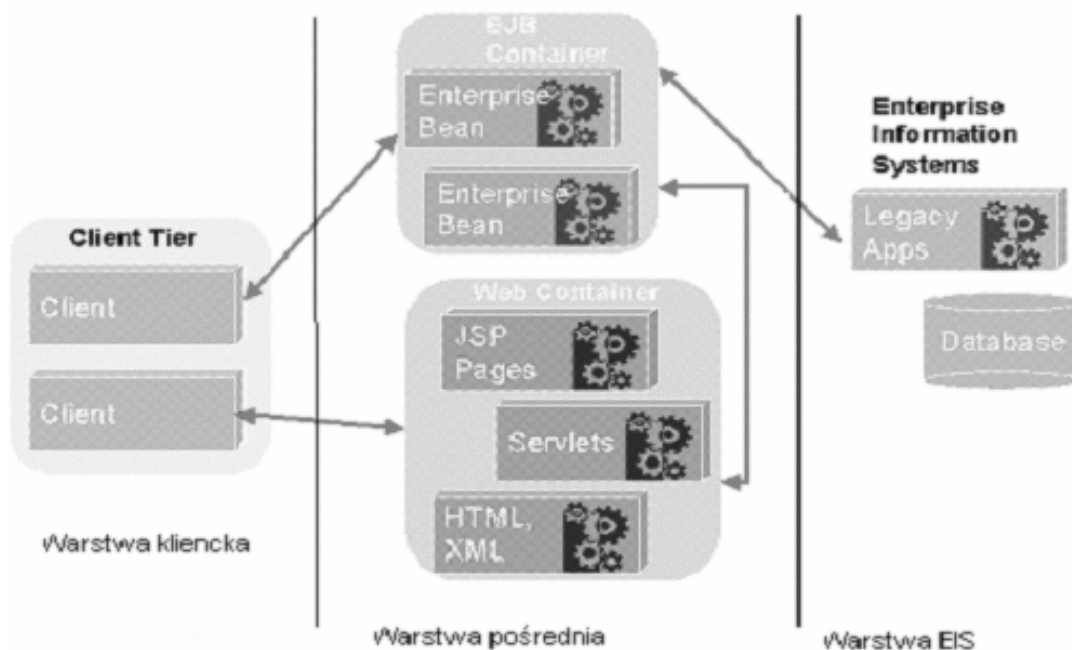
PWilk@PremiumTechnology.pl

1. Wprowadzenie

Tworzenie systemów informatycznych w oparciu o architekturę J2EE (Java Enterprise Edition) zdobywa coraz większą popularność. Liczne zastosowania praktyczne tej architektury dowiodły też, że systemy tworzone w jej oparciu są wydajne i zaspokajają potrzeby biznesowe, dla których zostały stworzone. Budowanie coraz bardziej skomplikowanych systemów w oparciu o tę architekturę wymaga jednak stosowania systematycznego podejścia do analizy i projektowania oraz przestrzegania procesu wytwórczego, tak aby systemy te były budowane w sposób powtarzalny, aby powstawały na czas i zaspokajały potrzeby klientów, dla których zostały zbudowane. Artykuł ten opisuje w jaki sposób zastosować proces wytwórczy RUP (Rational Unified Process) oraz notację UML w celu analizy i projektowania systemów informatycznych stworzonych w architekturze J2EE.

2. Wprowadzenie do architektury J2EE

Architektura J2EE umożliwia programistom tworzenie rozproszonych transakcyjnych systemów informatycznych wykorzystujących szybkość, bezpieczeństwo i niezawodność technologii serwerowych. Architektura J2EE wykorzystuje wielowarstwowy model aplikacji. Oznacza to, że cały system podzielony jest na pewien zbiór współpracujących ze sobą warstw pokazanych na poniższym rysunku:



Rys. 1. Architektura J2EE

Elementy wchodzące w skład warstwy klienckiej odpowiadają za obsługę interfejsu użytkownika, najczęściej stosowane rozwiązania to tworzenie tzw. cienkiego klienta, czyli umieszczenie po stronie klienckiej tylko i wyłącznie mechanizmów związanych z jego wyświetlaniem i obsługą operacji dokonywanych na interfejsie przez użytkownika. W przypadku architektury J2EE mamy możliwość zbudowania warstwy klienckiej na dwa sposoby. Warstwa kliencka naszego systemu może być zbudowana w oparciu o technologię WEB bądź też może być standardową aplikacją wykorzystującą graficzny interfejs użytkownika.

Elementy wchodzące w skład warstwy pośredniej odpowiadają przede wszystkim za obsługę logiki biznesowej naszego systemu. To właśnie w tym miejscu zgłoszenia użytkownika proszącego system o wykonanie jakiejś operacji za pośrednictwem warstwy klienckiej są obsługiwane i przetwarzane za pomocą komponentów biznesowych zbudowanych w oparciu o technologię EJB (Enterprise Java Beans).

Elementy wchodzące w skład warstwy EIS(Enterprise Information Systems) odpowiadają przede wszystkim za realizację mechanizmu trwałości dla naszego systemu informatycznego. To w tym miejscu obiekty biznesowe przetwarzane po stronie warstwy pośredniej są utrwalane z wykorzystaniem np. relacyjnej bazy danych. Tutaj też mogą znaleźć się istniejące już aplikacje biznesowe, z którymi nasz system powinien współpracować.

3. Komponenty EJB

Komponent EJB jest napisany w języku Java komponentem działającym po stronie serwera zawierającym logikę biznesową aplikacji. Komponenty te są tworzone w oparciu o specyfikację EJB (Enterprise Java Beans) firmy Sun. Specyfikacja ta opisuje model komponentów działających po stronie serwera.

Komponenty EJB działają w ramach kontenera EJB, który jest środowiskiem uruchomieniowym i odpowiada za obsługę ich cyklu życia oraz dostarcza im pewnych usług takich jak np. transakcje rozproszone, trwałość, bezpieczeństwo, obsługa wielowątkowości itp. Pełen zakres odpowiedzialności kontenera EJB opisany jest w specyfikacji EJB, co umożliwia uniezależnienie się od konkretnego producenta kontenerów. Sam komponent EJB składa się z następujących części:

- *Enterprise Bean* – klasa napisana w języku Java, która zawiera implementację metod biznesowych udostępnianych przez komponent EJB. Zgodnie ze specyfikacją EJB autor tej klasy podczas jej implementowania nie musi przejmować się zapewnieniem takich elementów jak autoryzacja klientów korzystających z komponentu, czy też np. wielowątkowości. Zapewnienie takich usług spada na barki kontenera EJB co wydatnie upraszcza konstrukcję klasy Enterprise Bean i pozwala skupić się programiście tylko i wyłącznie na logice biznesowej.
- *Interfejs Home* - interfejs ten definiuje metody, które pozwalają klientowi zlokalizować, stworzyć bądź usunąć instancję komponentu EJB.
- *Interfejs Remote*- interfejs ten definiuje wszystkie operacje wchodzące w skład logiki komponentu EJB. Klasa Enterprise Bean dostarcza realizację tych operacji.
- *Deployment Descriptor* – jest to plik zapisany w formacie XML, zawierający informacje o komponencie EJB, jak również ustawienia pewnych parametrów konfiguracyjnych, które mają wpływ na pracę komponentu EJB.

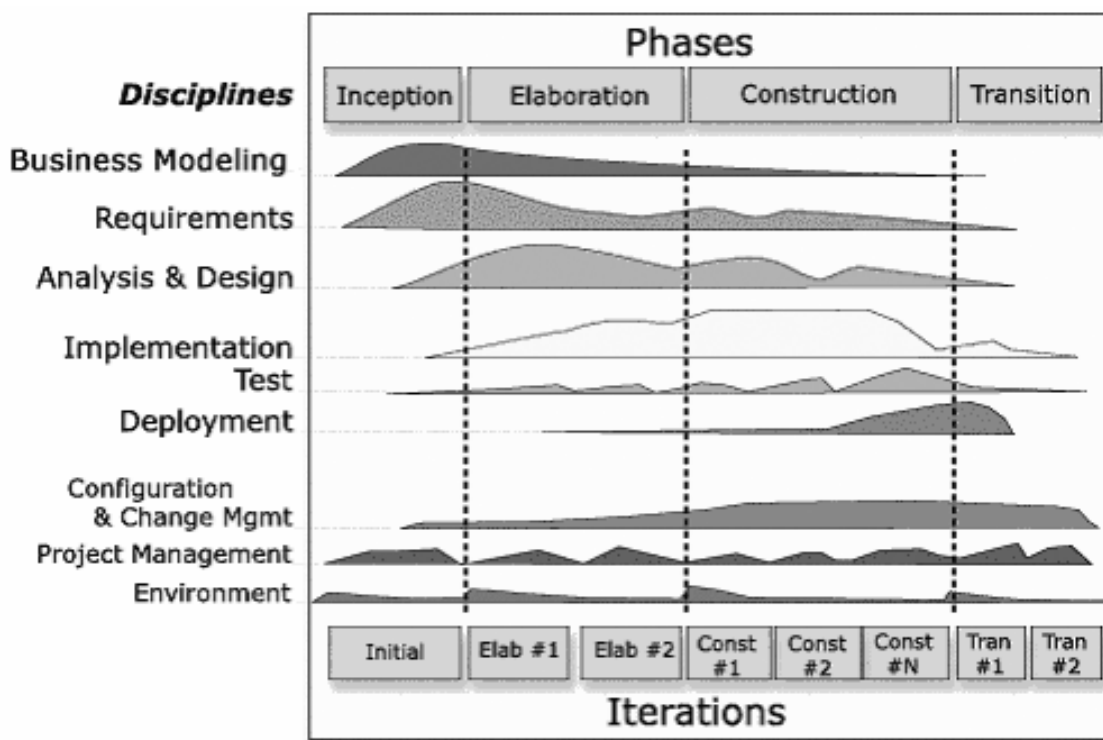
Specyfikacja EJB definiuje następujące typy komponentów EJB:

- *Session Beans* – komponenty te wykorzystywane są do reprezentacji procesu biznesowego w imieniu klienta. Komponenty te reprezentują działania na danych, ale nie same dane. Istnieją dwa typy komponentów Session Bean: Stateless – komponenty bezstanowe oraz Statefull – komponenty stanowe. Komponenty Stateless Session Beans nie utrzymują stanu konwersacji z klientami pomiędzy kolejnymi wywołaniami ich metod biznesowych. Statefull Session Beans utrzymują pełną informację na temat stanu pomiędzy kolejnymi wywołaniami metod biznesowych. Są one związane z klientem, którego żądania obsługują.

- *Entity Beans* – komponenty te reprezentują dane przetwarzane i przetrzymywane przez system. Istnieją dwa typy komponentów Entity Beans: komponenty z trwałością zarządzaną przez kontener (CMP) oraz z trwałością zarządzaną przez sam komponent (BMP).

4. Wprowadzenie do procesu wytwórczego RUP (Rational Unified Process)

Chcąc budować systemy informatyczne w oparciu o architekturę J2EE w sposób powtarzalny, przewidywalny oraz zgodnie z wymaganiami klienta potrzebny jest proces wytwórczy. Dobry proces wytwórczy definiuje **Kto** powinien wykonywać **Jakie** czynności w **Jaki** sposób oraz **Kiedy**, aby osiągnąć z góry określony cel. Ten z góry określony cel to wyprodukowanie nowego bądź zmodyfikowanego systemu za każdym razem gdy pojawiają się nowe bądź zmodyfikowane wymagania. Proces wytwórczy RUP firmy Rational spełnia te wymagania. RUP jest procesem interakcyjnym zakładającym zbudowanie systemu w kilku iteracjach. Po zakończeniu każdej iteracji produkowany jest system spełniający część wymagań klienta, jest on mu następnie udostępniany. W ten sposób zespół analityczno-projektowy otrzymuje szybko sygnał zwrotny od klienta, który umożliwia bieżącą oceną ryzyka niepowodzenia projektu jak również pozwala stwierdzić czy zespół analityczno-projektowy dobrze zrozumiał wymagania klienta wobec systemu. W razie wystąpienia problemów zostaną one szybko wykryte i zespół analityczno-projektowy będzie mógł wdrożyć odpowiednie postępowanie zaradcze. Cecha ta powoduje gwałtowną redukcję ryzyka niepowodzenia projektu już po zakończeniu pierwszej iteracji. Struktura procesu wytwórczego pokazana jest na poniższym rysunku.



Rys. 2. Struktura procesu wytwórczego RUP

Można powiedzieć, że proces RUP posiada dwa wymiary:

- Na osi X znajduje się czas i pokazane są aspekty związane z cyklem życia projektu.

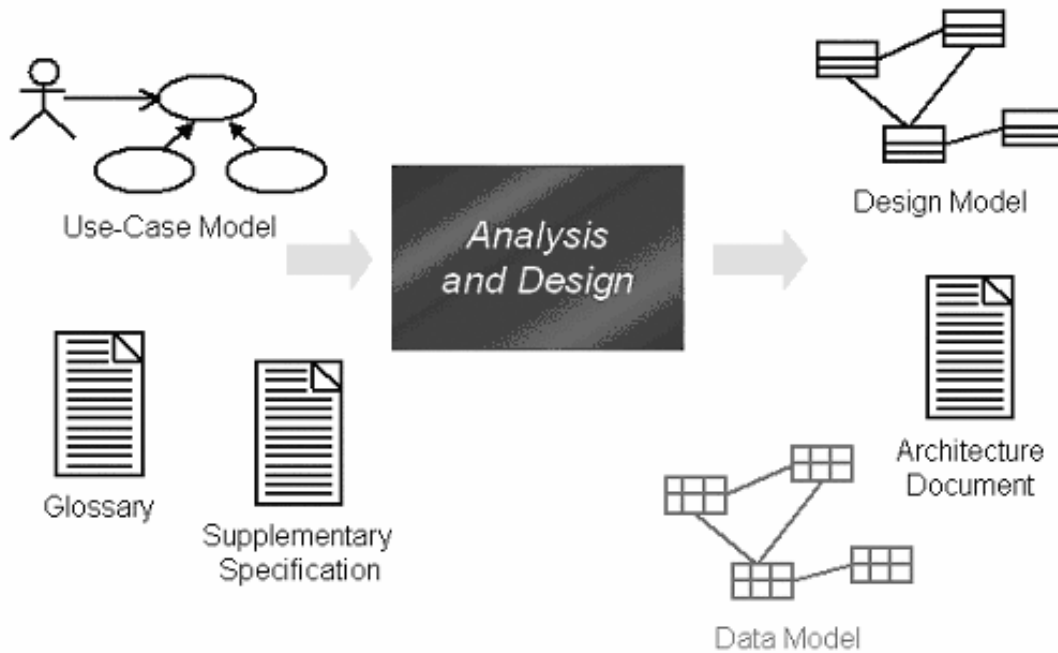
- Na osi Y znajdują się natomiast podstawowe dyscypliny jakie powinny być zrealizowane podczas prac nad projektem.

W dalszej części artykułu dokładniej zostanie omówiona dyscyplina o nazwie Analysis and Design w odniesieniu do systemów tworzonych w architekturze J2EE.

Głównym celem tej czynności jest:

- Dokonanie transformacji wymagań względem systemu informatycznego na jego projekt.
- Wypracowanie niezawodnej architektury systemu.
- Zaadoptowanie w projekcie wszelkich ograniczeń projektowych związanych np. ze środowiskiem, w którym będzie działał system.

Gdyby dyscyplinę Analysis and Design potraktować jako czarną skrzynkę, wówczas na jej wejściu mielibyśmy wymagania funkcjonalne i нефункционалне względem systemu informatycznego, wyrażone w postaci modelu przypadków użycia oraz dokumentu specyfikacji dodatkowej, jak również słownik dziedziny problemu. Na wyjściu natomiast otrzymalibyśmy model projektowy naszego systemu, dokument specyfikujący wytworzoną przez nas architekturę systemu i być może model danych.



Rys. 3. Dyscyplina „Analysis and Design”

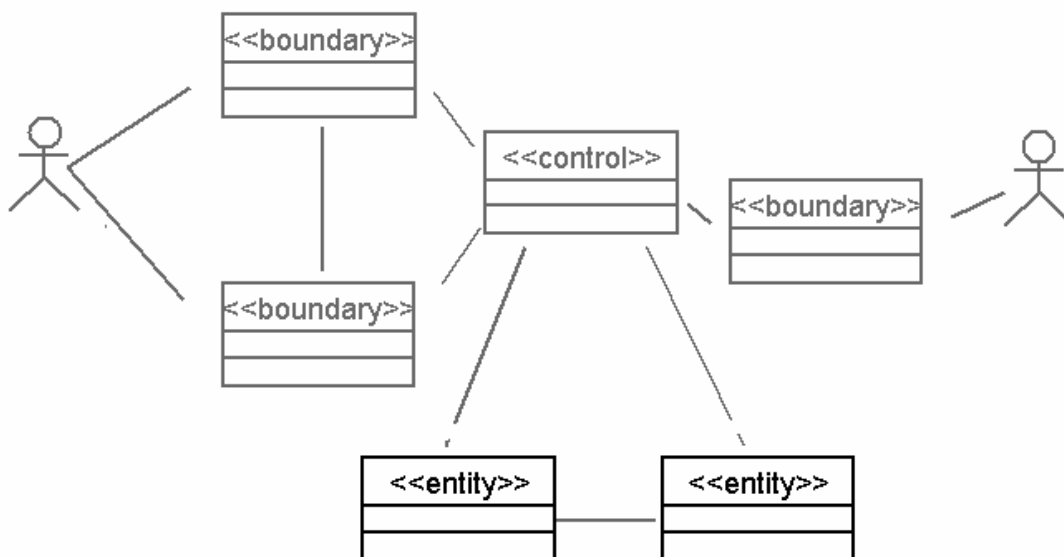
Proces wytwórczy RUP opisuje jakie czynności powinniśmy wykonać aby przejść od wyspecyfikowanych wymagań względem systemu informatycznego do jego projektu. RUP zakłada, że podczas naszych prac posługiwać się będziemy dwoma poziomami abstrakcji:

- Analitycznym – na tym poziomie stosujemy zasadę uproszczonego modelowania, abstrahujemy od środowiska programistycznego, w którym zostanie stworzony nasz system, oraz obsługujemy tylko i wyłącznie wymagania funkcjonalne wobec systemu.
- Projektowym – na tym poziomie stosujemy już dokładne projektowanie, z uwzględnieniem środowiska programistycznego, w którym zostanie utworzony system. Na tym poziomie musimy również obsłużyć wymagania нефункционалне wobec systemu informatycznego.

Czynnością przewidzianą na etapie analitycznym przez RUP jest czynność o nazwie Analiza Przypadków Użycia. Podczas trwania tej czynności naszym głównym zadaniem jest zidentyfikowanie klas analitycznych, niezbędnych przy realizowaniu określonego przypadku użycia, przyporządkowanie im niezbędnych odpowiedzialności oraz pokazanie za pomocą diagramów interakcji oraz klas aspektów dynamicznego oraz statycznego sposobu realizacji danego przypadku użycia. Prace wykonywane w trakcie trwania tej czynności koncentrują się na pokazaniu realizacji przez system pojedynczego przypadku użycia, tak więc najpierw patrzymy na system w skali mikro przez pryzmat konkretnej funkcjonalności udostępnianej przez system, a dopiero na końcu prac uogólniamy ich wyniki próbując zidentyfikować pewną część wspólną pojawiającą się w realizacji kilku bądź wszystkich przypadków użycia. Prace wykonywane w tej czynności opierają się na dokumencie specyfikacji przypadku użycia, który zawiera opis interakcji pomiędzy użytkownikiem a systemem w trakcie realizacji danego przypadku użycia. Czytając taki dokument staramy się zidentyfikować z jego treści klasy analityczne. Klasa analityczna to podstawowy budulec jakim możemy się posługiwać na poziomie analitycznym. Posługiwać się możemy trzema typami klas analitycznych:

- Klasa o stereotypie *boundary* – nazywana w polskiej literaturze czasami klasą interfejsu, odpowiada za organizację styku pomiędzy naszym systemem a jego światem zewnętrznym. Klasa ta może modelować np. element interfejsu użytkownika zdefiniowany w naszym systemie. Aktor reprezentujący świat zewnętrzny nie ma prawa kontaktować się z naszym systemem w inny sposób niż tylko za pomocą obiektu będącego instancją klasy *boundary*.
- Klasa o stereotypie *entity* – nazywana w polskiej literaturze czasami klasą aplikacji, odpowiada za modelowanie danych przechowywanych i przetwarzanych przez system.
- Klasa o stereotypie *control* – nazywana w polskiej literaturze czasami klasą sterującą, odpowiada za takie wysterowanie pozostałymi elementami składowymi systemu, aby zrealizować poprawnie dany przypadek użycia.

Na poniższym rysunku pokazano w sposób poglądowy role poszczególnych typów klas analitycznych.



Rys. 4. Klasy analityczne

Gdy udało nam się już zidentyfikować klasy analityczne określonych typów kolejny etap naszej pracy to stworzenie diagramów sekwencji bądź współpracy pokazujących w jaki sposób obiekty będące instancjami zidentyfikowanych przez nas klas analitycznych komunikują się ze sobą w celu zrealizowania danego przypadku użycia. Na tym etapie identyfikujemy i opisujemy również odpo-

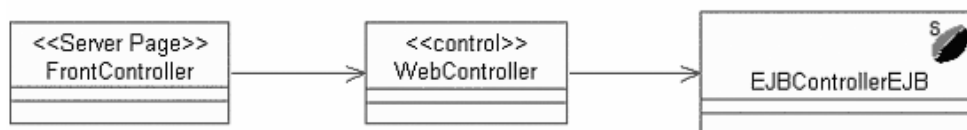
wiedzialności poszczególnych klas niezbędne w celu zrealizowania danego przypadku użycia. Kolejny krok to stworzenie diagramu klas, który ukazuje statyczne związki występujące pomiędzy klasami, których obiekty komunikują się ze sobą. Na tym etapie identyfikuje się takie związki pomiędzy klasami jak asocjacja, agregacja, kompozycja czy też generalizacja. Istnieją pewne reguły, które pozwalają zidentyfikować takie związki w zależności od postaci diagramu sekwencji czy też współpracy.

Po zakończeniu analizy przypadków użycia wiemy na poziomie analitycznym w jaki sposób system będzie realizował poszczególne przypadki użycia. Jednak wiedza ta nie jest jeszcze wystarczająca aby zbudować system w określonej architekturze. Teraz musimy wzbogacić ten opis o szczegóły związane z wykorzystaną technologią. W tym celu w dyscyplinie Analysis and Design procesu wytwórczego RUP zdefiniowana jest czynność o nazwie Identyfikacja elementów projektowych. Głównym celem tej czynności jest dokonanie transformacji ogólnego pojęcia klasy analitycznej na elementy projektowe, z których będzie zbudowany nasz system przy założeniu stosowania odpowiedniej wybranej przez nas technologii (w naszym przypadku będzie to J2EE). W celu zrealizowania tego celu powinniśmy kolejno przyrzeć się wszystkim zidentyfikowanym przez nas klasom analitycznym i w zależności od ich typów oraz odpowiedzialności dokonać transformacji tych elementów na odpowiednie elementy projektowe. W naszym przypadku klasa analityczna o stereotypie boundary może przekształcić się w stronę kliencką wykonaną w technologii HTML, może stać się stroną wykonaną w technologii JSP, może stać się serwletem bądź też konglomeratem tych trzech technologii.

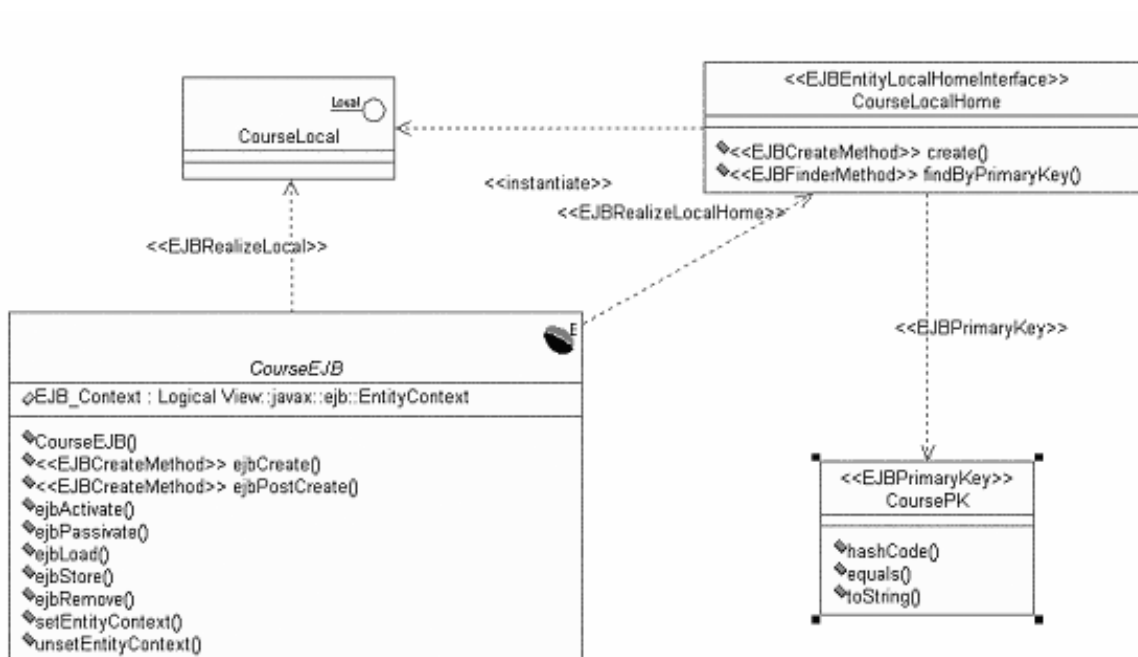


Rys. 5. Przejście od klasy analitycznej do klasy projektowej na przykładzie klasy o stereotypie Boundary

W przypadku klas o stereotypie control przy założeniu budowy klienta w oparciu o WEB można zastosować dwa wzorce projektowe: Front Controller oraz Session Facade. W wyniku zastosowania takiej operacji dostaniemy trzy elementy projektowe, serwlet o nazwie FrontController, klasę WebController oraz Session Bean o nazwie EJBController.



Rys. 6. Przejście od klasy analitycznej do klasy projektowej na przykładzie klasy o stereotypie Control



Rys. 7. Przejście od klasy analitycznej do klasy projektowej na przykładzie klasy o stereotypie Entity

Po zakończeniu czynności Identyfikacja elementów projektowych pozostaje nam zmodyfikować realizację poszczególnych przypadków użycia, w ten sposób aby wyrazić ją za pomocą nowo wprowadzonych elementów projektowych. Następnie musimy popracować w celu dokładnego określenia atrybutów, operacji i związków w poszczególnych klasach naszego systemu. Po wykonaniu tej operacji możemy dokonać automatycznej generacji kodu bezpośrednio z naszego środowiska CASE do środowiska programistycznego i tam zapełnić treścią poszczególne operacje.

Stopień skomplikowania systemów budowanych w oparciu o technologię J2EE wymaga, aby podczas ich budowy posługiwać się określonym procesem wytwórczym pozwalającym produkować systemy w sposób powtarzalny, przewidywalny oraz zgodnie z wymaganiami użytkownika. Co więcej, budując takie systemy powinniśmy posługiwać się notacją UML pozwalającą na zobrazowanie aspektu statycznego oraz dynamicznego budowanego systemu. Ważne jest, że notacja UML pozwala na zobrazowanie nie tylko modelu analitycznego naszego systemu, ale również modelu projektowego, podczas tworzenia którego posługujemy się pojęciami występującymi w danej technologii. Mechanizm rozszerzeń języka UML pozwala na wprowadzenie do niego takich elementów jak serwet komponent EJB, czy też strona HTML.