IBM Rational Software Architect

uproszczona instrukcja użytkowania

TOMASZ ŁUKASZUK

na podstawie Software Developer's Jurnal Extra Nr 18

STRESZCZENIE: Dokument przedstawia ogólne informacje na temat narzędzia IBM Rational Software Architect. Omawia pokrótce kolejne etapy budowy aplikacji w tym środowisku: od modelu przypadków użycia, poprzez model analityczny, model projektowy do kodu aplikacji. Zamieszczono tu także przykład pokazujący projektowanie i implementację aplikacji „Książka adresowa" w IBM RSA.

1. WPROWADZENIE

IBM Rational Software Architect (IBM RSA) jest zintegrowanym narzędziem projektowym wspomagającym projektowanie i tworzenie oprogramowania z wykorzystaniem techniki programowania modelowego w języku UML. Umożliwia opracowywanie aplikacji i usług za pomocą języka UML. Pozwala na ujednolicenie wszystkich aspektów projektowania i tworzenia oprogramowania. Dzięki temu znacznie zwiększa się produktywność i wzrasta efektywność pracy zespołu programistów. Ważnym faktem jest precyzyjne określenie i utrzymanie wszystkich aspektów architektury projektowanej aplikacji. IBM Rational Software Architect zapewnia możliwość sprawdzenia zgodności architektury z projektem i jej odporność na zmiany. W znacznym stopniu podwyższa to jakość tworzonego oprogramowania.

IBM RSA jest środowiskiem dla rozwoju aplikacji, które umożliwia projektowanie aplikacji w języku **UML 2.0** oraz ich implementację. RSA to zbiór narzędzi wspierających procesy wytwórcze. W skład RSA wchodzą moduły *Rational Software Modeler* (narzędzie do wizualnego modelowania i projektowania aplikacji), *Rational Web Developer* (wizualne narzędzie przeznaczona dla twórców aplikacji web tworzących strony WWW i usługi sieciowe) i *Rational Aplication Developer for WebSphere Software* (zintegrowane środowisko programowania, które pozwala projektować, tworzyć, analizować, testować, profilować i wdrażać aplikacje internetowe i portalowe, aplikacje w technologii Java oraz J2EE oraz aplikacje wykorzystujące Web Services).

Oto niektóre możliwości programu:

- wsparcie dla modelowania UML 2.0 w procesie projektowania i analizy (wykorzystanie diagramów: przypadków użycia, przebiegu, klas, komponentów, aktywności, maszyny stanowej, komunikacji i wdrożenia),
- uproszczona praca z diagramami, modelowanie wizualne z wykorzystaniem rozbudowanej pomocy, możliwość stosowania i promowania przez firmy "recept" - sprawdzonych przepisów i schematów zwiększających przewidywalność i powtarzalność procesu projektowania aplikacji,
- edycja diagramu klas UML dla C++, Java, Enterprise Java Beans i obiektów bazodanowych - modelowanie ER (edytor diagramów IE i IDEF),
- wizualizacja ciała metod klas Java poprzez zastosowanie diagramów przebiegu UML 2.0,

- wykorzystanie przekształceń do automatycznego generowania kodu Java, C++, EJB,
- wstawianie znaczników, dzięki którym możliwe jest wykrywanie schematów w kodzie Java i ich wizualizacja, oparte na szablonach zasady do monitorowania i wymuszania struktury aplikacji,
- IDE wspierane przez Eclipse,
- zgodne z WS-I usługi sieciowe, narzędzia, kreatory i wykorzystanie technologii drag-and-drop i point-and-click do szybkiego tworzenia aplikacji (Rapid Application Development),
- zautomatyzowane narzędzie i użyte standardy poprawiające jakość kodu, wbudowane narzędzie raportowania Crystal-Reports,
- środowisko projektowania w języku C++ z edytorem wykorzystującym podświetlanie składni, dzięki któremu w jednym środowisku możliwe jest tworzenie aplikacji w różnych językach, kierowanie i pomoc w procesie tworzenia jest dostarczane dynamicznie - w zależności od potrzeb użytkownika,
- otwarte API wspierające rozszerzalność środowiska,
- możliwość tworzenia raportów i eksportowania ich do formatów HTML, PDF i XML, generowanie i integrowanie diagramów do Javadoc, tworzenie skryptów.

RSA jest narzędziem wspierającym podejście **Model Driven-Development**, które ukierunkowane jest na modele. Z tej racji podczas analizy i projektowania systemów za pomocą RSA duży nacisk kładziony jest na specyfikację architektury systemu. Metodyką, którą bezpośrednio wskazuje RSA jest **Rational Unifield Process**. Proces inicjujemy przez utworzenie modelu przypadków użycia, który stanowi wizualny obraz wymagań funkcjonalnych systemu. Następnie tworzymy kolejno model analityczny, model projektowy i kod systemu w konkretnym języku programowania.

RSA poza możliwością budowania projektów w języku UML lub kodowania aplikacji pozawala na transformacje artefaktów. Te transformacje to przede wszystkim możliwości generowania szkieletów kodu systemu za pomocą metod **inżynierii wprzód** oraz tworzenie na podstawie kodu aplikacji równoważnej mu specyfikacji wyrażonej w języku UML za pomocą metod **inżynierii wstecz**.

2. PROJEKT I IMPLEMENTACJA APLIKACJI W ŚRODOWISKU IBM RATIONAL SOFTWARE ARCHITECT

A. Model przypadków użycia

Zgodnie z zasadami inżynierii oprogramowania pracę nad projektem informatycznym zaczynamy od identyfikacji i udokumentowania wymagań. W tym celu należy zbudować diagram przypadków użycia, który jest wizualnym repozytorium wymagań funkcjonalnych.

Budowę modelu przypadków użycia rozpoczynamy od zdefiniowania aktorów systemu. Następnie z każdym aktorem kojarzymy przypadki użycia. Należy podkreślić, że pełny model przypadków użycia składa się również z tekstowego opisu przypadków użycia i ich scenariuszy. RSA umożliwia współpracę z programem IBM Rational RequsitePro – narzędziem, które jest przeznaczone do zarządzania wymaganiami podczas całego cyklu życia aplikacji. Możliwe jest również dodanie opisu przypadku użycia we właściwościach elementu. Do wybranych przypadków użycia można dołączyć diagramy czynności schematycznie pokazujące przebieg reprezentowanej przez przypadek użycia funkcji systemu.

B. Model analityczny

Po zdefiniowaniu wymagań przychodzi kolej na budowę modeli, które opiszą elementy składowe systemu i ich wzajemne współdziałanie. Modele te wchodzą w skład modelu analitycznego.

Model analityczny powinien pokazywać realizacje wszystkich przypadków użycia w postaci diagramów klas obejmujących klasy potrzebne do zrealizowania modelowanego wymagania, diagramów interakcji pokazujących podstawowy i alternatywne przebiegi przypadku użycia. Następnie z klas występujących na diagramach pokazujących realizacje przypadków użycia tworzymy diagram klas całego systemu (modułu systemu) i uszczegóławiamy go określając niezbędne a wcześniej pominięte relacje.

C. Model projektowy

Po zbudowaniu modelu analitycznego można przejść do budowy modelu projektowego będącego podstawą do wygenerowania szkieletu kodu aplikacji.

Model projektowy jest kompletnym modelem klas powstałym z rozszerzenia diagramu z modelu analitycznego o wszystkie atrybuty i metody klas.

D. Implementacja

Po wstępnym uszczegółowieniu klas można przystąpić do generowania szkieletu aplikacji.

Utworzony projekt staje się repozytorium kodu aplikacji. Model implementacji jest jednoznaczny z jego implementacją wyrażoną w języku programowania. Od tej pory każda zmiana w modelu powoduje zmianę w kodzie aplikacji.

Kolejnym krokiem jest uzupełnienie pliku z kodem źródłowym odpowiednimi treściami, które pozwolą na funkcjonowanie aplikacji.

E. Uruchomienie

Po dodaniu kodu aplikacji należy przejść do jej debugowania i uruchomienia. Najpierw należy sprawdzić poprawność kodu przez validator. W przypadku poprawnej walidacji można uruchomić aplikację. W trakcie pierwszego uruchomienia należy określić typ aplikacji i jej środowisko uruchomienia.

F. Dokumentacja po implementacji

W celu uniknięcia trudności związanych między innymi ze zrozumieniem w przyszłości własnoręcznie napisanego programu, dobrze jest po implementacji zsynchronizować kod aplikacji z modelem w języku UML, co stanowić będzie dodatkowe elementy repozytorium projektu.

W RSA synchronizacja kodu aplikacji z modelem UML odbywa się poprzez budowę diagramu klas.

3. KOLEJNE ETAPY PRACY NAD PROJEKTEM W ŚRODOWISKU IBM RSA NA PRZYKŁADZIE APLIKACJI „KSIĄŻKA ADRESOWA”

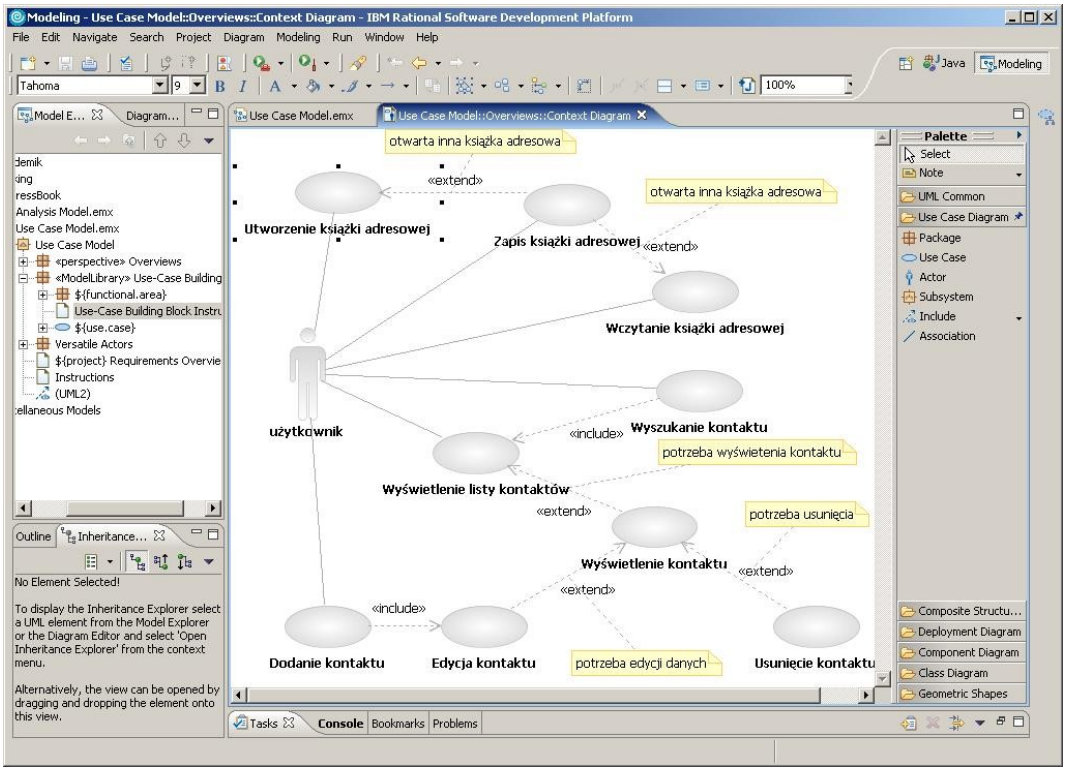
Pracę nad projektem zaczynamy od budowy modelu aplikacji zapisanego w języku UML. W tym celu należy wybrać w menu *File->New Project* i wybrać projekt UML. Następnie należy nadać nazwę nowemu projektowi (tutaj: AddressBook) i określić pierwszy model projektowy.

A. Model przypadków użycia

Pierwszym modelem zgodnie z zasadami inżynierii oprogramowania jest model przypadków użycia. W RSA model przypadków użycia tworzony jest na podstawie szablonu. Wybór szablonu projektowego powoduje udostępnienie szeregu pakietów, które grupują elementy składowe modelu przypadków użycia ze względu na zastosowanie.

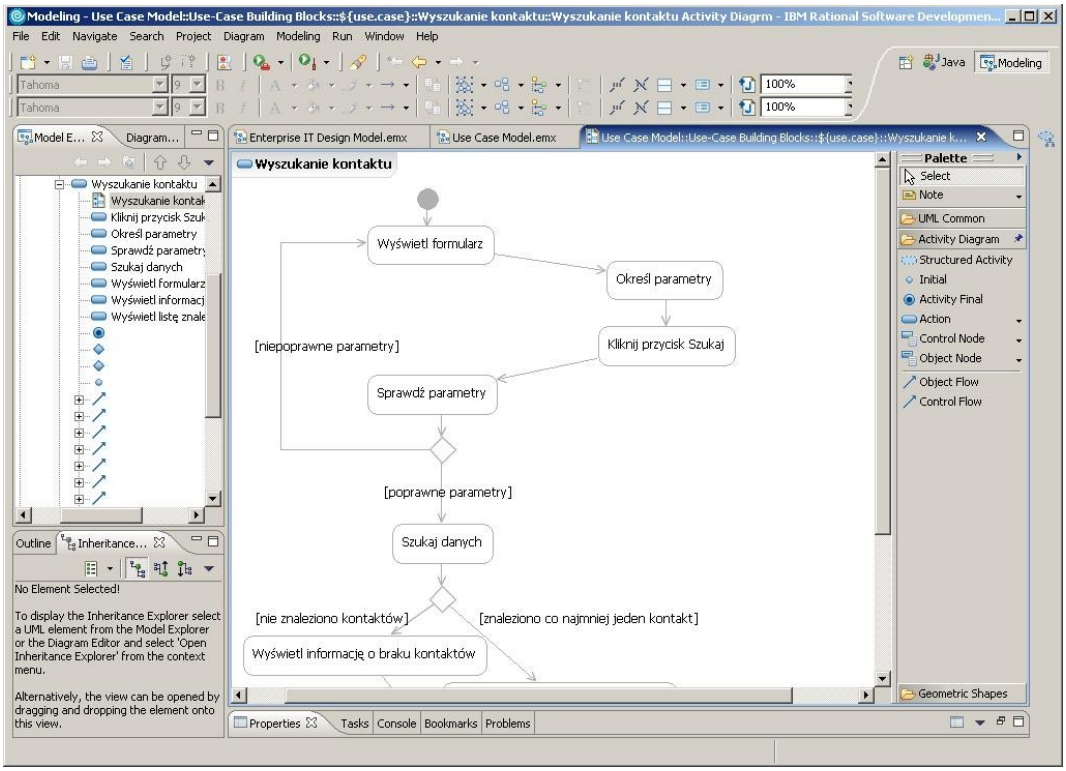
Budowę modelu przypadków użycia rozpoczynamy od zdefiniowania aktorów systemu. Następnie dla każdego aktora określamy funkcje (przypadki użycia), które system powinien mu udostępniać. Aktorów jak i inne elementy systemu można dodać wykorzystując menu kontekstowe lub paletę elementów.

Rysunek 1 przedstawia diagram przypadków użycia obrazujący wymagania, które ma realizować aplikacja „Książka adresowa”



Rysunek 1: Diagram przypadków użycia - Książka adresowa

Każdy z przypadków użycia powinien być opisany. Opis można dołączyć we właściwościach elementu. Niektóre przypadki użycia mogą być szczegółowo przedstawione na diagramie czynności pokazującym przebieg procesu, scenariusz podstawowy lub alternatywny.



Rysunek 2: Diagram czynności - przypadek użycia Wyszukiwanie kontaktu

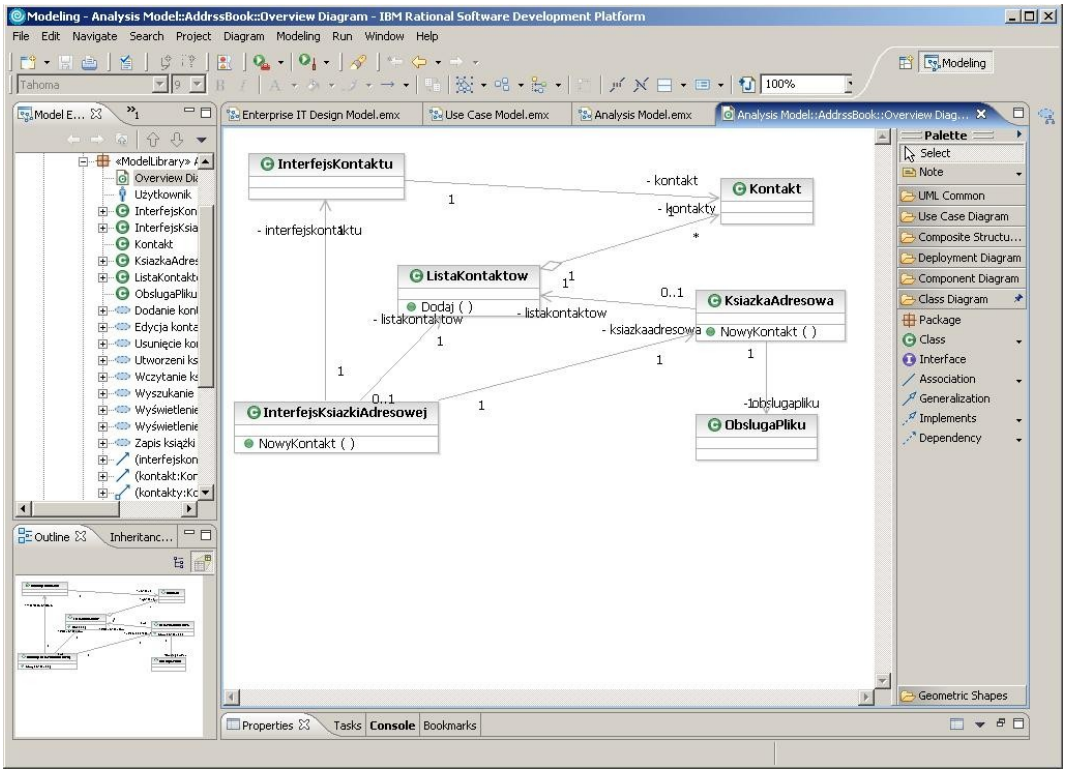
B. Model analityczny

Model analityczny dodajemy do projektu w podobny sposób jak model przypadków użycia (wybieramy szablon *Analysis Model*). Model analityczny jest opcjonalnym elementem projektu. W przypadku prostych systemów można od razu budować model projektowy.

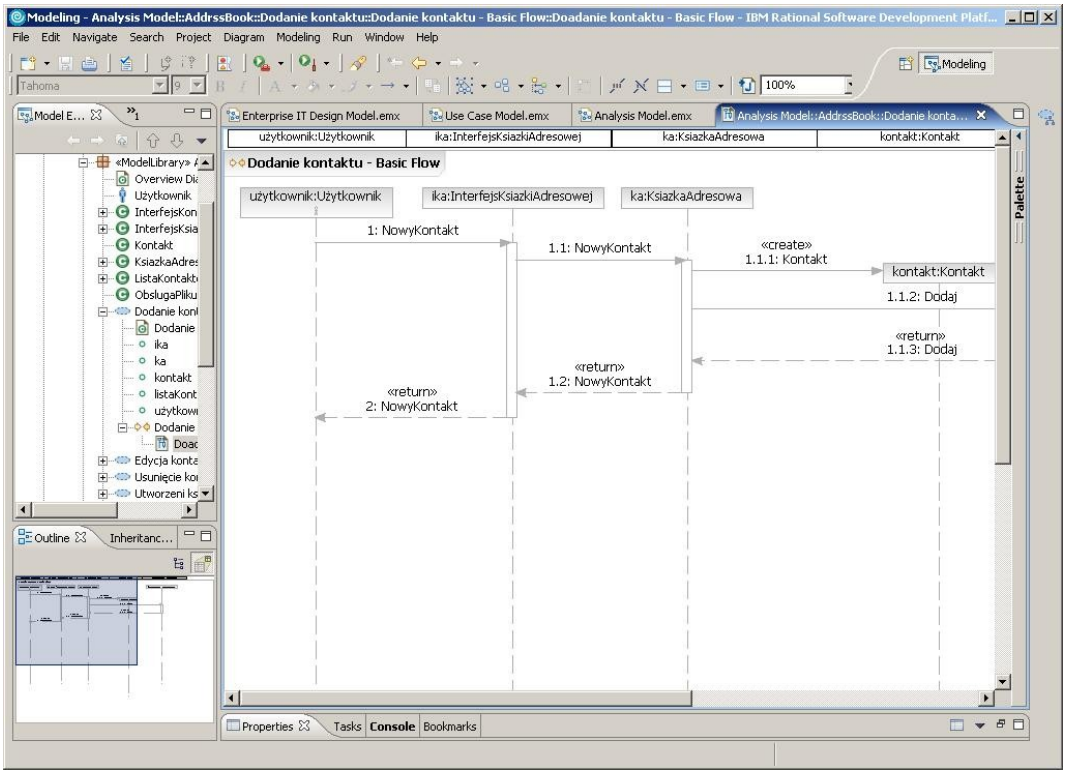
W zdefiniowanym modelu analitycznym należy utworzyć tak zwane realizacje przypadków użycia. Są to elementy współpracy (*Collaboration*), które powinny nosić nazwę przypadku użycia, którego dotyczą. Element współpracy może być dodany za pomocą menu kontekstowego.

Dla każdego przypadku użycia należy dołączyć diagram klas zawierający elementy pozwalające na zrealizowanie danej funkcjonalności. Realizacja bardziej skomplikowanych przypadków użycia może zostać przedstawiona na diagramie sekwencji. RSA zapewnia spójność diagramów klas i sekwencji, tzn. automatycznie umieszcza w odpowiednich klasach metody użyte na diagramie sekwencji.

Po opracowaniu wszystkich przypadków użycia i utworzeniu związanych z mini diagramów klas należy sporządzić jeszcze jeden klas diagram, pokazujący cały system lub moduł. Na tym diagramie należy umieścić wszystkie klasy wcześniej zidentyfikowane (metoda przeciągnij i upuść). Po dodaniu klasy RSA zatroszczy się o uwzględnienie także wcześniej określonych relacji. Następnie można uszczegółowić diagram dodając wcześniej pominięte atrybuty, metody, relacje.



Rysunek 3 Model analityczny - Książka adresowa



Rysunek 4 Diagram sekwencji - przypadek użycia Dodawanie kontaktu

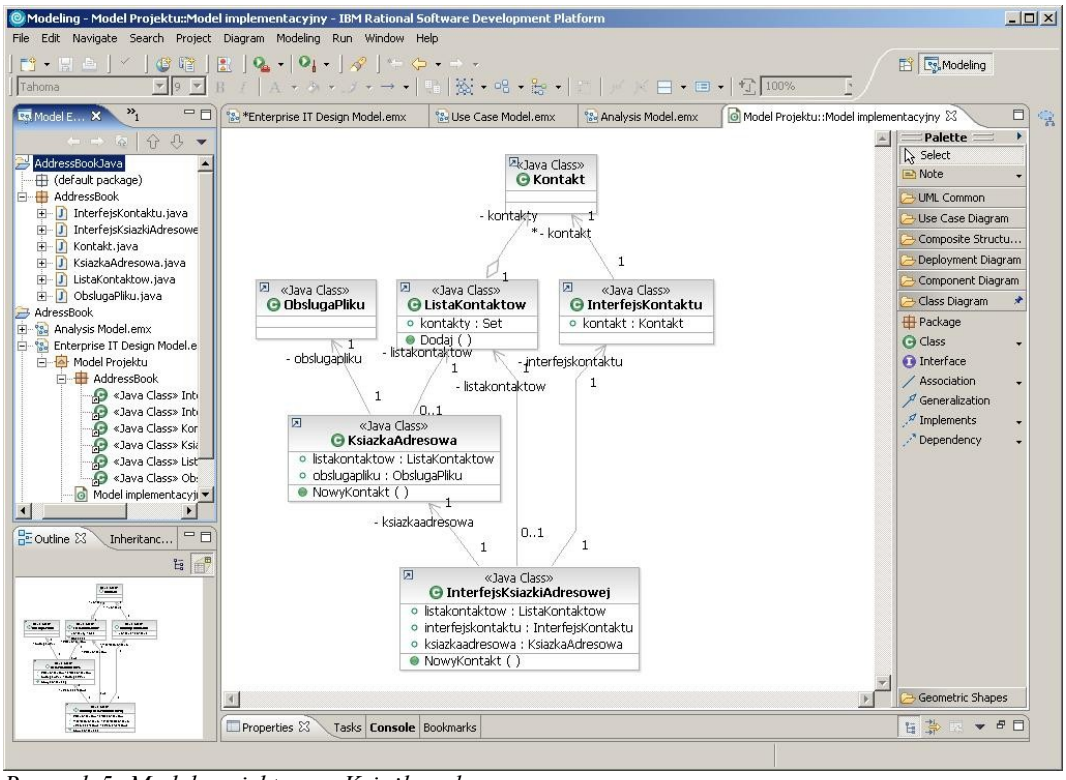
C. Model projektowy i implementacja

Po zbudowaniu modelu analitycznego można przejść do budowy modelu projektowego będącego podstawą do wygenerowania szkieletu kodu aplikacji. Pierwszym krokiem będzie dodanie do projektu pustego szablonu projektowego. Model ten dodaje się w sposób analogiczny jak poprzednie modele.

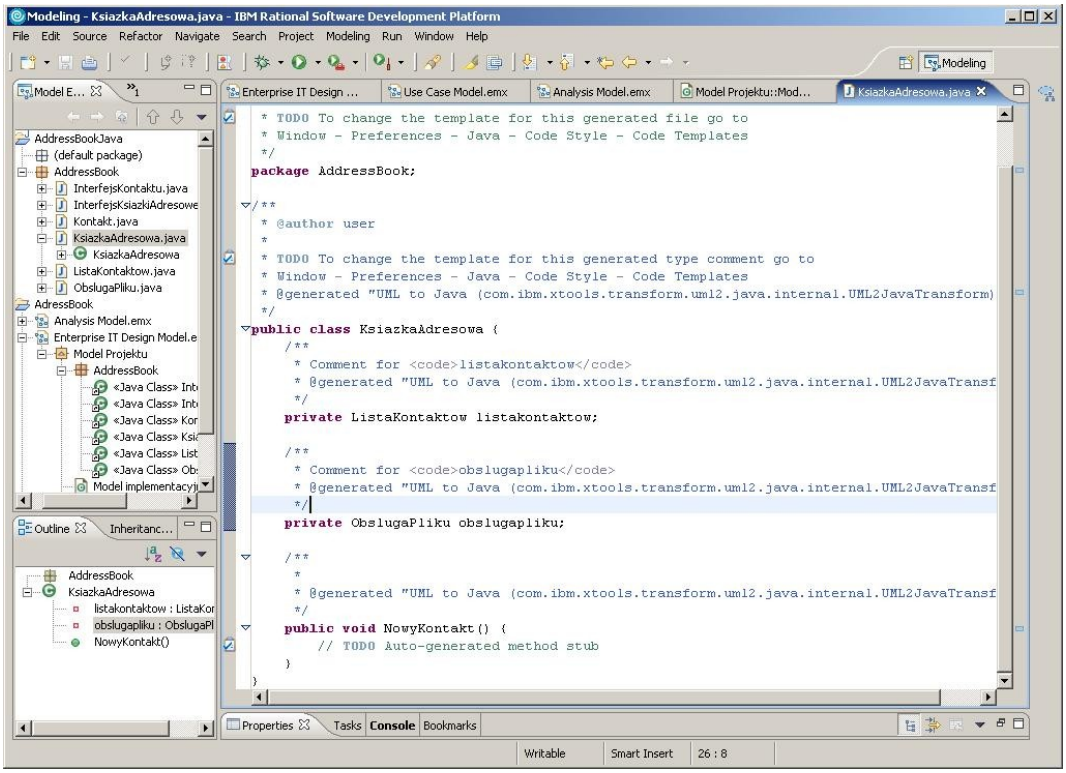
Następnie kopiujemy do modelu projektowego wszystkie klasy z modelu analitycznego. Po tym uzupełniamy klasy poprzez dodanie wszystkich atrybutów i metod. Dodawane elementy powinny uwzględniać specyfikację języka implementacji (w naszym przypadku Java).

Następnie przystępujemy do generowania szkieletu aplikacji. W tym celu należy kliknąć prawym klawiszem myszy na pakiecie gdzie znajdują się klasy projektowe i w menu kontekstowym wybrać *Tranform->Run Transformation->UML* to Java. Uruchomiony zostanie kreator, który krok po kroku pozwala na zdefiniowanie szkieletu javowego aplikacji. Pierwszym widokiem kreatora jest ekran umożliwiający wybór projektu aplikacji lub utworzeni nowego projektu (*Create New Target Container*), w którym umieszczony zostanie kod. Istotnym jest ustawienie aby podczas transformacji klasy projektowe w modelu projektowym zostały zamienione przez klasy kodowe (ustawienie na zakładce *Common*). Model implementacji jest jednoznaczny z jego implementacją wyrażoną w języku Java. Od tej pory jakakolwiek zmiana w modelu powoduje zmianę w kodzie.

Kolejnym krokiem jest uzupełnienie plików z kodem źródłowym klas odpowiednimi treściami, które pozwolą na funkcjonowanie aplikacji.



Rysunek 5: Model projektowy - Książka adresowa



Rysunek 6: Szkielet aplikacji wygenerowany w wyniku transformacji UML to Java

---

Politechnika Białostocka
Katedra Oprogramowania
ul. Wiejska 45A
15-351 Białystok

18 grudnia 2005