САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

Соковикова Наталья Алексеевна

Реализация коллаборативной разработки в QReal

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

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

Рецензент:
ст. преп. Журавлев М. М.

Санкт-Петербург
2014

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

Natalia Sokovikova

Implementation of collaborative development in QReal

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Andrey Terekhov

Scientific supervisor:
senior lect. Yurii Litvinov

Reviewer:
senior lect. Maksim Zhuravlev

Saint-Petersburg
2014

Оглавление

Введение **4**

Постановка задачи **7**

1. Обзор существующих решений **8**

1.1. Cacoo 9

1.2. Creately 10

1.3. LucidChart 11

1.4. Сравнение 13

2. Архитектура QReal **14**

3. Обеспечение синхронности изменения данных **16**

3.1. Обеспечение сетевого взаимодействия 16

3.1.1. Буферизация 17

3.2. Протокол передачи данных 17

3.3. Синхронизация данных 19

3.4. Механизм разрешения конфликтов 19

4. Отображение изменений в модели **21**

5. Апробация новой функциональности **22**

6. Обсуждение **23**

Заключение **24**

Введение

Появление Интернета открыло для человечества множество новых возможностей. Помимо прочего, появились системы контроля версий, которые позволили разработчикам программного обеспечения работать над одним проектом с разных устройств, а затем проводить слияние всех изменений. В дальнейшем развитие технологий дало возможность изменять проект одновременно с другими пользователями, в режиме онлайн, и видеть все вносимые ими изменения. Такое взаимодействие называется коллаборативной разработкой.

Когда нет возможности общаться с партнерами вживую, каждый из участников подстраивает свою работу под свои представления о том, каков должен быть результат. В конечном итоге могут возникнуть проблемы с тем, чтобы совместить части работы, выполненные разными людьми. На решение этой проблемы уйдет дополнительное время, что может быть критично. Чтобы избежать подобной нерациональной траты времени, партнеры вынуждены постоянно поддерживать связь, задействуя различные средства, от мобильных телефонов, до программ, их заменяющих и позволяющих показывать экран собеседнику (Skype, TeamViewer, и т.д.).

Постоянные попытки скоординировать деятельность так же отнимают значительное количество времени и сил. К тому же, даже видя экран партнера, не всегда можно внятно и быстро объяснить ему, на что бы, на ваш взгляд, следовало обратить внимание. На такой случай существуют программы, предоставляющие доступ к удаленному компьютеру (TeamViewer, Microsoft Lync, и т.д.). Имея такую функциональность, вполне можно синхронизировать работу над некоторыми проектами.

Однако, все перечисленные выше решения предполагают, что в каждый момент времени лишь один из участников может совершать какие-то действия, а остальные вынуждены только смотреть и, возможно, разговаривать с этим единственным или общаться между собой. Несомненно, эти “остальные” могли бы более эффективно использовать свое время. Кроме того, разрабатываемое решение хранится на устройстве лишь одного из собеседников. В лучшем случае, остальные могут скачать обновленную версию из сети.

Тут следует рассказать о том, что представляет собой коллаборативная разработка, и почему она так удобна. Имея программу, поддерживающую режим совместной

разработки, те же люди могли бы закончить проект намного быстрее, так как они все могли бы одновременно редактировать его, видеть, чем заняты остальные и обсуждать сложные моменты с другими. Более того, изменения в проекте появлялись бы сразу у всех его участников. Таким образом, коллаборативная разработка позволяет людям, находящимся далеко друг от друга, максимально удобно и эффективно организовать совместную работу.

При разработке программных продуктов активно используются системы контроля версий. Как известно, при слиянии разных веток даже малейшие конфликты в коде приводят к необходимости разрешать их вручную. Коллаборативная разработка не может заменить всю функциональность системы контроля версий, однако с ее помощью можно было бы избежать некоторых задач, воникающих из-за недостаточной осведомленности о действиях других участников проекта. Поддержку коллаборативной разработки было бы удобно иметь во всех средах программирования.

Одной из таких сред является QReal [7] - metaCASE система, разрабатываемая на кафедре системного программирования СПбГУ. QReal предназначен для создания специализированных языков визуального программирования [8]. Система ориентирована на удобство программиста, разбирающегося в предметной области [4], и повышение эффективности его работы. В частности, на основе QReal была создана среда для программирования роботов QReal:Robots, которая теперь используется в некоторых российских школах для обучения детей основам программирования.

В QReal возможность совместной работы над задачей была бы особенно удобна, так как наряду с наглядностью визуальных языков программирования и возможностью создавать для каждого конкретного класса задач свой язык и среду программирования, она предоставила бы возможность эффективно организовать командную разработку проекта, не тратя дополнительные усилия на организацию взаимодействия между разработчиками. Кроме того, реализация поддержки коллаборативной разработки в ядре QReal позволит использовать коллаборативность во всех средах программирования, созданных на его основе. В частности, такая функциональность позволила бы ученикам и учителям совместно проводить разработку программ для управления роботами в QReal:Robots.

Существует множество редакторов визуальных языков, поддерживающих коллаборативную разработку, однако, они поддерживают лишь стандартные языки, не поз-

воляют создавать новые. Именно возможность создания специализированных языков, подходящих под предметную область задачи, является важной частью функциональности QReal. Например, язык программирования роботов сильно отличается от обычного UML. Из вышесказанного можно сделать вывод, что для поддержки совместного решения задач, в которых было бы предпочтительно использование специализированных визуальных языков программирования, необходимо реализовать механизм коллаборативной разработки именно в QReal.

Постановка задачи

Целью данной работы является реализация коллаборативного режима в системе QReal. Для достижения этой цели были сформулированы следующие задачи:

1. провести обзор существующих решений в области коллаборативной разработки UML диаграм;
2. обеспечить синхронность изменения моделей, в том числе разработать протокол общения нескольких экземпляров QReal;
3. поддержать отображение изменений, выполняемых другими пользователями;
4. предоставить средства коммуникации пользователей;
5. провести апробацию реализованной функциональности.

1. Обзор существующих решений

Было принято решение реализовать поддержку коллаборативной разработки в QReal. Следовало, в первую очередь, составить более подробный список требований к механизму, ознакомившись с инструментами, в которых поддерживается коллаборативность. Для начала стоило выяснить, какие требования предъявляют пользователи к подобным программам, и насколько те соответствуют этим требованиям.

Обозначим круг программ, с поддержкой совместной разработки, подходящих для рассмотрения. Самым распространенным и часто используемым визуальным языком является UML, поэтому существует множество редакторов UML-диаграмм, и они более зрелые, чем полные аналоги QReal, особенно такие, которые поддерживают коллаборативную разработку. При этом, программирование в QReal принципиально не отличается от рисования UML-диаграмм. Поэтому для составления требований было решено исследовать редакторы UML-диаграмм с поддержкой коллаборативной разработки. Для дальнейшего рассмотрения были отобраны следующие редакторы:

- Cacoo;
- Creately;
- LucidChart.

После анализа нескольких источников было составлено описание функциональности, относящейся к коллаборативной разработке, на которую пользователи рассчитывают, имея дело с подобными программами. Следует упомянуть, что любая программа с поддержкой коллаборативной разработки должна предоставлять пользователям возможность разрабатывать диаграмму одновременно, причем изменения, вносимые каждым участником, должны отображаться в режиме реального времени у всех остальных.

Ниже представлены основные моменты, на которые обращают внимание пользователи UML-редакторов.

- Возможность продолжать работу при потере соединения, не утрачивая при этом изменения, сделанные во время работы оффлайн.

- Наличие дополнительных средств коммуникации, таких как чат или возможность комментирования частей диаграмм.
- Наличие механизма отображения информации о том, кто из пользователей занимается той или иной частью проекта, часто может быть полезно при выполнении задачи в команде. Он дает возможность обсудить интересующую подзадачу с человеком, непосредственно ею занимающимся, согласовать свои решения.
- Средство для просмотра истории изменений с их подробным описанием, включающим в себя дату и время изменения и имя сделавшего их пользователя.

Далее следовало узнать, насколько выбранные редакторы соответствуют выделенным требованиям, и как, с точки зрения пользователя, эти требования в них реализованы.

1.1. Cacoo

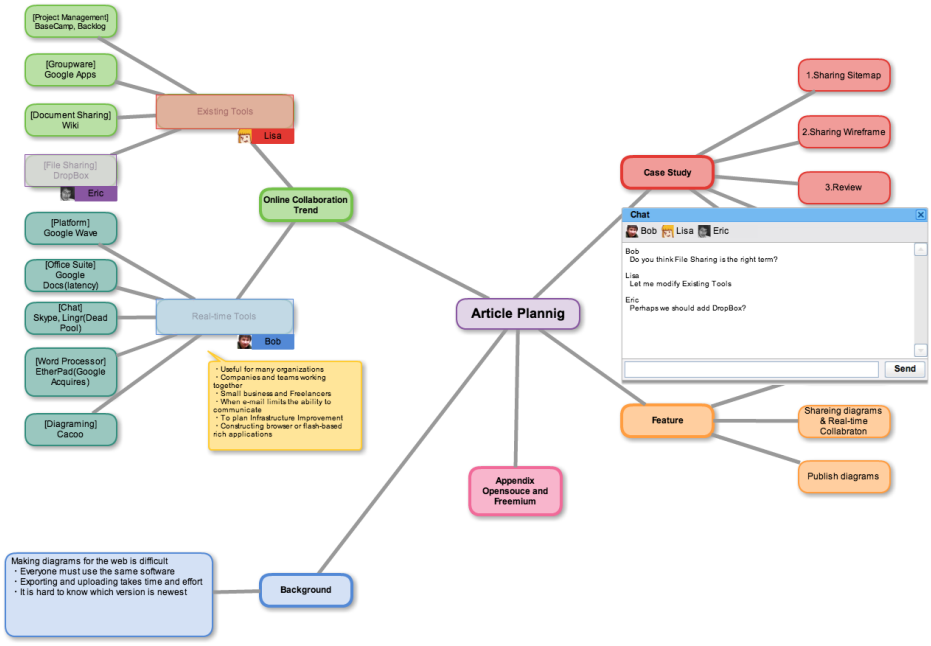

Рис. 1: Cacoo

Cacoo [1] - онлайн сервис для создания, совместного использования и публикации различных диаграмм, таких как карты сайта, каркасные схемы, UML-диаграммы и многие другие. В результате ознакомления с данным редактором, было составлено описание того, как выглядят в нем выделенные аспекты.

Все действия остальных пользователей без труда прослеживаются - когда кто-нибудь открывает редактируемую диаграмму или покидает ее, на экране появляется уведомление с соответствующей информацией и именем пользователя. В момент, когда другой участник начинает редактировать какой-нибудь элемент диаграммы, у остальных этот элемент выделяется цветной рамкой, около которой на том же фоне написано имя пользователя. Рамка сохраняется на протяжении всего времени редактирования.

В качестве средства коммуникации пользователей в Cacoo выступает чат. Окно чата отображается на экране, если диаграмма открыта одновременно у двух или более людей. В него по умолчанию добавляются все, открывшие диаграмму. Когда кто-то покидает диаграмму, его имя из чата удаляется.

Cacoo не поддерживает возможность работать с диаграммами оффлайн: при утрате сетевого соединения пользователь может продолжить редактирование диаграммы, однако, его достижения будут потеряны. Многие пользователи отмечают, что хотели бы своевременно видеть уведомления о временной пропаже связи.

Просмотр истории изменений, с возможностью восстановления состояния проекта в момент любого сохранения, доступен лишь в платной версии продукта

1.2. Creately

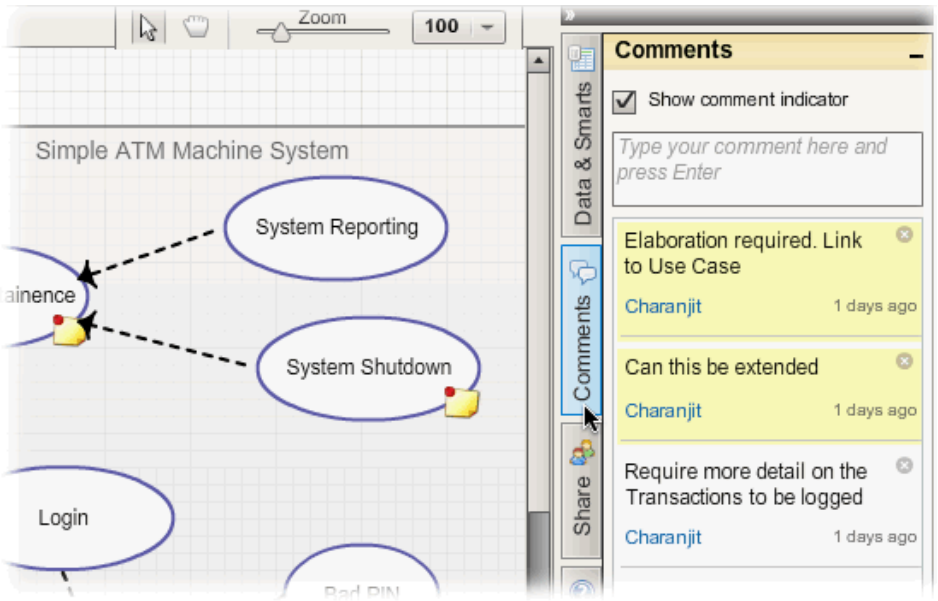

Рис. 2: Creately

Creately [2] — браузерное приложение для совместной работы с различными диаграммами. Среди диаграмм, которые можно создавать и редактировать в Creately, есть и UML.

Когда пользователь начинает редактировать элемент диаграммы, тот выделяется цветной рамкой, рядом с которой написано имя пользователя. При продолжении работы с элементом, имя исчезает. Для наглядности у каждого пользователя свой цвет рамки.

Система комментариев вполне заменяет отдельный чат. Здесь можно оставлять комментарии как об отдельном элементе, так и о части диаграммы.

Creately предупреждает пользователя о том, что соединение с сервером теряется, и некоторые функции могут быть недоступны. Существует также десктопная версия приложения, которая позволяет сохранить все сделанные изменения и при восстановлении соединения синхронизировать диаграмму. Однако в таком режиме коллаборативная разработка невозможна.

В этом редакторе можно просмотреть историю редактирования диаграммы.

1.3. LucidChart

LucidChart [3] - это визуальная онлайн программа для совместной работы над диаграммами разных видов, в том числе UML.

В качестве средства коммуникации предусмотрен чат, также им предоставляется возможность комментировать элементы диаграммы. Стоит отметить, что чат в LucidChart доступен, даже если диаграмма редактируется лишь одним пользователем.

Здесь нет отдельных уведомлений о прибавлении и уходе людей, поэтому, пока они не начали работу, можно узнать о появлении новых участников лишь заметив новую иконку на заголовке чата (аналогично, если пользователь покидает диаграмму, его имя из чата удаляется). Когда же человек начинает редактирование какого-нибудь элемента диаграммы, вокруг этого элемента рисуется цветная рамка с подписью имени.

При потере соединения данный редактор не уведомляет пользователя. Изменения, сделанные оффлайн, можно сохранить, скопировав получившуюся диаграмму и вста-

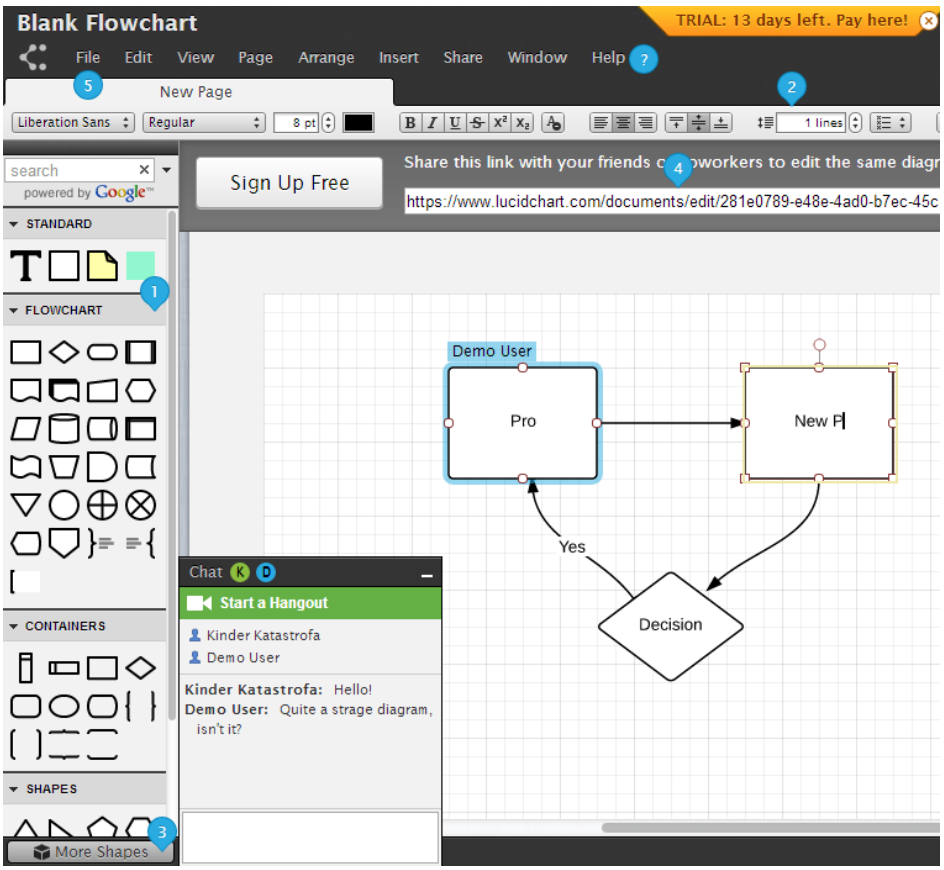

Рис. 3: Lucid Chart

вив ее в новое окно. Однако, если работа велась совместно с другими пользователями, такой способ не годится.

Как и предыдущие инструменты, LucidChart хранит историю изменений диаграммы.

1.4. Сравнение

	Cacoo	Creately	LucidChart
Возможность работы оффлайн	Нет	Нет	Нет
Общение разработчиков	Чат, комментарии	Комментарии	Чат, комментарии
Отображение изменений	Цветная рамка, имя пользователя	Цветная рамка, имя пользователя	Цветная рамка, имя пользователя
История изменений	Есть	Есть	Есть

Таблица 1: Обзор функциональности схожих инструментов

2. Архитектура QReal

Для описания деталей реализации коллаборативной разработки в QReal рассмотрим часть архитектуры системы [5], в которую требовалось внести изменения, а именно компоненту qrgui. Полная архитектура QReal представлена на Рис. reffig:GeneralScheme [9].

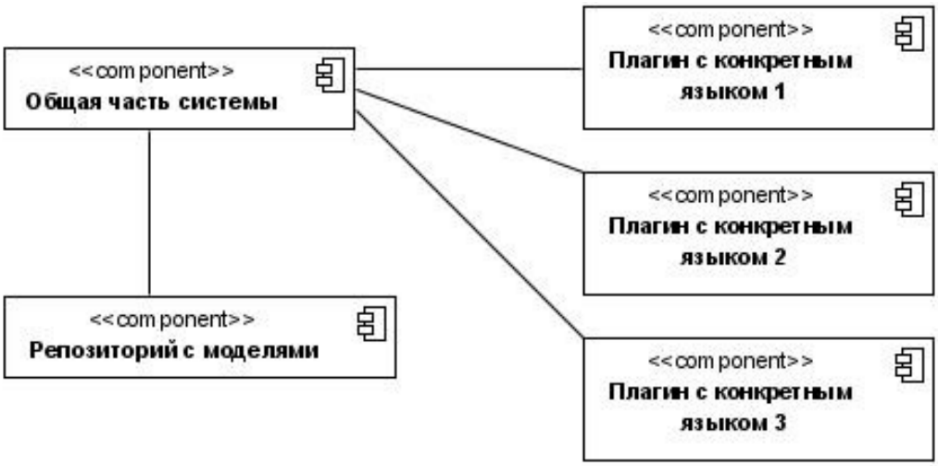

Рис. 4: Общая архитектура QReal

Все компоненты, которые могут отображать или модифицировать память, где хранятся данные, с которыми работает QReal, получают доступ к ней через компонент Models. Она реализует функциональность модели из стандартного Qt фреймворка Model/View. Этот класс отвечает за согласованность всех данных и их отображений и видов.

Все объекты, которые пользователь может видеть на диаграмме, являются элементами моделей. QReal разделяет понятия “элемент” и “представление элемента на диаграмме”. Первое хранится в так называемой логической модели, а второе — в графической. Модели связаны между собой, их данные синхронизируются.

Такое разделение сделано по той причине, что некоторые элементы могут вовсе не иметь графического представления (например, некоторые настройки), другие же, напротив, нужны на диаграмме, но не несут никакого логического значения, их логическое представление бесполезно, а третьи могут иметь как логическое представление, так и множество графических. В этом случае наличие раздельных моделей особенно удобно, так как позволяет связать все графические объекты лишь с одним логическим.

Логическая и графическая модели работают с одним и тем же репозиторием. Однако они получают доступ к нему через разные интерфейсы. Графической модели предоставляется интерфейс репозитория, содержащий методы для работы с чисто графическими свойствами объектов, в то время как из интерфейса для логической модели все такие методы убраны.

Все основные операции выполняются через компонент MainWindow, так как в нем хранятся ссылки на основные компоненты системы. При необходимости они запрашивают у MainWindow нужную для своей работы информацию.

Для того, чтобы создать новый язык, нужно, так или иначе, определить его абстрактный и конкретный синтаксис. Абстрактный синтаксис определяет, какие в языке есть элементы, свойства этих элементов и то, как они взаимодействуют между собой. Для визуальных языков абстрактный синтаксис чаще всего задается с помощью метамодели [10].

Конкретный синтаксис определяет, как будут отображаться элементы языка на диаграммах. На данном этапе разработчику предлагается либо создать изображение, либо использовать уже готовое.

Доступ к метамоделям предоставляется через интерфейс, в реализациях которого, помимо прочего, описаны методы для добавления, редактирования и удаления элементов языка. В QReal реализовано метамоделирование на лету, то есть язык можно модифицировать во время создания диаграммы.

Чтобы поддержать отображение изменений, выполняемых другими пользователями, было необходимо изменить методы, которые отвечают за отрисовку элементов на сцене и саму сцену. Классы, в которых находятся такие методы, расположены в компонентах umllib (здесь расположены классы, отвечающие за визуальное представление элементов - “сущностей” и “связей”) и view (эта компонента представляет сцену).

Чтобы встроить механизм коллаборативной разработки, было необходимо внести изменения в описанные выше классы.

На Рис. reffig:Scheme приведена схема описанной части архитектуры.

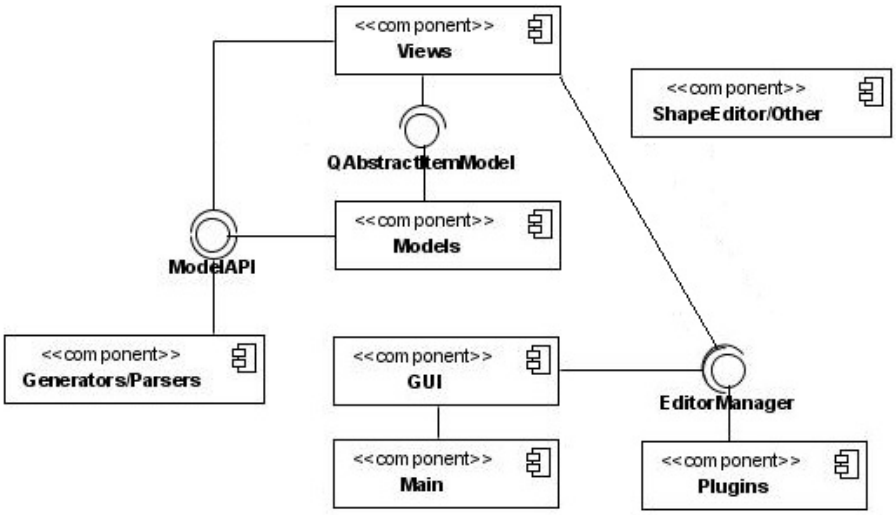

Рис. 5: Архитектура QReal

3. Обеспечение синхронности изменения данных

Задача обеспечения синхронизации изменения данных разбивается на несколько подзадач. В первую очередь, необходимо наладить соединение между несколькими экземплярами проекта. Затем, следует понять, какие данные один экземпляр должен отправлять другому, чтобы они находились в синхронизированном состоянии. Кроме того, нужно найти подходящий формат для передачи этих данных. Наконец, решение всех перечисленных выше задач будет бесполезно, если в QReal не будет реализована функциональность оповещения соответствующего модуля о необходимости отправки сообщения с подходящими параметрами. В дополнение к этому, необходим механизм, способный по пришедшему сообщению повторить изменения, происшедшие на экземпляре-отправителе.

3.1. Обеспечение сетевого взаимодействия

С целью реализации сетевого взаимодействия было создано приложение — сервер. Оно реализовано с использованием стандартных классов QTcpServer и QTcpSocket [6].Приложение реализует следующую функциональность:

- приниятие сообщений об основных изменениях диаграммы на каждом из подключенных экземпляров QReal и рассылка сообщений о них всем остальным подключенным клиентам;

- поддержка актуальных данных об элементах, которые в текущий момент редактируются на каждом из экземпляров;
- хранение истории изменений совместно разрабатываемой диаграммы;

Для того, чтобы поддерживать коммуникацию с приложением-сервером в компоненту Модель был встроен класс Client. Данный класс, имея ip-адрес сервера, подключается к нему. Далее, в этом классе имеются методы, формирующие сообщения об изменениях. Эти сообщения он отправляет серверу. Кроме того, он принимает уведомления об изменениях в других экземплярах и обрабатывает их. Данный класс реализован с применением стандартного Qt класса QTcpSocket.

По иготам анализа архитектуры системы было выяснено, что для поддержки синхронизированности данных нескольких экземпляров QReal, необходимо передавать сообщения о всех возможных изменениях моделей и метамоделей, а именно о добавлении, удалении и изменении элементов моделей (графической и логической) и метамодели, а также о создании новой диаграммы.

3.1.1. Буферизация

Для предотвращения неполной передачи данных был применен механизм буферизации. То есть при получении очередной порции данных, они записываются в буфер, а обрабатываются данные из буфера лишь после того, как его размер станет равным (или превысит) размер первого сообщения. После чего первое сообщение из буфера стирается.

Реализация данного механизма требовала учета размера передаваемого сообщения при разработке протокола передачи данных.

3.2. Протокол передачи данных

Данные об изменениях моделей и метамоделей было решено представлять в виде сообщения в обычном строчном формате. Такой выбор был частично мотивирован наличием в проекте методов генерации строкового представления для необходимых классов. Отказ от использования общепринятых методов представления данных в строковом формате (таких как json, xml) был обоснован малым размером передаваемых данных и простотой их структуры. В конечном итоге был использован следу-

ющий формат сообщений: в первых двух байтах указывался его размер. Затем сокращенное название функции, о вызове которой составлялось сообщение. Далее шли параметры функции, приведенные к строкам, и, в некоторых случаях, дополнительные необходимые данные. Все данные, за исключением размера и названия функции, разделяются "|".

```
<размер>senderFunctionName|param1|…|paramN|someExtraData|
```

Здесь `<размер>` — размер сообщения в символах, кодируемый двумя байтами. В качестве `senderFunctionName` могут выступать следующие сокращения:

addElem — соответствует методу для добавления элементов в логическую и графическую модели;

setData — метод для изменения свойств элементов диаграммы;

grElemRemoved — метод для удаления элементов из графической модели;

logElemRemoved — метод для удаления элементов из логической модели;

addNode — один из методов редактирования метамодели; отвечает за создание новой "сущности" в палитре;

addEdge — соответствует функции создания новой "связи" в метамодели языка;

updShape — изменений формы элементов палитры (QReal позволяет менять визуальное представление как "сущностей", так и "связей");

delElem — метод удаления составляющих языка;

addProp — добавление свойства некоторому элементу метамодели;

updProp — редактирование свойства некоторого элемента метамодели;

delProp — удаление свойства некоторого элемента метамодели;

diagrCr — создание диаграммы;

elemLocked — соответствует функции-запросу на установление или снятие блокировки;

chatMsg — передача текстовых сообщений чата;

3.3. Синхронизация данных

Процесс создания и обработки сообщений основан на механизме сигналов и слотов библиотеки Qt. При первом запуске QReal по умолчанию открывается как отдельная рабочая станция, без поддержки коллаборативности. Для использования режима совместной разработки, пользователю нужно выбрать соответствующий пункт в настройках и указать ip-адрес работающего сервера и имя, под которым его увидят другие участники проекта. Важную роль в механизме коллаборативной разработки играет компонент MainWindow. После установления соединения с сервером, MainWindow инстанциирует классы, участвующие в коллаборативной разработке, и соединяет необходимые сигналы с соответствующими слотами.

На экземпляре QReal происходят следующие события: функции из классов логической и графической моделей, а также одного из классов, реализующих интерфейс для метамодели, при вызове посылают сигнал со всеми параметрами, нужными для точного повторения этих действий в другом экземпляре QReal. Такой сигнал ловит один из слотов экземпляра класса Client. Функции-слоты приводят все свои параметры к строкам, составляют из них сообщения в описанном выше формате и отправляют их на сервер. При получении сообщения сервер действует в зависимости от типа этого сообщения (запрос на блокировку элемента или уведомление о происшедших изменениях). Получив сообщение о некоторых действиях на одном из клиентов, он рассылает его остальным подключенным экземплярам и записывает на диск. Экземпляр QReal, получая сообщение об изменениях, вносимых на других экземплярах, обрабатывает их следующим образом: выделяет название функции, которую нужно вызвать, и список ее параметров и посылает сигнал, на который реагируют подходящие слоты. Функция-слот повторяет действия, сделанные на другом клиенте, таким образом поддерживая данные нескольких экзепляров проекта синронизированными.

3.4. Механизм разрешения конфликтов

Для реализации механизма разрешения конфликтов на сервере и всех клиентах хранятся записи обо всех объектах модели, которые в данный момент редактируются кем-либо. Имея такую информацию, для того, чтобы предотвратить конфликты, можно запретить редактирование всех объектов, изменяемых другими пользователя-

ми.

Это осуществимо лишь при условии, что данные в локальной копии таблицы, хранящейся на каждом экземпляре QReal, синхронизированы с той, что хранится на сервере. Выполнение этого условия реализуется следующим образом: при попытке пользователя выбрать незаблокированный элемент диаграммы, отсылается сообщение на сервер с id этого элемента. В случае, если сервер, проверив локальную копию упомянутой таблицы, не находит в нем элемент с таким id, он вносит туда данный элемент, сопоставляя ему адрес и порт устройства, от которого получил запрос. Далее он рассылает всем клиентам сообщение о новой блокировке, и те обновляют свои локальные копии таблицы.

Таким образом, все клиенты узнают о блокировке данного элемента, а отправитель получает подтверждение того, что он успешно заблокировал элемент и может теперь изменять его.

Если два (или более) клиента одновременно посылают запрос на блокировку одного элемента, сервер подтверждает это действие для того из них, чье сообщение дойдет первым. Все остальные клиенты, подключенные к данному серверу, в том числе и те, кто претендовал на этот элемент, получают одинаковые уведомления о его блокировке.

Когда пользователь пытается редактировать элемент, проверяется наличие id этого элемента в локальной копии таблицы. Если в нем уже есть элемент с таким id, проверяется, совпадают ли адрес и порт с теми, что указаны в таблице. При совпадении, пользователю разрешается вносить в элемент изменения, иначе, он его не может выделить. Если же такого элемента в отображении нет, посылается описанный выше запрос на сервер о блокировке этого элемента.

4. Отображение изменений в модели

Помимо организации синхронного изменения моделей стояла задача реализации механизма для выделения изменений, вносимых другими пользователями.

Все элементы, отображаемые на экране, представляются двумя классами (классы условно соответствуют “сущностям” и “связям”). Для этих классов был создан новый флаг, который выставляется, если элемент редактируется каким-либо пользователем, кроме данного. Метод, отвечающий за отрисовку элемента на экране, учитывает значение флага и, если флаг выставлен, рисует вокруг элемента цветную рамку, а рядом подписывает имя пользователя, редактирующего данный элемент.

Для выставления значения флага используется тот же класс, который отвечает за блокировку элементов. Как было сказано выше, одно из полей этого класса хранит необходимые данные о редактируемых элементах. Флаг выставляется в случае, если id элемента присутствует в локальной копии таблицы, и ему сопоставлены данные другого пользователя.

Данный механизм позволяет хранить и визуализировать актуальную информацию о действиях каждого пользователя на диаграмме.

5. Апробация новой функциональности

Возможности коллаборативной разработки были проверены в ходе апробации. Двух человек, знающих систему QReal, попросили совместно решить предложенную задачу, которая заключалась в том, чтобы реализовать язык описания детерминированных конечных автоматов и создать автомат для разбора строк, удовлетворяющих следующему регулярному выражению:

$(abb(b)?(c)?d) + z$

Поскольку это был лишь эксперимент, для удобства наблюдения участники были размещены в одной комнате так, чтобы они не могли видеть экраны друг друга. Кроме того, они не общались между собой вне QReal. За действиями участников эксперимента велось наблюдение в ходе всего процесса решения задачи.

Разработчикам удалось справиться с поставленной задачей. Они отметили, что заниматься проектом вместе, не имея при этом возможности лично общаться, вполне удобно и продуктивно в коллаборативном режиме. Кроме того, были получены некоторые замечания и пожелания к созданной функциональности:

- в первую очередь, пользователям хотелось бы иметь возможность просматривать историю изменений и возвращать диаграмму к прошлым состояниям;
- было замечено, что часто было бы удобно иметь возможность прокомментировать часть диаграммы.

6. Обсуждение

Удобство подобной организации сетевого взаимодействия, когда сервер написан в виде отдельного приложения, заключается в возможности относительно простой реализации возврата к предыдущим версиям диаграммы. Поскольку на сервере хранится журнал, содержащий полный список всех изменений, начиная с создания диаграммы. Если один из клиентов захочет восстановить состояние на некоторый указанный им момент времени, сервер должен будет предпринять следующие действия:

- удалить текущую диаграмму;
- отметить в журнале последнее действие, после которого диаграмма окажется в нужном пользователю состоянии;
- разослать всем клиентам все команды из журнала, до отмеченной на прошлом шаге, включая ее;
- выполняя у себя соответствующие действия, клиенты в результате получат диаграмму, тождественную некоторому состоянию прошлой;

Несмотря на простоту реализации, такой подход имеет очевидные недостатки. При достаточно сложной диаграмме, на восстановление потребуется много ресурсов.

Хранение полной истории изменений позволяет присоединиться к редактированию диаграммы в любой момент. В этом случае новому клиенту отсылается история изменений. После обработки полученных сообщений, он имеет ту же диаграмму, которую его партнеры разработали без его участия.

Заключение

В ходе данной работы были достигнуты следующие результаты:

1. разработан механизм синхронизации изменений моделей;
   (a) реализован протокол общения нескольких экземпляров QReal;
   (b) организовано сетевое взаимодействие нескольких экземпляров QReal, на основе стека `tcp/ip` с применением буферизации полученных сообщений;
   (c) реализован механизм блокировки редактируемых элементов модели;
2. предложен механизм отображения изменений, выполненных другим пользователем;
3. предоставлен способ общения между разработчиками;
4. была проведена апробация разработанного решения.

Список литературы

[1] Cacoo website. — 2014. — mar. — URL: https://cacoo.com/.

[2] Creatly website. — 2014. — mar. — URL: https://creatly.com/.

[3] Lucid Chart website. — 2014. — mar. — URL: https://lucidchart.com/.

[4] Ouraiba El Amine, Choquet Christophe, Cottier Philippe. Domain-Specific Modeling Approach To Support Instructional Design Rationale // Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on / IEEE. — 2011. — P. 205–206.

[5] QReal project wiki. — URL: https://github.com/qreal/qreal/wiki.

[6] Qt project documentation. — URL: http://qt-project.org/doc/.

[7] V.Polyakov A.Kuzenkova A.Deripaska T.Bryksin Y.Litvinov. QReal DSM Platform: An Environment for Creation of Specific Visual IDEs. — 2013. — P. 251–257.

[8] Кознов Дмитрий Владимирович. Основы визуального моделирования // М.: Изд-во Интернетуниверситета информационных технологий, ИНТУИТ. ру, БИНОМ, Лаборатория знаний. — 2008.

[9] Т.А.Брыксин А.С.Кузенкова Ю.В.Литвинов. Метамоделирование: современный подход к созданию средств визуального проектирования // Материалы второй научно-технической конференции молодых специалистов «Старт в будущее», посвященной 50-летию полета Ю.А. Гагарина в космос. — 2011. — P. 228–231.

[10] Т.А.Брыксин А.С.Кузенкова А.О.Дерипаска К.С.Таран А.В.Подкопаев Ю.В.Литвинов. Средства быстрой разработки предметно-ориентированных решений в metaCASE-средстве QReal // Научно-технические ведомости СПбГПУ, Информатика, телекоммуникации, управление. — no. 4. — P. 142–145.