BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 32, 2011

Wytwarzanie serwisów informacyjnych z wykorzystaniem koncepcji modelowania dziedzin. Budowa języka dziedzinowego (KsiML)

Agata KOSIOR, Andrzej STASIAK

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

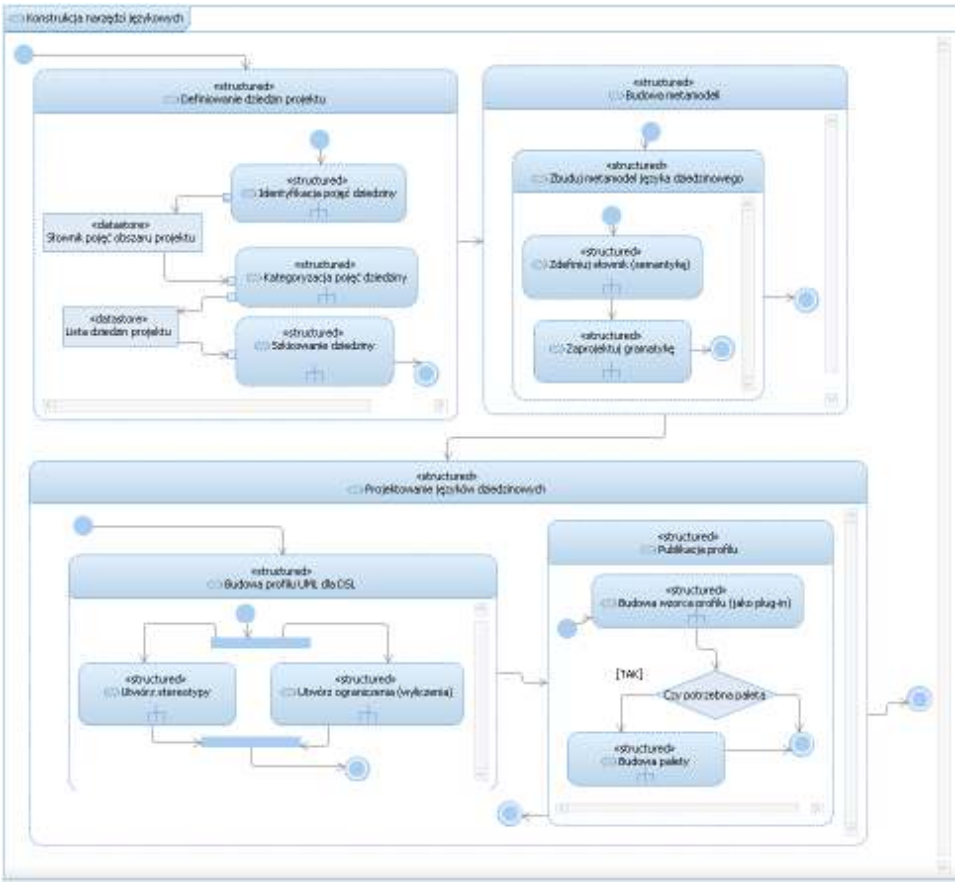
STRESZCZENIE: W artykule przedstawiono opis procesu budowy języka dziedzinowego KsiML z wykorzystaniem MDE, bazując na modelowaniu dziedzin (DSM) i autorskiej metodzie KMS. Proces ten wykorzystano do budowy serwisu informacyjnego o projektach studenckich. W założeniach, zastosowanie zaproponowanego procesu powinno prowadzić do wzrostu jakości i reużycia tworzonego kodu, powodując obniżenie kosztów budowanych systemów.

SŁOWA KLUCZOWE: język dziedzinowy (DSL), modelowanie dziedzin (DSM), języki modelowania dziedzin (DSML), metody wytwarzania SI, UML

1. Wstęp

Modelowanie dziedzin (DSM, ang. *Domain Specific Modeling*) [13] jest techniką inżynierii opartej na modelach (MDE, ang. *Model Driven Engineering*) [23], w której najczęściej poprzez graficzne języki bazujące na piktogramach, staramy się uchwycić semantykę i syntaktykę języka opisu danej dziedziny.

Specyficzne dla dziedziny języki modelowania (DSMLs, ang. *Domain-Specific Modeling Languages*) [8],[10],[13], dostarczają twórcom modeli kluczowe (intuicyjne) abstrakcje, pozyskane w wyniku analizy dziedziny problemu, które dobrze ją opisują, a modele tych abstrakcji pozwalają na formalizację języka opisu dziedziny, tj. jego składni i semantyki.



Rys. 1. Działania prowadzące do opracowania języka dziedzinowego (w metodzie KMS)

Jednym z pierwszych skutecznych środowisk wspierających procesy modelowania języków dziedzinowych (wg. koncepcji „fabryk oprogramowania") było rozwiązanie dostępne jako SDK [3] dla środowiska Microsoft Visual Studio (Visual Studio Visualization and Modeling SDK (VMSDK))[1]. Jednak obecnie rozwój tej koncepcji wytwarzania oprogramowania zasadniczo koncentruje się na platformie Eclipse Modeling Framework [2], jako Generic Eclipse Modeling System (GEMS) [24].

W swojej pracy nad budową serwisów informacyjnych, których z założenia miało powstać kilka (w tym będący przedmiotem projektu w [16]),

[1] Artykuły na temat Visualization and Modeling SDK – Domain-Specific Languages dostępne są na platformie MSDN (http://msdn.microsoft.com/en-us/library/bb126259(v=vs.110).aspx).

postanowiliśmy odwołać się do koncepcji języków dziedzinowych (DSL) głównie ze względu na możliwość weryfikacji utworzonych modeli (zgodnych z DSMLs) dla docelowej dziedziny przez szybkie konstruowanie rozwiązania (ang. *correct by construction*).

Z definicji, język określany jest jako zbiór wyrażeń, którym przyporządkowane są reguły semantyczne i syntaktyczne. Budowa każdego nowego języka, również dziedzinowego wiąże się z opracowaniem i dostarczeniem jego słownika oraz gramatyki. Słownik określa r*eguły semantyczne* i zawiera kompletny zbiór elementów dziedziny wraz z opisem ich znaczenia, a gramatyka stanowi zestaw precyzyjnych reguł opisujących dozwolone połączenia elementów dziedziny (r*eguły* składniowe).

W tym punkcie przedstawiony zostanie opis metody KMS ograniczony do procesu wytwarzania języków dziedzinowych i opracowania narzędzi językowych (rys. 1). Proces budowy języka dziedzinowego rozpoczynamy od opracowania jego szkicu, następnie definiujemy metamodel języka, opracowujemy profil oraz wymagane narzędzia (jako plugin'y profilu, palety języka, edytory). Działania te zostaną przedstawione w kolejnych podpunktach.

Cały przedstawiony proces zilustrowano na przykładzie wytwarzania jednego z trzech, z opracowanych w ramach pracy [16] języka dziedzinowego „KsiML Serwis", który ze względu na swoją złożoność może stanowić reprezentatywny przykład języka DSL[2].

W ramach pracy [16] powstał język dziedzinowy KsiML (Kosior Information Service Modeling Language), który tworzą trzy jego dialekty („KsiML Serwis", „KsiML Projekt, „KsiML Komunikacja"**)**, jako języki dziedzinowe DSL bazujące na języku UML i opisane za pomocą trzech profili:

1. **KsiML Serwis** − definiuje terminy i gramatykę języka projektowania serwisów informacyjnych (portali, vortali);
2. **KsiML Projekt** − definiuje język modelowania dziedziny, którą stanowią projekty prowadzone za pomocą serwisu informacyjnego;
3. **KsiML Komunikacja** − definiuje język pozwalający na budowanie mechanizmów tzw. dziedziny komunikacyjnej serwisów WWW. Jest on bezpośrednią odpowiedzią na potrzebę zapewnienia funkcji umożliwiających komunikację między użytkownikami serwisu, tj.: newsletter, e-mail, SMS, forum, czat, news, komunikator, księga gości itd.

Wyspecjalizowany język dziedzinowy (KsiML) ma za zadanie usprawniać realizację zadań związanych z modelowaniem serwisów informacyjnych o projektach studenckich. Jego stosowanie pozwala zatem przyspieszyć budowę specjalistycznych modeli i ułatwia ich dalszą

[2] Do wytworzenia pozostałych DSL'i zastosowano również ten sam proces.

transformację, prowadzącą ostatecznie do otrzymania szkieletu kodu rozwiązania.

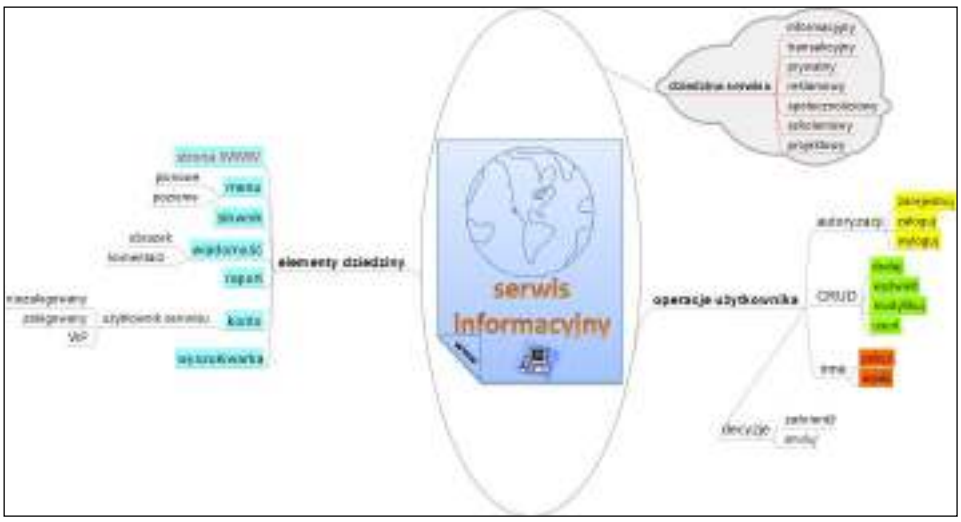
Środowisko wytwarzania języka KsiML stanowiło narzędzie CASE, Rational Software Architect v. 8.0 [5],[6],[7],[11],[12], które korzysta z rozszerzenia (Profile Tools) i automatyzuje działania na platformie Eclipse [2],[24].

2. Szkic dziedziny

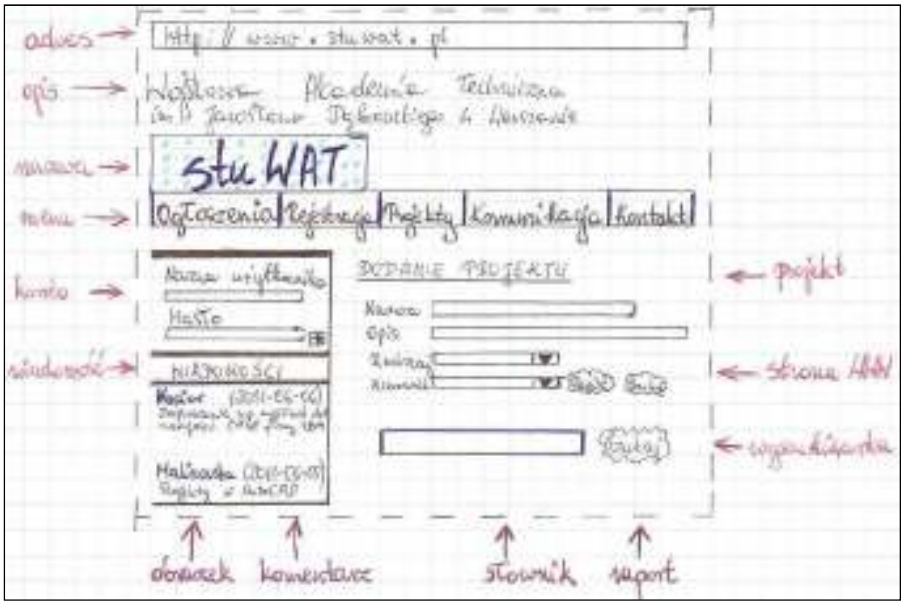
Zaprojektowanie nowego języka DSL wymaga przeprowadzenia szczegółowej analizy dziedziny [3],[10],[16] w wyniku której powinien powstać wstępny szkic modelu dziedziny [12].

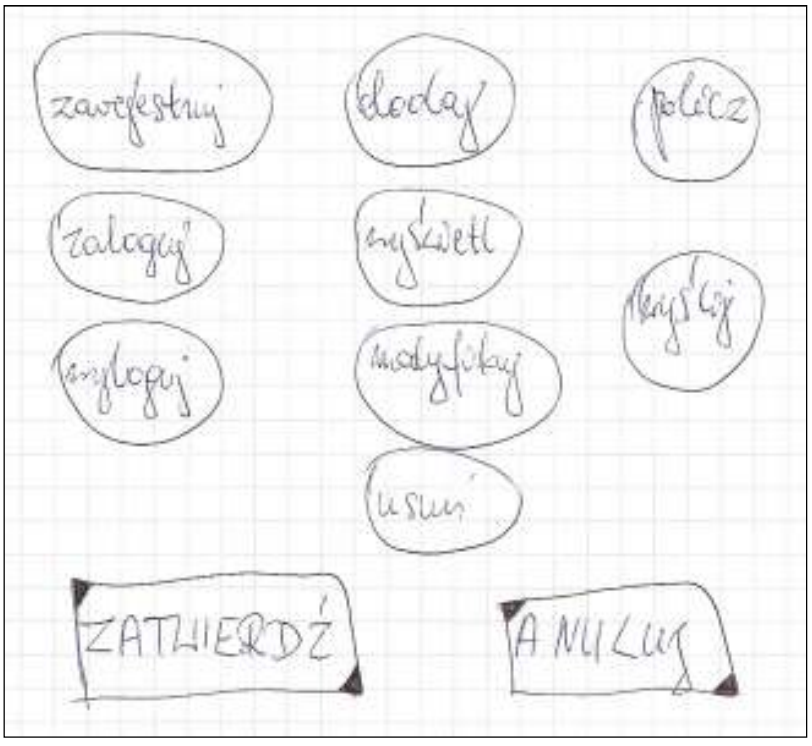
Szkicowanie pozwala na określenie „głównych ram" dziedziny, ponieważ ma na celu zobrazowanie wszystkich elementów języka, zawierających pewne wspólne cechy i właściwości, które muszą zostać włączone do składni nowego języka DSL. Zwykle szkic dziedziny można wykonać odręcznie (z pomocą kartki papieru i długopisu), ale również przy pomocy elektronicznych aplikacji (chociażby tych, które umożliwiają projektowanie tzw. „map myśli" (ang. *Mind Mapping*)).

Przykłady takich szkiców zostały zaprezentowane na rys. 2 i rys. 3.



Rys. 2. Szkic modelu dziedziny (elektroniczna mapa myśli)





Rys. 3. Szkic modelu dziedziny

3. Metamodel. Wprowadzenie

Na podstawie analizy utworzonych szkiców (np.: rys. 2 i rys. 3), stanowiących nieformalny opis dziedziny, budowany jest metamodel określany często mianem modelu koncepcyjnego. Opisuje on podmioty (byty) jako obiekty (koncepcje) - ich atrybuty, relacje oraz reguły ich łączenia i ograniczenia. Jego reprezentacją graficzną będzie diagram klas języka UML, który zobrazuje słownictwo i podstawowe pojęcia z dziedziny problemu. Metamodel jest wzorem, weryfikatorem oraz walidatorem budowanych modeli, które przedstawiają konkretny problem projektowy z danej dziedziny. Poprawnie przemyślana struktura dziedziny określa obraz pojęciowy badanego problemu i może być wykorzystana do weryfikacji i walidacji znajomości dziedziny.

Zgodnie z definicją OMG [20], budowa oprogramowania opiera się na wspólnym metamodelu opisującym sposób zapisu modelu konkretnego systemu.

Martin Fowler w [9] opisuje model domenowy jako sieć połączonych ze sobą obiektów, przy czym każdy obiekt reprezentuje jednostkę istniejącą w realnym obszarze biznesowym. Wyznacznikiem poprawnego metamodelu będzie diagram klas z przemyślaną strukturą pokazującą kontekst pojęciowy badanego problemu i będący jednocześnie zapewnieniem, że język jest semantycznie kompletny, jednoznaczny i skoncentrowany wyłącznie na dziedzinie.

3.1. Reguły semantyczne

Budowę metamodelu [20] opiera się na szkicu dziedziny [12], ale również na repozytorium wymagań [4], a w szczególności słowniku projektu będącym naturalnym źródłem informacji o dziedzinie problemu i pozwalającym na precyzyjniejsze modelowanie elementów języka dziedzinowego.

Projektowanie języków dziedzinowych związane jest najczęściej z konkretnym portfelem projektów, lub projektem informatycznym, który ma ściśle określony zakres funkcjonalny. Z uwagi na potrzebę reużycia elementów języka dziedzinowego w innych projektach należy pamiętać, że kompletny słownik języka, to taki który umożliwia rozwiązywanie wszystkich problemów projektowych w określonej dziedzinie. Wynika z tego, że należałoby określić możliwie największy, tzn. pełny zbiór terminów tworzących opis dziedziny, co jednak powodowałoby znaczny wzrost złożoności języka i trudności w jego poprawnym stosowaniu. W praktyce, więc poszukujemy minimalnego zbioru terminów, który pozwala opisać nam maksymalną (lub przynajmniej określoną w wymaganiach na modelowany system) liczbę problemów projektowych w dziedzinie projektu.

Język dziedzinowy ma dostarczać wielowymiarowy słownik do budowy rozwiązań w danej dziedzinie (np. serwisów informacyjnych) [16], dlatego podczas jego projektowania, należy pamiętać o kilku właściwościach [12]:

- zakres: przechwytywanie do modeli tylko istotnych informacji;
- kompletność: kompleksowy słownik do modelowania w danej dziedzinie, wspierany transformacjami każdego z elementów modelu;
- szczegółowość: znalezienie równowagi pomiędzy szczegółowością modeli i detalami transformacji;
- przyjazność narzędzi: możliwość wygenerowania dedykowanego oprzyrządowania do modelowania w danym języku dziedzinowym;
- łatwość obsługi: dostarczenie edytorów pozwalających na intuicyjne wykorzystywanie języka i łatwą nawigację po modelach.

Z analizy powyższych cech wynika, że opracowanie kompletnego języka dziedzinowego będzie można skutecznie przeprowadzić odwołując się do procesu iteracyjnego, w którym stopniowo rozbudowujemy jego zakres zapewniając wymaganą szczegółowość i kompletność, nie zapominając jednak o przyjazności i łatwości jego obsługi.

W pierwszym cyklu zaleca się określenie krytycznego słownictwa, a w kolejnych iteracjach jego rozwinięcie i doprecyzowanie (wykorzystując do tego celu opinie ekspertów dziedzinowych, deweloperów oraz wyniki testów modeli). Używając informacji zwrotnych od użytkowników języka należy ciągle udoskonalać DSL, sprawdzając w szczególności:

- zasady i założenia języka;
- użyteczność języka;
- kompletność języka;
- zakres języka.

Metamodel dziedziny „s*erwis informacyjny*" przedstawiono na rys. 4. Szczegółowy opis elementów słownikowych (terminów) w notacji EBNF[3] zawarto w [16], a ich uproszczone przykłady przedstawiono w tab. 1.

Tab. 1. Przykładowe terminy słownika języka KsiML

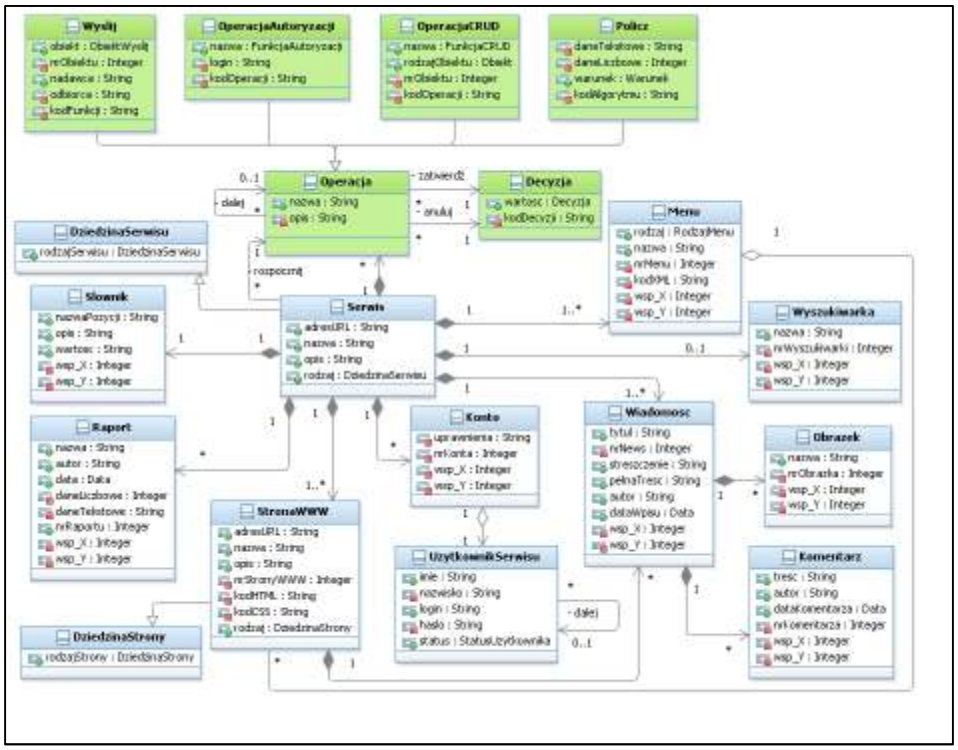
Nazwa		Opis	Rodzaj
D	decyzja	decyzja= wartość, kod decyzji; (*element odpowiedzialny za potwierdzenie operacji w serwisie [zatwierdź| anuluj]*)	II

[3] Rozszerzona notacja Backusa-Naura (ang. *Extended Backus-Naur Form*).

	dziedzina serwisu	dziedzina serwisu= rodzaj serwisu; (*element definiujący możliwe dyscypliny, rodzaje portali internetowych [informacyjna| transakcyjna| prywatna| reklamowa| społecznościowa| rozrywkowa| szkoleniowa| produktowa| korporacyjna| ogłoszeniowa| projektowa]*)	I
	dziedzina strony	dziedzina strony= rodzaj strony; (*element określający charakter strony WWW [kontaktowa| newsowa| projektowa| komunikacyjna]*)	I
K	komentarz	komentarz= treść, autor, data komentarza, nr komentarza, wsp_X, wsp_Y; (*element definiujący formę wypowiedzi użytkownika serwisu, najczęściej krótki tekst o charakterze opiniotwórczym*)	I
	konto	konto= uprawnienia, nr konta, wsp_X, wsp_Y; (*element reprezentujący zbiór zasobów, uprawnień w ramach serwisu, które przypisane są zarejestrowanemu użytkownikowi, każdy użytkownik ma swój login i hasło, które uwierzytelniają go podczas logowania do systemu*)	I
M	menu	menu= rodzaj, nazwa, nr menu, kod XML, wsp_X, wsp_Y; (*element odpowiedzialny za nawigację serwisu WWW, może występować jako [pionowe| poziome]*)	I
O	obrazek	obrazek = nazwa, nr obrazka, wsp_X, wsp_Y; (*element serwisu w postaci [pliku graficznego| ikony| obiektu multimedialnego*)	I
	operacja	operacja= nazwa, opis; (*element z którego dziedziczą elementy: wyślij, operacja CRUD, operacja autoryzacji, policz*)	II
	operacja autoryzacji	operacja autoryzacji= nazwa, login, kod operacji; (*element odsługujący w serwisie informacyjnym funkcje autoryzacji: zarejestruj, zaloguj, wyloguj*)	II
	operacja CRUD	operacja CRUD= nazwa, rodzaj obiektu, nr obiektu, kod operacji; (*element, który pozwala na obsługę w serwisie informacyjnym czterech funkcji CRUD: dodaj, wyświetl, usuń, modyfikuj*)	II

P	policz	policz= dane tekstowe, dane liczbowe, warunek, kod algorytmu; (*element odpowiedzialny za funkcje obliczeniowe, raportowe itd.*)	II
R	raport	raport= nazwa, autor, data, dane liczbowe, dane tekstowe, nr raportu, wsp_X, wsp_Y; (*element reprezentujący zdefiniowane zestawienie statystyczne, analityczne *)	I
S	serwis	serwis= adres URL, nazwa, opis, rodzaj; (*element specyfikujący usługę, która dostarcza treści w określonej dziedzinie tematycznej, np. informacyjnej, reklamowej, społecznościowej, projektowej itd.,*)	I
	słownik	słownik= nazwa pozycji, opis, wartość, wsp_X, wsp_Y; (*element definiujący agregaty danych słownikowych, tzn. danych z list rozwijanych, dostępnych w systemie*)	I
	strona WWW	strona WWW= adres URL, nazwa, opis, nr strony WWW, kod HTML, kod CSS, rodzaj; (*element reprezentujący dokument hipertekstowy udostępniany w sieci Internet, w zależności od dziedziny możemy wyróżnić strony kontaktowe, newsowe projektowe, komunikacyjne*)	I
U	użytkownik serwisu	użytkownik serwisu= imię, nazwisko, login, hasło, status; (*element odpowiedzialny za role określająca osobę korzystającą z serwisu informacyjnego: vip, niezalogowany, zalogowany*)	I
W	wiadomość	wiadomość= tytuł, nr news, streszczenie, pełna treść, autor, data wpisu, wsp_X, wsp_Y; (*komunikat umieszczony najczęściej na stronie serwisu, która prezentuje [aktualności| ogłoszenia| informacje| newsy]*)	I
	wyszukiwarka	wyszukiwarka= nazwa, nr wyszukiwarki, wsp_X, wsp_Y; (*element odpowiedzialny za mechanizm ułatwiający użytkownikom poszukiwanie informacji w serwisie*)	I
	wyślij	wyślij= obiekt, nr obiektu, nadawca, odbiorca, kod funkcji; (*element reprezentujący funkcjonalność wysyłania do określonych odbiorców [mail| newsletter| sms| raport| zadanie| wiadomość]*)	II

Terminy, oznaczone jako „I" w tab. 1, opisują dziedzinę serwisu informacyjnego w kontekście strukturalnym (na rys. 4) przedstawiono je w kolorze niebieskim), zaś te oznaczone jako „II" prezentują aspekt behawioralny (na rys. 4 przedstawiono je w kolorze zielonym) i zostały wyodrębnione po dogłębnej analizie wymagań funkcjonalnych oraz kilku iteracji prac nad metamodelem (co pozwoliło na określenie generycznych elementów specyfikujących późniejsze operacje).



Rys. 4. Metamodel dziedziny SERWIS INFORMACYJNY

3.2. Metamodel – reguły syntaktyczne

Wyodrębnione klasy określiły słownictwo dziedziny, zaś atrybuty, relacje i ograniczenia (stanowiące reguły budowy i łączenia elementów dziedziny), doprowadziły do jego uszczegółowienia. Zaprojektowane reguły tworzą wspólnie tzw. model ograniczeń, który można uszczegółowić za pomocą modeli języka UML z ograniczeniami doprecyzowanymi w języku OCL.

OCL (ang. *Object Constraint Language*) − deklaratywny język formalnych specyfikacji ograniczeń w modelach obiektowych [14],[21], który odnosi się

bezpośrednio do UML [1],[4] – pozwala na jego uściślenie (tzn. uszczegółowienie, doprecyzowanie). OCL wykorzystywany jest najczęściej do wspierania definicji dynamiki systemu, co podnosi tym samym precyzję modeli w języku UML. OCL posiada zestaw wbudowanych operatorów, predykatów, ma możliwość definiowania własnych funkcji, warunków i niezmienników. Może być używany przy niemal wszystkich elementach występujących w UML. Język OCL można scharakteryzować następującymi własnościami:

- wyraża dowolną regułę logiczną: warunki wstępne, końcowe, niezmienniki, wyniki metod itd.;
- nie może modyfikować modelu, jedynie go sprawdzać;
- można go związać z dowolnym elementem modelu (klasą, operacją, atrybutem, asocjacją itd.).

Język OCL jest wykorzystywany zarówno w procesach metamodelowania jak i modelowania. Przykładem wykorzystania OCL w procesie tworzenia metamodelu dziedziny „serwis informacyjny", przedstawionego na (rys. 4), mogą być ograniczenia dotyczące poprawności serwisu:

1) serwis tworzą tylko te strony www dla których określono ich adres URL;

```
context Serwis;
inv: self.stronaWWW->exists(adresURL<>"")
```

2) wszystkie strony www serwisu mają określony adres URL;

```
context Serwis;
inv: self.stronaWWW->forAll(adresURL<>"")
```

3) wymagamy aby serwis tworzyły jedynie te strony www, które posiadają różny adres URL;

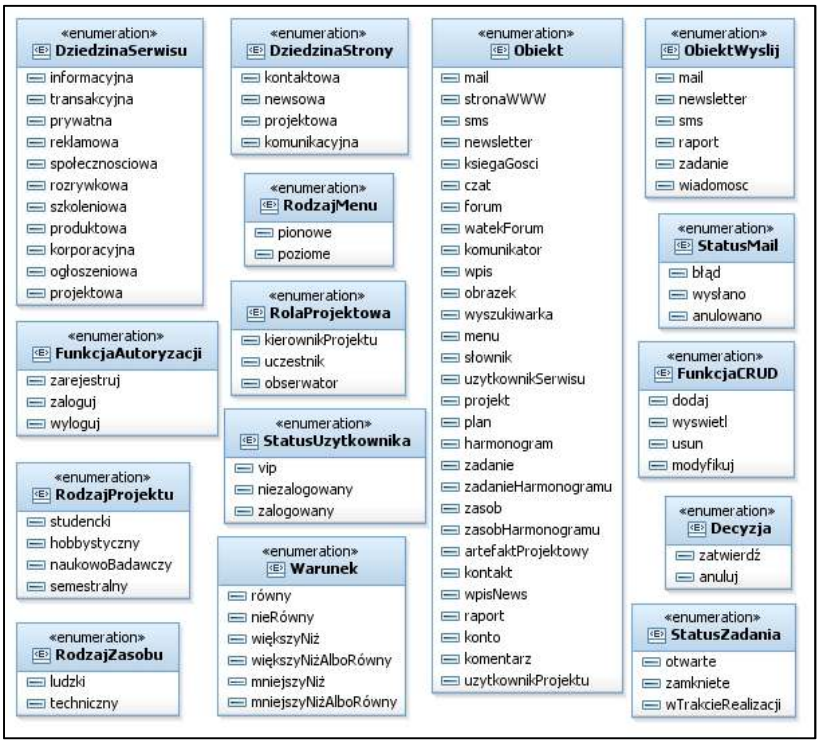
```
context Serwis;
inv: self.stronaWWW->forAll(www1, www2 | www1<>www2
implies www1.adresURL<> www2.adresURL)
```

Zbudowany metamodel, oprócz szablonowych typów (tzw. „*UMLPrimitiveTypes*") [11], może zostać wzbogacony autorskimi wyliczeniami (tab. 2).

Tab. 2 Metamodel języka KsiML. Przykłady definicji wyliczeń modelu ograniczeń

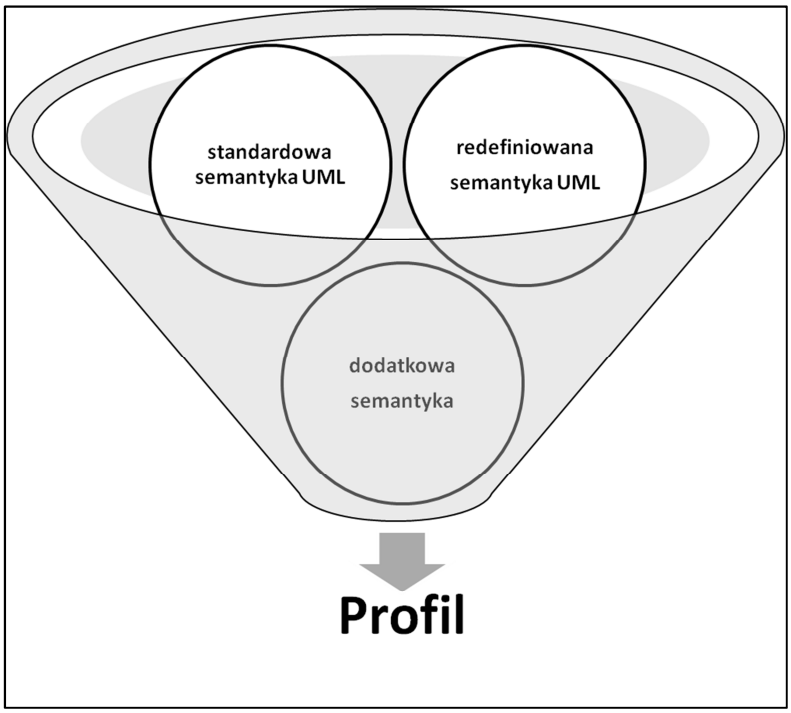
Nazwa		Opis
D	decyzja	decyzja= [zatwierdź| anuluj]; (*wyliczenie zaprojektowane dla potrzeb potwierdzeń wykonywanych operacji, które umożliwia wybór jednej z opcji: zatwierdź, lub anuluj*)
	dziedzina serwisu	dziedzina serwisu = [informacyjna| transakcyjna| prywatna| reklamowa| społecznościowa| rozrywkowa| szkoleniowa| produktowa| korporacyjna| ogłoszeniowa| projektowa]; (*wyliczenie reprezentujące możliwe dziedziny serwisów informacyjnych*)
	dziedzina strony	dziedzina strony= [kontaktowa| newsowa| projektowa| komunikacyjna]; (*wyliczenie reprezentujące rodzaje stron WWW serwisu informacyjnego*)
F	funkcja autoryzacji	funkcja autoryzacji= [zarejestruj| zaloguj| wyloguj]; (*wyliczenie dedykowane dla stereotypów, których logika działania wykorzystuje funkcje autoryzacji w serwisie informacyjnym*)
	funkcja CRUD	funkcja CRUD= [dodaj| wyświetl| usuń| modyfikuj]; (*wyliczenie opracowane dla potrzeb „klas-operacji", które odpowiadają za realizację czterech funkcji CRUD*)
O	obiekt	obiekt= [mail| strona WWW| SMS| newsletter| księga gości| czat| forum| wątek forum| komunikator| wpis| obrazek| wyszukiwarka| menu| słownik| użytkownik serwisu| projekt| plan| harmonogram| zadanie| zadanie harmonogramu| zasób| zasób harmonogramu| artefakt projektowy| kontakt| wpis news| raport| konto| komentarz| użytkownik projektu]; (*wyliczenie, które reprezentuje możliwe rodzaje elementów występujących w serwisie informacyjnym i tym samym definiujących dziedzinę problemu*)
	obiekt wyślij	obiekt wyślij= [mail| newsletter| SMS| raport| zadanie| wiadomość]; (*wyliczenie dedykowane dla stereotypu „Wyślij", który reprezentuje możliwe rodzaje elementów, które można przesłać na skrzynkę e-mailową za pomocą funkcjonalności dostępnych w serwisie informacyjnym*)
R	rodzaj menu	rodzaj menu= [pionowe| poziome]; (*wyliczenie dostarczające dwa rodzaje elementów nawigacyjnych serwisu informacyjnego*)
	rodzaj projektu	rodzaj projektu= [studencki| hobbystyczny| naukowo- badawczy| semestralny]; (*wyliczenie pozwalające określić charakter projektu, realizowanego w serwisie*)
	rodzaj zasobu	rodzaj zasobu= [ludzki| techniczny]; (*wyliczenie pozwalające na określanie zasobu projektowego*);

	rola projektowa	rola projektowa= [kierownik projektu| uczestnik| obserwator] (*wyliczenie odpowiedzialne za różnicowanie uprawnień w projekcie według trzech możliwych ról projektowych*)
S	status mail	status mail= [błąd| wysłano| anulowano]; (*wyliczenie wykorzystywane w obsłudze mechanizmu e-mail, które pozwala na określenie aktualnego statusu wiadomości*)
	status użytkownika	status użytkownika= [vip| niezalogowany| zalogowany]; (*wyliczenie pozwalające administratorowi serwisu na różnicowanie uprawnień *)
	status zadania	status zadania= [otwarte| zamknięte| w trakcie realizacji]; (*wyliczenie zaprojektowany dla potrzeb mechanizmu obsługującego projekty, umożliwia określanie poziomu zaawansowania prac zadania*)
W	warunek	warunek= [równy| nie równy| większy niż| większy niż albo równy| mniejszy niż| mniejszy niż albo równy]; (*wyliczenie powstało dla potrzeb stereotypu „Policz", którego logika działania, skupia się na obsłudze operacji matematycznych, w zależności od zdefiniowanego warunku*)



Rys. 5. Wyliczenia modelu ograniczeń metamodelu języka KsiML

Na rys. 6 dokonano przeglądu wszystkich wyliczeń modelu dziedzinowego, powstałych w ramach dziedziny „serwis informacyjny". Wyliczenia jako elementy typu *enumeration* zostały zdefiniowane w celu dokładniejszego odzwierciedlenia dziedziny modelowanego problemu i prawidłowego jej określenia w ramach wyodrębnionych atrybutów.



Rys. 6. Profil jako mechanizm rozszerzania semantyki

Możliwość korzystania z autorskich wyliczeń znacznie ułatwia budowanie metamodeli i pozwala na ukrywanie ich złożoności, dzięki temu na jednym diagramie możliwe jest prezentowanie nawet bardzo złożonych (pojęciowo) dziedzin. Reprezentatywnym przykładem w tym zakresie jest klasa *„OperacjaCRUD"*, która dzięki argumentowi *„nazwa"* i wyliczeniu *„FunkcjaCRUD"* do minimum pozwala redukować złożoność modelu. Cztery klasy: *dodaj*, *wyświetl*, *usuń*, *modyfikuj* zostały zastąpione jednym, generycznym elementem: *„OperacjaCRUD"* umożliwiającym projektowanie wszystkich funkcji CRUD.

Tworząc definicję semantyki i syntaktyki języka dziedzinowego należy pamiętać, że budowany DSL musi tworzyć spójną „abstrakcyjną ramę" [9] pozwalającą na zamknięcie decyzji projektowych przy przejściu na wyższy poziom modeli (z metamodelu do modelu).

4. Budowa narzędzi językowych

4.1. Definiowanie i budowa profili

Metamodel jest formalną definicją języka dziedzinowego [10],[12],[19]. Określa jego słownik oraz gramatykę, czyli pokazuje kontekst pojęciowy badanego problemu, przy jednoczesnym zapewnieniu, że DSL jest semantycznie kompletny, jednoznaczny i skoncentrowany wyłącznie na specyficznej (konkretnej) dziedzinie.

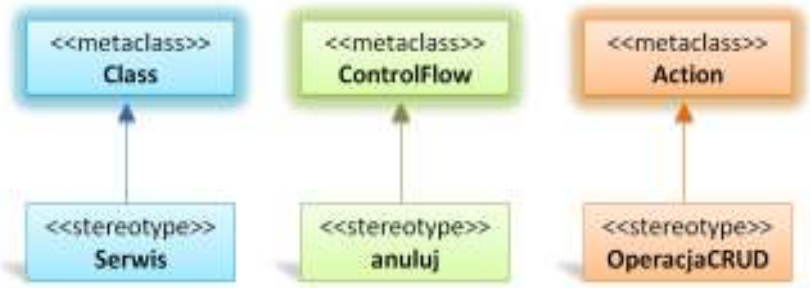
Wykorzystywanie nowych języków dziedzinowych w środowiskach CASE wymaga wcześniejszego wdrożenia odpowiednich mechanizmów za pomocą możliwości profilu UML.

Warto podkreślić, że język UML posiada trzy rodzaje mechanizmów rozszerzeń (profile, stereotypy, metki) [1],[4], które pozwalają na zwiększenie zakresu zastosowania tego języka poprzez redefiniowanie pojęć i doprecyzowanie ich znaczenia do nowych obszarów zastosowań.

Budowane rozszerzenia muszą być zgodne ze standardem języka UML, nie mogą naruszać jego standardowej semantyki, a jedynie ja uściślać dla specyficznych dziedzin.

- **Profil** – to zbiór stereotypów i metek definiujących znaczenie bytów projektowych związanych z dziedziną problemu, stanowi zatem rozszerzenie dla języka UML [1],[4],[22].
- **Stereotyp** – służy do zmieniania lub doprecyzowania semantyki elementów modelu. Stereotypy wykorzystuje się do klasyfikowania, oznaczania istniejących modeli oraz wprowadzania nowych elementów [1]. Do bazowych elementów notacji UML (m.in.: klas, atrybutów, asocjacji, itd.) dodają nową semantykę. Stereotypy zapisywane są w charakterystyczny sposób, tzn. ich nazwa umieszczana jest w specjalnych nawiasach syntaktycznych (znakach cudzysłowu ostrokątnego). Każdy stereotyp posiada własną ikonę (tzw. piktogram), która jednoznacznie go identyfikuje. Definiowanie stereotypów jest realizowane na diagramach klas. Klasy ze stereotypem <<stereotype>> określają nowe pojęcia, rozszerzając zbiór pojęć podstawowych języka UML, których formalną specyfikację opisano w MOF. Aby skorzystać z tych formalnych specyfikacji w definiowaniu nowych pojęć wystarczy je połączyć związkiem <<Extension>> z metaklasą reprezentującą pojęcie języka UML (rys. 7).
- **Metka** - opisuje dodatkowe właściwości elementów dziedziny. Definiowana jest wewnątrz profilu (tzn. znajdujących się w nim

stereotypów) [1]. Pojawia się w postaci przyporządkowanych par *nazwa - wartość*, zapisywana jest w nawiasach klamrowych, dołączana jest do opisywanych elementów w postaci notek. W większości narzędzi CASE są one jednak zapisywane w postaci metadanych, zawartych wewnątrz elementu, ponieważ pozwala to na łatwiejsze ich przetwarzanie. Przykład.: informacja o autorach modelu {autorzy = "Agata Kosior i Andrzej Stasiak"}.



Rys. 7. Związki między metaklasami a stereotypami

Tab. 3 prezentuje właściwości trzech przykładowych stereotypów: „anuluj", „Serwis", „OperacjaCRUD".

Tab. 3. Właściwości przykładowych stereotypów

Stereotype	Extensions Metaclass	Attributes: Type	Icon Shape Image
anuluj	ControlFlow	`base_ControlFlow: <<metaclass>>ControlFlow`	

Serwis	Class	`adresURL: String`
		`base_ Class: <<metaclass>> Class`
		`nazwa: String`
		`opis: String`
		`rodzaj: DziedzinaSerwisu`
			

OperacjaCRUD	Action	`base_ Action: <<metaclass>> Action`
		`kodOperacji: String`
		`nazwa: FunkcjaCRUD`
		`nrObiektu: Integer`
		`rodzajObiektu: Obiekt`
			

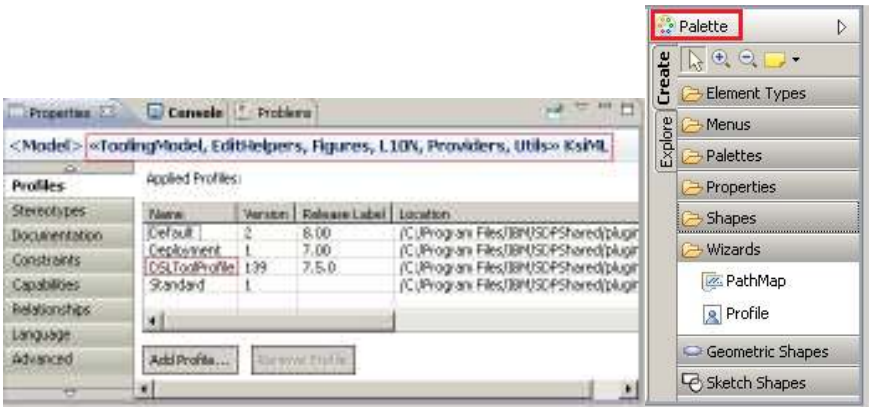
Profil jest więc reprezentacją języka dziedzinowego w narzędziu CASE. Jest on określony za pomocą zbioru stereotypów, definiujących znaczenie bytów projektowych związanych z dziedziną problemu.

Autorskie profile powstają na podstawie metamodeli, a ich semantykę definiują elementy typu: „stereotype" (wyodrębnione na podstawie klas

metamodeli) oraz elementy „enumeration” (dodatkowe wyliczenia charakterystyczne dla danego języka dziedzinowego).

Proces budowy nowego profilu UML należy rozpocząć od analizy metamodelu dziedziny, ponieważ naturalnymi kandydatami na stereotypy są klasy w nim określone.

Budowa profili UML w środowisku CASE − IBM RSA odbywa się w dedykowanym projekcie typu „*Profile Project*” (rys. 8), który jest niczym innym jak specjalnym profilem narzędzia RSA („*DSLToolProfile*”) [5],[6],[12] dostarczającym wymagane „oprzyrządowanie”– narzędzia dla budowy nowych języków dziedzinowych.



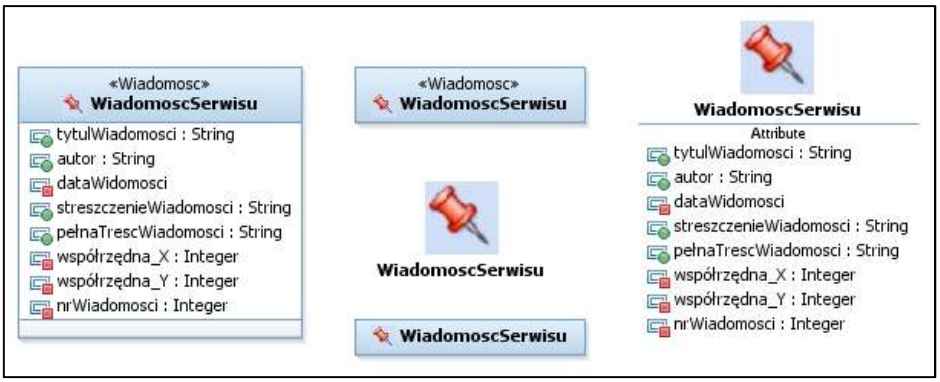
Rys. 8. Projekt z profilem „DSL ToolProfile” i dedykowaną paletą

4.2. Stereotypy

Profil UML jest definiowany za pomocą stereotypów (ang. *stereotype*), które określają znaczenie bytów projektowych dziedziny problemu. Nowe elementy definiujemy poprzez określenie ich cech, wyglądu i zachowania (rys. 9). W narzędziach CASE stereotypy języka dziedzinowego określa się za pomocą:

- właściwości (*properties*): nazwa, ikona, kształt itd.;
- ograniczeń (*constraints*).

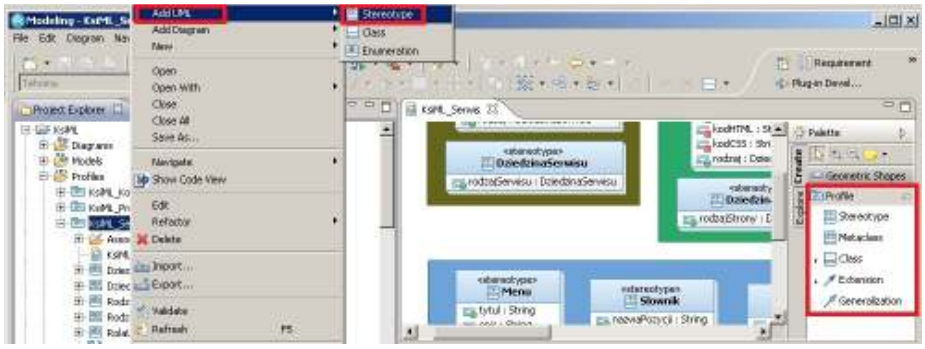
Atrybuty stereotypów definiuje się za pomocą standardowych typów danych: String, Boolean, Integer oraz tych zdefiniowanych w autorskich wyliczeniach (*enumeration*).



Rys. 9. Przykłady sposobów prezentacji elementów typu „stereotype” (postać kanoniczna i piktograficzna)

Dodanie stereotypu do profilu w środowisku RSA można wykonać na dwa sposoby (rys. 10):

1. Bezpośrednio w *„Project Explorer”* wykorzystując funkcję *„Add UML − Stereotype”*.
2. Na diagramie typu *„Class diagram”* wykorzystując dostępne elementy z palety *„Profile”*.



Rys. 10. Dodawanie elementu typu „Stereotype” do projektu profilu

Rys. 11. Przykład elementu typu „enumeration”

4.3. Wyliczenia

Profile uszczegóławia się poprzez dodanie wyliczeń (ang. *enumeration*), które stanowią nowe typy danych dla definiowanego języka dziedzinowego. Określenie wartości wyliczenia polega na dodaniu tzw. literałów wyliczeniowych (ang. *enumeration literals*). Proces wyszukania wyliczeń w większości przypadków kończy się na pełnym odwzorowaniu tych elementów z metamodelu do profilu.

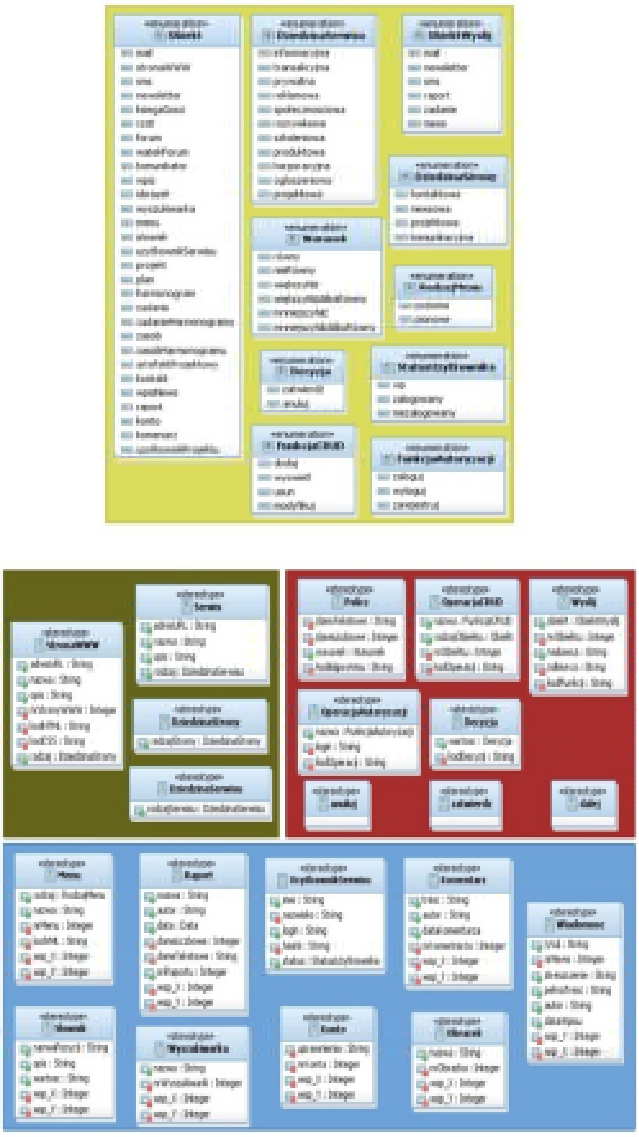
4.4. Proces budowy profilu. Podsumowanie

Podsumowując treści podpunktów 4.2 i 4.3, w proponowanej metodzie KMS, proces budowy profilu przebiega w trzech krokach:

1. Utworzenie projektu na podstawie właściwego wzorca (*„Profile Project”*).
2. Wstawienie zdefiniowanych w metamodelu pojęć języka dziedzinowego jako stereotypowanych klas − przypisując im wymagane cechy (tab. 4).
3. Wstawienie ograniczeń do profilu na podstawie:
    a. Relacji w metamodelu (rys. 4).
    b. Zdefiniowanych wyliczeń w metamodelu (rys. 4).

Tab. 4. Cechy elementu „stereotype” (w środowisku RSA)

Lp	Nazwa		Opis zawartości
1.	General	ogólne	Nazwa stereotypu, jego widoczność (public, private, protected, packed), zdefiniowana ikona i kształt obrazu (przypisany plik graficzny).
2.	Extensions	rozszerzenia	Element UML może zostać rozszerzony o wybór jednej lub kilku metaklas z dostępnej listy.
3.	Attributes	atrybuty	Lista z atrybutami określonymi poprzez: nazwę, typ, wartość.
4.	Stereotypes	stereotypy	Możliwość przypisania do obiektu stereotypu.
5.	Documentation	dokumentacja	Cecha stanowiąca opis – komentarz danego elementu.
6.	Relationships	relacje	Informacje o relacjach wybranego elementu z innymi (dostępnymi w projekcie).
7.	Appearance	wygląd	Edytor pozwalający na dostosowanie wyglądu (czcionki, koloru wypełnienia, obramowania) oraz ukrycia/ odkrycia szczegółów, jakie mają być prezentowane na diagramie.
8.	Advanced	zaawansowane	W tej kategorii znajdują się informacje zgrupowane w trzy obszary: *Presentation*, *UML*, *View*. Podsumowują one wszystkie właściwości modelowanego elementu.



Rys. 12. Profil „KsiML Serwis”

Ta łatwość obsługi i użytkowania profili przekłada się na szersze zastosowanie nowego DSL (rys. 12).

Przyjazny dla użytkownika profil UML zwiększa prawdopodobieństwo, że będzie on wykorzystywany przez nowych odbiorców. Ważnym aspektem jest możliwość przystosowania środowiska CASE do dziedziny modelowania.

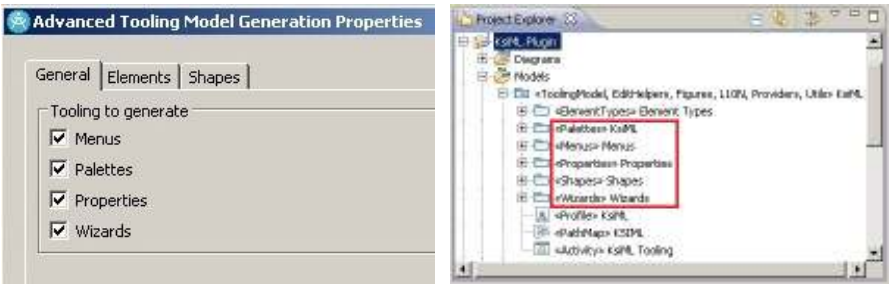
Opracowanie dedykowanych elementów profilu w dedykowanej palecie sprawia, że DSL jest łatwiejszy w użyciu. Ponadto należy zapewnić mechanizmy automatycznej walidacji (ograniczenia pozwalają uniknąć błędów podczas modelowania).

Po zbudowaniu profilu, w narzędziach CASE, takich jak *Rational Software Architect*, można rozwijać stereotypy ich właściwości, budować edytory diagramów oraz opracowywać modele zgodne ze zdefiniowanym profilem.

5. Budowa narzędzi językowych. Dystrybucja i zarządzanie profilami

Zaprojektowanie elementów profilu UML nie kończy jeszcze prac związanych z rozpowszechnieniem notacji nowego języka dziedzinowego. Wykorzystywanie DSL w narzędziach CASE [5],[6] wymaga wcześniejszego dostosowania tych środowisk do modelowania w nowym języku, co wiąże się z opracowaniem dedykowanych wzorców (*templates*): palet, edytorów, menu itd. Środowisko RSA ułatwia wytworzenie specjalistycznych narzędzi profili, ponieważ dostarcza dedykowany do tych zadań projekt typu *„UML Profile Tooling Plug-in Project”* (rys. 8) dający możliwość określenia wzorców (rys. 13):

- Menus;
- Palettes;
- Properties;
- Wizards.



Rys. 13. Widok projektu z dedykowanymi „templates” (profil „DSL ToolProfile”)

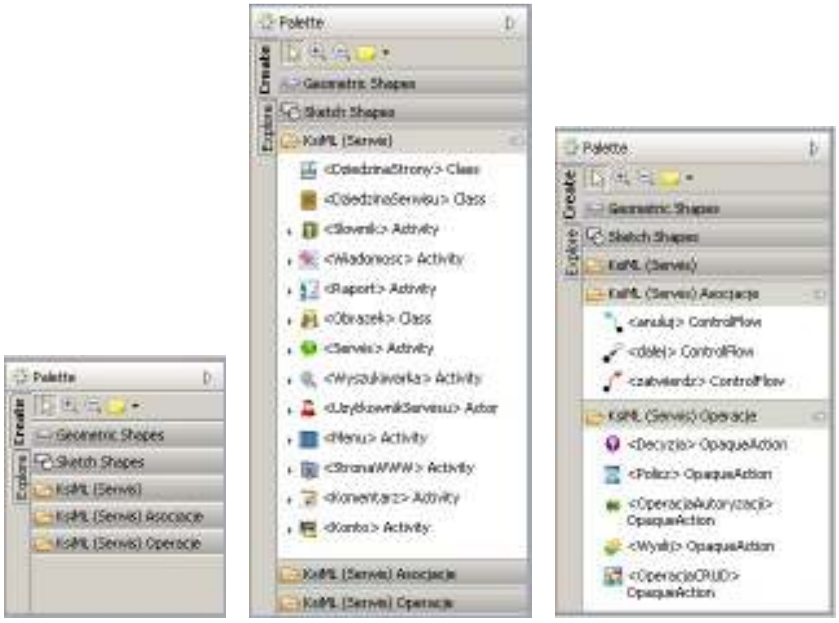
W tab. 5 zostały przedstawione kluczowe informacje o elementach profilu *„DSL ToolProfile”*.

5.1. Paleta

Dostarczenie wraz z notacją DSL dedykowanej palety zwiększa przyjazność języka i powoduje, że jest on łatwiejszy w użyciu. Dużym atutem języków dziedzinowych wydaje się być ich „obrazkowa notacja" (rys. 14), która jednoznacznie identyfikuje elementy dziedziny w postaci piktogramów (jako ich metafor) przez co zwiększa ich czytelność.

Tab. 5. Właściwości elementów profilu „DSL ToolProfile"

Element palety	Właściwości elementu- przykład zastosowania
Palette	Property / Value Palette children: Entries: 5 description: KsiML diagramKind: Entries: 2 editorId: Entries: 1 factoryClassName: KsiMLPaletteFactory id: KsiML largeIcon: /icons/default.gif providerClassName: KsiMLPaletteProvider smallIcon: /icons/default.gif
PaletteDrawer	Property / Value PaletteDrawer children: Entries: 5 description: id: initialState: 0 - INITIAL_STATE_CLOSED largeIcon: smallIcon:
PaletteStack	Property / Value PaletteStack activeTool: null children: Entries: 2 description: Projekt id: Projekt largeIcon: smallIcon:
PaletteCreation ToolEntry	Property / Value PaletteCreationToolEntry description: Creates a OpaqueAction with stereotype Policz elementType: Entries: 1 id: KsiML._Policz__OpaqueAction largeIcon: smallIcon:



Rys. 14. Palety dla profilu „KsiML_Serwis"

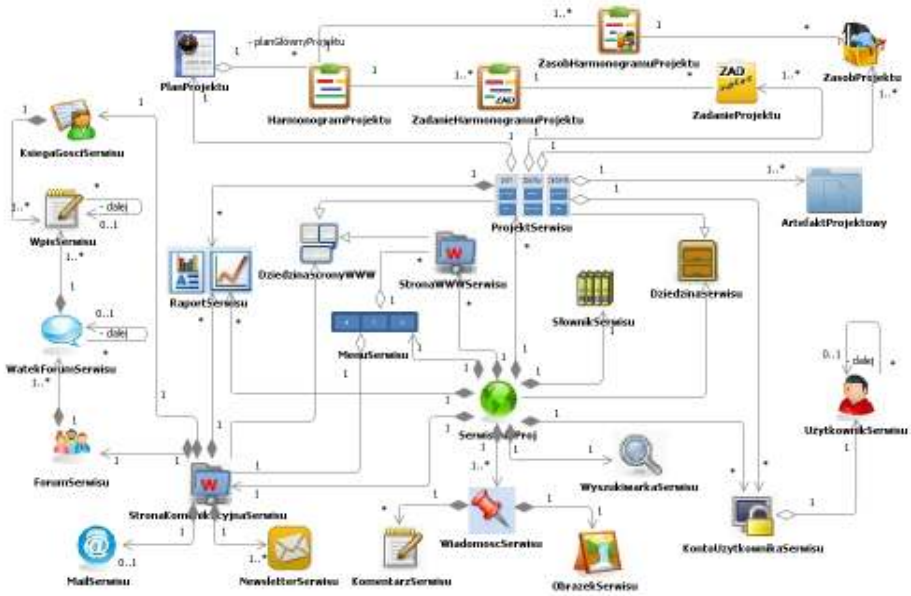
5.2. Wzorce modeli i diagramów

Projektant języka dziedzinowego musi wybrać typy diagramów, które są najlepiej dostosowane do jego specyfiki.

UML [1] oferuje szereg strukturalnych i behawioralnych schematów, które mogą być użyte jako podstawa do budowy modeli za pomocą języka dziedzinowego. Wybór typu diagramów UML jest uzależniony od tego co nowy język ma obsługiwać.

W ramach języka do modelowania serwisów informacyjnych zostały udostępnione dwa rodzaje diagramów: *„ClassDiagram"* oraz *„ActivityDiagram"* (możliwość stereotypowania elementów modeli, jako *„Action"* oraz *„Control Flow").*

Słownictwo dostarczone w profilach powinno umożliwiać budowanie dowolnych serwisów informacyjnych. Przeprowadzenie kompletnej weryfikacji języka dziedzinowego wiąże się z przetestowaniem poprawności i kompletności składni DSL. Należy zatem zbudować model dla konkretnego problemu projektowego i sprawdzić możliwość zapisania w nim wyrażeń odnoszących się do struktury dziedziny (rys. 15) i opisu zachowania elementów tej struktury.



Rys. 15. Przykładowy model serwisu informacyjnego o projektach studenckich w języku KsiML

Budowa modelu to faza swobodnego posługiwania się składnią DSL i dowolnego łączenia elementów wraz z nim dostarczanych. Dopiero na etapie walidacji model podlega sprawdzeniu i ocenie, a jej wynikiem jest informacja o poprawnym albo błędnym zastosowaniu reguł językowych określonych w metamodelu.

Przeprowadzone testy (opisane w [16]) języka dziedzinowego do budowy serwisów informacyjnych pozwalają twierdzić, że jest on językiem generycznym - możliwym do użycia w szerokim obszarze zastosowań. Jego reguły pozwalają na przetwarzanie danych różnych typów oraz budowę rozwiązań poziomu PSM (w modelu MDA) w różnych technologiach i na wielu platformach.

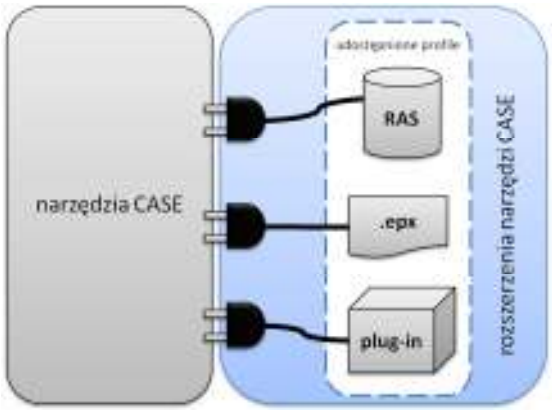
5.3. Udostępnianie profili

Proces budowy profilu UML związany jest najczęściej z konkretnym projektem informatycznym. Jednak dzięki możliwościom wytworzenia uniwersalnych narzędzi, opisanym w podpunktach 5.1 oraz 5.2, profil może być dystrybuowany na nowe stanowiska projektowe i stosowany wielokrotnie w projektach (których zakres ograniczony jest do dziedzin, dla których zdefiniowano języki dziedzinowe (DSL) i odpowiadające im profile).

Dystrybucja języków dziedzinowych w narzędziach CASE jest możliwa dzięki trzem mechanizmom:

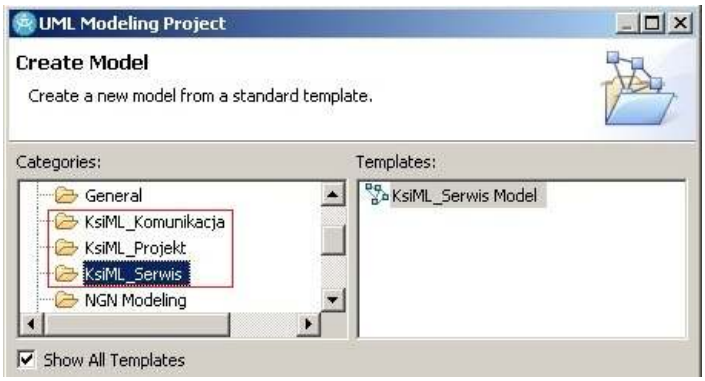
- plik RAS (ang. *Reusable Asset Specification*);
- plug-in;
- plik .epx.

Wymienione mechanizmy pozwalają na „utrwalenie" profili i rozszerzenie standardowych możliwości narzędzi CASE (rys. 16), poprzez dostosowanie ich do nowych zadań (determinowanych dziedziną problemu).



Rys. 16. Udostępnione profile jako rozszerzenia narzędzi CASE

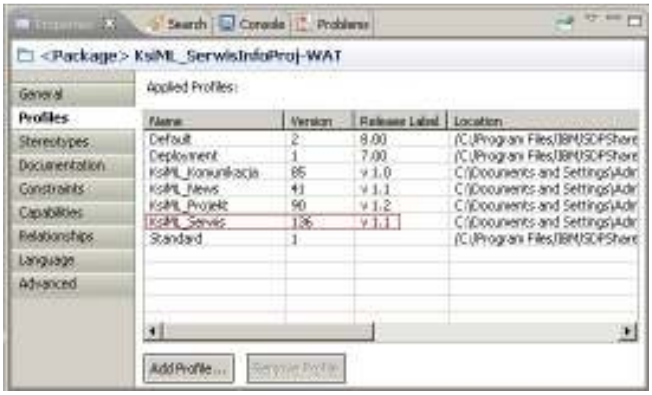
„Utrwalone" profile mogą być przenoszone między różnymi projektami. Możliwość wykorzystania nowego języka dziedzinowego w środowisku CASE (np. w RSA) wymaga zbudowania modelu (rys. 17) i zastosowania w nim profilu UML (rys. 18) (jego podłączenia).



Rys. 17. Widok nowych kategorii profili wraz z odpowiadającymi im wzorcami modeli

Rys. 18. Podłączenie profilu w środowisku RSA (wybór jednaj z trzech możliwości)

Kontrola i zarządzanie zmianami w profilu zbudowanego języka dziedzinowego jest możliwa dzięki opcji wersjonowania. W RSA każdy profil jest związany z określonym numerem wersji, co przedstawia (rys. 19).



Rys. 19. Widok podłączonych profili wraz z numerem wersji i wydania

5.4. Komponowanie i dekomponowanie profili

Profile UML można zarówno komponować jak i dekomponować. Środowisko CASE, firmy IBM (*Rational Software Architect*), dostarcza dedykowany profil (*„DSL ToolProfile”*) umożliwiający kastomizację budowanych języków dziedzinowych (DSL'i) za pomocą palety z dedykowanymi elementami, co pokazuje rys. 8.

Umiejętne łączenie profili pozwala na wytwarzanie precyzyjnych języków dziedzinowych do modelowania złożonych dziedzin problemowych (odwołujących się do wielu dialektów języków dziedzinowych). Takie podejście, czyli podział dziedziny problemu na opisujące ją mniejsze specyficzne profile, jest bardzo przyjazne i otwarte na nowo powstające dziedziny np. portali webowych.

Zaletą tej metody jest redukcja czasu potrzebnego na zbudowanie nowego języka dziedzinowego, którą osiągniemy dzięki reużyciu już istniejących profili, zarówno UML jak i wcześniej zbudowanych języków dziedzinowych.

Konsolidacja kilku profili w jeden wymaga prac, których celem jest dostosowanie wyglądu palety, tzn. dokonanie podziału na pakiety funkcjonalne, wykorzystując przy tym właściwości elementów pakietu o nazwie *„Palettes”*- dostarczonego w profilu *„DSL ToolProfile”* pokazanych na rys. 8.

5.5. Język KsiML

W artykule posłużono się przykładami artefaktów wytworzonych podczas budowy języka dziedzinowego KsiML, którego pełną specyfikację przedstawiono w [16]. Język KsiML został formalnie wyspecyfikowany za pomocą trzech profili UML: „KsiML Serwis”, „KsiML Projekt”, „KsiML Komunikacja” (rys. 20) dla których opracowano palety (rys. 21).

Rys. 20. Koncepcja języka dziedzinowego KsiML

Rys. 21. Palety języka KsiML

6. Podsumowanie

Języki dziedzinowe buduje się, aby ułatwić opisywanie zagadnień w pewnej, dobrze określonej dziedzinie. Opracowanie kompletnego DSL wymaga przeprowadzenia szeregu iteracyjnych prac, które zostały opisane w tym artykule. Podczas udziału w przedsięwzięciach informatycznych można się spotkać nie tylko z potrzebą budowy nowych języków dziedzinowych, ale również z koniecznością ich rozszerzania lub udoskonalania.

1. **Budowa nowego DSL.** Zaletą tego podejścia jest maksymalna elastyczność i swoboda w projektowaniu. Możliwość wyboru składni, semantyki języka oraz jego wizualizacji. Jednocześnie, takie podejście wymaga największego doświadczenia i umiejętności projektowania nowych języków dziedzinowych, ponieważ wiąże się ono z koniecznością wypracowania bazowej struktury DSL i przeprowadzenia licznych testów badających jego kompletność, użyteczność oraz przyjazność.
2. **Rozszerzanie/udoskonalanie istniejącego DSL.** Podejście to polega na opracowaniu dodatkowych elementów prowadzących do rozszerzenia semantyki istniejącego języka, albo poprawie już istniejących elementów języka. Prace te nie wymagają aż tak dużego doświadczenia jak podczas budowy nowego języka. Ważna jest tu jednak doskonała znajomość notacji modyfikowanego DSL (tzn. słownika i gramatyki).

W artykule zaprezentowano kompletny proces definiowania języka dziedzinowego zgodnie z metodą KMS. Proces ten został w pełni zweryfikowany, a jego szczegółowe wyniki zamieszczono w [16].

Proces budowy serwisów informacyjnych wyznaczony przez metodę KMS kończy utworzenie transformacji, które pozwolą na automatyczne przekształcenie opracowanych modeli, zapisanych w języku dziedzinowym, w rozwiązanie. Rozwiązanie to może mieć postać modelu lub kodu co zaprezentujemy w artykule [15].

Literatura

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

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

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

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

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

[6] DUSKO M., *Authoring UML Profiles: Part 1. Using Rational Software Architect, Rational Systems Developer, and Rational Software Modeler to create and deploy UML Profiles*, *2008*.

[7] DUSKO M., *Authoring UML Profiles: Part 2. Using Rational Software Architect, Rational Systems Developer, and Rational Software Modeler to create and deploy UML Profiles*, 2008.

[8] ETHAN K., SZTIPANOVITS J.J., *Formalizing the Structural Semantics of Domain-specific Modeling Languages,* Journal of Software and System Modeling, 2009.

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

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

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

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

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

[14] KLEPPE A., WARMER J., *OCL. Precyzyjne modelowanie w UML,* WNT, 2003.

[15] KOSIOR A., STASIAK A., *Wytwarzanie serwisów informacyjnych w oparciu o koncepcję modelowania dziedzin. Budowa transformacji.* Biuletyn IAiR, NR 32, 2012.

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

[17] *OMG. Catalog of UML Profile Specifications*, OMG 2011.

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

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

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

[21] *OMG. Object Constraint Language, Version 2.2,* OMG 2010.

[22] *OMG. UML Superstructure Specification v2.4,* OMG 2011 (chapter 18: Profiles).

[23] SCHMIDT D.C., *Model-Driven Engineering.* IEEE Computer 39(2), (2006).

[24] WHITE J., SCHMIDT D.C., NECHYPURENKO A., WUCHNER E.: *Introduction to the Generic Eclipse Modeling System.* Eclipse Magazine 7 (2007).

Production of information services using the concept of domain modeling . A systematic approach for building domain specific language (KsiML)

ABSTRACT: The paper describes a process of building the domain language KsiML using the MDE approach, based on modeling domains (DSM) and the authors' KMS method . The process was used to build an information service of students' projects. In assumptions, the application of the proposed process should lead to height of quality and re-use of the source code, as well as decrease of construction systems costs.

KEYWORDS: domain specific language (DSL), domain specific modeling (DSM), domain specific modeling languages (DSML), UML

Praca wpłynęła do redakcji: 11.05.2012