Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный университет»

Кафедра системного программирования

Храмышкина Юлия Сергеевна

Поддержка диаграммы классов языка UML 2.5 в QReal

Курсовая работа

Научный руководитель:
ст.преп. кафедры СП Литвинов Ю.В.

Санкт-Петербург
2016

Содержание

Введение

При разработке программного обеспечения часто требуется значительное количество времени на проектирование и анализ требований к будущему продукту. В частности, для того, чтобы непосредственно реализовать придуманное, существуют различные языки программирования и другие вспомогательные инструменты, которые не подходят для этапов проектирования и анализа. Часто требуется выделять наиболее важную информацию о разрабатываемой системе и работать непосредственно с ней, не углубляясь в детали реализации. Поэтому вводят понятие модели системы, суть которого в том, что выделяется наиболее важная информация о системе, не содержащая лишних деталей.

Наиболее наглядными и удобными для восприятия являются визуальные модели, которые часто представляются в виде диаграмм, что позволяет разработчикам быстро понять/объяснить друг другу какую-либо идею или решение. Однако часто овладение навыками визуального моделирования вызывает сложности.

Поэтому стоит отметить, что обучение визуальному моделированию вызывает интерес у исследователей. Так, например, в работе [1] был поставлен масштабный эксперимент, доказавший, что студенты, освоившие визуальное моделирование, успешнее остальных справляются с выполнением поставленных заданий.

Одним из самых распространенных сейчас визуальных языков является унифицированный язык моделирования (UML). Сейчас в нем существует 14 видов диаграмм. Наиболее широко известной из них является диаграмма классов UML.

На кафедре системного программирования СПбГУ существует научно-исследовательский проект QReal, который представляет собой систему, поддерживающую создание своих собственных предметно-ориентированных визуальных языков. Причем это некоммерческий проект с открытым исходным кодом. В свою очередь, UML — это визуальный язык, и поэтому его можно поддержать в QReal для того, чтобы использовать для обучения студентов визуальному моделированию.

Целью данной курсовой работы является поддержка диаграммы классов UML 2.5 в среде QReal. Для реализации поставленной цели предстоит решить следующие задачи:

- изучить спецификацию UML 2.5;
- проанализировать другие программные средства, поддерживающих UML;
- усовершенствовать ядро QReal и его графическую подсистему для удобства работы в QReal и его работоспособности в целом;
- реализовать диаграмму классов UML 2.5 в соответствии со спецификацией.

Обзор

UML 2.5

UML (Unified Modeling Language) — язык, представляющий собой семейство нотаций, основанное на единой метамодели. С его помощью можно описывать и проектировать программные системы, в частности те, в построении которых используются объектно-ориентированные технологии. UML является открытым стандартом, который находится под управлением OMG (Object Management Group). Он служит, например, для упрощения обмена информацией об аспектах системы между её разработчиками.

Этот язык был разработан в 1996 году и с тех пор в нем произошло большое количество изменений. На данный момент актуальной версией этого языка является UML 2.5, опубликованная в июне 2015 года. Сам язык UML 2.5 подробно изложен в соответствующей спецификации, на основе которой в рамках данной работы и будет реализована диаграмма классов.

Для того, чтобы выявить наиболее приоритетные направления в усовершенствовании ядра QReal, а также перенять опыт и оценить достоинства и недостатки других систем, поддерживающих создание UML-диаграмм, в рамках данной работы были рассмотрены следующие средства.

Visual Paradigm for UML

Visual Paradigm for UML (Visual Paradigm Int'l Ltd.) — кросс-платформенное средство для создания диаграмм UML версии 2.0. Включает в себя возможность кодогенерации в такие языки, как Java, C#, C++ и некоторые другие. Поддерживает 13 типов диаграмм UML. Исходный код закрыт, есть возможность бесплатного использования: Free Community Edition.

Особенности: возможность указывать свойства на концах каждой связи, n-арные связи, громоздкий интерфейс.

UMLet

UMLet (The UMLet Team) — средство для создания UML-диаграмм, которое можно использовать на следующих платформах: Windows, OS X, Linux. Данное средство не поддерживает создание UML-диаграмм версий 2.x, однако имеет открытый исходный код и возможность бесплатного использования: GPL. Сам инструмент предназначен для преподавания UML и для быстрого создания диаграмм, однако не является инструментом для моделирования, так как не имеет необходимой базы и каталогов объектов. Поддерживает шесть основных видов UML-диаграмм, включая диаграмму классов.

Особенности: неудобный интерфейс, неаккуратно нарисованные классы, связи между ними, надписи на связях часто перекрывают сами связи.

StarUML

StarUML (MKLab) — средство для создания UML-диаграмм, с которым можно работать на следующих платформах: Windows, OS X, Linux. Данный инструмент поддерживает создание большинства типов UML-диаграмм версий 2.0. Также StarUML поддерживает генерацию кода в Java, C#, C++. Однако его исходный код закрыт и этот инструмент не является бесплатным.

Особенности: возможность указывать свойства на концах каждой связи, древовидное представление свойств, удобный интерфейс.

ArgoUML

ArgoUML (Tigris.org) — кросс-платформенное средство для создания UML-диаграмм. Данный инструмент не поддерживает создание UML-диаграмм версий 2.x. В нем можно создавать девять типов диаграмм UML версии 1.4. Этот продукт является бесплатным, и его исходный код открыт. Также он поддерживает генерацию кода в C++, C#, Java и некоторые другие языки.

Особенности: неаккуратно нарисованные классы и неаккуратное отображение надписей на связях.

Eclipse Papyrus

Eclipse Papyrus — средство для создания UML-диаграмм, с которым можно работать на платформах Windows, Linux. Данное средство базируется на среде Eclipse, поддерживает версии UML 2.x, обладает открытым исходным кодом. Может быть использован как самостоятельное средство или как плагин к Eclipse. Поддерживает генерацию кода в C/C++, Java.

Особенности: громоздкий интерфейс, красиво отрисованные классы, аккуратные подписи на связях

Microsoft Visio

Microsoft Visio (Microsoft) — средство для создания UML-диаграмм, предназначенное для использования на платформе Windows. Подходит только для коммерческого использования, исходный код закрыт.

Особенности: привычный для многих интерфейс, отсутствие n-арный связей.

Enterprise Architect

Enterprise Architect (Sparx Systems) — средство для создания UML-диаграмм, предназначенное для использования на платформах Windows, Linux, Mac OS. Поддерживает UML 2.5 и генерацию кода в Java, C#, C++ и некоторые другие языки. Однако исходный код данного средства закрыт и оно не является бесплатным.

Особенности: древовидное представление свойств, возможность указывать свойства на концах каждой связи, приятный, но громоздкий интерфейс.

UModel

UModel (Altova) — средство для создания UML-диаграмм, предназначенное для использования на платформе Windows. Поддерживает генерацию кода в Java, C#, Visual Basic. Подходит только для коммерческого использования, исходный код закрыт. На данный момент поддерживает создание 14-ти видов UML-диаграмм версии 2.0.

Особенности: отсутствие древовидного представления свойств, неудобный интерфейс.

GenMyModel

GenMyModel — онлайн-средство для создания UML-диаграмм. Поддерживает создание пяти видов UML-диаграмм, включая диаграмму классов версии 2.0. Существует возможность бесплатной работы с этим средством. Также можно генерировать код в Java, SQL.

Особенности: возможность указывать свойства на концах каждой связи, аккуратный внешний вид и приятный интерфейс.

QReal

Как уже упоминалось ранее, визуальное программирование упрощает разработку программного обеспечения и делает ее более наглядной. Для того, чтобы использовать этот подход, требуются специализированные средства, называемые CASE-системами, однако для упрощения создания CASE-систем используются metaCASE-системы, благодаря которым можно автоматически генерировать CASE-системы по их формальному описанию.

QReal, в свою очередь, является metaCASE-системой, которая позволяет пользователям создавать свои собственные редакторы визуальных языков. Для этого используется метаязык — язык для описания всех других языков, и метаредактор — редактор, с помощью которого можно создавать визуальные языки на основе метаязыка. Он предоставляет для моделируемого языка такие элементы, как, например, сущности и связи (Node, Edge), являющиеся сущностями самого метаязыка, а также он предоставляет возможность устанавливать между сущностями различные отношения, например, отношения наследования. Также у элементов создаваемого языка можно задавать различные свойства, такие как name, displayedName, shape, defaultValue, которые означают имя, имя, отображаемое на палитре, форму и значение по умолчанию для данного свойства соответственно.

После создания модели нового визуального языка для последующей работы с ним необходимо сгенерировать код, отвечающий за редактор визуального языка.

На данный момент в QReal это можно сделать двумя способами: с использованием промежуточного xml-описания (QReal Xml Compiler) или же сразу сгенерировать код по метамодели требуемого визуального языка (QReal Metamodel Compiler). Стоит отметить, что существует еще и интерпретативный способ создания визуальных языков, который позволяет изменять визуальный язык непосредственно во время работы с ним, однако этот способ больше подходит для экспериментов с создаваемым языком, когда еще неизвестно, что именно от него будет требоваться.

Что касается архитектуры архитектуры самого проекта, она имеет довольно сложное устройство.

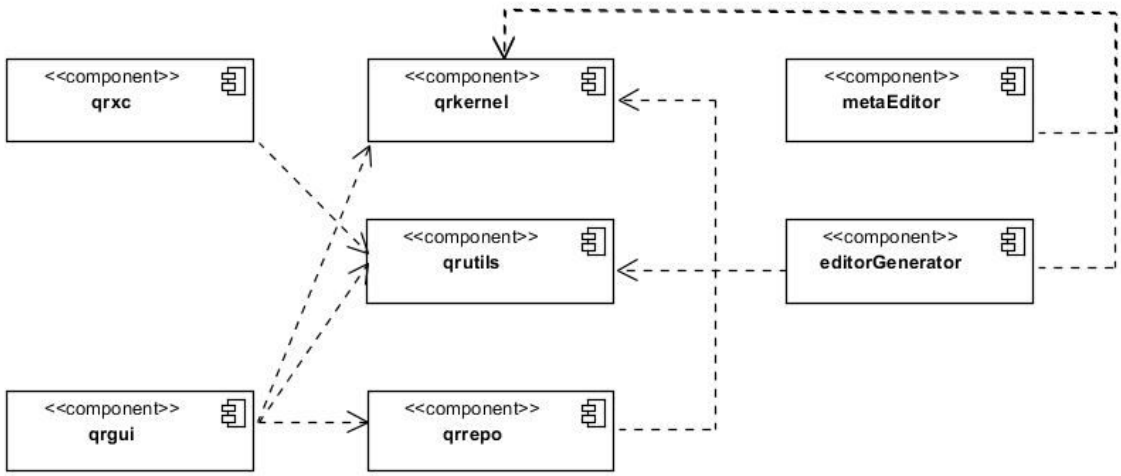

Рис. 1. Диаграмма компонентов QReal

На Рис. 1 представлена диаграмма компонентов среды QReal, а именно — тех ее частей, взаимодействие с которыми частично или косвенно производилось в рамках данной курсовой работы. Стоит отметить, что в связи с особенностями архитектуры данного проекта для того, чтобы поддержать в нем ту или иную функциональность часто требуется поддержать ее сразу в нескольких частях системы.

Реализация

В результате анализа других программных средств, поддерживающих создание UML-диаграмм, были выявлены некоторые ключевые моменты, реализация которых необходима для создания диаграммы классов UML.

Поддержка ролей

В частности, для корректного отображения UML-диаграмм требуется, чтобы у каждой связи, которая соединяет узлы, на каждый из концов можно было бы добавлять надписи со значениями соответствующих свойств (например, кратность, ограничения). В связи с тем, что подобная функциональность в QReal отсутствовала, было принято решение это исправить.

Для этого была реализована новая сущность “роль”. Сначала она была поддержана в метаредакторе и также из него были удалены “ассоциации”, которые прежде отвечали за иллюстрацию свойства у связи. Идейно “роль” представляет из себя неграфический тип, у которого может быть какой-либо набор свойств, а также у ролей можно указывать то, как именно будет рисоваться связь (поля arrowType, end, navigability). В свою очередь, связи (edge), для которых роли и были созданы, имеют два свойства, ссылающиеся на роли: beginRole, endRole (Рис. 2).

```
<role name="role1" arrowType="filled_arrow">
    <properties>
        <property displayedName="propertyForInt" type="int" name="property1">
            <default>1</default>
        </property>
        <property displayedName="propertyForString" type="string" name="property2">
            <default>hello</default>
        </property>
    </properties>
</role>
...
<edge name="Test">
    <logic>
        <beginRole role="role1"/>
        <endRole role="role2"/>
    </logic>
</edge>
```

Рис. 2. Роли в метамодели визуального языка

Для осуществления корректной работы ролей они были также поддержаны в части кода, отвечающей за генерацию и сборку редакторов — QReal Xml Compiler и EditorGenerator.

Принцип их работы заключается в следующем:

- с помощью EditorGenerator по модели, созданной в метаредакторе, генерируется XML-представление метамодели нового визуального языка;
- из полученной метамодели извлекается информация и передается в QReal Xml Compiler;
- на основе извлеченной информации автоматически генерируется код на C++;
- после чего код компилируется, и в QReal загружается новый редактор.

В частности, требовалась поддержка на этапе получения метамодели нового визуального языка в формате XML. Далее, требовалось поддержать разбор содержимого полученной метамодели, а после этого сгенерировать код самого редактора.

После этого этапа потребовалось поддержать их непосредственно в ядре QReal для того, чтобы при работе с получившимся редактором информация, получаемая из сгенерированного кода, была доступной и корректной. Так, например, требуется обращаться к сгенерированному коду за тем, чтобы узнать, какие имена у различных свойств данной связи.

Поддержка множественных надписей у связей

После поддержки ролей потребовалось осуществить поддержку множественных надписей у связей для того, чтобы свойства связей могли отображаться на своих концах, как изначально и задумывалось.

Для этого потребовалось поддержать их взаимодействие с ролями в ядре и коде, отвечающем за генерацию и сборку редакторов.

Поддержка древовидного представления свойств элементов

После проделанной работы и анализа полученных результатов была выявлена следующая проблема: отображение свойств связей в редакторе свойств стало слишком громоздким, поскольку каждое свойство стало состоять из имени соответствующей роли, знака разделителя и непосредственного имени конкретного свойства.

В связи с этим было принято решение поддержать древовидное представление для свойств, поскольку предыдущее (табличное) оказывается слишком громоздким (Рис. 3).

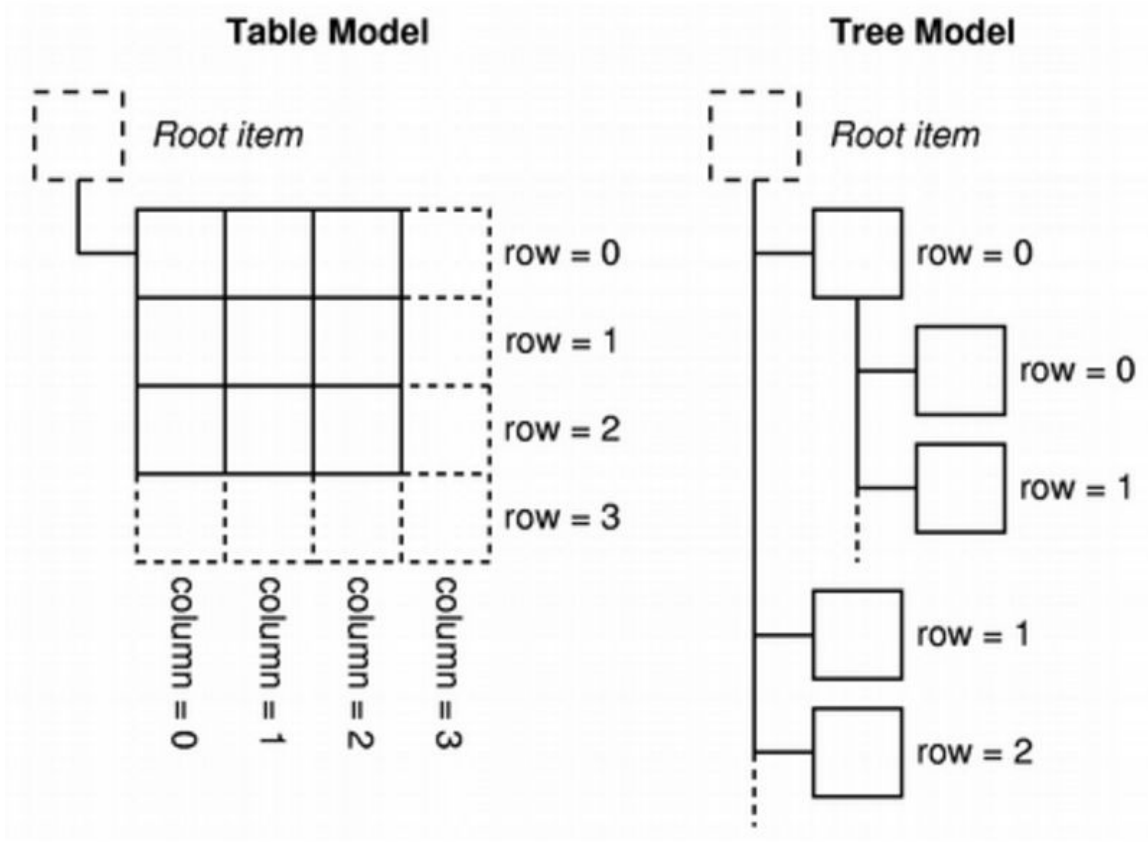

Рис. 3 Варианты организации модели редактора свойств (из [2])

Для этого в ядре системы были внесены изменения в код, отвечающий за редактор свойств. Если говорить подробнее, то у редактора свойств существуют свои model и view, благодаря которым происходит отображение информации из графической и логической моделей и ее изменения в случае необходимости.

Ранее, как уже говорилось выше, model и view были устроены так, что можно было отображать свойства только в виде двумерной таблицы, причем в виде подряд идущих пунктов списка без возможности группировки.

В итоге, это было изменено и реализована древовидная модель свойств, благодаря которой в редакторе свойств, который видит пользователь, свойства, имеющие одинаковое общее имя, группируются, и при желании их можно свернуть или развернуть. (Рис. 4)

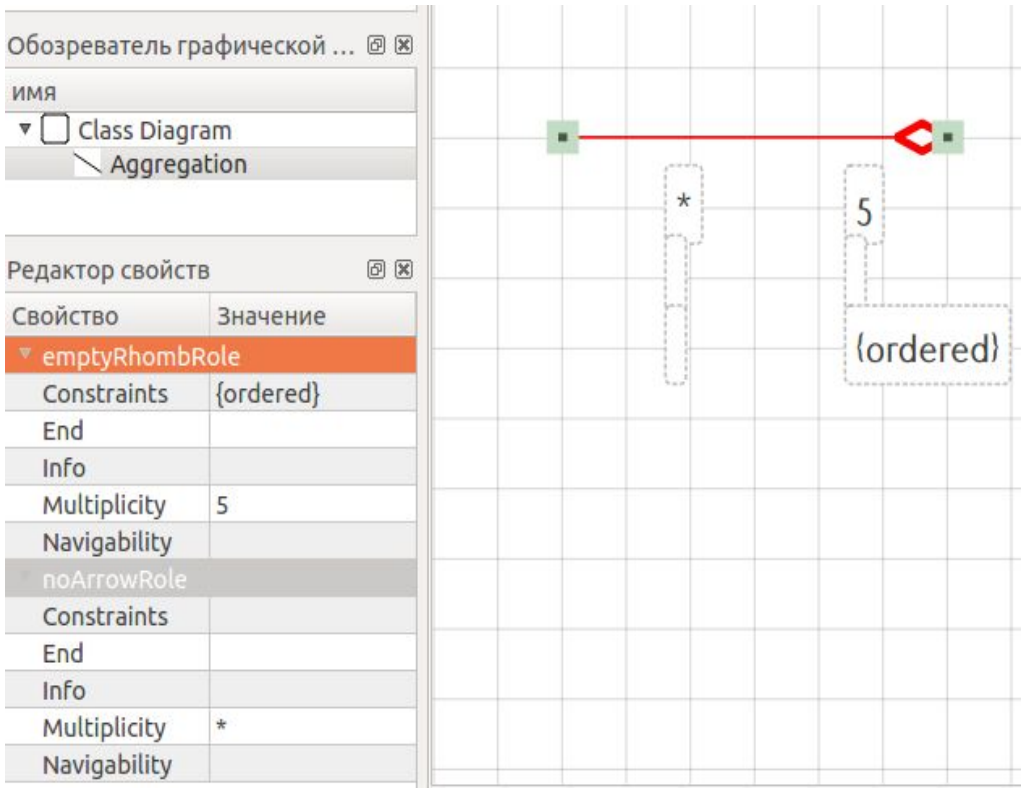

Рис. 4

Создание редактора диаграммы классов

После реализации всех этих пунктов был создан редактор диаграммы классов UML. (Рис. 5)

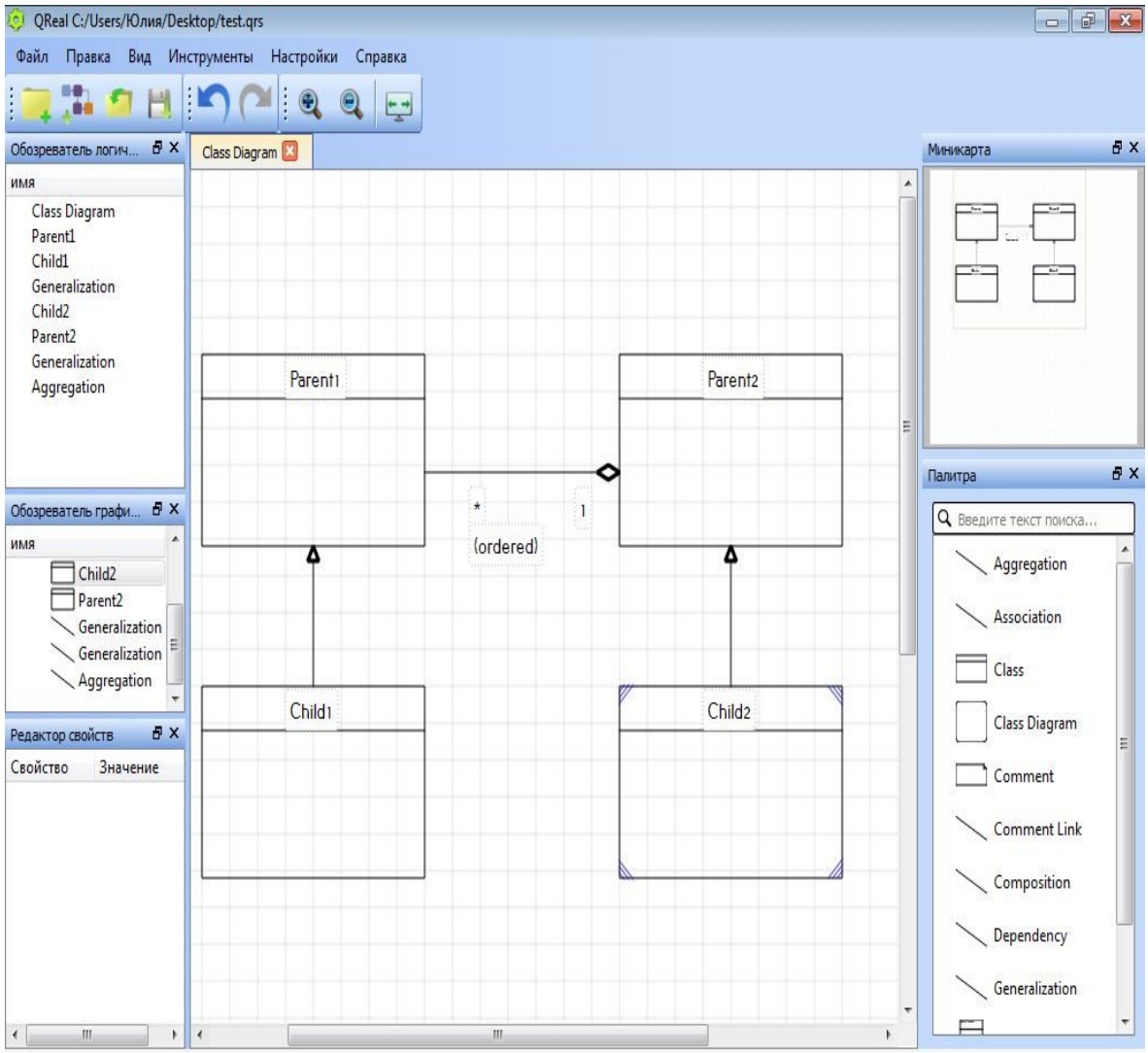
QReal C:/Users/Юлия/Desktop/test.qrs
Файл
Правка
Вид
Инструменты
Настройки
Справка
Обозреватель логич...
имя
Class Diagram
Parent1
Child1
Generalization
Child2
Parent2
Generalization
Aggregation
Обозреватель графи...
имя
Child2
Parent2
Generalization
Generalization
Aggregation
Редактор свойств
Свойство
Значение
Class Diagram
Parent1
Parent2
*
1
{ordered}
Child1
Child2
Миникарта
Палитра
Введите текст поиска...
Aggregation
Association
Class
Class Diagram
Comment
Comment Link
Composition
Dependency
Generalization

Рис. 5

Апробация

В рамках данной работы после создания редактора диаграммы классов UML были выполнены измерения субъективного восприятия редактора пользователями по шкале System Usability Scale (SUS) для того, чтобы иметь данные для проведения дальнейшей работы над ним.

Апробация проводилась на пяти студентах. Им было дано задание: “Нарисовать диаграмму классов UML для хэш-таблицы со сменными хэш-функциями” (Рис. 6). После чего они заполнили анкету SUS, состоящую из 10 вопросов, каждый из которых оценивался по 5-балльной шкале, а затем полученные результаты переводились в 100-балльную шкалу. Опросник SUS предполагает, что приложение, набравшее 68 баллов, считается удовлетворительным. Результаты данного опроса таковы, что средний балл данного редактора оказался равным 80-ти баллам (Таблица 1).

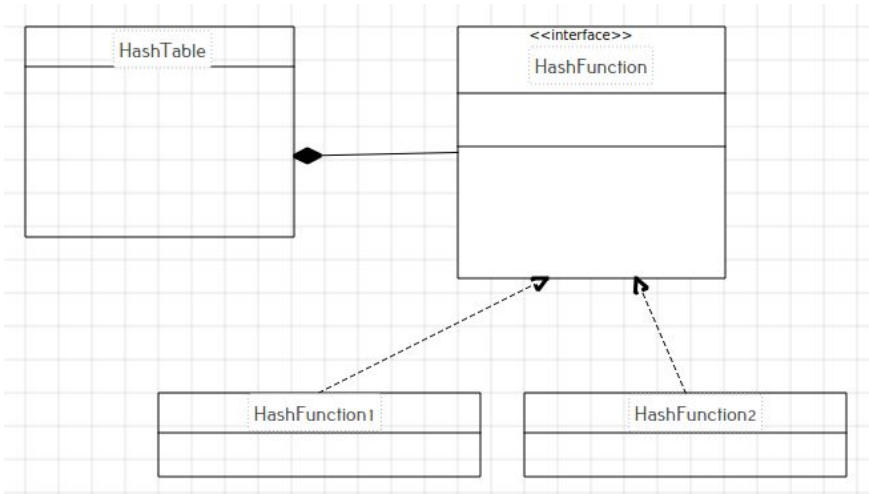

Рис. 6. Результат выполненного задания

	№1	№2	№3	№4	№5
SUS Score	80	82,5	77,5	75	80

Таблица 1

Заключение

В рамках данной курсовой работы были получены следующие результаты:

- проанализированы различные программные средства, поддерживающие создание UML-диаграмм и выявлены ключевые особенности;
- поддержана новая сущность для концов связей;
- поддержано отображение нескольких надписей у каждого из концов связей;
- поддержано древовидное свойств элементов;
- создан редактор диаграммы классов языка UML 2.5;
- проведена апробация.

Список литературы

[1] Lano K., Yassipour-Tehrani S., Alfraihi H. Experiences of Teaching Model-based Development, 2015.

[2] Qt Documentation. URL: http://doc.qt.io/qt-4.8/modelview.html (дата обращения: 22.05.2016)

[3] Фаулер, М. UML. Основы. Третье издание. / М. Фаулер. – М.: Символ-Плюс, 2006. – 192 с.

[4] Unified Modeling Language (UML), v2.5. Release date: March 2015. URL: http://www.omg.org/spec/UML/2.5/ (дата обращения: 29.09.2015)

[5] Терехов А.Н., Брыксин Т.А., Литвинов Ю. В., Смирнов К.К., Никандров Г.А., Иванов В.Ю., Такун Е.И. Архитектура среды визуального моделирования QReal. Системное программирование. Т. 4. СПб.: Изд-во СПбГУ. 2000. С. 165–169.

[6] Тарасова П.М., Храмышкина Ю.С., Литвинов Ю.В. Сравнение способов получения редакторов по метамодели и скорости их работы // Материалы научно-практической конференции студентов, аспирантов и молодых учёных "Современные технологии в теории и практике программирования". СПб.: Изд-во Политехн. ун-та, 2015. С. 79-80.

[7] Храмышкина Ю.С., Литвинов Ю.В. Поддержка диаграммы классов языка UML 2.5 в среде QReal // Материалы научно-практической конференции студентов, аспирантов и молодых учёных "Современные технологии в теории и практике программирования". СПб.: Изд-во Политехн. ун-та, 2016. С. 86-87.