Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Черниговская Лидия Александровна

Реализация задания правил генерации в WMP

Курсовая работа

Научный руководитель:
к. т. н., доцент каф. СП Ю. В. Литвинов

Санкт-Петербург
2017

Оглавление

Введение

Одним из подходов к разработке программных систем является визуальное моделирование. При данном подходе программа описывается с помощью моделей на графическом языке, содержащем набор элементов и набор связей между ними. Решение задачи сводится к созданию диаграммы в специальном редакторе данного визуального языка.

Существуют инструментарии – CASE-системы, которые имеют заранее определенный набор редакторов визуальных языков для создания новых систем. Но зачастую создание системы с использованием языков общего назначения оказывается довольно сложным, и требуется создать свой визуальный язык под конкретную предметную область или конкретную задачу, и уже для этого языка добиться полной генерации кода по визуальным моделям. Такой подход называется предметно-ориентированным визуальным моделированием (Domain-Specific Modelling, DSM), а созданный язык, соответственно, предметно-ориентированным.

Создание своего визуального языка, редакторов, генераторов и других инструментов для этого языка “с нуля” под каждую конкретную задачу – трудоёмкая операция. Поэтому существуют инструменты, позволяющие автоматизировать этот процесс. Такие инструменты называются DSM-платформами или metaCASE-системами. MetaCASE-системы содержат специальный визуальный язык – метаязык, который позволяет описывать сущности и связи требуемого языка. Модель, построенная с помощью метаязыка, называется метамоделью. По ней генерируется новый язык.

Некоторые MetaCASE-системы содержат дополнительные инструменты. В частности, часто бывает необходимо не просто формальное описание программы с помощью визуального языка, но и отображение диаграммы этой программы в некоторый код на текстовом языке, который можно потом запустить и получить результат. Для того, чтобы не писать генератор кода для каждого нового языка заново, в metaCASE-системах может присутствовать способ формального задания правил

генрации кода.

В настоящее время почти все DSM-платформы должны быть установлены на компьютер, что не всегда удобно, так как это требует времени пользователя и вычислительной мощности персонального компьютера. В связи с этим было принято решение о создании браузерной DSM-платформы.

В конечном итоге хотим иметь полноценную DSM-платформу в вебсреде, где для создания нового визуального языка программисту необходимо будет формально описать синтаксис этого языка, используя метаредактор, реализующий вышеупомянутый метаязык, и задать правила генерации для этого нового языка.

Разработка ведется в проекте Web Modelling Project (WMP)[1].

Данная задача разбивается на два этапа. Первый включает в себя подготовку метаредактора. Второй – реализацию задания правил генерации.

[1]WMP – `https://github.com/qreal/wmp`

1. Постановка задачи

Целью курсовой работы является реализация задания правил генерации.

В рамках курсовой работы требуется решить следующие задачи.

1. Доработать клиентскую часть, а именно: отделить ядро, реализующее работу с абстрактными диаграммами от кода, связанного с роботами.

2. Добавить в существующую архитектуру проекта WMP метаредактор, реализованный в рамках курсовой второго курса.

3. Сделать обзор существующих решений.

4. Выбрать подходящий инструмент для реализации генерации.

5. Опробовать выбранный инструмент для генерации кода в отдельном приложении.

6. Интегрировать данное решение в проект WMP.

2. Обзор существующих решений

В данном разделе рассматриваются существующие решения задания правил генераций, веб-приложение, позволяющее генерировать код по моделям, а также архитектура проекта WMP.

2.1. QReal

В данном подразделе рассматриваются решения задания правил генерации в проекте, разрабатываемом на кафедре системного программирования, QReal

2.1.1. QReal:Geny

Для системы QReal была разработана система задания правил генерации: текстовый язык Geny, редактор для него и интерпретатор [4].

На рис. 1 представлен пример задания правил генерации на языке Geny.

```
1  Task JavaClass
2  / Produce Java class from repo
3
4  foreach Class in elementsByType(Class)
5              toFile name.java
6  class name {
7                          foreach MethodsContainer in children
8                                      foreach Method in children
9              methodVisibility methodReturnType
10             methodName( @@!task MethodParameters@@ ) {
11             }
12                                     }
13                         }
14
15                         foreach FieldsContainer in children
16                                     foreach Field in children
17             fieldVisibility fieldType fieldName;
18                                     }
19                         }
20             }
21 }
22 }
```

Рис. 1: Пример описания генератора на языке Geny в редакторе из [4]

Предложенное решение совмещает два подхода реализации: с использованием шаблонов генерации и без, с большим количеством управляющих конструкций. Оно явно содержит скелет итоговой программы,

что увеличивает наглядность и понятность итогового кода, и позволяет использовать управляющие конструкции, что расширяет класс визуальных языков, для которых можно будет задать таким способом правила генерации.

Данное решение не было до конца реализовано и не встроено в систему QReal.

2.1.2. Визуальный язык задания правил генерации

В системе QReal также было реализовано визуальное средство задания правил генерации – специальный визуальный язык, позволяющий описывать формально генератор для структурных языков, т.е. тех языков, которые не совершают действия над потоками данных или управления.

Основным элементом для задания правил генерации является элемент, позволяющий задавать тип элемента визуального языка (для которого создается генератор), и текстовый код, который будет генерироваться при обходе элемента этого типа [5].

Также данный язык позволяет задавать:

- правила генерации для всех элементов, вложенных в некоторый элемент;
- текстовые шаблоны, по которым будет сгенерирован файл или метка с определенным именем;
- конвертеры, которые позволяют преобразовывать строковые представления.

Визуальное средство также не было встроено в систему QReal.

2.1.3. Генерация кода в режиме "метамоделирования на лету"

Средство для задания правил генерации в режиме "метамоделирования на лету" позволяет задавать правила для каждого элемента разрабатываемого языка. Можно использовать ключевые конструкции языка для задания правил генерации, а также обращаться к свойствам

элемента, для которого задается правило. Обращение к текущему элементу происходит по имени этого элемента в языке. Текст, который без изменений попадет в итоговый код, выделяется кавычками (‘’). Текст самого генератора не выделяется никак [8].

Общая схема генерации итогового кода по диаграмме выглядит следующим образом. У каждого элемента метамодели в её контекстном меню есть пункт “Добавить правило генерации”. Если выбрать данный пункт, то будет показано окно для задания правила генерации для данного элемента (рис. 2). В нём есть список шаблонов языка, список имён свойств данного элемента и текстовое поле для задания правила. В метамодели создаваемого языка для каждого элемента имеется свойство generationRule, которое содержит в себе текст правила для данного элемента. Разбор правил начинается с корневого элемента. Правило сначала разбирается на лексемы, а затем по нему строится абстрактное синтаксическое дерево. Если в правиле встречается строка callGeneratorFor (Element, GeneratorName), где Element – имя элемента метамодели, а GeneratorName – имя генератора, то в метамодели ищется правило генерации для элемента с именем Element, затем оно разбирается на лексемы и по нему тоже строится абстрактное синтаксическое дерево.

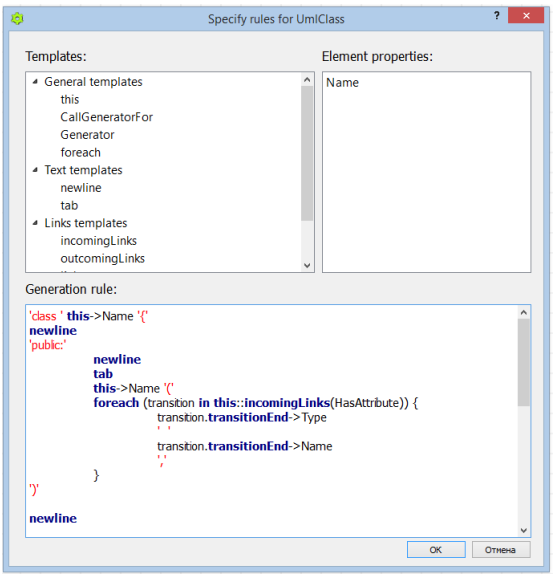

Рис. 2: Внешний вид окна для задания правил генерации из [8]

2.2. GenMyModel

GenMyModel[2] – веб-приложение, позволяющее пользователям создавать модели на известных языках, таких как UML, Ecore, BPMN и т.п., и генерировать код по этим моделям, используя либо уже имеющиеся в проекте генераторы (рис. 3), либо созданные самостоятельно.

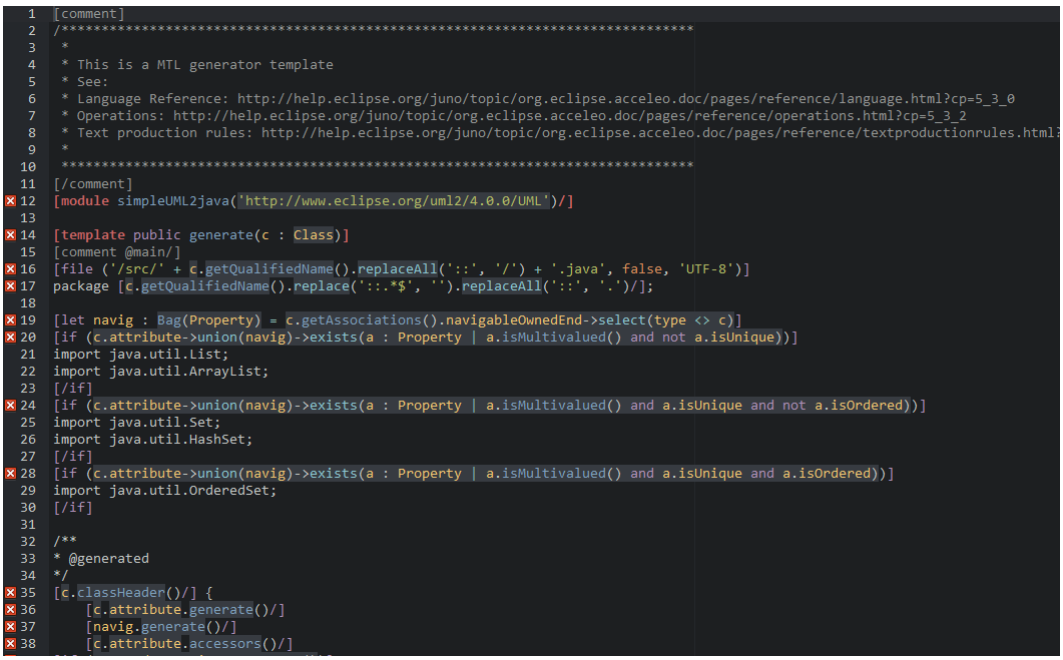

Рис. 3: Пример генератора из UML в Java

Модели в данном приложении можно создавать, используя только стандартные языки, в то время как мы бы хотели уметь создавать свои собственные языки и для этих языков задавать правила генерации.

2.3. Acceleo

Acceleo [6] – продукт Eclipse. Данный инструмент позволяет создавать свой собственный генератор для определенной метамодели.

Acceleo довольно популярен для реализации генерации, например, в рассмотренном ранее примере – GenMyModel используется Acceleo. Также он имеет широкий набор функциональности. И еще одним преимуществом является возможность использования Acceleo вне Eclipse.

[2]Основной сайт GenMyModel – `https://www.genmymodel.com/`

2.4. JET (Java Emitter Template)

JET [3] также является продуктом Eclipse, который позволяет генерировать текстовый вывод на основе модели EMF(Eclipse Modelling Framework). Использует технологию шаблонов. Пользователь определяет шаблоны (рис. 4), которые затем используются для создания Java-классов (рис. 5). Данный шаг называется трансляцией. Затем на этапе генерации из этих Java-классов создаются конечные файлы (рис. 6).

```
<%@ jet package="hello" imports="java.util.*" class="XMLDemoTemplate" %>

<% List elementList = (List) argument; %>
<?xml version="1.0" encoding="UTF-8"?>
<demo>
<% for (Iterator i = elementList.iterator(); i.hasNext(); ) { %>

  <element><%=i.next().toString()%></element>
<% } %>
</demo>
```

Рис. 4: JET, шаблон

```
List data = new ArrayList();
data.add("first");
data.add("second");
data.add("third");

XMLDemoTemplate generateXml = new XMLDemoTemplate();
String result = generateXml.generate(data);
System.out.println(result);
```

Рис. 5: JET, сгенерированный Java-класс

```
<?xml version="1.0" encoding="UTF-8"?>

<demo>
  <element>first</element>
  <element>second</element>
  <element>third</element>

</demo>
```

Рис. 6: JET, конечный XML файл

2.5. WMP (Web Modelling Project)

Основным архитектурным подходом в Web Modeling Project является микросервисный подход. Функциональность проекта разделена на сервисы, запускаемые отдельно и связанные друг с другом с помощью сетевых соединений (рис.7). Интеграция между микросервисами в WMP осуществляется с помощью Remote Procedure Call (RPC), для чего используется технология Apache Thrift.[3]

Сервисы, имеющиеся в проекте на момент начала работы.

1. Authentication service – сервис OAuth аутентификации.
2. Dashboard service – сервис панели управления редакторами.
3. Editor service – сервис редактора, предоставляющий пользователю возможность работать с визуальными языками.
4. User DB Service – сервис хранения пользователей.
5. Diagram DB Service – сервис хранения диаграмм.

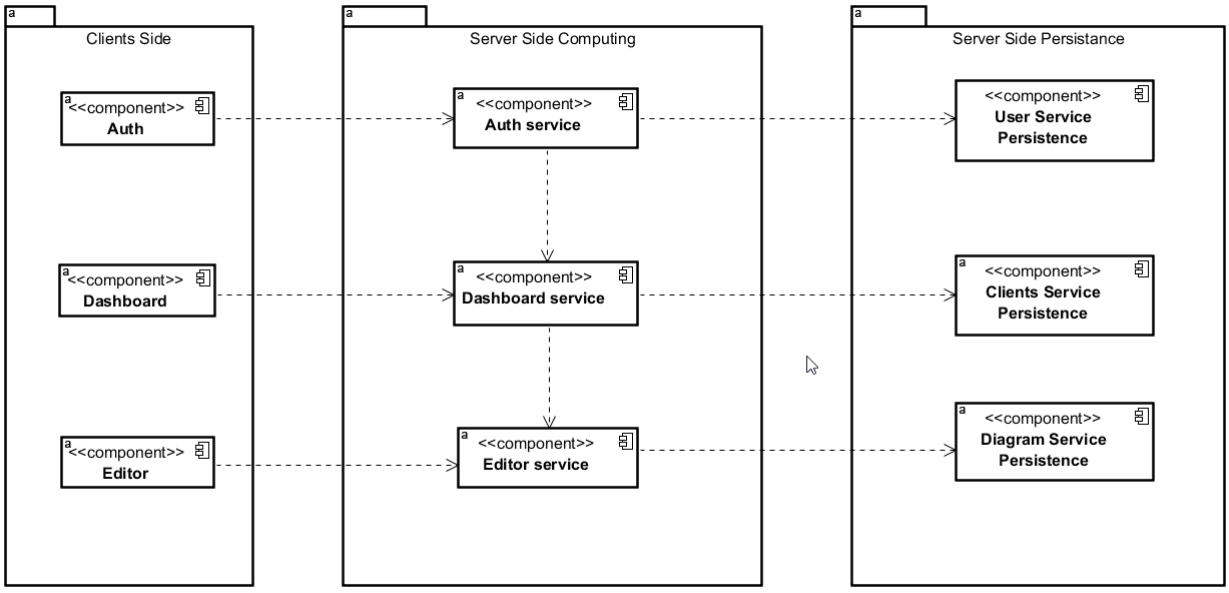

Рис. 7: Архитектура WMP

[3]Основной сайт Apache Thrift – `https://thrift.apache.org`

3. Клиентская часть

Изначально велась разработка онлайн среды графического программирования роботов. На момент начала работы в проекте WMP была реализована поддержка визуального языка программирования роботов. И первостепенной задачей являлось отделение ядра в клиентской части проекта, отвечающего за работу с абстрактным редактором, от кода, отвечающего за язык роботов. Логика, связанная с роботами, перенесена в отдельную папку, и произведено удаление ненужного кода.

Такое разделение было необходимо для того, чтобы при создании "вручную" нового визуального языка, можно было использовать ядро редактора и расширять его так, как нужно для создаваемого визуального языка.

Функциональность ядра:

- перемещение элементов с палитры на сцену;
- отмена и возврат действий пользователя, связанных с элементами редактора;
- хранение информации о текущем расположении роботов.

На данный момент в проекте имеется три визуальных языка, использующих данное ядро – язык программирования роботов, язык BPMN (Business Process Model and Notation) и метаязык.

4. Метаредактор

В рамках курсовой работы второго курса был реализован метаредактор, позволяющий создавать новые визуальные языки. Метаязык состоит из двух элементов: “сущность” и “свойство” [7].

Следующей задачей было добавление данного метаредактора в существующую архитектуру проекта WMP.

К уже имеющимся сервисам потребовалось добавить сервис для работы с базой данных, отвечающей за хранение новых визуальных языков, – Palette DB Service (рис.8).

В сервис редактора была добавлена функциональность для метаязыка.

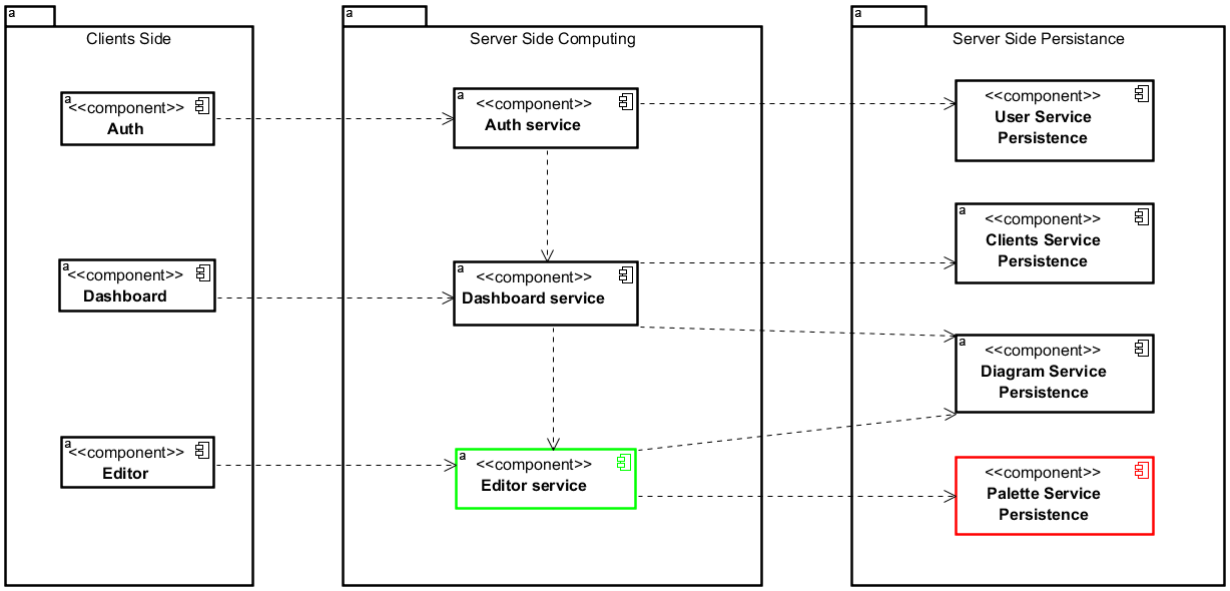

Рис. 8: Изменения в архитектуре WMP для метаредактора

Теперь на главной странице у пользователей имеется возможность выбрать, какой язык им использовать. Если был выбран метаязык, то с помощью него можно нарисовать метамодель нового языка. На рис. 9 представлена модель, созданная с помощью метаязыка, по которой генерируется новый визуальный язык.

При нажатии на кнопку Create и ввода имени нового языка от клиента на сервер отправляется запрос на сохранение, который обрабатывается в сервисе редактора и затем отправляется на сервис хранения

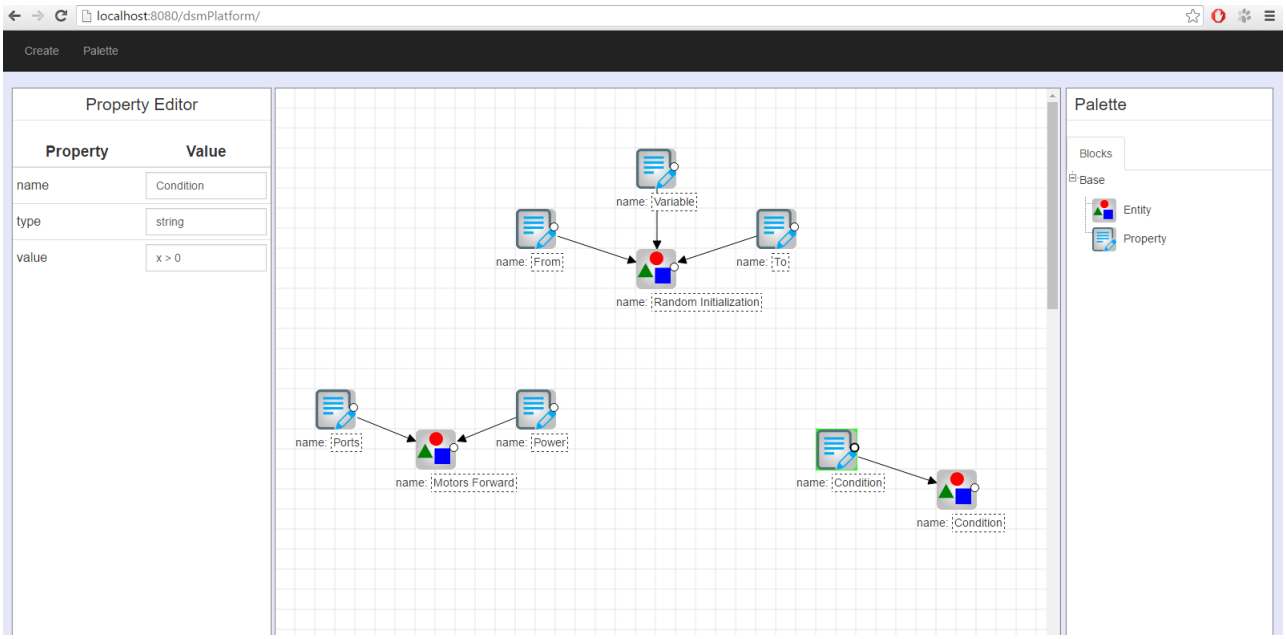

Рис. 9: Пример описания нового языка в метаредакторе WMP

языков. В палитру загружаются элементы нового языка, с помощью которых можно рисовать диаграммы на созданном языке. Также имеется возможность выбора другого созданного языка.

5. Генерация

В даном разделе описаны этапы разработки задания правил генерации.

5.1. Acceleo

Для реализации генерации был выбран Acceleo. Правила генерации в Acceleo показаны на рис. 10. Для начала было проведено знакомство с данным инструментом в среде Eclipse. Написан свой генератор для языка UML, генерирующий Java-классы с полями и методами по классам UML.

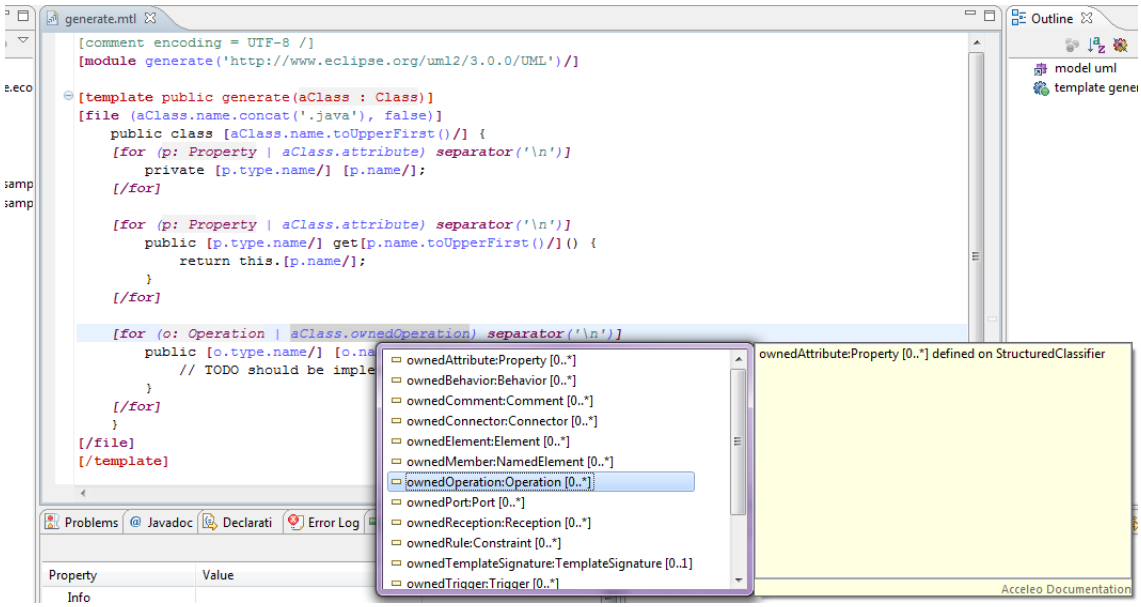

Рис. 10: Пример описания правил генерации в Acceleo из [1]

Затем требовалось научиться использовать генератор вне Eclipse, а также создавать файл генерации вне Eclipse. На данном этапе возникли сложности, в связи с набором библиотек, необходимых для использования Acceleo standalone, и их соответствием друг другу.

В ходе поиска решения проблемы была найдена статья разработчиков из Альбертского университета, в которой рассказывается о проекте MTCLauncher [2]. В данном проекте присутствует следующая функциональность: по метамодели, представленной в формате Ecore, модели, соответствующей этой метамодели и описанной в формате XMI, а

также правилам генерации, описанных в файле .mtl, генерируется код, соответствующий правилам генерации. Для реализации данной функциональности используется Acceleo.

Было решено использовать данную разработку в нашем проекте.

В проекте WMP метамодель и модель хранятся соответственно в базе данных метамоделей и базе данных диаграмм и доступны в виде Java-объектов. В MTCLauncher метамодель должна передаваться как файл Ecore, а модель – XMI. В связи с этим необходимо было реализовать генерацию файлов данных форматов из Java-объектов.

5.2. Интеграция в WMP

Следующим этапом стало добавление полученной в прошлом пункте функциональности в проект WMP. Для этого в микросервисную архитектуру проекта был добавлен новый сервис – Acceleo Generator, в котором содержится функциональность MTCLauncher, а также осуществляется создание необходимых для MTCLauncher файлов по метамодели и модели, которые передаются в данный сервис из сервиса редактора диаграмм при помощи RPC (рис.11).

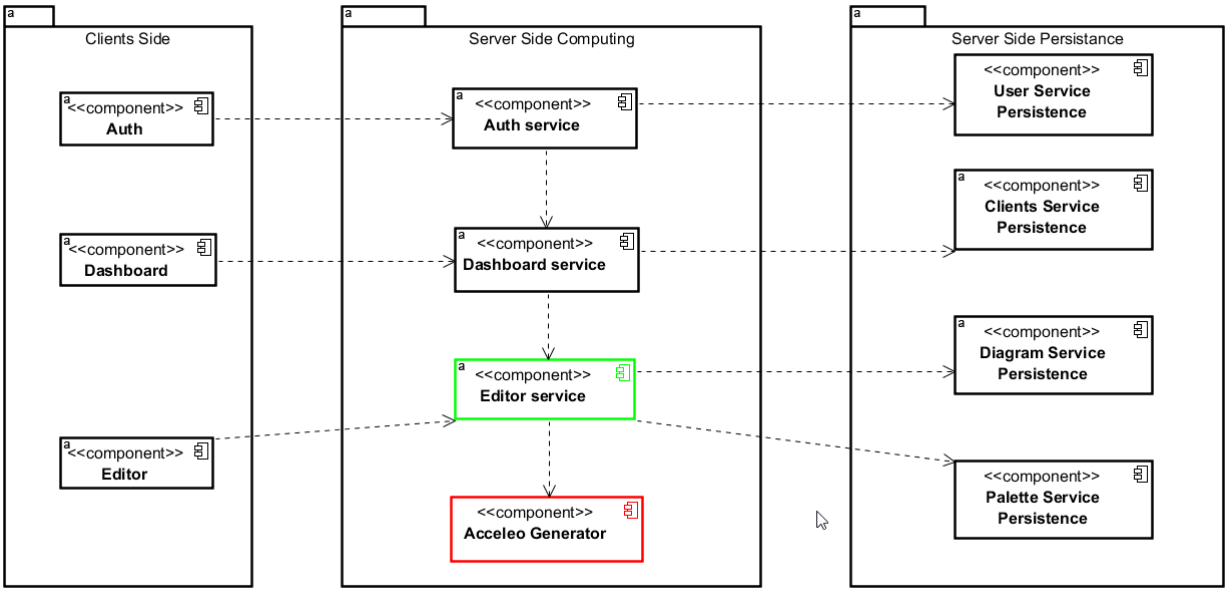

Рис. 11: Добавление сервиса генерации

При создании нового языка в сервис Acceleo Generator из Editor

Service отправляется информация об этом новом языке, а именно: название, элементы этого языка и свойства этих элементов. В данном сервисе создается новый файл Ecore с именем созданного языка, в который заносится полученная информация об этом языке.

Далее пользователь рисует модель с помощью созданного им языка. Если он хочет сгенерировать по этой модели код, то ему предлагается сохранить свою диаграмму, затем информация о модели отправляется на сервис Acceleo Generator, там она сохраняется в виде файла XMI с именем этой модели.

Таким образом у нас есть три файла, необходимые для генерации – метамодель, модель и правила генерации, которые на данный момент лежат отдельным файлом в проекте и не могут быть заданы из браузера. По этим трем файлам с помощью MTCLauncher генерируются нужные файлы, которые на данный момент сохраняются в проекте в папке gen. В будущем планируется передавать сгенерированные файлы пользователю.

Заключение

В рамках курсовой работы было выполнено следующее.

1. В клиентской части проекта WMP отделено ядро, реализующее работу с абстрактными диаграммами от кода, связанного с роботами.

2. В существующую архитектуру проекта WMP интегрирован метаредактор, реализованный в рамках курсовой второго курса.

3. Сделан обзор существующих решений.

4. В качестве инструмента для реализации генерации был выбран Acceleo.

5. С помощью MTCLauncher, использующего Acceleo, реализована генерация кода.

6. Генерация интегрирована в проект WMP, но еще не в основную ветку разработки.

Возможные варианты развития:

- передача сгенерированных файлов пользователю;
- развитие функциональности метаредактора:
  - возможность задавать типы свойств (на данный момент доступен только строковый тип);
  - возможность создавать разные типы связей между элементами;
- генерация кода по нелинейным диаграммам;
- сохранение моделей и метамоделей в базе данных, вместо создания новых файлов.

Список литературы

[1] Acceleo tutorial. — URL: https://wiki.eclipse.org/Acceleo/Getting_Started (online; accessed: 20.05.2017).

[2] Guana Victor. Running Acceleo and ATL Transformations Programmatically. — 2016. — URL: http://victorguana.blogspot.ru/2016/05/running-acceleo-and-atl-transformations.html (online; accessed: 20.05.2017).

[3] JET tutorial. — URL: https://eclipse.org/articles/Article-JET/jet_tutorial1.html (online; accessed: 20.05.2017).

[4] А.В. Подкопаев. Средства описания генераторов кода для предметно-ориентированных решений в metaCASE-средстве QReal. — 2012. — URL: http://se.math.spbu.ru/SE/YearlyProjects/2012/YearlyProjects/2012/345/345_Podkopaev_report.pdf (дата обращения: 20.12.2016).

[5] А.О. Дерипаска. Средства задания правил генерации в QReal. — 2014. — URL: http://se.math.spbu.ru/SE/diploma/2014/s/DeripaskaAnna_Diploma.pdf (дата обращения: 20.12.2016).

[6] Домашняя страница Acceleo. — URL: https://www.eclipse.org/acceleo/ (online; accessed: 20.05.2017).

[7] Л.А. Черниговская Ю.В. Литвинов. Разработка онлайн-метаредактора QReal-Web // Современные технологии в теории и практике программирования. — 2016. — С. 29–31.

[8] М.В. Тихонова. Генерация кода в режиме "метамоделирования на лету" в системе QReal. — 2015. — URL: http://se.math.spbu.ru/SE/diploma/2015/bmo/444-Tikhonova-report.pdf (дата обращения: 20.12.2016).