BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 32, 2012

Wytwarzanie serwisów informacyjnych z wykorzystaniem koncepcji modelowania dziedzin. Budowa transformacji

Agata KOSIOR[1], Andrzej STASIAK[1], Włodzimierz DĄBROWSKI[2]

1. Instytut Teleinformatyki i Automatyki WAT,
ul. Gen. S. Kaliskiego 2, 00-908 Warszawa
agata.kosior@gmail.com, astasiak@wat.edu.pl
2. Instytut Sterowania i Elektroniki Przemysłowej, PW
ul. Koszykowa 75, 00-662 Warszawa
w.dabrowski@ee.pw.edu.pl

STRESZCZENIE: W artykule przedstawiono klasyfikację transformacji modeli (M2M, M2T, T2M, JET) oraz modeli mapowań. Szczegółowo omówiono proces wytwarzania generycznych mechanizmów transformacji umożliwiających wielokrotne przekształcenia modeli. Dodatkowo zaprezentowano przykładowe transformacje opracowane w ramach projektu serwisu informacyjnego.

SŁOWA KLUCZOWE: transformacje (M2M, M2T, T2M, JET), języki opisu transformacji (ATL, QVT, Tefkat), model mapowań (Model Type Mapping, Model Instance Mapping), mapa powiązań.

1. Wstęp

Niniejsze opracowanie omawia procesy generycznego wytwarzania transformacji służących do przekształcania modeli powstających w trakcie procesu budowy oprogramowania. Wcześniejsze kroki służące do wytwarzania serwisów informacyjnych wykorzystujące koncepcję modelowania dziedzin można znaleźć w [17]. Opis metody KMS jest ograniczymy do procesu budowy transformacji i przedstawimy go w kolejnych punktach.

Wytworzenie generycznych mechanizmów transformacji wymaga opracowania wzorców (w metodzie KMS traktowanych jako modele mapowań) oraz narzędzi umożliwiających wielokrotne wykorzystywanie automatycznych przekształceń. Działania te zostaną przedstawione w kolejnych podrozdziałach.

Za punkt wyjścia przyjmiemy definicję OMG [21], według której budowa oprogramowania została oparta na wspólnym metamodelu przedstawionym w tab. 1. Metamodel ten opisuje sposób opisu modelu budowanego systemu. Metamodel jest wzorem, weryfikatorem oraz walidatorem budowanych modeli, które przedstawiają konkretny problem projektowy z danej dziedziny. Metoda KMS podkreśla potrzebę wykorzystywania wzorców nie tylko na poziomie modelowania, ale również w procesie przekształceń modeli. Dzięki temu, że kontekst prac dotyczy dobrze znanej i określonej dziedziny, mechanizmy transformacji można oprzeć na sprawdzonych rozwiązaniach, jednocześnie umożliwiając swobodne wykorzystywanie zbioru dostępnych przekształceń. W praktyce budowane są metatransformacje (modele mapowań/ wzorce przekształceń/ szablony przekształceń), a w metodzie KMS projektowane są one podczas budowy modeli mapowań (rys. 4). Należy pamiętać, że poprawny wzorzec, to taki który umożliwia rozwiązywanie wszystkich problemów projektowych określonych w specyfikacji wymagań na system. Zapewnia on wymaganą szczegółowość i kompletność opisu dziedziny problemu, nie zapominając jednak o przyjazności i łatwości jego obsługi.

Tab. 1 Wzorce i narzędzia wspierające transformacje modeli

Model	Wzorzec	Narzędzie
Model wymagań	repozytorium wymagań + model usług 	notacja UML oraz standardowe mechanizmy dostępne w narzędziach CASE (np. RSA) 
Model dziedziny	metamodel + model ograniczeń 	język dziedzinowy KsiML repezentowany trzema profilami (dla których opracowano dedykowane palety i wzorce modeli) 

Model systemu	model mapowań M2M (dla elementu wiadomość) 	mechanizm transformacji M2M *(KsiML_Transformacja_M2M Transformation)* 
Kod	model mapowań M2T (dla elementu wiadomość) 	standardowy mechanizm M2T dostępny w narzędziach CASE (w RSA wykorzystano transformację *UML to C#*) 

Zaletą takiego wytwarzania oprogramowania jest możliwość automatycznego przekształcania zbioru prostych opisów w językach dziedzinowych w elementy bardziej skomplikowane, co pozwala na minimalizację kosztów i czasu potrzebnego na dostosowanie rozwiązań do nowych wymagań (możliwość przekształcania modelu dziedziny w kilka modeli systemu [21]). Dodatkową przewagą tego podejścia jest możliwość wielokrotnego wykorzystywania mechanizmów przekształceń i seryjnego generowania rozwiązań z danej dziedziny problemowej.

Wytwarzanie serwisów informacyjnych z wykorzystaniem koncepcji modelowania dziedzin odwołuje się do generacji wykonywalnego kodu źródłowego, który budowany jest automatycznie w wyniku transformacji opracowanych modeli zapisanych w językach dziedzinowych. W metodzie KMS przyjęto, że wszystkie opracowywane zgodnie z koncepcją MDA (ang. *Model-Driven Architecture*) modele zostaną połączone transformacjami: manualnymi (*Manual*) bądź automatycznymi (*Model-to-Model* [M2M] lub *Model-to-Text* [M2T]) (rys. 1).

Zastosowanie metody KMS zostało zilustrowane przykładami artefaktów projektowych. Przykłady te zostały wykonane w środowisku IBM RSA (*IBM Rational Software Architect ver. 8.0*) i pochodzą z projektu wykonanego w pracy [16].

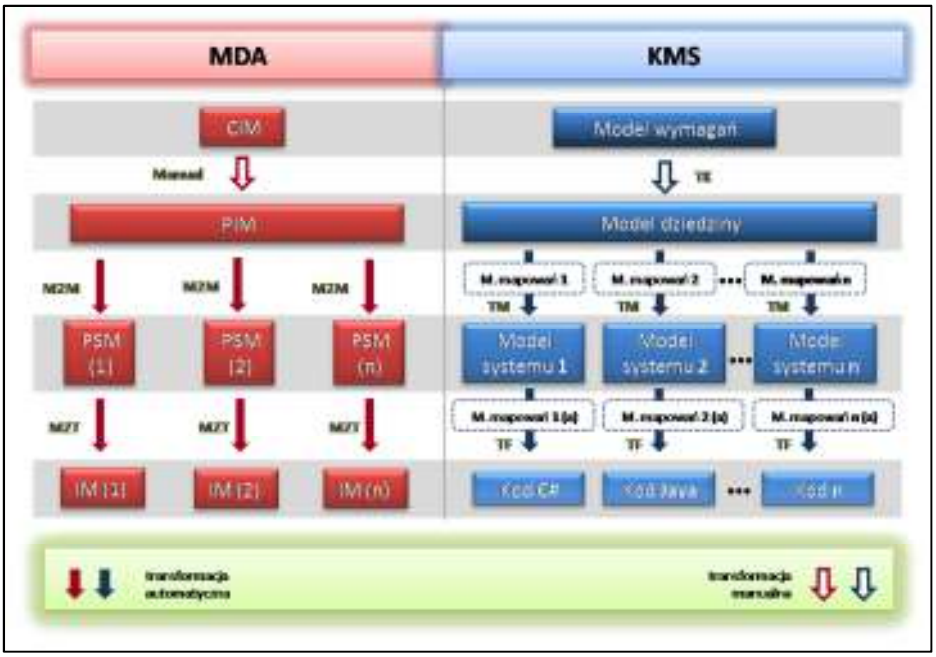
2. Transformacje – wprowadzenie

Transformacja oznacza przekształcenie danych wejściowych (wejście) w dane wyjściowe (wyjście) i zgodnie z przyjętymi regułami tego przekształcenia może być zapisana w postaci zależności (1). Należy zwrócić uwagę, że dane wejściowe i dane wyjściowe mogą oznaczać artefakty na różnym poziomie abstrakcji.

$$wy = f(we), \tag{1}$$

gdzie: *we* – dane wejściowe, *wy* – dane wyjściowe, *f* – transformacja.

Transformacja ma istotne znaczenie w koncepcji MDA oraz metodzie KMS. W podejściach tych modelowanie zaczyna się na wysokim poziomie abstrakcji i poprzez kolejne przekształcenia modele są uszczegóławiane, a zmianom ulega sposób ich prezentacji oraz struktura determinowana wybraną technologią (rys. 1).

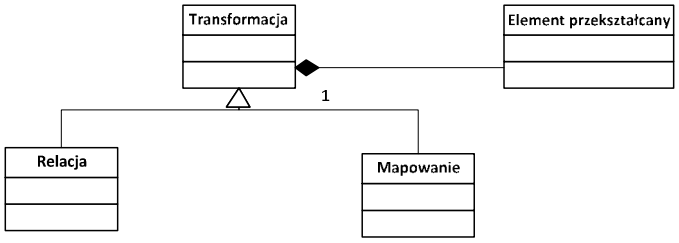


Rys. 1. Koncepcja MDA vs KMS

W metodzie KMS zakłada się, że modele zostaną wygenerowane dzięki transformacjom, które zostały podzielone na:

1) **transformacje manualne** - które dotyczą przekształceń konceptualnych wykonywanych przez analityka, czy projektanta bez użycia specjalistycznych narzędzi;
2) **transformacje automatyczne** - wspierane przez narzędzia minimalizujące nakłady pracy ludzkiej. W proponowanej metodzie wyróżnia się cztery typy tych transformacji:
   a) M2M - transformacja typu model-model (*Model- to- Model)*;
   b) M2T - transformacja typu model-tekst *(Model- to-Text)*;
   c) T2M - transformacja typu tekst-model *(Text- to- Model)*;
   d) JET - transformacja typu *Java Emitter Template*.

Definicję transformacji według podejścia MOF [22] pokazuje (rys. 2), wynika z niej, że wyróżniamy dwa odrębne podtypy transformacji: *relacja* i *mapowanie*.



Rys. 2. Transformacja, relacja i mapowanie w hierarchii MOF

- **Relacje** w modelu opisywane są przez połączenie (ang. *link*) i nie są zakończone grotem. W tym sensie tworzą specyfikację wielokierunkową połączenia między elementami transformacji. Relacje nie są wykonywalne w tym sensie, że nie są w stanie utworzyć lub zmienić modelu. Pozwalają na sprawdzenie spójności dwóch lub więcej modeli. Zazwyczaj używane są w fazie specyfikacji systemu lub podczas sprawdzania poprawności mapowania.
- **Mapowania** są implementacją transformacji i w odróżnieniu od relacji są jednokierunkowe. Mapowania mogą udoskonalać dowolną liczbę relacji, ale muszą być zgodne z ich definicją [7], [22].

Zarówno w metodzie KMS jak i MDA podkreśla się znaczenie separacji zasadniczej logiki warstwy biznesowej od specyfiki warstwy implementującej, a szczególny nacisk kładzie się na dynamiczne przejście pomiędzy platformami

PIM - (ang. *Platform Independent Model*) (w KMS określonej przez model dziedziny) i PSM (ang. *Platform Specific Model*) (w KMS mapowanej na model systemu) (Rys. 1).

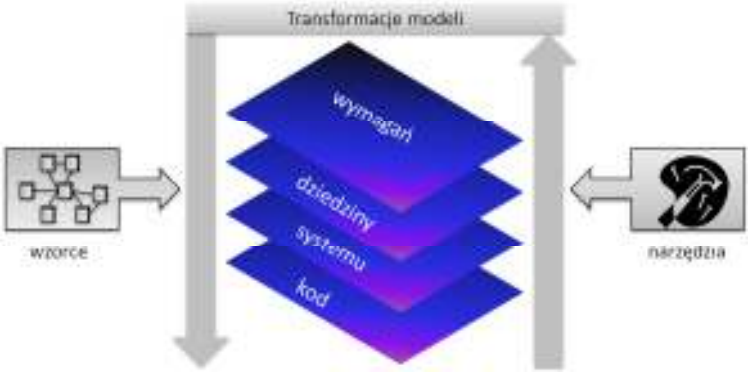
Wykonanie transformacji w tym przypadku polega na uzupełnieniu modelu PIM o dodatkowe informacje determinowane rodzajem wskazanej platformy, co w konsekwencji pozwala na otrzymanie modelu docelowego-PSM, który może być zaimplementowany przy użyciu dowolnej z istniejących technologii (C#, Java, C++, itd.).

W ramach metody KMS wymaga się opracowania kilku transformacji:

- transformacji manualnej umożliwiającej przejście z modelu wymagań do modelu dziedziny – transformacja TK;
- transformacji umożliwiającej przejście z modelu dziedziny do modelu systemu – transformacja TM;
- transformacji umożliwiającej przejście z modelu systemu do kodu rozwiązania – transformacja TF.

Dodatkowo można budować transformacje uzupełniające (TJ), opierając się na zasadzie przekształceń JET. Proces przekształcania modeli w metodzie KMS wspierany jest dedykowanymi wzorcami oraz narzędziami. Perspektywę transformacji modeli KMS przedstawiono na rys. 3.

W pracy [16] można znaleźć zestawienie, które jest przykładem bezpośredniego przełożenia tej koncepcji na wytwarzane artefakty projektowe.



Rys. 3. Perspektywa transformacji modeli w metodzie KMS

W pracy szczególną uwagę zwrócono na transformacje automatyczne, ponieważ przyspieszają one wytwarzanie nowych rozwiązań informatycznych, minimalizując nakład pracy manualnej. Dzięki unikalnym narzędziom sam proces docelowego przekształcenia (rys. 1) wymaga jedynie wskazania danych wejściowych (modelu wejściowego) i wyjściowych (modelu wyjściowego).

Przeniesienie ciężaru przekształcenia na wzorzec wymaga wcześniejszego opracowania reguł transformacji, które w metodzie KMS umieszcza się w modelu mapowań (rys. 4), będącym wzorcem przekształcenia. Wynikiem precyzyjnego opisania zasad przekształceń (umieszczonych w modelu mapowań) jest tzw. język opisu transformacji.

Język opisu transformacji powinien zapewnić precyzyjne zdefiniowanie przekształcenia (1), być kompletny, spójny i łatwy w użyciu. W tym celu powinien on dostarczać kompletny słownik dziedzinowy (przeznaczony do budowy rozwiązań w danej dziedzinie). Kompletny słownik języka, to taki który definiuje wszystkie terminy danej dziedziny problemowej, ich powiązania oraz strukturę.

Reguły transformacji można projektować za pomocą własnych języków (ich przykład zostanie zaprezentowany w rozdziale czwartym) oraz za pomocą kilku powszechnie dostępnych języków na przykład ATL, QVT, Tefkat.

- **ATL**

ATL - jest językiem transformacji modeli i jest dostarczany wraz z zestawem narzędzi opracowanych przez ATLAS Group (INRIA & LINA). Język ATL ma zastosowanie w syntaktycznych i semantycznych tłumaczeniach. Pozwala on na definiowanie zbioru zasad, które określają sposób mapowania elementów wejściowych na wyjściowe. W celu ułatwienia pracy z językiem ATL platforma Eclipse została wzbogacona o zintegrowane środowisko programistyczne (ATL IDE)- dostarczające szereg standardowych narzędzi programistycznych[1].

- **QVT**

Query/View/Transformation Language - jest standardowym zestawem języków transformacji modeli zaproponowanym przez organizację Object Management Group (OMG). QVT jest kluczową techniką transformacji stosowaną w modelowaniu architektury, definiuje zapytania, widoki i transformacje. Język QVT określa trzy modele transformacji:

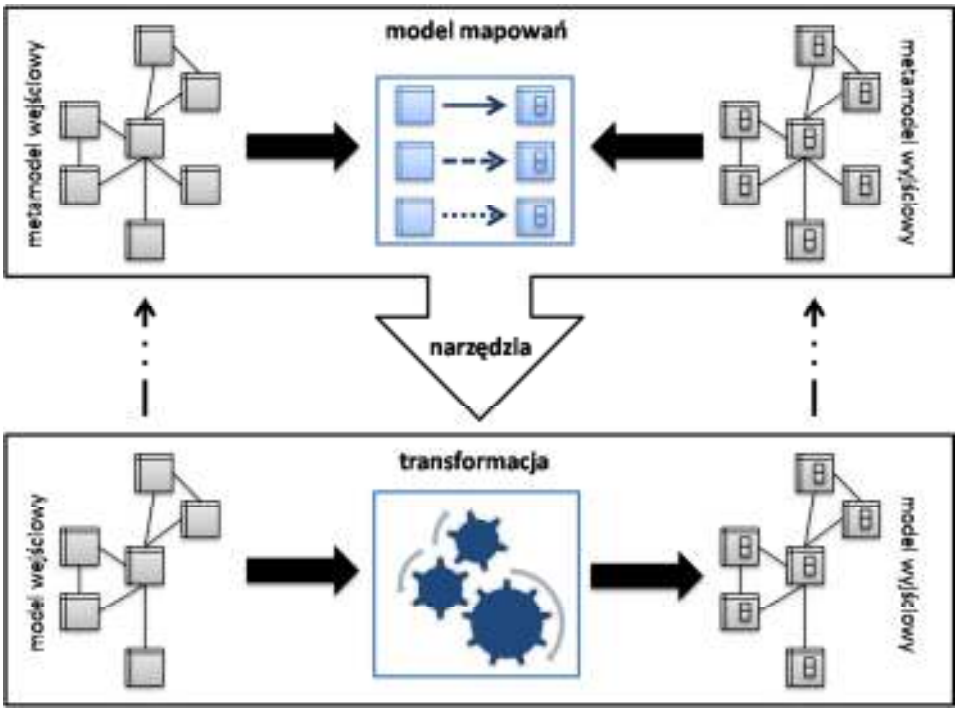
- QVT-Operational - imperatywny język przeznaczony do pisania jednokierunkowych transformacji;
- QVT-Relations - deklaratywny język (zarówno tekstowy jak i graficzny) umożliwiający jednokierunkowe i dwukierunkowe transformacje modelu;
- QVT-Core - deklaratywny język do tłumaczenia modeli zapisanych w QVT-Relations.

Dodatkowo wyróżnia się jeszcze narzędzie QVT-BlackBox - jako mechanizm służący do wywoływania obiektów transformacji [20].

[1] http://www.eclipse.org/m2m/atl/atlTransformations/

- **Tefkat**

Tefkat jest deklaratywnym językiem transformacji modeli oraz danych wykorzystywanym na platformie Eclipse. Realizowany jest przez specjalizowany plug-in platformy Eclipse i wykorzystuje Eclipse Modeling Framework (EMF) do obsługi modeli opartych na specyfikacjach: MOF, UML2 i XML Schema. Tefkat definiuje mapowanie zestawu metamodeli wejściowych na zestaw metamodeli wyjściowych. Transformacje zbudowane są z reguł, wzorców i szablonów. Tektaf zaprojektowany został z myślą o pisaniu skalowalnych specyfikacji transformacji, które umożliwiają wielokrotne ich wykorzystywanie przy wysokim poziomie koncepcji domeny zamiast bezpośredniego operowania na składni języka XML[2].

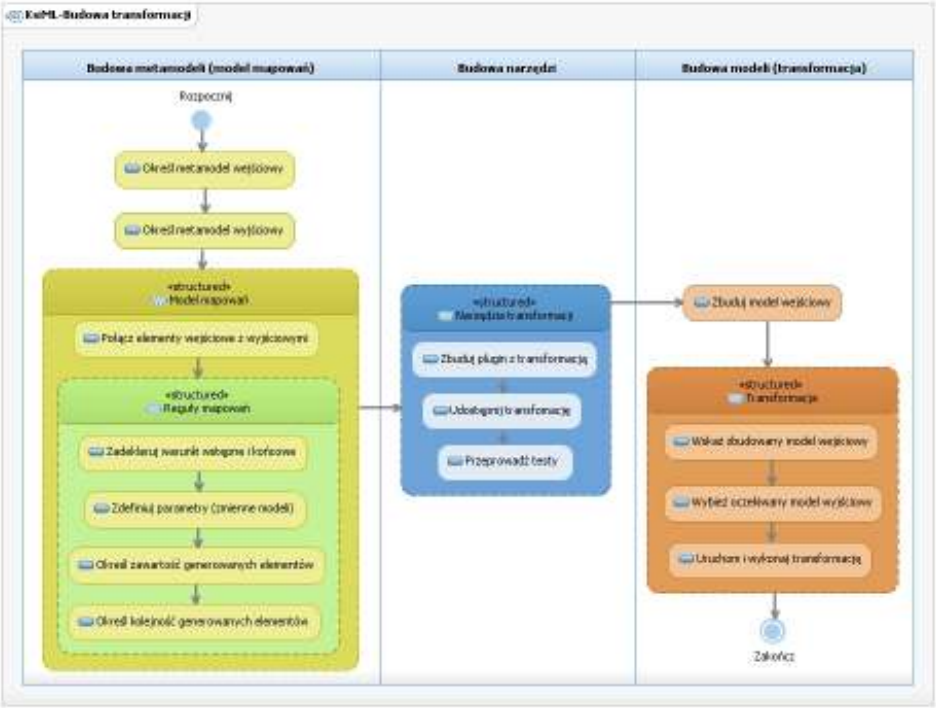


Rys. 4. Koncepcja transformacji modeli według metody KMS

[2] http://tefkat.sourceforge.net/

3. Proces transformacji według KMS

Budowa transformacji w kontekście wytwarzanie serwisów informacyjnych wykorzystujących modelowanie dziedzin ma kluczowe znaczenie, ponieważ pozwala na automatyzowanie zadań projektowych (przekształceń modeli) i minimalizuje czas potrzebny na wytworzenie rozwiązań informatycznych. Poprawna realizacja tego procesu wymaga wykonania szeregu zależnych od siebie czynności (rys. 5).



Rys. 5. Proces budowy transformacji zgodny z metodą KMS

Przekształcenie, w ramach metody KMS, wiążę się z realizacją całego procesu budowy transformacji (rys. 4). Przed rozpoczęciem budowy transformacji wymaga się wytworzenia lub wykorzystania dostępnych wzorców i zastosowania ich w modelu mapowań. Udostępnienie specjalistycznych narzędzi umożliwia wielokrotne wykorzystywanie automatycznych przekształceń i pozwala w końcu na wykonywanie docelowych transformacji modeli (rys. 1).

Transformacje modeli pozwalają na realizację podstawowych celów stawianych przed architekturą systemu w MDA [21] i KMS [16]:

- *Przenośność* - możliwość wykorzystywania tego samego rozwiązania na wielu platformach;
- *Interoperacyjność* - tworzenie systemów, które można łatwo ze sobą zintegrować i zapewnić skuteczną komunikację między nimi (te same informacje powinny być tak samo rozumiane przez wszystkich uczestników procesów ich przetwarzania);
- *Ponowne użycie* - budowa mechanizmów, które mogą być wielokrotnie użyte.

Proces budowy transformacji w metodzie KMS został pokazany na rysunku 5.

Zasadnicze prace dotyczą tu zbudowania *modelu mapowań* – będącego wzorem, który za pomocą dedykowanych narzędzi umożliwia automatyczne transformowanie elementów wejściowych na elementy wyjściowe.

Należy pamiętać, że przed transformacją modelu, trzeba wykonać jego walidację (sprawdzenia zgodności z metamodelem). Dopiero pozytywnie zweryfikowany model wejściowy może podlegać transformacji i w konsekwencji prowadzić do generacji modelu wyjściowego [11].

Budowa mechanizmów przekształceń wielokrotnego użytku wiąże się z opracowaniem wzorca i zaimplementowaniem go w modelu mapowań (ang. *Mapping Model*). W szczególności należy zdefiniować *język opisu transformacji*, traktowany jako kompletny i spójny zbiór reguł dotyczący powiązań i przekształceń elementów modeli. W wyniku mapowania określone zostają związki między modelami oraz zasady transformacji, determinowane przez wybór platformy, dla której budowany jest model wyjściowy. Działania te wymagają:

- szczegółowego określenia modelu wejściowego i docelowego;
- połączenia elementów modelu wejściowego z elementami modelu wyjściowego;
- deklaracji warunków wstępnych i końcowych przekształceń;
- definicji parametrów (traktowanych jako zmienne modeli);
- określenia zawartości i kolejności generowanych elementów.

Poprawny model mapowań to taki, który jest kompletnym repozytorium reguł mapowań w danej dziedzinie (tj. szablon z zasadami transformacji modeli). Zasady transformacji w narzędziach CASE przybierają postać tzw. *map powiązań* elementów źródłowych i docelowych, wzbogaconych o ograniczenia nałożone na model wynikowy.

Można wyróżnić dwa główne rodzaje mapowań: *Model Type Mapping* i *Model Instance Mapping* [21]. Ich charakterystyka została przedstawiona w tab. 2.

Tab. 2. Zestawienie informacji o Model Type Mapping i Model Instance Mapping

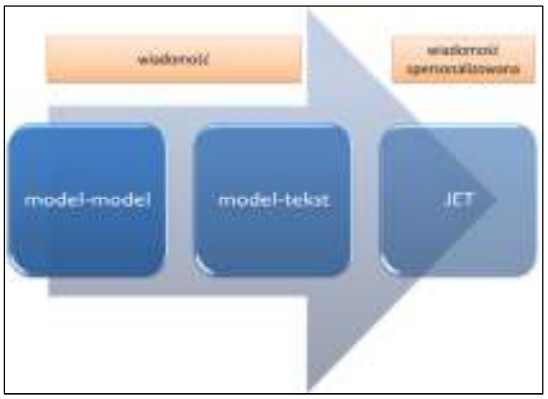
Nazwa	***Model Type Mapping***	***Model Instance Mapping***
Opis	**Mapowanie na poziomie metamodelu.** Przekształcenie to odnosi się nie do samego elementu modelu, lecz do jego typu. Tworząc model PIM (model dziedziny) używa się typu elementu istniejącego w danym języku. Przekształcenie to odbywa się według zdefiniowanych reguł i prowadzi do powstania typu elementu istniejącego w innym języku. Mapowanie definiuje się pomiędzy elementami tego samego typu (np. klasy z klasą, atrybutów z atrybutami, itd.).	**Mapowanie instancji modelu.** Przekształcenie odnosi się do elementu modelu. Realizacja takiego mapowania wymaga od projektanta wcześniejszego oznaczenia elementów w modelu źródłowym PIM (model dziedziny). Podczas wykonywania przekształcenia następuje identyfikacja elementów (określonych indywidualnym znakiem) i sposobu transformacji, a następnie zastosowanie jej i przekształcenie do odpowiedniego elementu w modelu docelowym PSM (model systemu).
Przykład	Przykładem jest przekształcenie klasy w encję. Specjalnym przykładem jest mapowanie metamodelu (tzw. Metamodel Mappings). W tym przypadku zarówno elementy modelu PIM (model dziedziny) jak i PSM (model systemu) są zdefiniowane w repozytorium MOF (Meta-Object Facility). Celem mapowania jest przekształcenie typów obiektów należących do jednego metamodelu (na jego podstawie określany jest model PIM - model dziedziny) na typy elementów należących do innego (na podstawie którego zbudowany został model PSM - model systemu).	Istnieje wiele sposobów oznaczania modelu PIM. Jednym z nich jest zastosowanie tzw. profili stanowiących rozszerzenie języka UML. Stereotypy jako elementy modelu dziedziny oznaczone są indywidualnymi cechami (nazwą, ikoną, kształtem, atrybutami, ograniczeniami). Podczas transformacji następuje identyfikacja stereotypów i ich przekształcenie, zgodnie z ustalonymi zasadami (w modelu mapowań), do odpowiedniego elementu w modelu docelowym PSM (model systemu) - w ramach pracy wytworzone zostały mechanizmy w języku C#.

Dozwolone jest wykorzystywanie w jednym projekcie dwóch mapowań (*Model Type Mapping* i *Model Instance Mapping*) i tworzenie za ich pomocą specjalistycznych modeli łączonych [9], [21].

4. Przykładowe transformacje modeli

W kolejnych podrozdziałach zostaną zaprezentowane przykładowe transformacje modeli typu: M2M, M2T, JET, zdefiniowane w [16] i dotyczące reprezentatywnej funkcjonalności serwisu informacyjnego, tzn. mechanizmu ogłoszeń (ograniczonego do pojęcia wiadomość (rys. 6)). W trakcie prac projektowych wykorzystano materiały szkoleniowe IBM ([1], [11], [13], [14]).

Budowa modeli mapowań została oparta o koncepcję typu *Model Instance Mapping*, co było konsekwencją decyzji oznaczania elementów źródłowych (modelu dziedziny) za pomocą profili UML.



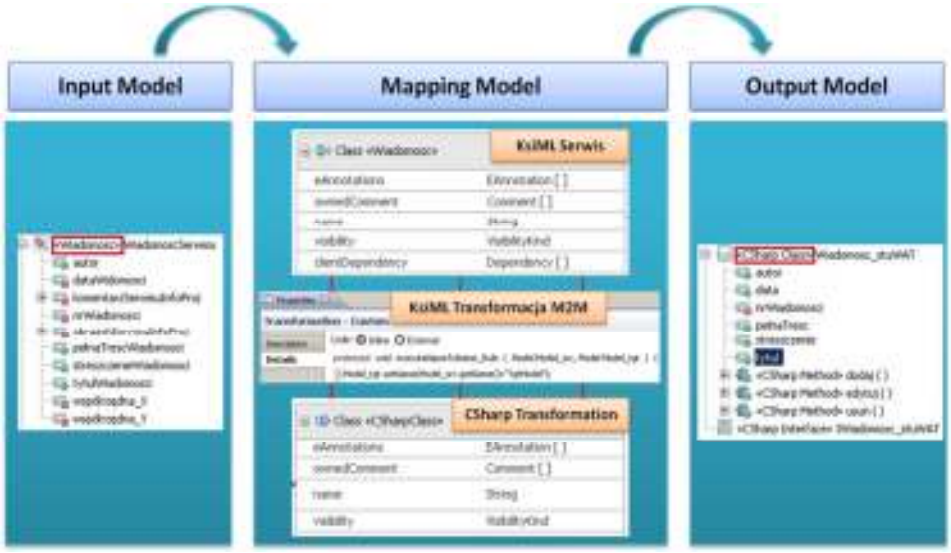
Rys. 6. Przegląd trzech transformacji wytworzonych w [16]

4.1. Transformacje typu M2M

Transformacja typu model-model (M2M) jest rodzajem przekształcenia automatycznego. Uogólnioną koncepcję jej działania prezentuje rys. 4. Odpowiednikiem przekształcenia M2M w metodzie KMS jest transformacja TM.

Transformacja TM zbudowana została w celu dostarczenia mechanizmów umożliwiających przekształcenie modelu dziedziny w model systemu serwisu informacyjnego. Składnikiem modelu mapowań dla tej transformacji był wzorzec programowy *„MójWydział”*.

Wejściem w transformacji TM (rys. 1) jest jeden z elementów modelu dziedziny - element *„WiadomośćSerwisu”*, wyjściem jest również jeden element modelu systemu *„Wiadomość_stuWAT”*. Schemat wraz z wizualizacją dla omawianej transformacji TM przedstawiono na rys. 7.



Rys. 7. Transformacja TM

W tab. 3 zostały zaprezentowane czynności algorytmu budowy transformacji TM w środowisku IBM Rational Software Architect.

Tab. 3. Czynności algorytmu budowy transformacji M2M w środowisku IBM RSA

Lp.	Nazwa czynności
1.	Budowa nowego szablonu modelu mapowań M2M (wzorca transformacji) w środowisku CASE.
2.	Nadanie jednoznacznej nazwy dla nowego modelu mapowań M2M.
3.	Określenie metamodelu wejściowego (ang. *Mapping input*), tzn. wskazanie dedykowanego profilu UML.
4.	Określenie metamodelu wyjściowego (ang. *Mapping output*), tzn. wskazanie standardowego lub autorskiego profilu dostępnego w środowisku CASE.
5.	Budowa modelu mapowań za pomocą projektu typu *.mapping* (tzn. zaprojektowanie *map powiązań* między elementami modeli).
6.	Połączenie elementów modelu wejściowego z elementami modelu wyjściowego (tzn. mapowanie elementów modelu wejściowego na dedykowane elementy stereotypowe).

7.	Budowa języka opisu transformacji (tzw. reguł mapowań). Zdefiniowanie zasad dotyczących powiązań i przekształceń modeli): a) zadeklarowanie warunków wstępnych i końcowych, b) zdefiniowanie parametrów (zmiennych modeli), c) określenie zawartości generowanych elementów, d) określenie kolejności generowanych elementów.
8.	Budowa *plug-inu* z modelem mapowań (narzędzie umożliwiające wielokrotne wykorzystywania opracowanych przekształceń M2M).
9.	Testowanie otrzymanego modelu mapowań. Opracowanie i uruchomienie tzw. konfiguracji transformacji w nowej przestrzeni roboczej środowiska RSA, polegające głównie na wskazaniu lokalizacji elementów źródłowych i docelowych (tzw. *SourceModel* oraz *TargetModel)*.
10.	Uruchomienie transformacji M2M. W jej wyniku powinien powstać nowy projekt wraz z wygenerowanymi elementami modelu docelowego.

Realizację zadań projektowych w narzędziu IBM RSA rozpoczynamy od utworzenia szablonu i nadaniu mu nazwy (dla budowanej transformacji określono, że będzie to: *„KsiML_Transformacja_M2M”*). Model wejściowy określony został przez profil *„KsiML_Serwis”*, zaś model wyjściowy przez profil *„CSharp_Transformation”*. W kolejnych czynnościach dokonano mapowania elementów modelu dziedziny, polegającego na opracowaniu mapowania dla reprezentatywnego elementu *„WiadomoscSerwis”* (co pokazuje rys. 7) z elementami modelu systemu w języku C# (określonymi przez *C# profile*). Tab. 4 prezentuje stereotypy profilu *C# profile* wykorzystane do określenia właściwości elementów modelu wyjściowego transformacji TM.

Tab. 4. Elementy C# profile wykorzystane w transformacji TM

Name	Stereotype
Classes	<<CSharp Class>>
Structures (struct)	<<CSharp Struct>>
Delegates	<<CSharp Delegate>>
Enumerations (enum)	<<CSharp Enum>>

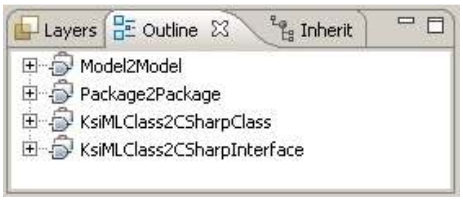
Definicje mapowań określane są pomiędzy elementami tego samego typu (np. klasy z klasami, pakiety z pakietami, atrybuty z atrybutami).

W omawianym przykładzie modelu mapowań (M2M) zbudowane zostały dwa oddzielne projekty:

1) KsiML_Transformacja_M2M.mapping

W projekcie *„KsiML_Transformacja_M2M.mapping”* określone zostały cztery mapy powiązań elementów modelu wejściowego z elementami modelu wyjściowego:

- Model2Model;
- Package2Package;
- KsiMLClass2CSharpClass;
- KsiMLClass2CSharpInterface.

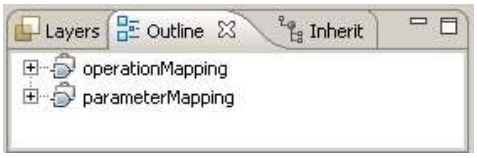


Rys. 8. Model KsiML_Transformacja_M2M - widok czterech map powiązań

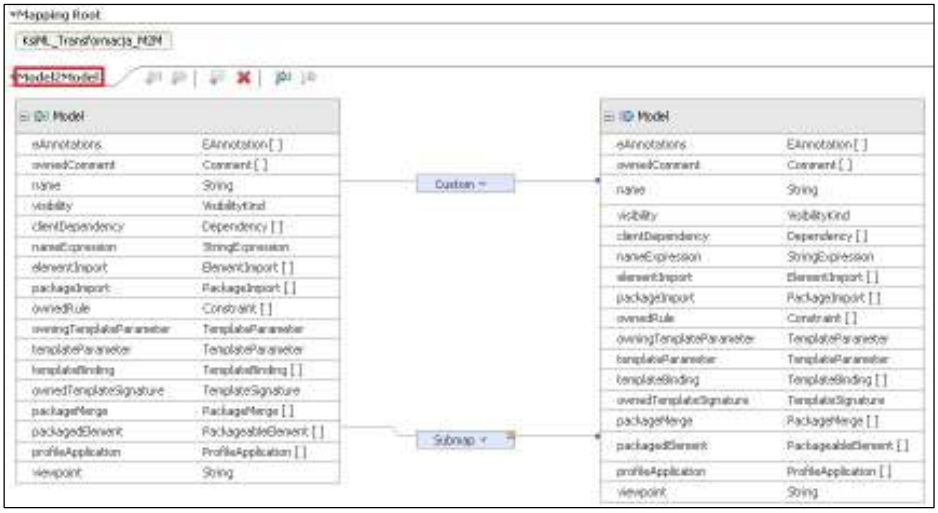
2) OperationMapping.mapping

W projekcie *„OperationMapping.mapping”* opracowane zostały dwie mapy powiązań:

- operationMapping;
- parameterMapping.



Rys. 9. Model OperationMapping – widok dwóch map powiązań



Rys. 10. Projekt KsiML_Transformacja_M2M.mapping z mapą powiązań Model2Model

Logika działania transformacji TM została zdefiniowana za pomocą języka opisu transformacji. Wykorzystując możliwości dedykowanego kreatora RSA zaprojektowane zostały reguły przekształceń (rys. 10), określone zostały ich własności (rys. 11) wraz z kodem odpowiedzialnym za transformacje.

Zasady mapowania definiują sposób przekształcenia elementów wejściowych w wyjściowe. W zależności od danych wejściowych można korzystać z różnych typów reguł mapowania elementów modeli w środowisku IBM RSA [14], wyróżnia się m. in.:

- **Move** – pozwala na kopiowanie jednej wartości atrybutu źródłowego do atrybutu docelowego. Należy pamiętać, aby typ danych atrybutu wejściowego był zgodny z typem danych elementu wyjściowego,
- **Submap i Custom submap** – pozwalają na mapowanie pojedynczego elementu modelu wejściowego z kilkoma elementami modelu wyjściowego,
- **Custom** – pozwala na obliczenie wartości właściwości elementu wyjściowego przy użyciu specjalistycznego kodu.

Wymienione reguły mapowania mogą być dodatkowo udoskonalane za pomocą predefiniowanych własności: **Condition**, **Input Filter**, **Output Filter**, **Extractors**, **Custom Feature**, **Custom Output**.

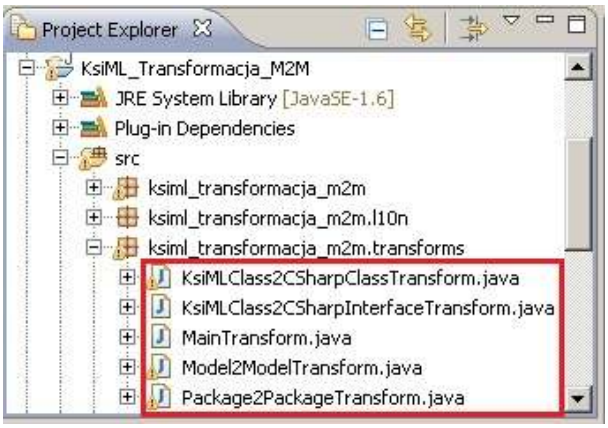
Rys. 11. Widok Outline- reguły przekształceń, widok Properties – szczegóły Custom

Rys. 12 prezentuje strukturę modelu mapowań z wyróżnionymi klasami języka Java odpowiedzialnymi za obsługę i realizację reguł transformacyjnych w środowisku IBM RSA.

Dalsze działania projektowe polegają na opracowaniu *plug-inu* z modelem mapowań jako narzędziem umożliwiającym wielokrotne wykorzystywanie mechanizmów transformacji TM.

Testowanie zaprojektowanego modelu mapowań wymaga opracowania i uruchomienia konfiguracji transformacji „*KsiML_Transformacja_M2M*". W nowej przestrzeni roboczej środowiska IBM RSA zbudowany został projekt „*Transformacja_M2M_Test"*, a w nim dwa modele:

- ***SourceModel,*** który określono profilami UML prezentującymi język dziedzinowy *KsiML* (zamodelowane zostały elementy i określone ich stereotypy);
- ***TargetModel_stuWAT***, dla którego zastosowany został profil dostępny w środowisku RSA tzn. *C# profile*.

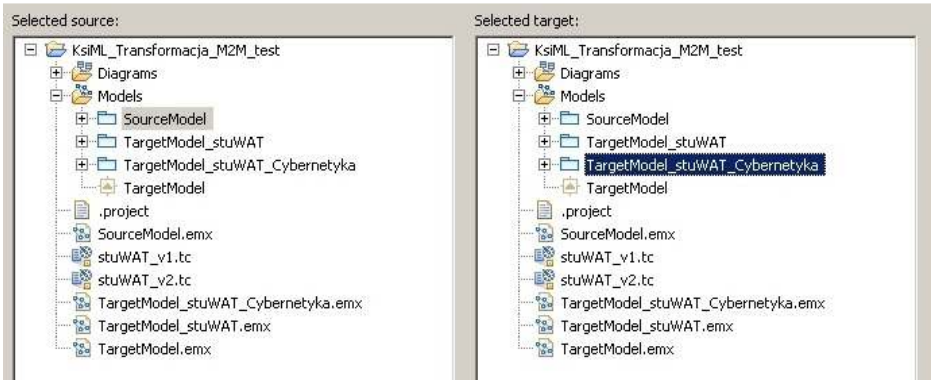


Rys. 12. Model mapowań - widok klas z kodem odpowiedzialnym za przekształcenia

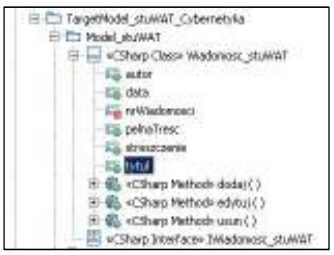
Rys. 13. Wzorzec KsiML_Transformacja_M2M- konfiguracja nowej transformacji

W celu przekształcenia modelu wejściowego (*SourceModel*) w wyjściowy (*TargetModel_stuWAT*), wykorzystana została konfiguracja transformacji TM z wzorca „*KsiML_Transformacja_M2M"* (rys. 13) i na jej podstawie wygenerowano nowy projekt o nazwie „*stuWAT_v1"*. Dalsze kroki polegały na doprecyzowaniu konfiguracji transformacji TM (rys. 14) i wskazaniu modelu wejściowego i wyjściowego.

Wynikiem uruchomienia i wykonania transformacji TM jest nowy projekt z wygenerowanymi elementami (rys. 15). Wszystkie przekształcenia zostały wykonane automatycznie na podstawie reguł określonych w modelu mapowań.



Rys. 14. Transformacja TM- proces wskazania modelu wejściowego i wyjściowego



Rys. 15. Model wyjściowy – wynik transformacji TM

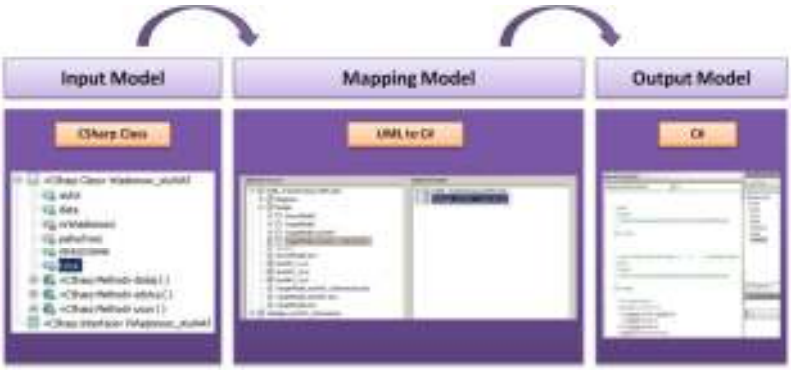
Kompletny model mapowań powinien określać wszystkie możliwe zależności między elementami modelu wejściowego i wyjściowego oraz zasady ich przekształceń.

Podsumowując, istotą działania transformacji typu *model-model* jest obsługa procesu przekształceń elementów modelu wejściowego (model dziedziny) na elementy modelu wyjściowego (model systemu).

4.2. Transformacje typu M2T

Transformacja typu model-tekst (M2T) jest rodzajem przekształcenia automatycznego, w którym model wejściowy zostaje przetransformowany w kod rozwiązania. Odpowiednikiem przekształcenia M2T w metodzie KMS jest transformacja TF. Transformacja TF (rys. 16) jest budowana w celu generacji kodu rozwiązania w jednym z języków programowania (C#, Java, itp.) w zależności od opracowanego modelu systemu.

Wejściem w transformacji TM jest jeden z elementów modelu systemu, element *„Wiadomość_stuWAT”*, wyjściem jest szkielet kodu w języku C#. Schemat wraz z wizualizacją dla omawianej transformacji TF prezentuje (rys. 16).



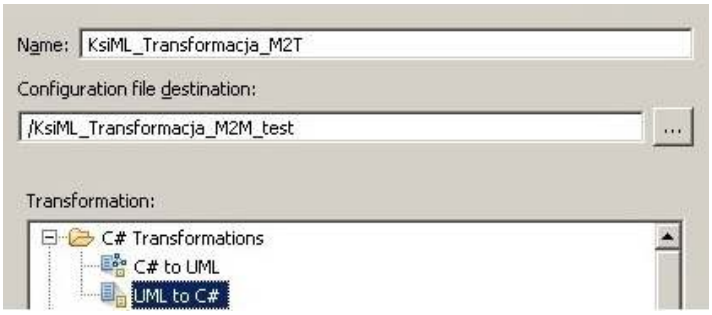
Rys. 16. Transformacja TF

W tab. 5 zostały przedstawione czynności algorytmu budowy transformacji TF w środowisku IBM RSA. Pierwsze czynności projektowe polegają na utworzeniu nowego projektu np. w środowisku Visual Studio 2008. Można wykorzystać wzorzec Visual Studio Solutions, do którego należy dodać nowy projekt (np.: typu ASP.NET Web Application).

Narzędzia CASE oferują szablony przekształceń z języka UML do kodu (m.in. w C#, Java itd.). W ramach przykładu zostanie zaprezentowany model mapowań wykorzystujący wzorzec *UML to C#* dostępny w środowisku IBM RSA.

Tab. 5. Czynności algorytmu budowy transformacji M2T w środowisku IBM RSA

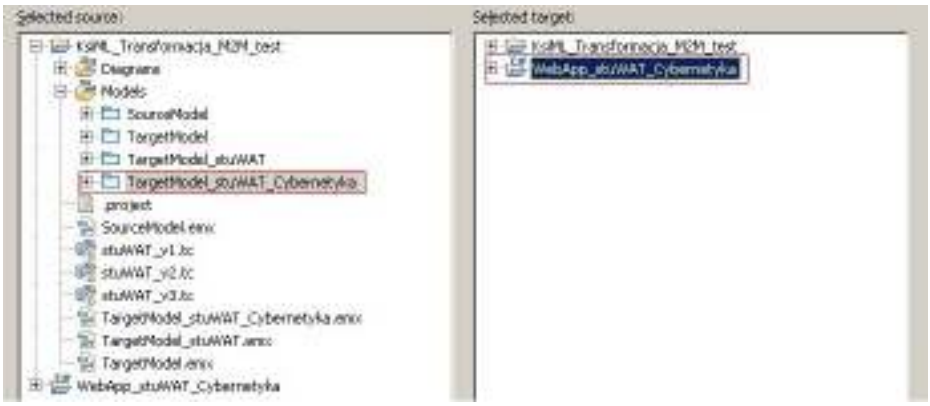
Lp.	Nazwa czynności
1.	Utworzenie nowego projektu w środowisku IDE (np. *Visual Studio*, *NetBeans*, itd.).
2.	Budowa nowego szablonu modelu mapowań M2T, opartego np. o wzorzec dostępny w środowisku CASE (*UML to C#*, *UML to Java*, itd.).
3.	Nadanie indywidualnej nazwy dla nowego modelu mapowań M2T.
4.	Kastomizacja wzorcowego modelu mapowań, tzn. wykorzystanie kreatora wbudowanego w IBM RSA do konfiguracji konkretnej transformacji M2T, polega w szczególności na wskazaniu: a) modelu wejściowego (przekształcanego), b) modelu wyjściowego, c) elementów przekształcanych.
5.	Uruchomienie i wykonanie transformacji M2T, co powinno skutkować wygenerowaniem szkieletu kodu np. *C#*, *Java*, itp. w projekcie IDE.



Rys. 17. Wzorzec UML to C#- konfiguracja nowej transformacji

Główne czynności algorytmu polegają na konfiguracji transformacji TF dla problemu projektowego. Modelem przekształcanym (wejściowym) został określony *„TargetModel_stuWAT_Cybernetyka”*, zaś modelem wyjściowym projekt *„WebApp_stuWAT_Cybernetyka”* wykonany w języku C#.

Widok kreatora umożliwiającego konfigurację transformacji TF pokazuje rys. 18. Wykonanie transformacji TF skutkuje wygenerowaniem szkieletu kodu w języku C# (dla opracowanej funkcjonalności ogłoszeń serwisu) w projekcie Visual Studio 2008 (rys. 19).



Rys. 18. Model wejściowy i wyjściowy- określenie zasad transformacji TF

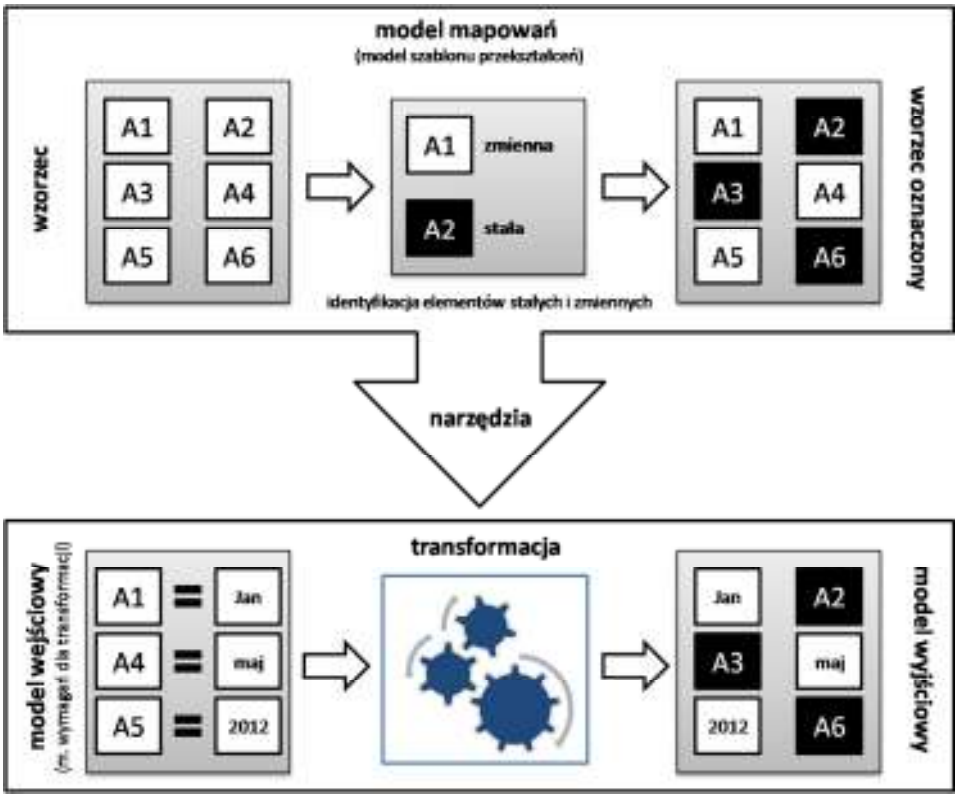
Rys. 19. Model wyjściowy - widok projektu VS po transformacji TF

Podsumowując istotą działania transformacji typu *model-tekst* jest obsługa procesu przekształceń elementów modelu wejściowego (model systemu), na kod rozwiązania (w jednej z wybranych technologii).

4.3. Transformacje typu JET

Transformacja TJ jest przykładem transformacji automatycznej typu *Java Emitter Template* (JET), która pozwala w szczególności na obsługę i automatyzację zmiennych części kodu języków skryptowych, m. in. HTML ([13], [14]). Raz zaprojektowany model mapowań (w metodzie KMS dla transformacji TJ określony mianem *modelu szablonu przekształceń* [16]) pozwala na przeprowadzenie wielu transformacji TJ i dołączenie wygenerowanych artefaktów do rozwiązań opracowanych w różnych technologiach.

Budowa mechanizmów transformacji typu JET wymaga wytworzenia specyficznych elementów i wykonania szeregu zadań (tab. 6). Sama koncepcja transformacji TJ różni się od dwóch pozostałych, (TM oraz TF (rys. 4)) i z tego powodu został dla niej zbudowany oddzielny schemat (rys. 20).



Rys. 20. Koncepcja transformacji JET według metody KMS

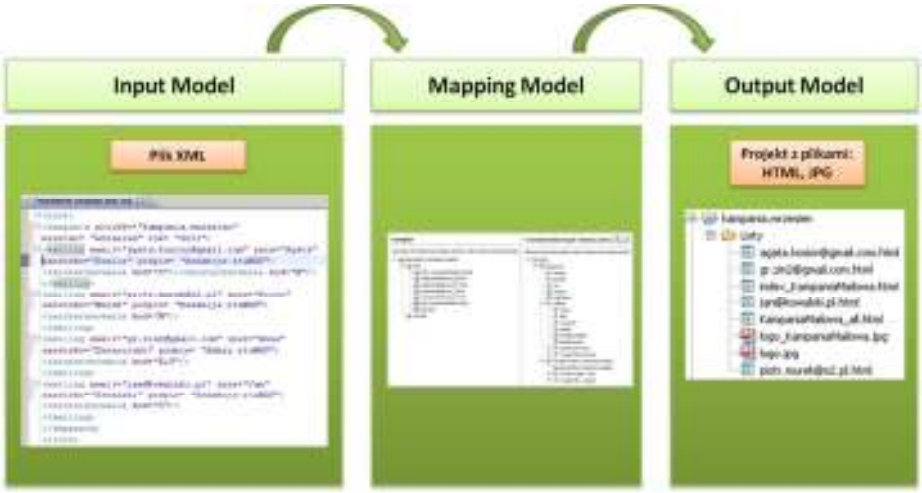
Wzorzec w ramach prac nad modelem mapowań JET wymaga analizy oraz identyfikacji jego elementów zmiennych i stałych, co pozwala na odpowiednie jego (wzorca) oznaczenie i w konsekwencji udostępnienie w narzędziach transformacji.

Docelowa transformacja (rys. 1) przebiega następująco: wejściem jest model wymagań dla transformacji (plik XML definiujący wartości dla elementów określonych jako zmienne), wyjściem jest model spersonalizowanego wzorca (dostosowany do wymagań danej transformacji). Schemat transformacji z wizualizacją dla omawianej transformacji TF pokazuje (rys. 21).

W dalszej części zostanie zaprezentowany przykład transformacji JET dla obsługi mechanizmu newslettera (opracowywanie listów elektronicznych do subskrybentów, którzy w sposób indywidualny określili swoje zainteresowania). Prace projektowe zostały wykonane z wykorzystaniem materiału szkoleniowego IBM [12] oraz prac [1] i [12].

Tab. 6. Czynności algorytmu budowy transformacji JET w środowisku RSA

Lp.	Nazwa czynności
1.	Opracowanie zestawu szablonów reprezentujących artefakty, które muszą być wygenerowane w wyniku przekształcenia (tzw. *wzorzec*, który jest najlepszym rozwiązaniem problemu projektowego): a) utworzenie nowego projektu transformacji, b) zaimportowanie wzorca do nowej przestrzeni roboczej IBM RSA.
2.	Zaprojektowanie zestawu szablonów odpowiedzialnych za kontrolę procesu i zadań transformacji (*model szablonu przekształceń*): a) budowa nowego projektu korzystającego z funkcji Exemplar Authoring, b) określenie stałych (niezmiennych elementów wzorca), c) określenie zmiennych, d) ustanowienie zasad obsługi zidentyfikowanych elementów.
3.	Zdefiniowanie modelu zawierającego informacje potrzebne jako wejście do transformacji (*model wymagań dla transformacji*).
4.	Wykonanie testów transformacji: a) budowa modelu wejściowego w postaci pliku XML, b) zastosowanie transformacji TJ, c) wygenerowanie modelu wyjściowego (artefaktów projektu).



Rys. 21. Transformacja TJ

Zbudowany został mechanizm wspierający obsługę kampanii mailowej serwisu *„stuWAT”*. Zadania projektowe rozpoczęto od przygotowania plików wejściowych (wzorcowych), które powstały w postaci projektu HTML o nazwie „*Newsletter.Kampania.szablon”*. Do budowy projektu wejściowego wykorzystany został wzorzec *Library Definitions Project* (dostępny w szablonach Web środowiska IBM RSA).

Opracowano wiadomości wzorcowe dla trzech dziedzin tematycznych. W celu sprawniejszego zarządzania tymi wiadomościami zbudowany został plik HTML („*KampaniaMailowa_all.html”*), który łączy w sobie treść trzech newsletterów. Przegląd zawartości *newsletterów* pozwolił na wyodrębnienie treści wymagających automatyzacji (elementów zmiennych). Wynik tych prac prezentuje rys. 22.

Stałym elementem wszystkich trzech wiadomości jest wstęp w postaci *„W tym miesiącu redakcja stuWAT szczególnie poleca następujące materiały edukacyjne”*.

Mechanizm newslettera powstał z myślą o obsłudze przyszłych kampanii wysyłkowych, dlatego podjęto decyzję o automatyzacji (oznaczenia jako zmienne) następujących elementów:

- imię subskrybenta,
- miesiąc,
- rok,
- podpis,
- treść.

Reguły przekształceń dla transformacji TJ zgodnie z metodą KMS zostały określone w modelu mapowań (*modelu szablonu przekształceń)* (rys. 1). W tym celu zbudowany został projekt typu *JET Project with Exemplar Authoring* o nazwie *„newsletter.kampania.transformacja1”*.

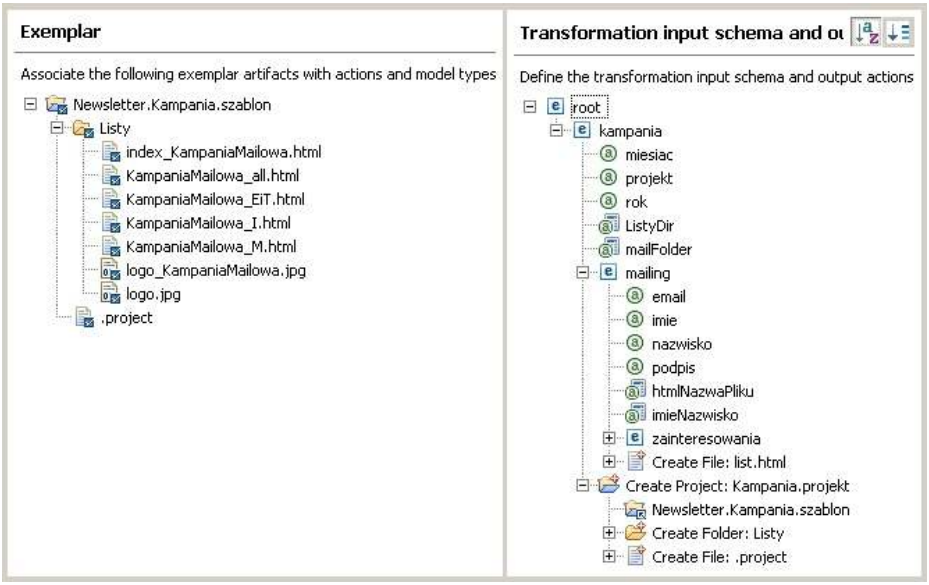
Środowisko IBM RSA udostępnia edytor transformacji JET (rys. 23), który umożliwia projektowanie reguł odpowiedzialnych za przekształcenia modeli. Edytor dostarcza elementy, które pozwalają na precyzyjne projektowanie zasad przekształceń; są nimi: **Type, Attribute, Project, Folder, File, Derived Attribute**.

Lewe okno edytora (rys. 23) odzwierciedla schemat systemu plików projektu wzorcowego, prawe przeznaczone jest na opis reguł transformacji (tworzą go elementy wiadomości zidentyfikowane jako wymagające automatyzacji i dodatkowo elementy pośrednie pozwalające na obsługę reguł transformacji).

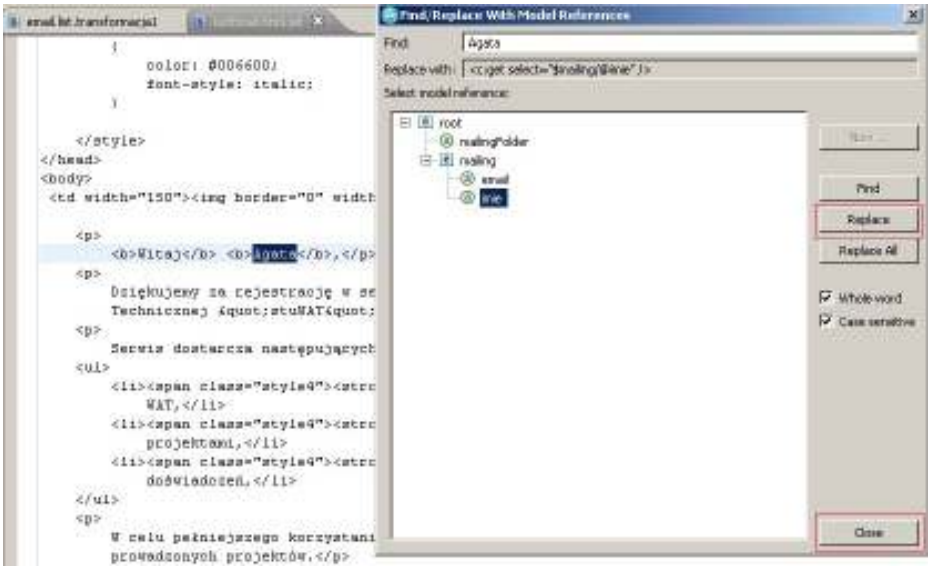
Budowa modelu szablonu przekształceń polegała na zaprojektowaniu reguł odpowiedzialnych za obsługę i automatyzację elementów zmiennych wiadomości newslettera. Ostateczny kształt modelu pokazuje (rys. 23).

Możliwość aktualizowania własności oraz dodawanie referencji w modelu szablonu przekształceń, pozwala na precyzyjne projektowanie elementów zmiennych. Przykładowy proces określania referencji prezentuje (rys. 24). Reguły generacji treści *newslettera* zostały określone za pomocą warunków definiujących zainteresowania subskrybenta.

Rys. 22. Identyfikacja elementów zmiennych w wiadomości Newslettera



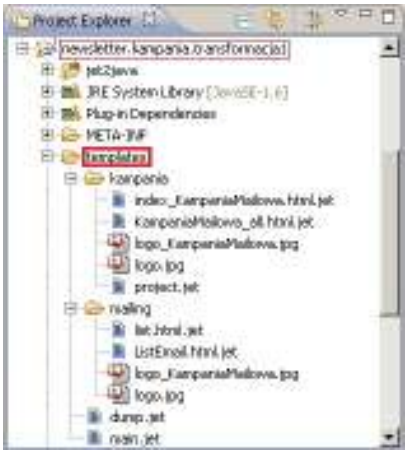
Rys. 23. Edytor transformacji JET dla Newslettera



Rys. 24. Określania referencji między modelem wejściowym a szablonem przekształceń

Schemat modelu szablonu przekształceń odpowiada za opis struktury modelu przekazywania danych do transformacji TJ. W wyniku zaprojektowania modelu szablonu przekształceń powstał dedykowany folder (templates)

przechowujący wszystkie wzorce wiadomości oraz informacje o strukturze projektu (rys. 25).



Rys. 25. Model szablonu przekształceń – zawartość templates

Testy transformacji TJ zapoczątkowane zostały budową pliku wejściowego XML, który zawiera wymagane dane do przekształcenia i wygenerowania nowego projektu (w metodzie KMS określany jako *model wymagań dla transformacji*). Zawartość przykładowego pliku testowego (*„newsletter.wrzesien.test.xml”*) przedstawiono w tab. 7.

Tab. 7. Zawartość pliku wejściowego do transformacji JET

```
<root>
<kampania projekt=kampania.wrzesien miesiac= wrzesień rok= 2011>
<mailing email=agata.kosior@gmail.com imie=Agata nazwisko=Kosior podpis= Redakcja stuWAT>
<zainteresowania kod=I/><zainteresowania kod=M/>
</mailing>
<mailing email=piotr.murek@o2.pl imie=Piotr nazwisko=Murek podpis= Redakcja stuWAT>
<zainteresowania kod=M/>
</mailing>
<mailing email=gr.zin2@gmail.com imie=Anna nazwisko=Zwierciado podpis= Admin stuWAT>
<zainteresowania kod=EiT/>
</mailing>
<mailing email=jan@kowalski.pl imie=Jan nazwisko=Kowalski podpis= Redakcja stuWAT>
<zainteresowania kod=I/>
</mailing>
</kampania>
</root>
```

Plik XML został poddany transformacji TJ, która wytworzyła nowy projekt wraz z artefaktami. Wygenerowany projekt o nazwie „*kampania.wrzesien”* zawiera folder „*Listy”*, a w nim dwa pliki graficzne „*logo_KampaniaMailowa.jpg”*, i „*logo.jpg”*, pięć spersonalizowanych

wiadomości newslettera (dostosowanych do zainteresowań odbiorcy) oraz plik „*KampaniaMailowa_all.html”* (raport wiadomości z kampanii wrzesień) 2011 prezentuje rys. 26.

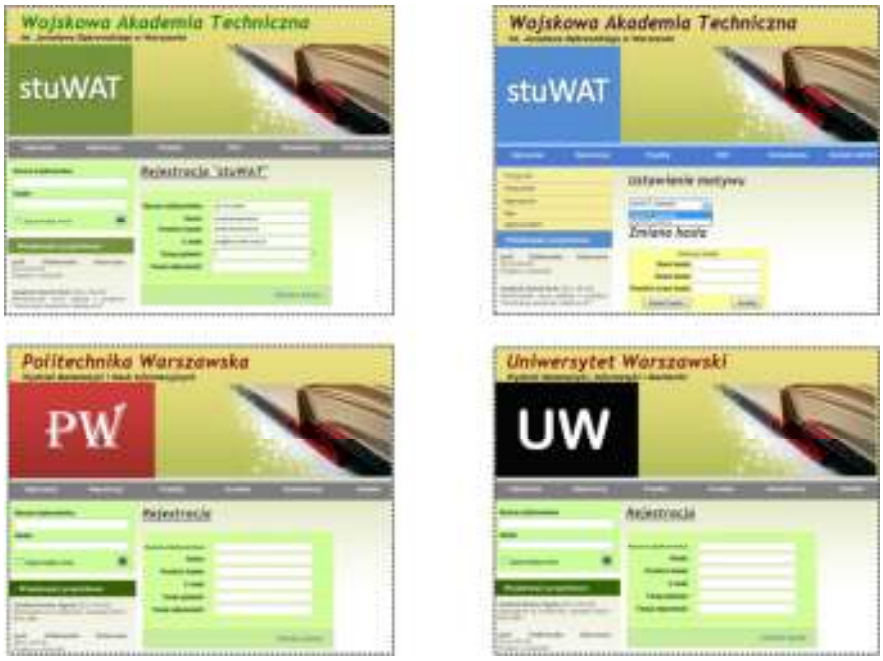
Rys. 26. Strona z raportem kampanii – widok Web Browser

Podsumowując, głównym celem budowy transformacji typu *JET* jest potrzeba automatyzowania powtarzalnych problemów występujących w rozwiązaniach informatycznych.

5. Podsumowanie

W artykule zaprezentowano kompletny proces budowy transformacji. Proces ten został w pełni zweryfikowany, a jego szczegółowe wyniki zamieszczono w [16].

Koncepcja wytwarzania serwisów informacyjnych polegająca na budowie ich oprogramowania poprzez rozwój opracowywanych w procesie wytwórczym modeli oraz transformacji, sprzyja powstawaniu generycznych rozwiązań. Raz zdefiniowane metamodele i metatransformacje pozwalają na ich wielokrotne wykorzystywanie i tym samym automatyzowanie budowy systemów informatycznych. Mechanizmy wytworzone w ramach projektu [16] pozwalają na seryjne generowane rozwiązań z danej dziedziny problemowej – serwisów informacyjnych (rys. 27).



Rys. 27. Transformacje automatyczne – możliwość seryjnej budowy serwisów

Proponowana metoda KMS jest wspierana autorskimi narzędziami i dzięki temu może zostać utrwalona i wykorzystana w nowych przedsięwzięciach. O przewadze metody KMS nad innymi świadczy to, że oprócz dedykowanego języka do budowy modeli (języka dziedzinowego) dostarcza również język opisu transformacji (zawarty w mechanizmach przekształceń), przez co stanowi alternatywną koncepcję wytwarzania systemów informatycznych i jest otwarta na nowe wdrożenia w innych dziedzinach zastosowań. Wykorzystanie doświadczeń i wynikowych artefaktów wytworzonych w ramach [16],[17] może ułatwić wdrożenie koncepcji wytwarzania systemów informatycznych z wykorzystaniem modelowania dziedzin w innych obszarach, niż zaprezentowany serwis informacyjny.

Dostawcy rozwiązań informatycznych coraz częściej spotykają się z zadaniami integracji, konsolidacji czy też rozwoju systemów spadkowych. Dużym problemem jest niewystarczająca dokumentacja, co pociąga za sobą konieczność reinżynierii (analizy kodu i dopiero realizacji bieżących prac). Wartym uwagi jest więc automatyzowanie rozwiązania opisanego problemu przez opracowanie dodatkowych mechanizmów rozszerzających KMS o transformacje typu T2M, które pozwoliłyby na zbudowanie ścieżki przekształceń modeli od kodu do modelu (rys. 1).

Literatura

[1] BERFELD M., COFFIN D., *Customize model-to-text transformation with Rational Software Architect editing tools: Create and run the JET transformation project*, 2010.

[2] BOOCH G., RUMBAUGH J., JACOBSON I., *Unified Modeling Language User Guide*, *2nd* Ed. Addison-Wesley, 2005.

[3] BUDINSKY F., STEINBERG D., MERKS E., ELLERSICK R., GROSE T.J., *Eclipse Modeling Framework.* Addison-Wesley, Reading, MA, 2003.

[4] COOK S., JONES G., KENT S., WILLS A.C., *Domain-Specific Development with Visual Studio DSL Tools,* Addison-Wesley, 2007.

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

[6] DIU W., *Custom domain modeling with UML Profiles: Part 1. Generating and deploying tooling.* IBM DeveloperWorks*,* 2008.

[7] EVANS A., SAMMUT P., WILLANS J. S. (praca zbiorowa), *Metamodelling for MDA*. First International Workshop*,* York, UK, November 2003, Proceedings.

[8] FOWLER M., *Architektura systemów zarządzania przedsiębiorstwem. Wzorce projektowe,* Helion, 2005.

[9] FOWLER M., PARSONS R., *Domain Specific Languages*. Addison Wesley, 2010.

[10] HOVATER S., *Implementing a domain-specific constraint in IBM Rational Systems Developer*. IBM DeveloperWorks, 2006.

[11] *Patterns: Model-Driven Development Using IBM Rational Software Architect (rozdział 6: Model-driven development in context)*.

[12] *Create model-to-text transformations with JET*, Help - Rational Software Architect version 8.0 releases, 2011.

[13] *Domain-Specific Modeling with IBM Rational Software Architect V7.5*, 2009.

[14] *Pattern Implementation Workshop with IBM Rational Software Architect*, 2007.

[15] KELLY S., TOLVANEN J.-P., *Domain-Specific Modeling: Enabling Full Code Generation,* NJ: Wiley, 2008.

[16] KOSIOR A., *Projekt serwisu informacyjnego o projektach studenckich realizowanych w WAT*, Praca dyplomowa, Wydział Cybernetyki WAT, 2011.

[17] KOSIOR A., STASIAK A., *Wytwarzanie serwisów informacyjnych w oparciu o koncepcję modelowania dziedzin. Budowa języka dziedzinowego (KsiML)*. Biuletyn IAiR, NR 32, 2012.

[18] MISIC D., *Authoring UML Profiles: Part 1. Using Rational Software Architect, Rational Systems Developer, Rational Software Modeler create and deploy UML Profiles*, 2008.

[19] MISIC D., *Authoring UML Profiles: Part 2. Using Rational Software Architect, Rational Systems Developer, Rational Software Modeler create and deploy UML Profiles*, 2008.

[20] *OMG. Documents associated with Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.1,* 2011.

[21] *OMG. MDA Specifications. The Architecture of Choice for a Changing World.*

[22] *OMG. Meta Object Facility (MOF) Core Specification Version 2.4.1,* 2011.

[23] PETERSON R., *Create powerful custom tools quickly with Rational Software Architect Version 7.0 (Introduction to the Transformation Authoring feature*), 2007.

Production of information services using the concept of domain modeling. Construction of transformation

ABSTRACT: The paper presents a classification of models transformations (M2M, M2T, T2M, JET) and mapping models. The process of producing generic transformation mechanisms allowing to multiple models transformations is discussed in details. In addition, examples of transformations are presented within the information service project.

KEYWORDS: transformations (M2M, M2T, T2M, JET), description of transformation languages (ATL, QVT, Tefkat), mapping model (Model Type Mapping, Mapping Model Instance), connections map

Praca wpłynęła do redakcji: 11.05.2012