Od wymagań do Javy w mgnieniu oka: modelowanie oprogramowania w praktyce

Michał Śmiałek

Politechnika Warszawska

Wyzwania Modelowania Inżynierskiego i Biznesowego
Warszawa, 5 kwietnia 2016

Prelegent…

- Czym się zajmowałem (od ok. 1991 r.)
  - Profesor nzw. (informatyka) – Politechnika Warszawska; UML, Inżynieria Wymagań, Projektowanie, MDD/MDA, …
  - Kierownik grupy SMoG (modelowanie oprogramowania)
  - Kierownik zespołu PW w projekcie REMICS (7PR)
  - Koordynator naukowy konsorcjum projektu ReDSeeDS (6PR): główny współautor języka RSL
  - Trener w wielu szkoleniach dla profesjonalistów (> 200 edycji szkoleń) z zakresu inżynierii oprogramowania
  - Konsultant w firmie Infovide-Matrix
  - Kierownik projektów z zakresu budowy oprogramowania
  - Inżynier procesu (pierwszy w Polsce certyfikat trenera RUP)
  - Analityk wymagań oprogramowania i biznesu
  - Projektant oprogramowania i programista

Treść seminarium

- Wprowadzenie – wymagania i MDD/MDA
  - Modelowanie wymagań i transformacje w „mgnieniu oka”
- Programowanie na poziomie wymagań
  - Wyjaśnienie semantyki przypadków użycia
  - Logika aplikacji i logika dziedziny/biznesu
  - Programowanie dla „zwykłych ludzi”?

- Od wymagań do kodu
  - Automatyczna transformacja wymagań (scenariuszy przypadków użycia) w kod logiki aplikacji i logiki dziedziny/biznesu
- Szczegóły transformacji
  - Metamodel języka (Requirements Specification Language)
  - Reguły transformacji z RSL do kodu
- Podsumowanie

:: WPROWADZENIE – WYMAGANIA I MDD/MDA ::

Marzenie kierownika projektu IT…

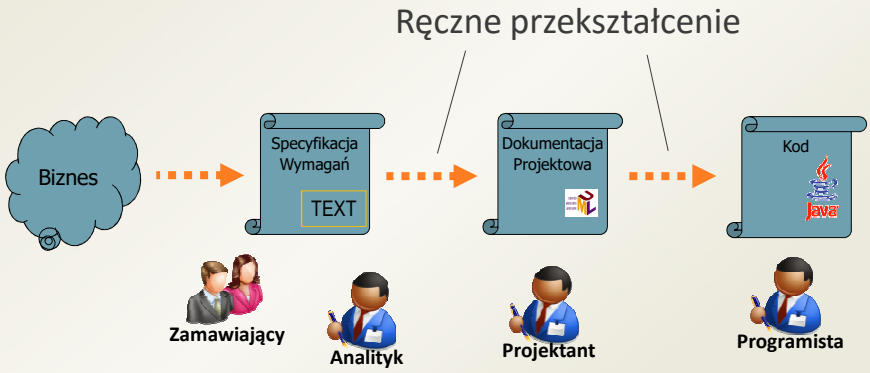
Czego potrzebujemy

Język specyfikowania wymagań

- Dobra komunikacja
- Automatyczna translacja

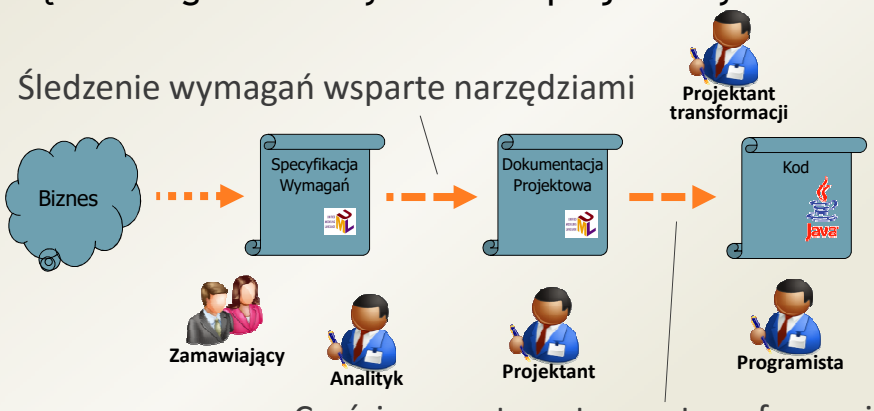
Od biznesu do kodu - tradycyjnie

Specyfikacja wymagań w języku naturalnym → specyfikacja projektowa wywiedziona z wymagań → kod wywiedziony z modeli projektowych



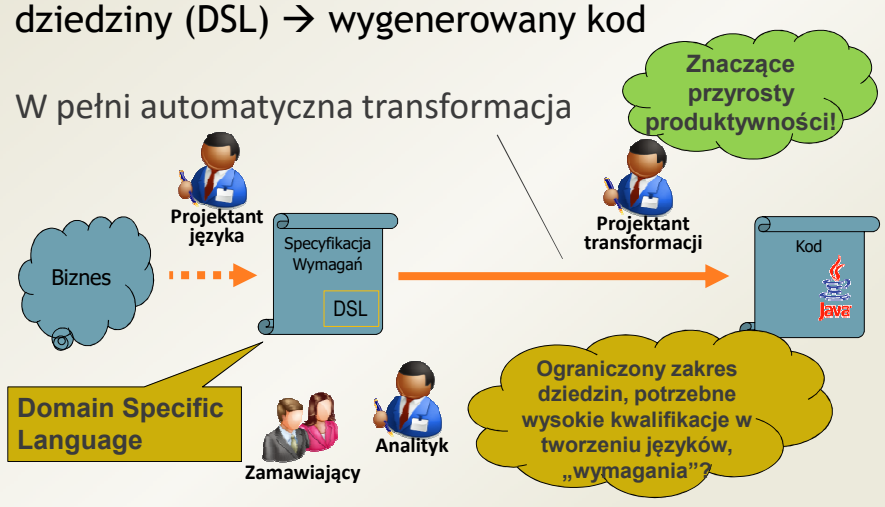
Od biznesu do kodu - sterowane modelami

Specyfikacja wymagań z półformalnymi modelami → modele projektowe ze śladem od wymagań → kod częściowo generowany z modeli projektowych



Od biznesu do kodu - języki dziedzinowe

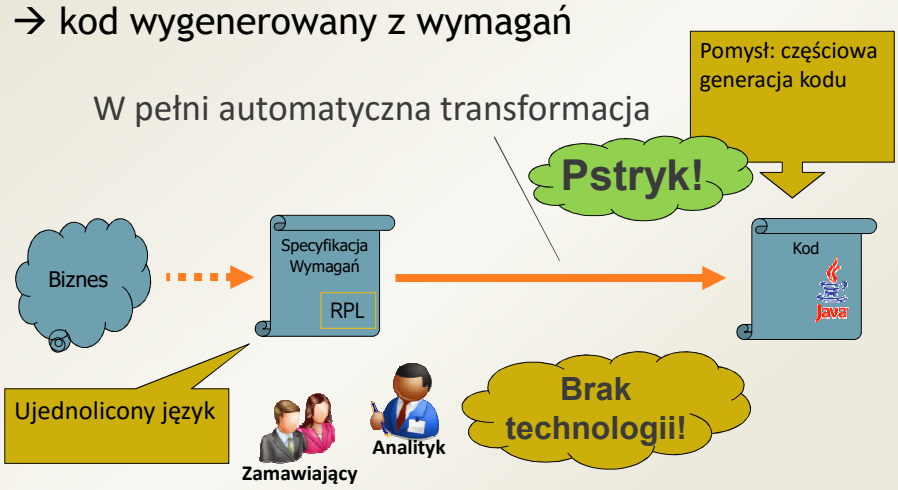
Specyfikacja systemu używając języków spec. dla dziedziny (DSL) → wygenerowany kod



Programowanie na poziomie wymagań?

Specyfikacja wymagań przy użyciu formalnych modeli → kod wygenerowany z wymagań

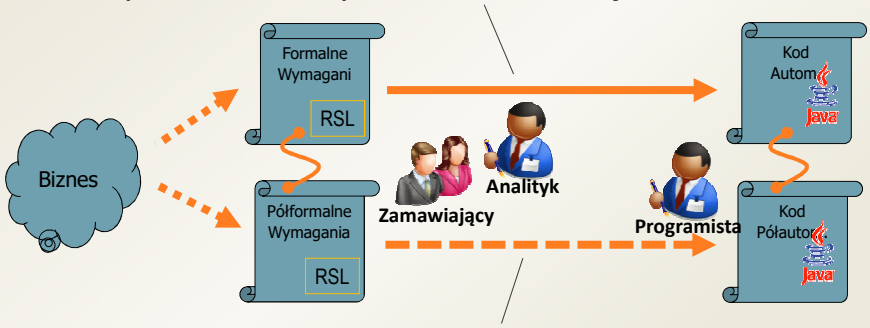
W pełni automatyczna transformacja



Proponowane rozwiązanie

Wymagania z częściowo formalnymi modelami → kod częściowo generowany z wymagań

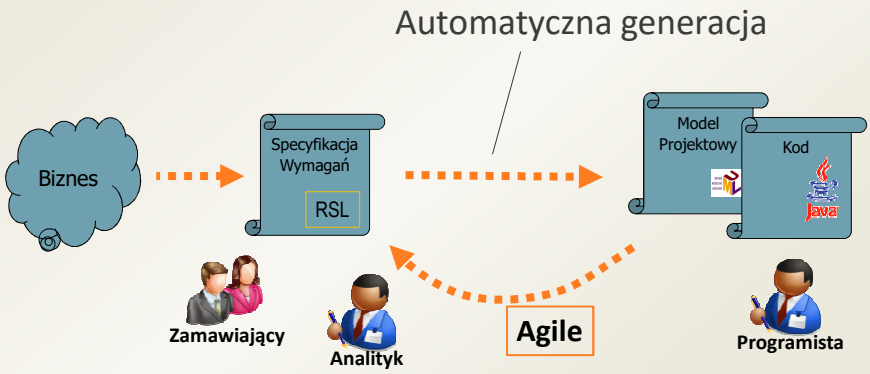
W pełni automatyczna transformacja



Częściowo automatyczna transformacja

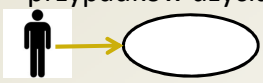
Od biznesu do kodu - ReDSeeDS

Specyfikacja wymagań przy użyciu precyzyjnych wymagań → kod wygenerowany z wymagań

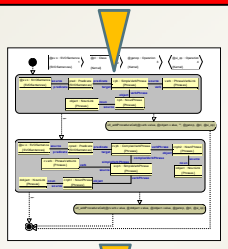


ReDSeeDS: jak to działa?

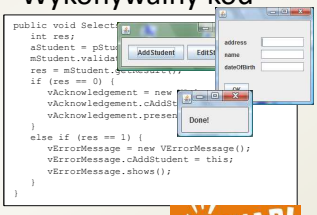
Model przypadków użycia

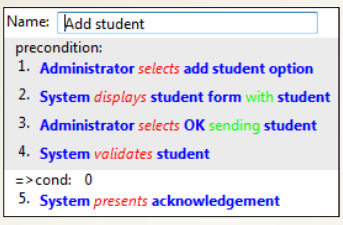


Reguły transformacji

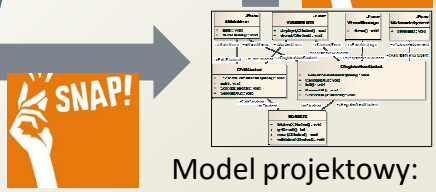


Wykonywalny kod



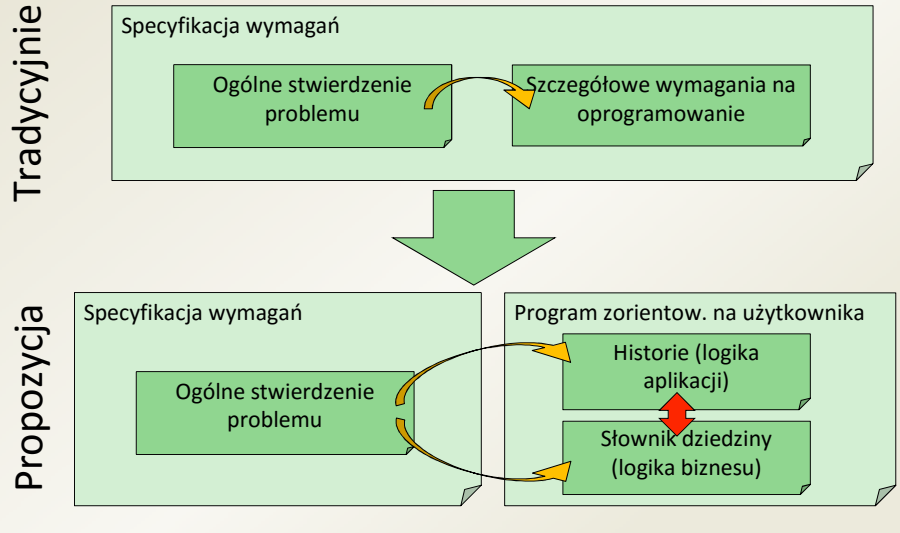


Model scenariuszy: RSL

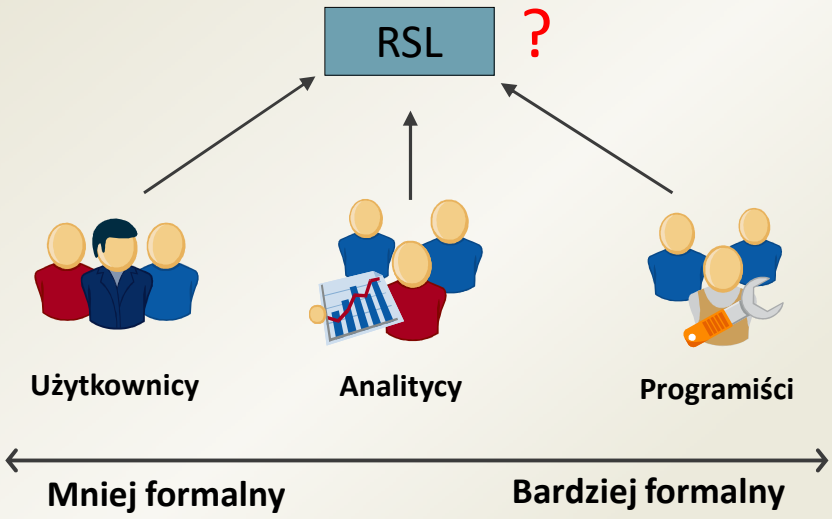


Model projektowy: UML + metody w Javie

Jeszcze wyższy poziom programowania?

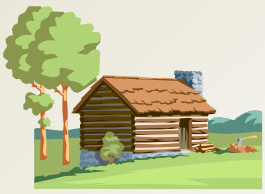


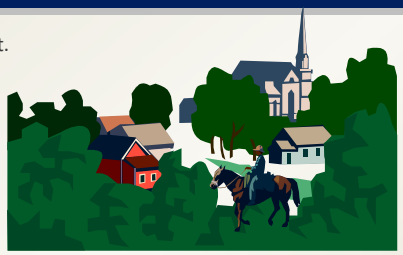
Precyzyjny język naturalny?



Historia i środowisko

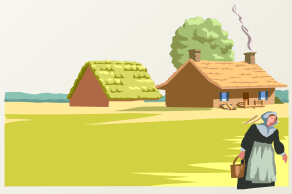
4. Tom rides to the old hut.





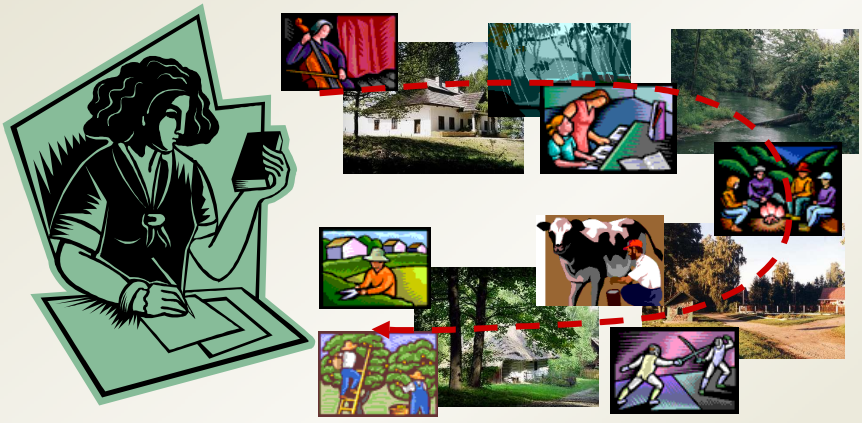
2. Lord Mark goes to the old hut.

3. Aunt Martha goes to Tom “the wise”.



1.a. Lord Mark runs to aunt Marta’s house.
b. He talks with aunt Martha.

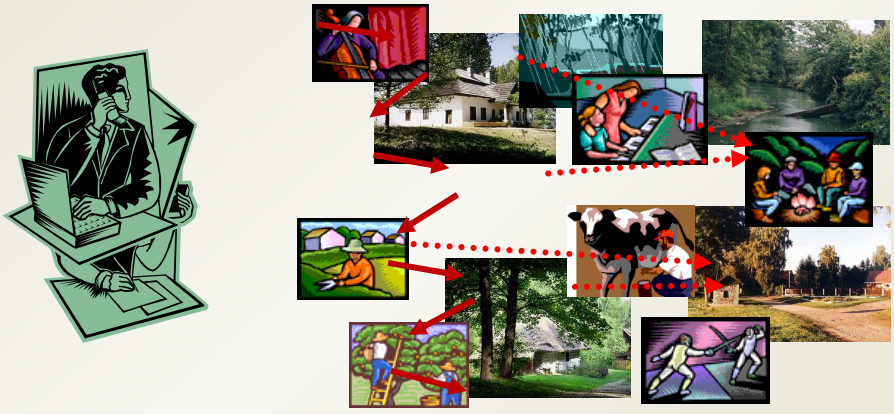
Pisanie dobrych opowiadań a pisanie dobrych wymagań



- Połączenie opowiadania (sekwencji zdarzeń) i opisów środowiska (ludzie, wnętrza, krajobraz…).

Rozdzielenie opowiadania od środowiska

- Zdarzenia a sceny



Styl pisania opowiadań

- Najprostsze możliwe zdania

Student enters the semester.

Teacher accepts the current marks.

Subject

Verb

Objects (1 or 2)

System assigns the student to the new semester.

Rozszerzenie opowiadania o „środowisko”

- Uwaga: opowiadania nie zawierają definicji pojęć

Student enters the semester ?

… where the semester is a number between 1 and 10 denoting the level of studies

ale gdzieś w innym odległym opowiadaniu:

Dean accepts the semester for the new student

… where the semester is a number denoting the current student’s status

Oddzielenie pojęć od opowiadań

- Osobny słownik pojęć, ale spójny z opowiadaniami

Student enters the semester

... where the semester is a number between 1 and 10 denoting the level of studies

Semester – number between 1 and 10 denoting the level of studies and the current student's status.

Pisanie scenariuszy

- Scenariusz jest sekwencją zdań

Dean adds new lecture to a course

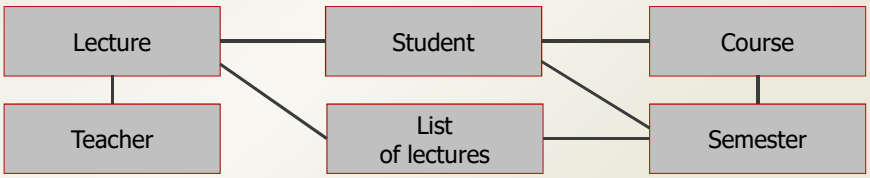
1. Dean *selects to add* new lecture to course
2. System *asks* for semester
3. Dean *enters* the semester
4. System *asks for data* of the lecture
5. Dean *enters the data* of the lecture
6. System *adds the lecture* to the list of lectures

Budowanie słownika

- Słownik jest „mapą terenu” dla użytkownika. Warto, aby mapa miała formę graficzną.

1. Dean wants to *add new lecture* to course
2. System *asks* for semester
3. Dean *enters* the semester
4. System *asks for data* of the lecture
5. Dean *enters the data* of the lecture
6. System *adds the lecture* to the list of lectures

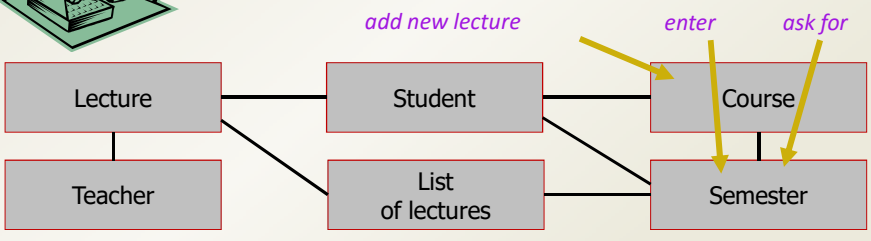
5. System *assigns* the teacher to the lecture



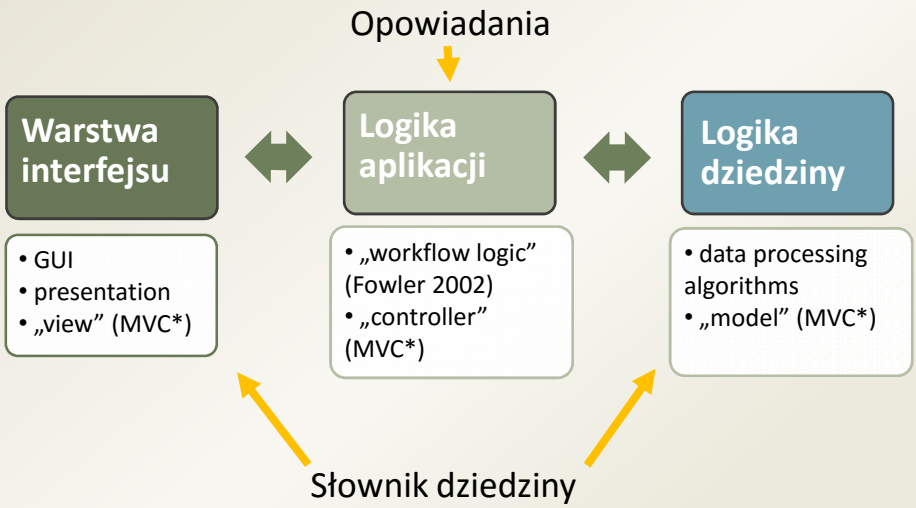
„Odgrywanie” opowiadań

- Spójność opowiadań można sprawdzić uzupełniając słownik

1. Dean wants to *add new lecture* to course
2. System *asks* for semester
3. Dean *enters* the semester
4. System *asks for data* of the lecture
5. Dean *enters the data* of the lecture
6. System *adds the lecture* to the list of lectures



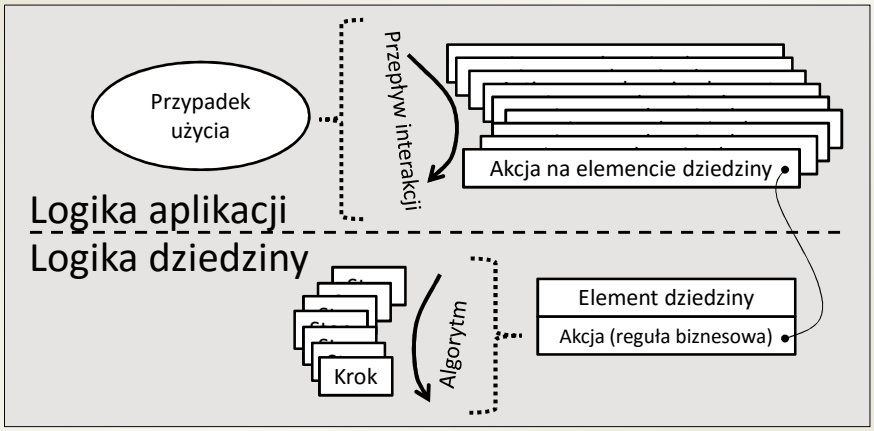
Mapowanie wymagań na strukturę systemu



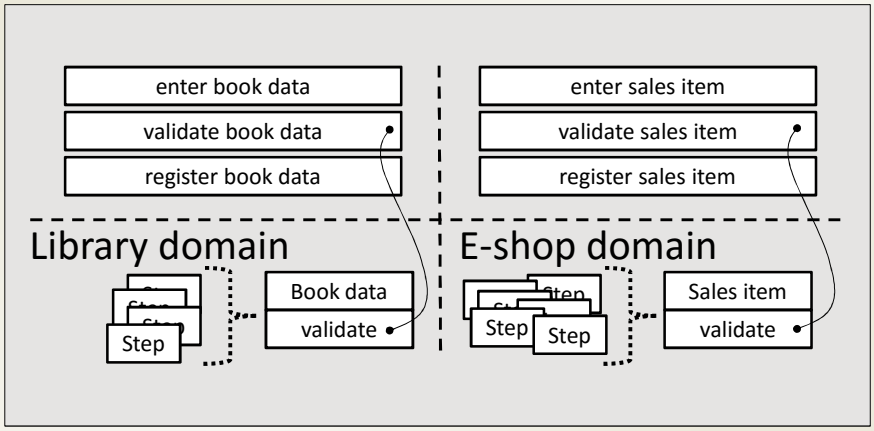
*Reenskaug 1977

Koncepcja: program na poziomie wymagań

Requirements Specification Language (RSL)



Dygresja: wzorce logiki aplikacji

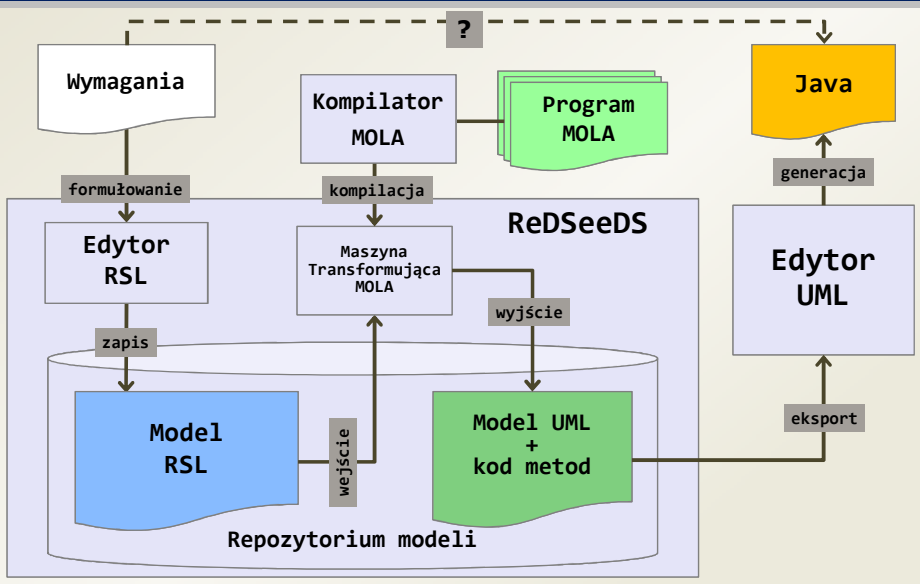


:: OD WYMAGAŃ DO KODU ::

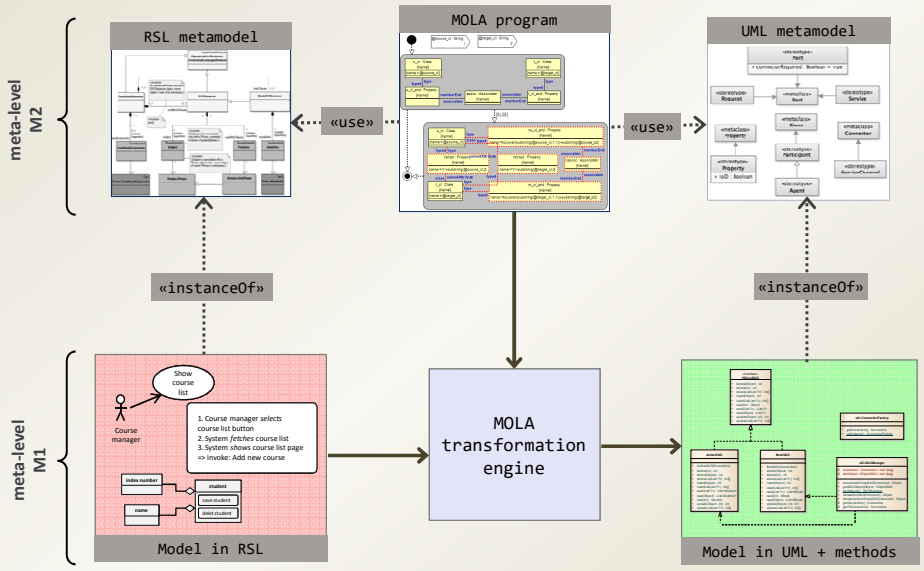
Całościowe rozwiązanie

- **Requirements Specification Language**: język formalny (DSL) dla specyfikowania typowej logiki aplikacji biznesowych + przetwarzane dane + elementy UI (brak logiki dziedzinowej)
  - Składnia abstrakcyjna oparta na precyzyjnym metamodelu
  - Składnia konkretna: graficzne diagramy + ograniczony język naturalny
  - Elastyczna semantyka czasu wykonania
- **W pełni automatyczna transformacja** z języka RSL do kodu dla wybranych technologii
- **Całościowe wsparcie narzędziowe** (edytor języka RSL, automatyczna, parametryzowalna transformacja)
  - Bardzo szybkie prototypowanie
  - Całościowa realizacja idei MDD/MDA

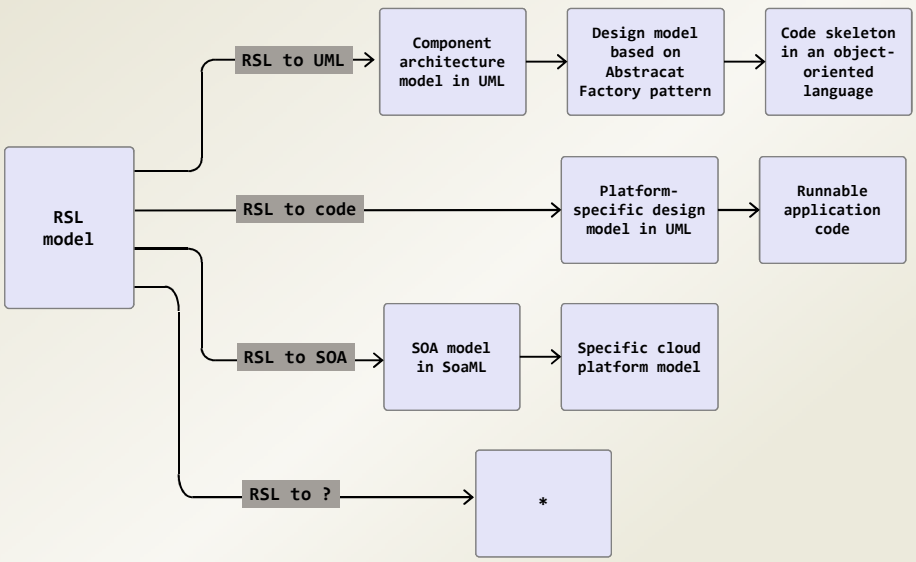
Zasada działania narzędzia



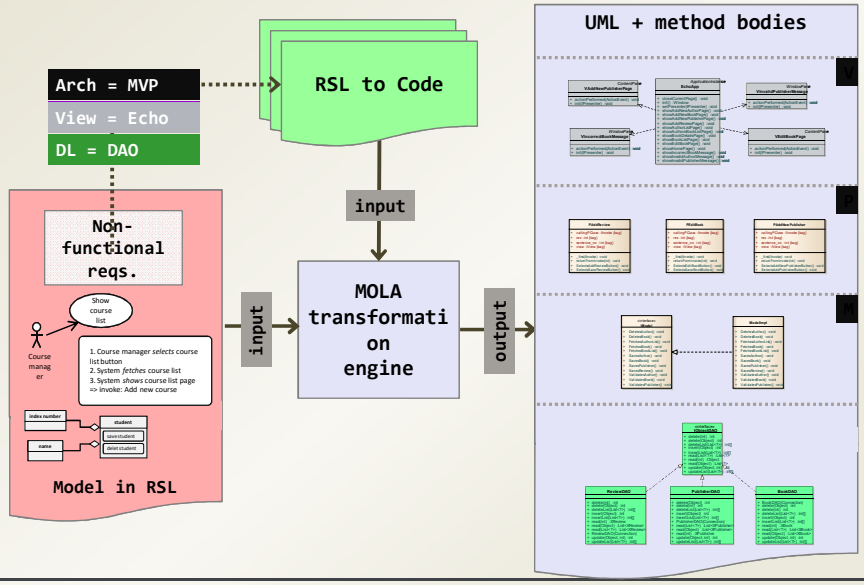
„Wnętrze” transformacji



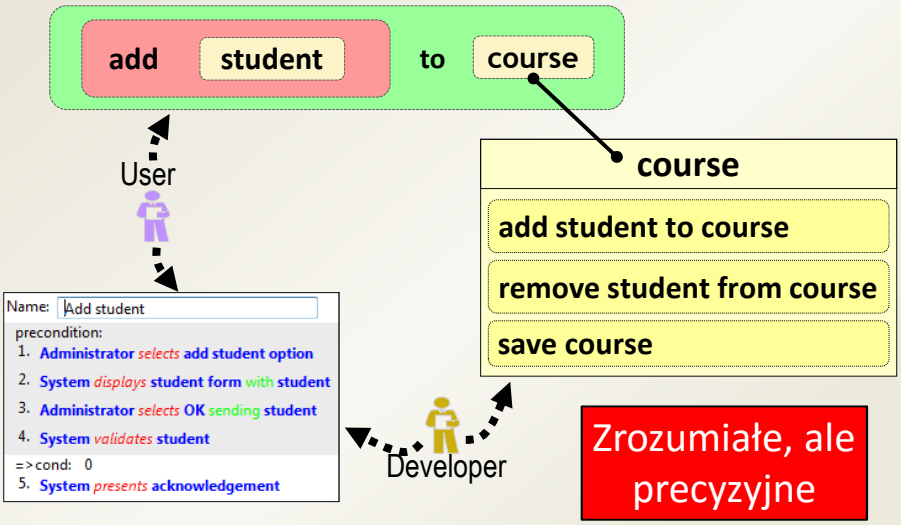
Możliwości transformacji



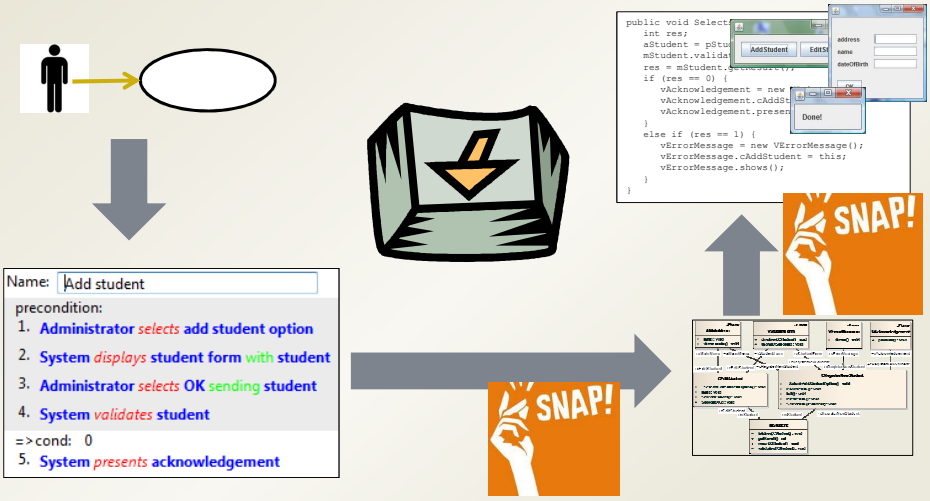
Reguły transformacji i sterowanie



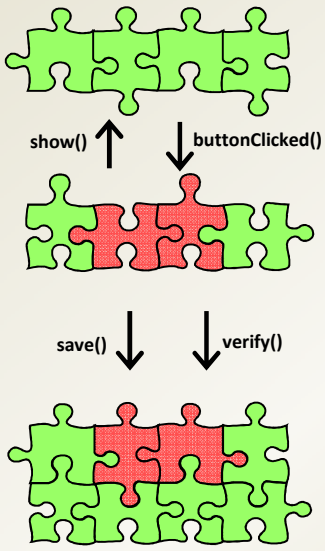
Najpierw: piszemy wymagania



Potem: naciskamy guzik



Wygenerowana struktura systemu

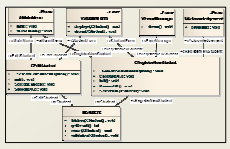


Interfejs użytkownika

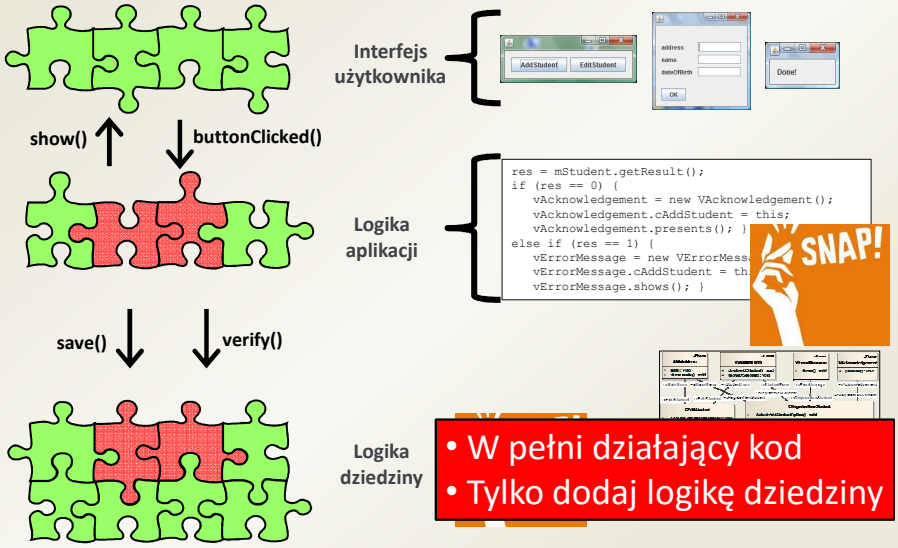
Logika aplikacji

Logika dziedziny

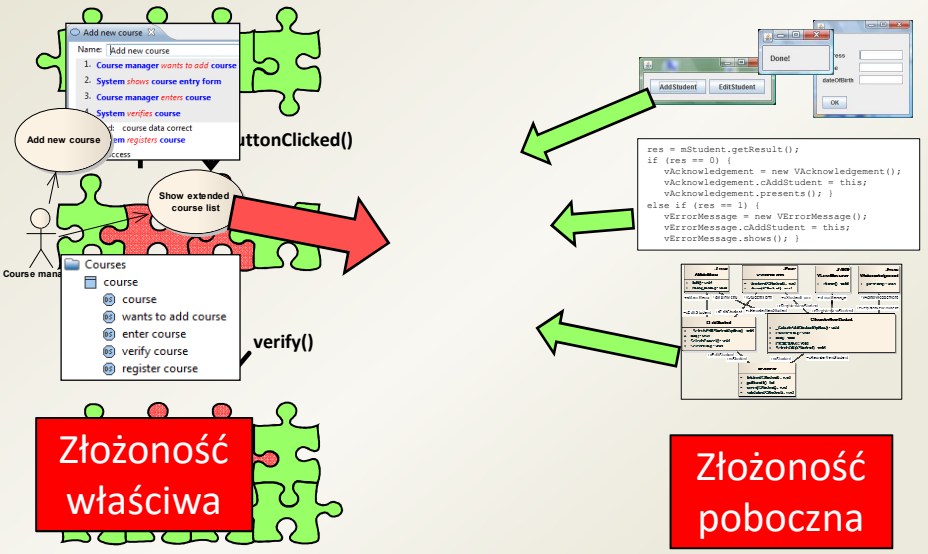
- Przejrzysta struktura logiczna
- Wybrana technologia



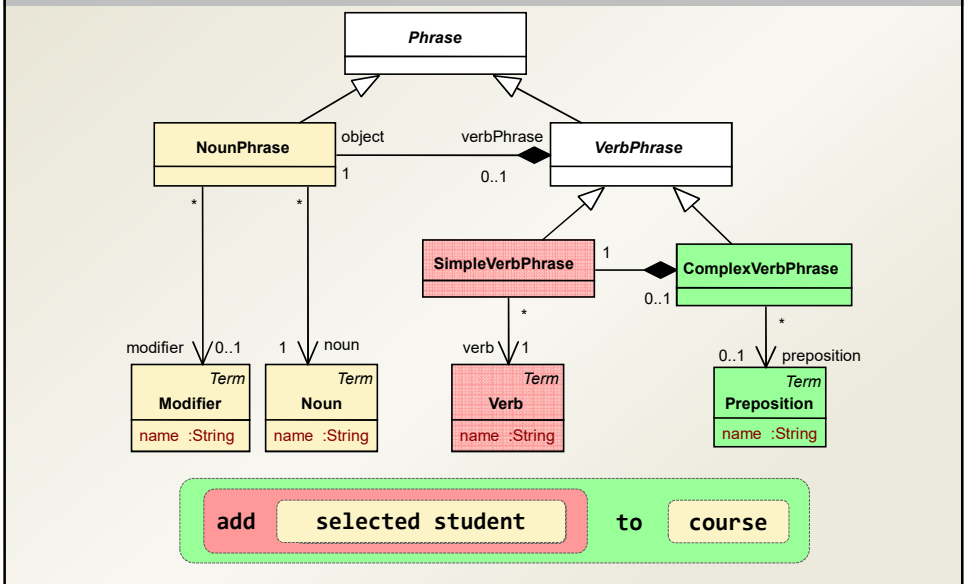
Wygenerowany działający kod aplikacji



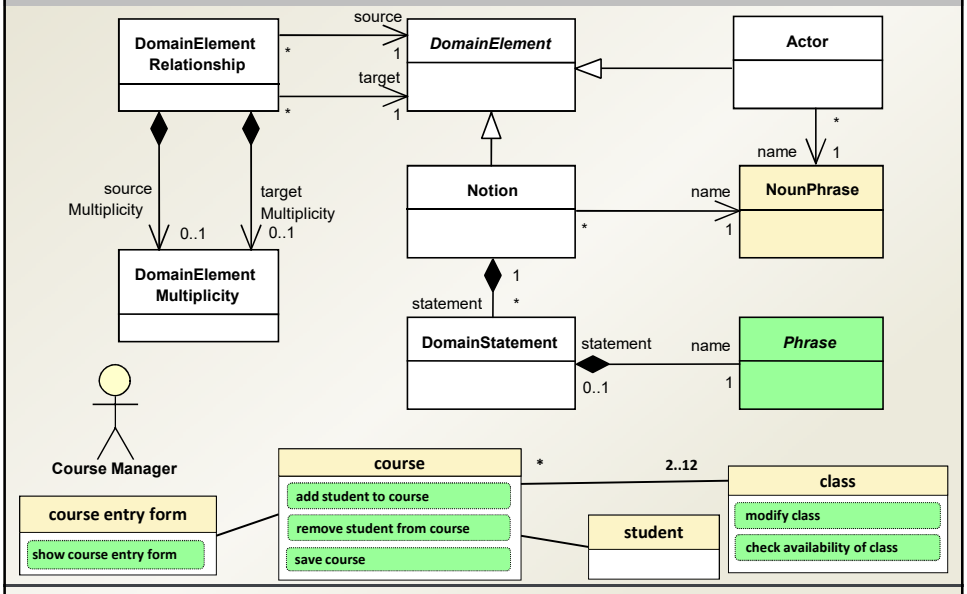
Wymagania + technologia = działający system



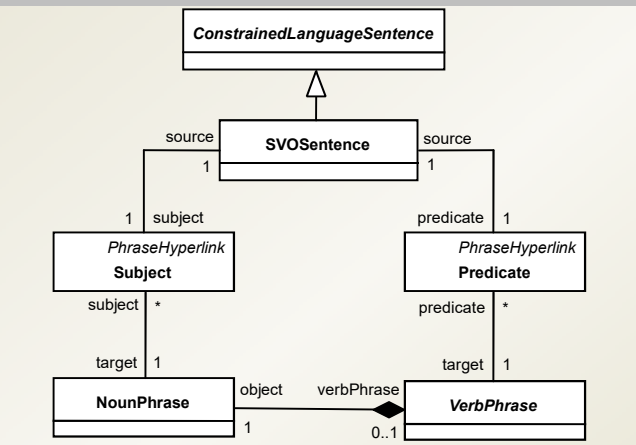
RSL metamodel: frazy (Phrase)



RSL metamodel: pojęcia (Notion)



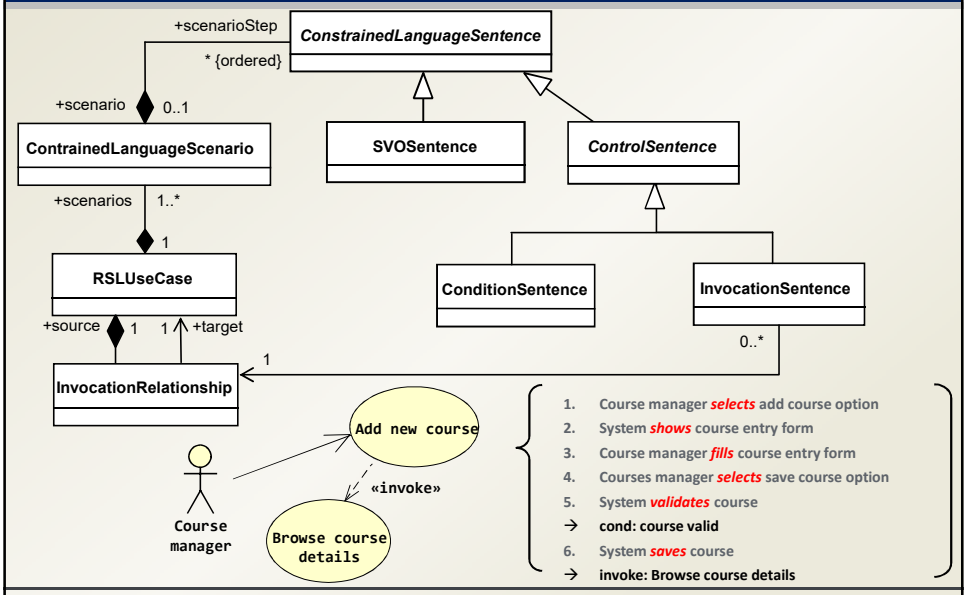
RSL metamodel: zdania SVO (SVOSentence)



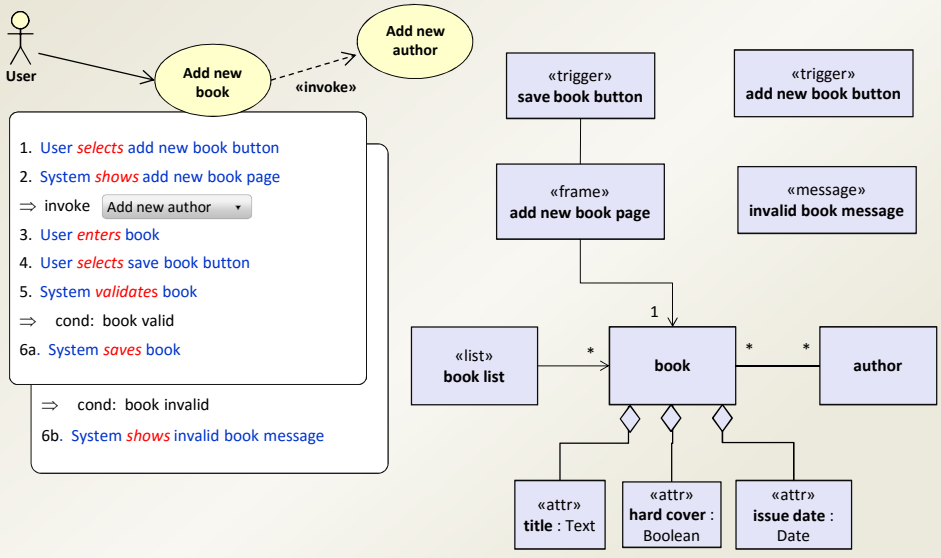
Course manager | *fills* course entry form

System | *adds* student to course

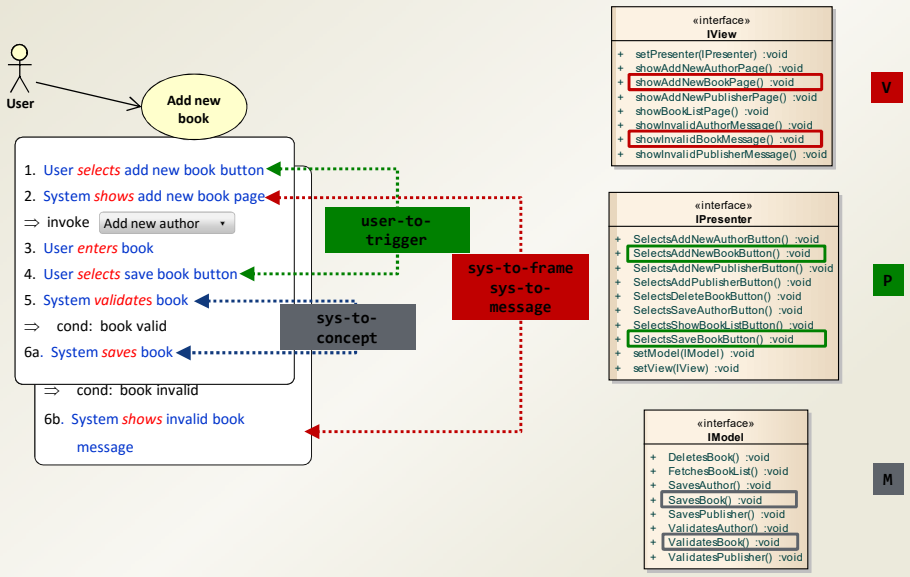
RSL metamodel: scenariusze



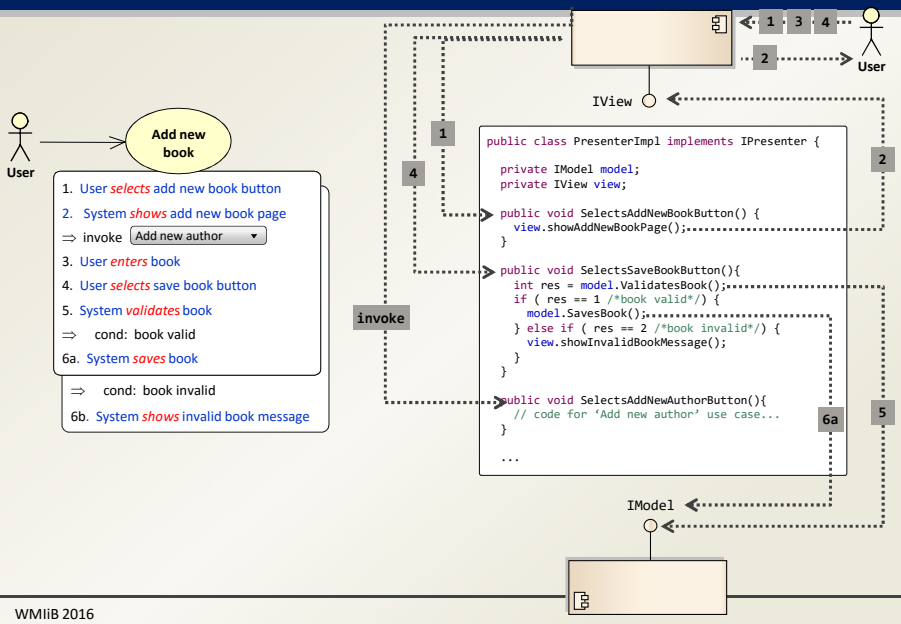
Przykład „RSL do kodu”



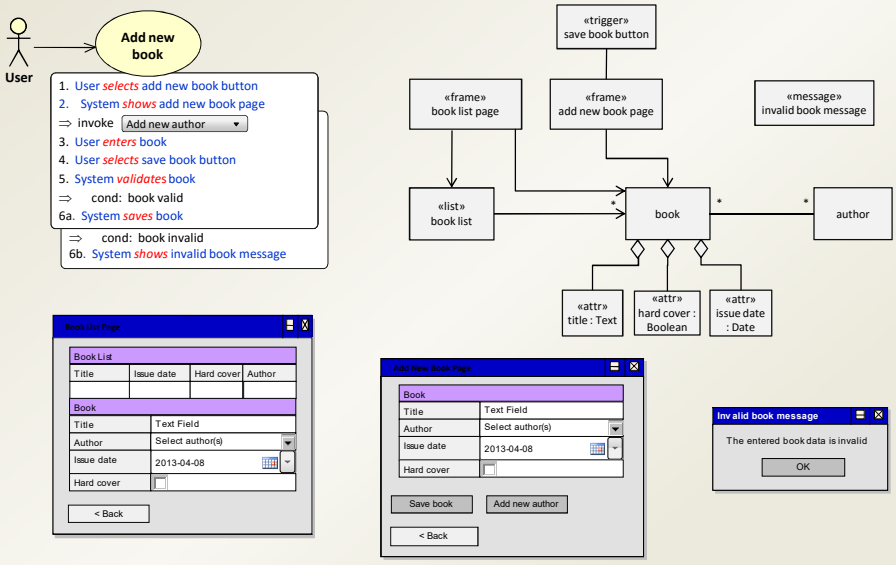
Generowanie interfejsów



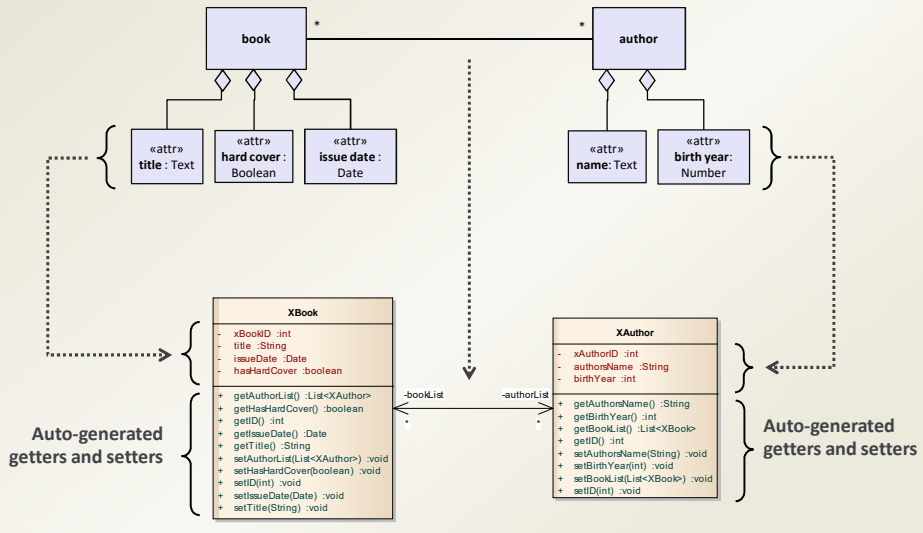
Semantyka scenariuszy



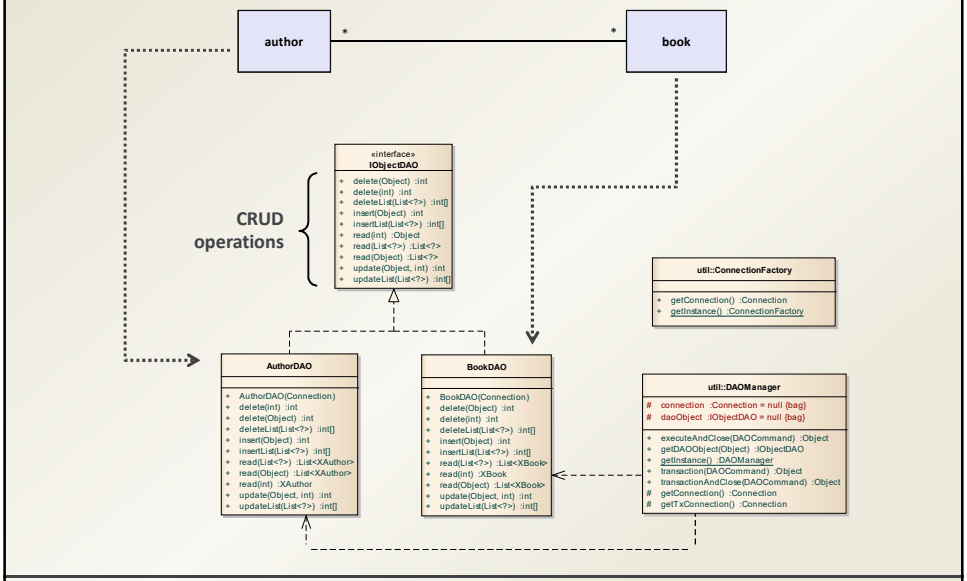
Generowanie UI



Generowanie DTO

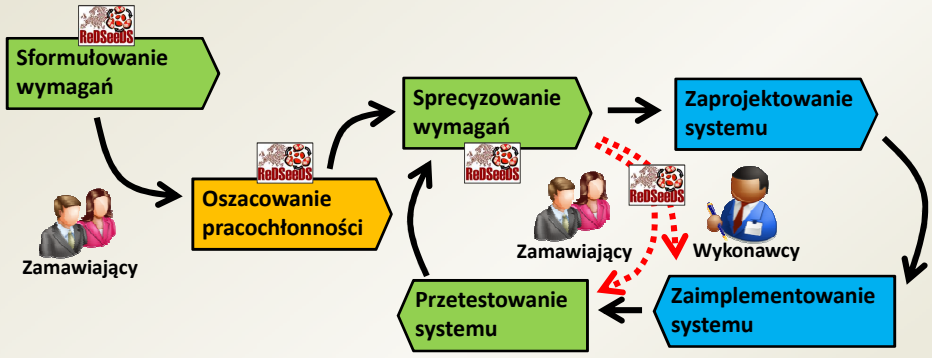


Generowanie DAO



:: PODSUMOWANIE ::

Dla kogo jest ReDSeeDS?



- Organizacje zamawiające oprogramowanie
- Firmy produkujące oprogramowanie
- Osoby amatorsko tworzące oprogramowanie

Aktualny status

- Dostepna pełna specyfikacja języka RSL
- ReDSeeDS Engine ver. 0.6.5 dostępna on-line
- Edytor RSL: przypadki użycia, scenariusze, pojęcia dziedzinowe
  - Edytory graficzne
  - Edytory tekstowe
  - Automatyczne tworzenie modelu dziedziny
  - Zintegrowana maszyna transformacji
- Integracja z Enterprise Architect i Modelio
- Generacja UML i kodu
  - Java + MVP + Swing
  - Java + MVP + Echo web framework

Przyszłość

- Rozszerzenie możliwości generacji kodu
- Rozszerzenie docelowych technologii UI (JavaFX, Flex, Android SDK, …)
- Rozszerzenie docelowych modeli architektury i języków
- Nowe zastosowania: odzyskiwanie i migracja wyeksploatowanych systemów
  - Projekt REMICS (www.remics.eu)
  - Odzyskiwanie na podstawie obserwowalnego zachowania (zapisywanie skryptów UI) do RSLa i generacja w nowej technologii
  - Niezależna od „pokręconego” wnętrza wyeksploatowanego systemu

Dziękuję za uwagę!