

<b>1.</b>	<b>SZYBKI START</b>	<b>2</b>
1.1	Podstawy obsługi narzędzia	2
1.2	Pisanie scenariuszy - Ogólne zasady	3
1.3	Oznaczanie elementów zdań i terminologia	4
1.4	Uporządkowanie Pojęć na potrzeby transformacji „RSL to C”	6
<b>2.</b>	<b>PRZYKŁAD</b>	<b>7</b>
2.1	Przypadki użycia	7
2.2	Definiowanie dziedziny aplikacji	8
2.3	Transformacja i generacja kodu	10
2.4	Wygenerowany kod	13

## 1. SZYBKI START

### 1.1 PODSTAWY OBSŁUGI NARZĘDZIA

#### 1.1.1 UTWORZENIE NOWEGO PROJEKTU (NEW PROJECT)

Aby utworzyć nowy projekt należy po otwarciu ReDSeeDSa wybrać opcję **File -> New Project**. Następnie z listy wybrać **New Software Case Project**, kliknąć **Next** i nadać nazwę projektu.

#### 1.1.1 UTWORZENIE PAKIETU WYMAGAŃ (REQUIREMENTS PACKAGE)

Aby utworzyć nowy pakiet wymagań, należy kliknąć prawym przyciskiem myszy na **Requirement Specification**, lub już istniejący pakiet pojęć, a następnie z menu kontekstowego wybrać **New -> Requirement Package**.

#### 1.1.2 UTWORZENIE PRZYPADKU UŻYCIA (USE CASE)


Aby utworzyć nowy przypadek użycia, należy kliknąć prawym przyciskiem myszy na pakiet wymagań, a następnie z menu kontekstowego wybrać **New -> Use case**.

#### 1.1.3 DODANIE ZDANIA (SVO SENTENCE)

Aby dodać nowe zdanie, w scenariuszu należy kliknąć **Create SVO sentence**



#### 1.1.4 DODANIE WARUNKU ROZGAŁĘZIENIA SCENARIUSZA

Aby dodać nowy warunek rozgałęzienia scenariusza, należy skorzystać z opcji **Fork scenario** . Po dodaniu rozgałęzienia i zapisaniu scenariusza, zostanie automatycznie utworzony scenariusz alternatywny.

#### 1.1.5 DODANIE SCENARIUSZA ALTERNATYWNEGO (ALTERNATIVE SCENARIO)

Patrz: Dodanie warunku rozgałęzienia scenariusza

#### 1.1.6 DODANIE PAKIETU POJĘĆ (NOTIONS PACKAGE)

Aby utworzyć nowy pakiet pojęć, należy kliknąć prawym przyciskiem myszy na pakiet **Notions** (znajdujący się w **Domain Specification**), lub już istniejący pakiet pojęć, a następnie z menu kontekstowego wybrać **New -> Notions Package**.

#### 1.1.7 DODANIE POJĘCIA (NOTION)

Aby utworzyć nowy przypadek użycia, należy kliknąć prawym przyciskiem myszy na pakiet pojęć, a następnie z menu kontekstowego wybrać **New -> Notion**.

#### 1.1.8 DODANIE POJĘCIA JAKO ATRYBUT (NOTION'S ATTRIBUTE)

Aby dodać jedno pojęcie jako atrybut drugiego, należy w widoku modelu (Project browser), metodą drag & drop, przeciągnąć jedno pojęcie na inne i wybrać opcję **Add as attribute**.

## 1.2 PISANIE SCENARIUSZY - OGÓLNE ZASADY

### 1.2.1 POLSKIE ZNAKI

Podczas całego procesu tworzenia specyfikacji należy **unikać używania polskich znaków**. ReDSeeDS obsługuje je prawidłowo, lecz ich użycie w specyfikacji, przeniesie się bezpośrednio na kod i może uniemożliwić jego kompilację.

### 1.2.2 ZDANIA SVO I SVOO

Podczas pisania scenariuszy należy używać zdań SVO, czyli Podmiot – orzeczenie – dopełnienie. Dopuszczalne są także zdania SVOO, które oprócz dopełnienia bezpośredniego (direct object), zawierają także dopełnienie pośrednie (indirect object).

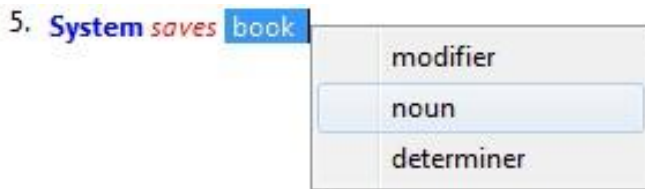
### 1.2.3 PAKIET WYMAGAŃ WEWNĄTRZ INNEGO PAKIETU WYMAGAŃ

Przypadki umieszczone w zagnieżdżonych pakietach wymagań (pakiet w pakiecie) nie będą generowane. Jest to błąd transformacji, który można wykorzystać, aby pominąć generację nieskończonych, lub niepotrzebnych w danym momencie przypadków.

## 1.3 OZNACZANIE ELEMENTÓW ZDAŃ I TERMINOLOGIA

### 1.3.1 OZNACZANIE ELEMENTÓW ZDAŃ

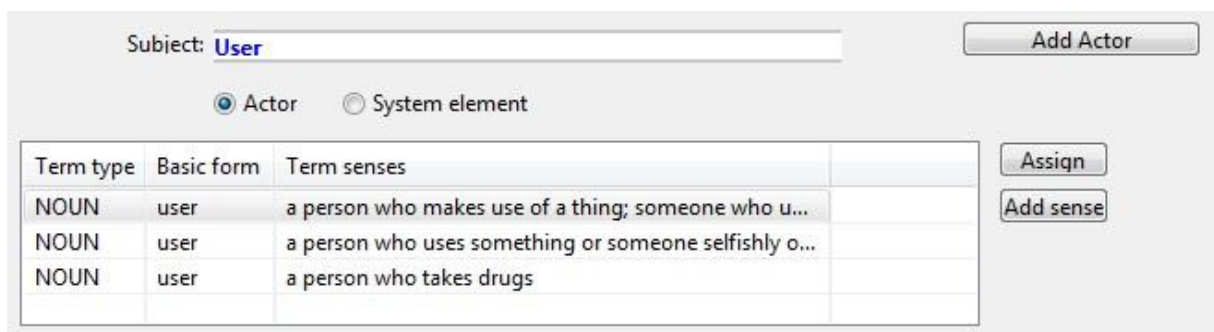
Aby oznaczyć elementy zdań takie jak rzeczownik (**noun**), czasownik (**verb**), czy przyimek (**preposition**), wystarczy kliknąć prawym przyciskiem myszy na końcu danego wyrazu, lub frazy w scenariuszu i wybrać odpowiednią pozycję z menu kontekstowego.



**Porada:** Elementy zdań można zaznaczać także poprzez metodą zaznaczania poprzez przeciągnięcie myszy, jednak metoda polegająca na klikaniu na końcu wyrazu/frazy jest znacznie szybsza.

### 1.3.2 DODANIE AKTORA

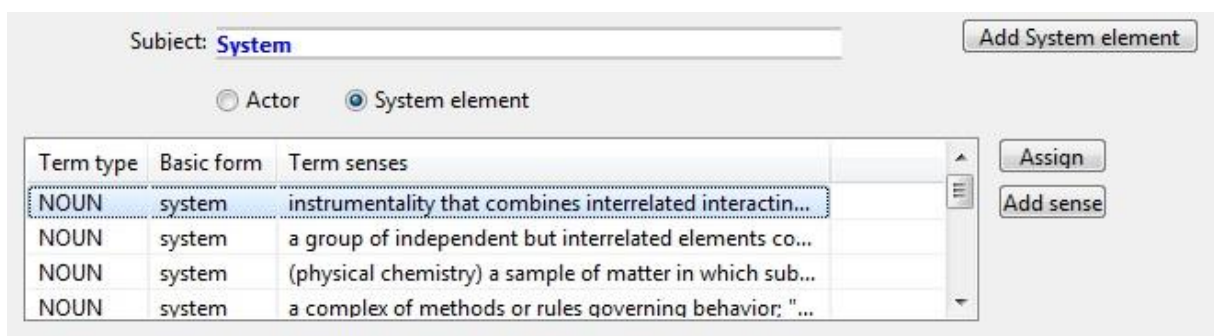
Aby dodać aktora, należy przypisać mu sens i kliknąć **Add Actor**.



Term type	Basic form	Term senses
NOUN	user	a person who makes use of a thing; someone who u...
NOUN	user	a person who uses something or someone selfishly o...
NOUN	user	a person who takes drugs

### 1.3.3 DODANIE ELEMENT SYSTEMOWEGO

Aby dodać element systemowy należy przypisać mu sens i kliknąć przycisk **Add System element**.



Term type	Basic form	Term senses
NOUN	system	instrumentality that combines interrelated interactin...
NOUN	system	a group of independent but interrelated elements co...
NOUN	system	(physical chemistry) a sample of matter in which sub...
NOUN	system	a complex of methods or rules governing behavior; "...

#### 1.3.4 PRZYPISANIE SENSU

Aby rozpocząć przypisanie sensu elementom, należy kliknąć dwukrotnie na oznaczony wcześniej element zdania. Zakładka **DomainStatementEditor** powinna wypełnić się danymi dla wybranego elementu. Jeżeli dane podświetlone są na czerwono, oznacza to, że jeszcze nie posiadają przypisanego sensu.

The screenshot shows the 'DomainStatementEditor' interface. At the top, there is a 'Predicate' field containing 'displays book details window' and an 'Add all' button. Below this, there are three rows for 'Direct object', 'Noun phrase', and 'Simple verb', each with a red-highlighted value and an 'Add' button. At the bottom, there is a table with three columns: 'Term type', 'Basic form', and 'Term senses'. The first row is highlighted in blue and contains 'NOUN', 'book det...', and 'book details window'. To the right of the table are 'Assign' and 'Add sense' buttons.

Predicate:	Direct object:	Noun phrase:	Simple verb:
displays book details window	book details window	book details window	displays book details wind

Term type	Basic form	Term senses
NOUN	book det...	book details window

W tym momencie należy wybrać element z pola **Predicate**, a następnie w tabeli poniżej zaznaczyć jeden z dostępnych sensów i kliknąć **Assign**. Gdy wszystkie pojęcia mają przypisane sensy, należy skorzystać z przycisku **Add all**. Po wykonaniu tej czynności na zostanie automatycznie utworzone odpowiednie pojęcie (lub pojęcia w przypadku zdań **SVOO**) w dziedzinie aplikacji (nowy **Notion** w **Domain specification** -> **Notions**).

Jeżeli tabela, po wybraniu elementu z pola **Predicate**, jest pusta, należy ręcznie dodać sens przy pomocy przycisku **Add sense**.

The screenshot shows the 'Add new sense' dialog box. It has a title bar with a close button. Inside, there are two main sections: 'Word's base form:' with a text field containing 'book', and 'Part of speech:' with a dropdown menu. The dropdown menu is open, showing a list of parts of speech: Adjective, Adverb, Cond. conjunction, Determiner, Noun (highlighted in blue), Preposition, and Verb. Below these sections is a text area labeled 'Enter sense:' containing the text 'A set of written, printed, or blank pages fastened alor between protective covers.' At the bottom, there are 'Add sense' and 'Cancel' buttons.

**Porada:** Sensy stosowane są głównie w celu ewentualnego reużywania specyfikacji. Na zajęciach WOSM, wpisywany sens nie ma dużego znaczenia i nie ma potrzeby wpisywania definicji ze słownika.

## 1.4 UPORZĄDKOWANIE POJĘĆ NA POTRZEBY TRANSFORMACJI „RSL TO C”

### 1.4.1 ELEMENTY INTERFEJSU UŻYTKOWNIKA

Pojęcia będące elementami interfejsu użytkownika takimi jak okna, formularze, komunikaty, należy umieścić w pakiecie pojęć, którego nazwa rozpoczyna się od **!UI** (np. **!UIElements**). Dla tych pojęć zostaną wygenerowane specjalne klasy wraz z importami i konstrukcjami charakterystycznymi dla elementów interfejsu użytkownika.

### 1.4.2 ATRYBUTY

Pojęcia będące atrybutami innych pojęć, należy umieścić w pakiecie pojęć, którego nazwa rozpoczyna się od **!AT** (np. **!ATtributes**). Dla tych pojęć nie będą generowane osobne klasy.

### 1.4.3 INNE PAKIETY ROZCZYNAJĄCE SIĘ OD „!”

Jeżeli nazwa pakietu pojęć rozpoczyna się od „!”, ale nie **!UI** i nie **!AT** to pojęcia z tego pakietu nie będą generowane jako klasy.

**Porada:** Zaleca się utworzyć pakiet **!Buttons** w pakiecie **!UI** i umieszczać w nim przyciski danej aplikacji, ponieważ przyciski są zazwyczaj elementami jakiegoś okienka, a nie osobnymi klasami. Zaleca się także utworzenie pakietu pojęć o nazwie **!NotGenerated**, w którym umieszczane będą inne pojęcia, które powstały podczas pisania specyfikacji wymagań, a nie ma potrzeby generowania dla nich osobnych klas.

## 2. PRZYKŁAD

W tym rozdziale zostanie przedstawiony przykładowy proces wytwarzania oprogramowania przy pomocy narzędzia ReDSeeDS.

### 2.1 PRZYPADKI UŻYCIA

Sposób pisania scenariuszy przypadków użycia w narzędziu **ReDSeeDS**, niewiele różni się od sposobu pisania scenariuszy przy użyciu **Structured Specification** w narzędziu **Enterprise Architect**. Tutaj także dodaje się kolejne zdania i scenariusze przy pomocy specjalnych przycisków. Ważne jest jednak, aby pamiętać o stosowaniu się do reguł i porad zawartych w poprzednim rozdziale.

**Porada:** Najlepiej najpierw napisać scenariusze, a dopiero później oznaczać elementy zdań (rzeczownik, czasownik) oraz nadawać sensy. Dzięki takiemu działaniu, nie następuje wytrącenie z ciągu myślowego, poprzez wykonywanie czasochłonnych zadań, które można wykonać później. Dodatkowo, późniejsze ponowne przejście scenariuszy pozwala wychwycić ewentualne błędy i niespójności.

#### 2.1.1 PRZYPADEK: ADD NEW BOOK

Scenariusz podstawowy:

Name:

precondition:

1. User <i>clicks</i> add new book button	system ▼
2. System <i>displays</i> add new book window	user ▼
3. User <i>fills</i> add new book window <i>with</i> book	system ▼
4. System <i>validates</i> book	system ▼

=> cond: 0 /\*Book data ok\*/

5. System <i>saves</i> book	user ▼
-----------------------------	--------

Scenariusz alternatywny:

Name:

precondition:

1. User <i>clicks</i> add new book button	system ▼
2. System <i>displays</i> add new book window	user ▼
3. User <i>fills</i> add new book window <i>with</i> book	system ▼
4. System <i>validates</i> book	system ▼

=> cond: 1 /\*Book data not ok\*/

4.1.1 System <i>displays</i> error message	user ▼
--	--------

### 2.1.2 PRZYPADEK: DISPLAY BOOK

Name:

precondition:

1. User *clicks* display books button
2. System *fetches* book list
3. System *displays* book list window *with* book list
4. User *chooses* book
5. System *displays* book details window

system	▼
system	▼
user	▼
system	▼
user	▼

## 2.2 DEFINIOWANIE DZIEDZINY APLIKACJI

### 2.2.1 DODANIE AKTORA, ELEMENTU SYSTEMOWEGO ORAZ NADANIE SENSÓW

Na tym etapie, należy dodać aktora, element systemowy oraz nadać sensy wszystkim pojęciom występującym w specyfikacji. Dokładny opis tego procesu został opisany w rozdziale **Oznaczanie elementów zdań i terminologia**.

**Porada:** Jeżeli system nie podpowiada sensów, dla podstawowych czasowników/rzeczowników to najprawdopodobniej nie został uruchomiony serwer terminologii, lub ReDSeeDS z jakiś powodów nie może się z nim połączyć. Aby prawidłowo połączyć ReDSeeDSa ze wspomnianym serwerem należy:

1. Uruchomić serwer terminologii (plik **start-jgwnl-server.bat**)
2. Począkać kilka sekund, aż pasek postępu uruchamiania serwera się zapełni.
3. Uruchomić ReDSeeDSa

### 2.2.2 DODANIE ATRYBUTÓW

Poszczególnym pojęciom można także nadać atrybuty (metoda opisana w podpunkcie **Dodanie pojęcia jako atrybut**). Aby podejrzeć atrybuty dla danego pojęcia należy kliknąć dwukrotnie na pojęcie i na nowo-otwartej zakładce odnaleźć tabelę **Attributes**. W przykładzie pojęciu **Book** nadano atrybut **Name**.

Name:  Type:

Path:

Attributes:

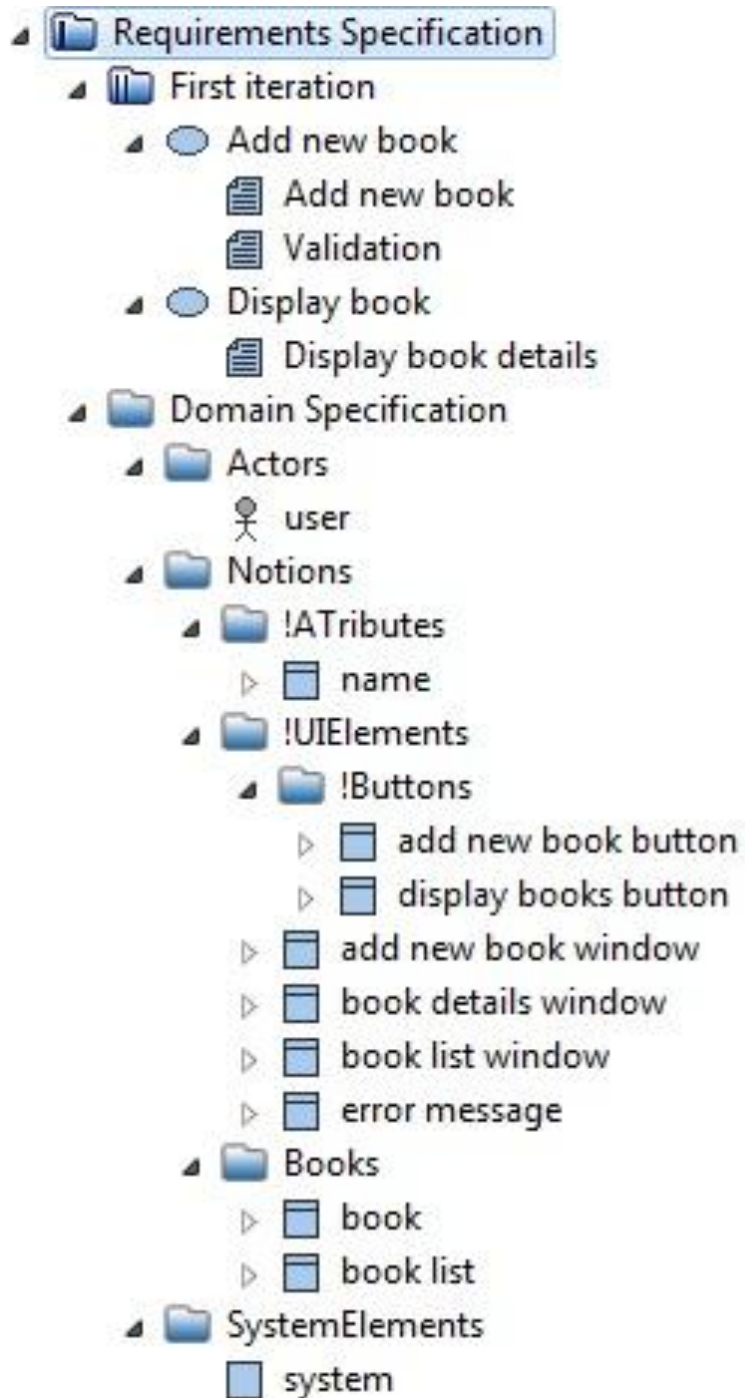
Attribute	Type	
<input checked="" type="checkbox"/> name	(none)	

Remove



### 2.2.3 PRZYKŁAD UPORZĄDKOWANEGO MODELU

Model dla przedstawionych scenariuszy, po uporządkowaniu, powinien prezentować się w następujący sposób:



## 2.3 TRANSFORMACJA I GENERACJA KODU

### 2.3.1 WALIDACJA

Przed wykonaniem transformacji warto wykonać walidację, poprzez kliknięcie prawym przyciskiem myszy na **Requirement Specification**, a następnie wybranie opcji **Validate**. Walidacja to proces polegający na sprawdzeniu poprawności modelu. Jeżeli wynik walidacji będzie zawierał tylko ostrzeżenia, bez błędów, to można przystąpić do wykonywania transformacji.

**Porada:** Transformacja **RSL to C** (i inne) zostanie wykonana nawet w przypadku, gdy model będzie zawierał błędy! Może to skutkować niewygenerowaniem pewnych klas, lub funkcji, a co się z tym wiąże – koniecznością poprawy modelu, ponownego wykonania transformacji i poprawy kodu w momencie odnalezienia błędu.

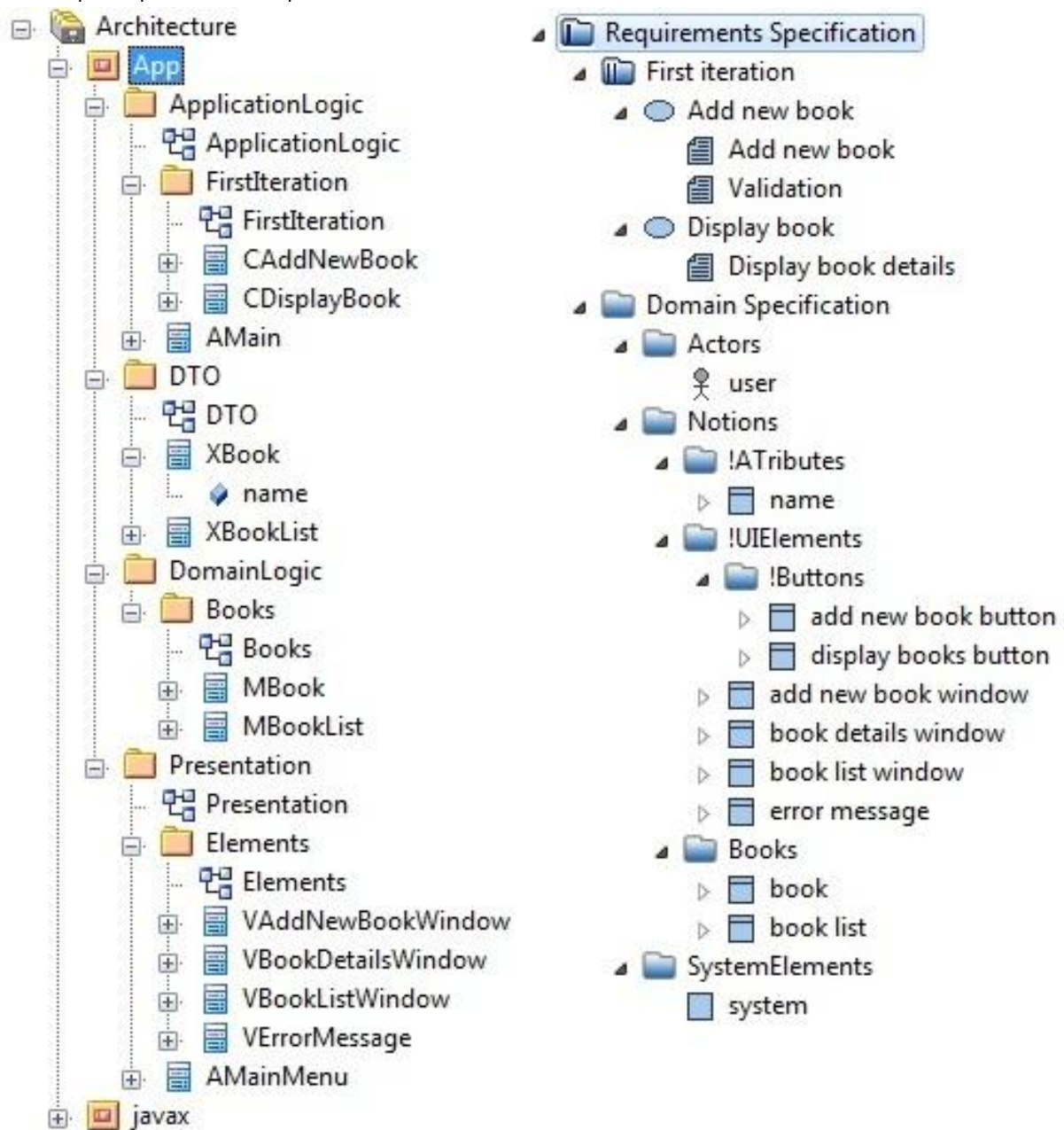
### 2.3.2 WYKONANIE TRANSFORMACJI

Aby wykonać transformację należy wybrać z menu opcję **ReDSeeDS -> Transformations...**, a następnie z nowo-otwartego okienka wybrać transformację, która ma zostać wykonana.

Po wykonaniu transformacji, które operują tylko na metamodelach (taką jest **RSL to C**), należy jeszcze wykonać transformację **\_UML\_TO\_EA**, która powoduje eksport modelu do Enterprise Architect'a. Niestety, ze względu na dość powolne API, ta transformacja może zająć od kilku sekund do nawet kilku minut, w zależności od rozmiaru modelu.

Model wynikowy transformacji **\_UML\_TO\_EA** można znaleźć w pliku **ea\_redseeds.eap**, w workspace'ie ReDSeeDS'a.

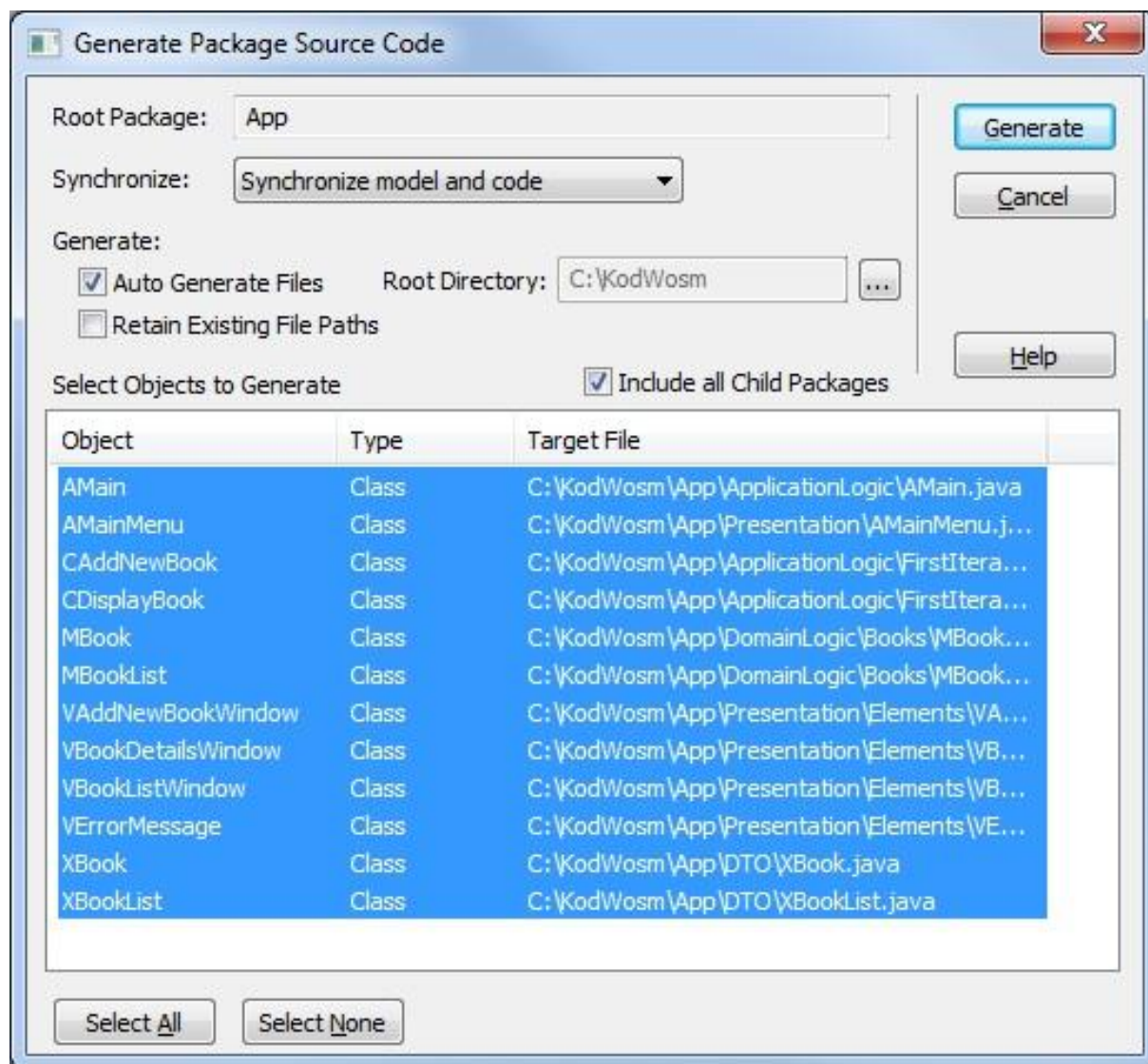
Poniżej przedstawiono zestawienie przedstawionej wcześniej specyfikacji wymagań, z wynikiem transformacji **RSL to C** po eksporcie do Enterprise Architect'a.



### 2.3.3 GENERACJA KODU W ENTERPRISE ARCHITECT

Aby wygenerować kod aplikacji z poziomu Enterprise Architect'a, należy kliknąć prawym przyciskiem w modelu na pakiet **App**, a następnie wybrać opcję **Code engineering -> Generate source code**. Następnie zaznaczyć checkboxy zgodnie ze ilustracją poniżej i kliknąć przycisk **Generate**.

**Porada:** Dokładny sposób wyboru opcji generacji kodu, może nieznacznie się różnić w zależności od wersji Enterprise Architect'a.



## 2.4 WYGENEROWANY KOD

### 2.4.1 KLASA CADDNEWBOOK

```
public class CAddNewBook {
    public XBook aBook;
    VAddNewBookWindow nAddNewBookWindow;
    MBook nBook;
    AMainMenu nMainMenu;
    public VAddNewBookWindow vAddNewBookWindow;
    public MBook mBook;
    public AMainMenu aMainMenu;

    public CAddNewBook() {
    }

    public void finalize() throws Throwable {
    }

    public void _ClicksAddNewBookButton() {
        int res=0;
        vAddNewBookWindow = new VAddNewBookWindow();
        vAddNewBookWindow.cAddNewBook = this;
        vAddNewBookWindow.displays();
    }
    /**
     *
     * @param pBook      redseeds_uid7671714083261269300--6917814001634927640--
     * 554483506756785203--8217886436700875814redseeds_uid
     */
    public void FillsAddNewBookWindow(XBook pBook) {
        int res=0;
        aBook = pBook;
        mBook.validates(aBook); res = mBook.getResult();
        if (res == 0 /*Book data ok*/) {
            mBook.saves(aBook); res = mBook.getResult();
        }
        else if (res == 1 /*Book data not ok*/) {
            vErrorMessage = new VErrorMessage();
            vErrorMessage.cAddNewBook = this;
            vErrorMessage.displays();
        }
    }

    public void init() {
        aBook = new XBook();
    }
}
```



```
public class CDisplayBook {  
  
    public XBookList aBookList;  
    VBookDetailsWindow nBookDetailsWindow;  
    MBookList nBookList;  
    VBookListWindow nBookListWindow;  
    AMainMenu nMainMenu;  
  
    public AMainMenu aMainMenu;  
    public MBookList mBookList;  
    public VBookListWindow vBookListWindow;  
    public VBookDetailsWindow vBookDetailsWindow;  
  
    public CDisplayBook() {  
    }  
  
    public void finalize() throws Throwable {  
    }  
  
    public void _ClicksDisplayBooksButton() {  
        int res=0;  
        mBookList.fetches(aBookList); res = mBookList.getResult();  
        vBookListWindow = new VBookListWindow();  
        vBookListWindow.cDisplayBook = this;  
        vBookListWindow.displays(aBookList);  
    }  
  
    public void ChoosesBook() {  
        int res=0;  
        vBookDetailsWindow = new VBookDetailsWindow();  
        vBookDetailsWindow.cDisplayBook = this;  
        vBookDetailsWindow.displays();  
    }  
  
    public void init() {  
        aBookList = new XBookList();  
    }  
}
```

### 2.4.3 WARUNKI

Warto zwrócić uwagę na sposób generacji warunków i tego jak przekładają się one na kod. Przykład przedstawiono poniżej:

4. System *validates* book

=>cond: 0 /\*Book data ok\*/

5. System *saves* book

=>cond: 1 /\*Book data not ok\*/

4.1.1 System *displays* error message

```
public void FillsAddNewBookWindow() {
    int res=0;
    mBook.validates(aBook); res = mBook.getResult();
    if (res == 0 /*Book data ok*/) {
        mBook.saves(aBook); res = mBook.getResult();
    }
    else if (res == 1 /*Book data not ok*/) {
        vErrorMessage = new VErrorMessage();
        vErrorMessage.cAddNewBook = this;
        vErrorMessage.displays();
    }
}
```

**Porada:** Transformacja RSL to C zawiera błąd, który występuje w przypadku zastosowania dwóch warunków w jednym przypadku użycia. W takim przypadku, kod dla jednego z warunków generuje się na złym poziomie zagnieżdżenia i wymaga ręcznej poprawy.

### 2.4.4 ZDANIE SVOO (OD SYSTEMU DO AKTORA)

W przypadku, gdy użyjemy zdania SVOO i następuje przepływ sterowania od systemu do aktora, klasa odpowiadająca rzeczownikowi występującemu po przyimku zostanie użyta jako parametr funkcji, pobrany z warstwy logiki biznesowej. Przykład poniżej:

3. System *displays* book list window *with* book list

```
public void _ClicksDisplayBooksButton() {
    int res=0;
    mBookList.fetches(aBookList); res = mBookList.getResult();
    vBookListWindow = new VBookListWindow();
    vBookListWindow.cDisplayBook = this;
    vBookListWindow.displays(aBookList);
}
```

**Porada:** Konstrukcja ta w przypadku nieprawidłowego użycia, może być kłopotliwa i generować kod niezgodnie z zamierzeniem twórcy. Np. **System adds book to database** spowoduje generację funkcji **add(aDatabase)** w klasie **MBook**, podczas gdy zamierzonym efektem było utworzenie **add(aBook)** w klasie **MDatabase**.

### 2.4.5 ZDANIE SVOO (OD AKTORA DO SYSTEMU)

W przypadku, gdy użyjemy zdania SVOO i następuje przepływ sterowania od aktora do systemu, klasa odpowiadająca rzeczownikowi występującemu po przyimku zostanie użyta jako parametr funkcji. Parametr ten zostanie przekazany przez odpowiedni element interfejsu użytkownika. Przykład poniżej:

3. User *fills* add new book window *with* book  
4. System *validates* book

```
public void FillsAddNewBookWindow(XBook pBook) {
    int res=0;
    aBook = pBook;
    mBook.validates(aBook); res = mBook.getResult();
}
```

**Porada:** W tym przypadku także należy wystrzegać się błędu, który został opisany w podpunkcie **Zdanie SVOO (od systemu do aktora)**.