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

Кафедра системного программирования

Никольский Кирилл Андреевич

Автоматическое тестирование пользовательского интерфейса системы QReal

Курсовая работа студента 344 группы

Допущен к защите.

Зав. кафедрой:

проф., д.ф.-м.н. Терехов.А.Н.

Научный руководитель:

ст. преп. Литвинов Ю.В.

Санкт-Петербург
2015

Оглавление

1. Введение

При активной разработке проекта, важной областью которого является графический пользовательский интерфейс, необходимо тестировать функциональность системы, имитируя действия пользователя. Особенно это актуально в проектах, функциональность которых расширяется постоянно, добавляются новые графические элементы, часто изменяются старые. Такой способ тестирования эффективно проверяет правильное взаимодействие различных частей сложных систем, доступ к функциональности, работоспособность программы.

При автоматизации такого тестирования отпадает необходимость затрачивать человеческие ресурсы для рутинной циклической работы проверки тестовых сценариев. Появляется возможность запускать соответствующие тесты графического пользовательского интерфейса неограниченное количество раз в любое требуемое время. Ресурсы тратятся только на написание и поддержку самих тестов, а также на разработку, настройку самой тестирующей системы.

На кафедре системного программирования СПбГУ существует проект QReal с открытым исходным кодом, который позволяет быстро создавать визуальные языки программирования и инструменты для них. Проект активно разрабатывается, добавляется новая функциональность. В нем сделан сильный упор на графический интерфейс пользователя, который активно меняется с развитием проекта. Для такого проекта актуально реализовать автоматическое тестирование пользовательского интерфейса. На базе проекта QReal создана среда обучения основам программирования и кибернетики TRIK Studio.

На данный момент в проекте QReal реализована возможность тестирования графического интерфейса при помощи системы Sikuli. В этом

инструменте распознавание графических элементов происходит с помощью заданных заранее изображений. Однако данный способ получения графических элементов затратный и неудобный в поддержке тестов. В случае тестирования пользовательского интерфейса системы QReal, в которой графические элементы меняются часто, этот способ неактуален.

Также на данный момент в проекте отсутствует общая статистика по проекту: количество функций, количество документированных функций, количество юнит-тестов. Такую информацию полезно было бы иметь для оценки тестового покрытия, проверки соответствия написанного кода принятым в проекте QReal наборам правил и рекомендаций.

2. Цели и задачи работы

Таким образом, целью данной работы является создание системы тестирования пользовательского интерфейса, повышение качества системы.

Работа над данным проектом рассчитана на один год.

Задачи:

- ➢ Выявить и проанализировать достоинства и недостатки нескольких инструментов для тестирования графического пользовательского интерфейса;
- ➢ Проверить, что уже настроенная для проекта система непрерывной интеграции подходит для автоматизации такого тестирования;
- ➢ Создать тестирующую систему, позволяющую писать тесты на скриптовом языке и запускать написанные, выдавая результатом запуска соответствующий отчет;
- ➢ Реализовать автоматический запуск тестов для проекта QReal (TRIK Studio);
- ➢ Написать набор тестов, проверяющих существование, работоспособность частей проекта QReal, правильное взаимодействие различных сложных систем;
- ➢ Реализовать программу, которая по конкретному проекту составляет html-отчет по его функциям, предоставляя количественную информацию, включающую подсчет юнит-тестов и документированных методов, и добавить возможность такую программу запускать автоматически, например, после очередного коммита.

3. Обзор

3.1 Основные понятия и определения

A. Инструмент QReal

a. Средство QReal – проект кафедры системного программирования СПбГУ с открытым исходным кодом, предназначенный для быстрого и удобного создания визуальных языков программирования[1].

b. Плагин в QReal – подключаемый модуль, расширяющий функциональность платформы[2].

c. ScriptAPI – скриптовый интерфейс приложения к графическому интерфейсу среды [3].

B. Тестирование

a. Непрерывная интеграция (CI, англ. Continuous Integration) – это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.

b. Графический интерфейс пользователя (Graphical user interface, GUI) – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

c. Qt[3] – кроссплатформенный инструментарий разработки ПО на языке программирования C++.

---

[1] Проект QReal, url: http://qreal.ru (дата обращения 25.08.2015)

[2] Документация проекта QReal, url: https://github.com/qreal/qreal/wiki/Плагины (дата обращения 25.08.2015)

[3] Официальный сайт Qt, url: http://www.qt.io/ (дата обращения 21.09.2015)

d. QtScript – скриптовый язык, который является составной частью Qt, начиная с версии 4.3.0. Язык основан на стандарте ECMAScript с некоторыми расширениями, такими как возможность соединения с сигналами и слотами объектов QObject[4].

C. Разработка ПО

a. GitHub[5] – веб-сервис для хостинга IT-проектов и их совместной разработки.

b. Travis[6] – распределённый веб-сервис для сборки и тестирования программного обеспечения, использующего GitHub в качестве хостинга исходного кода.

c. X-сервер (X Window System) – оконная система, обеспечивающая стандартные инструменты и протоколы для построения графического интерфейса пользователя. Подробно о X-сервере написано в соответствующей документации [11].

d. Xvfb[7] (X virtual framebuffer) – виртуальный X-сервер, который может быть запущен на виртуальной машине, использующий для вывода не видеокарту, а оперативную память.

[4] Документация Qt, url: http://doc.qt.io/qt-5/qtscript-index.html (дата обращения 25.08.2015)
[5] Официальный сайт сервиса, url: https://github.com/ (дата обращения 25.08.2015)
[6] Официальный сайт сервиса, url: https://travis-ci.org/ (дата обращения 25.08.2015)
[7] Сайт используемого Xvfb в Travis, url : http://www.xfree86.org/4.0.1/Xvfb.1.html (дата обращения 25.08.2015)

3.2 Обзор существующих инструментов для GUI-тестирования

Для языка C++ и инструментария Qt существует довольно мало бесплатных кроссплатформенных средств, позволяющих выполнять тестирование UI и умеющих это делать в стиле юнит-тестов с предоставлением соответствующего отчёта. В качестве такого средства на момент написания курсовой работы в QReal была выбрана[8] система Sikuli[9]. Ниже показан пример тестирования GUI с его помощью (информация взята из wiki-документации проекта [4]):

```
1 def setUp(self):
2     App.open("/home/metal/workspace/study/qreal/bin/qrgui")
3     wait(<image: Начальная страница / Недавние проекты / Новый проект на базе>, 200)
4
5 def tearDown(self):
6     App.close("/home/metal/workspace/study/qreal/bin/qrgui")
7
8 def testAA(self):
9     assert exists(<image: Выбрать>)
```

Инструмент позволяет тестировать GUI приложения, сравнивая заранее заданные изображения с экранными. Однако такой подход теряет свою актуальность в проекте, в котором графический интерфейс часто меняется (таким является проект QReal), и постоянно требуется много ресурсов для поддержки таких тестов.

[8] Документация GUI-тестирования QReal, url: https://github.com/qreal/qreal/wiki/Тестирование-GUI-в-QReal (дата обращения 25.08.2015)

[9] Официальный сайт проекта, url: http://www.sikuli.org/ (дата обращения 25.08.2015)

Для обзора существующих инструментов для тестирования в курсовой работе были рассмотрены некоторые популярные библиотеки, анализ проводился в сравнительном формате на основе как существующей информации о продукте, так и на личном опыте пользования данных инструментов[10]:

	Quick Test Professional	Coded UI Tests	White
Поддерживаемые технологии	Win32, WPF, WinForm, SWT, Silverlight, Web etc.	Win32, WPF, WinForm, SWT, Silverlight, etc.	Win32, WPF, WinForm, SWT, Silverlight 2.0
Скорость работы скриптов	Неудовлетвори-тельная	Удовлетворительная	Удовлетворительная
Среда разработки	QTP env.	VS 2010	Любая (включая VS)
Запуск тестов	Из QTP, Из HP QC	Как юнит-тесты (nunit, reSharper, и др.) Из TFS	Как юнит-тесты (nunit, reSharper, и др.)
Язык программирования	Vbscipt	C#	C#
Лицензия	Дорого	Дорого	Бесплатно
Record&Play	+	+	-

Record&Play – это технология записи действий пользователя (тестировщика) для повторного их воспроизведения в тестовых наборах в будущем.

[10] Сайты, содержащие информацию об анализируемых продуктах, urls:
https://github.com/TestStack/White (дата обращения 25.08.2013)
https://msdn.microsoft.com/en-us/library/dd286726(v=vs.100).aspx (дата обращения 25.08.2013)
http://www8.hp.com/ru/ru/software-solutions/unified-functional-automated-testing/
(дата обращения 25.08.2013)

3.3 Автоматизация запуска при помощи CI-системы

На момент выполнения курсовой работы в проекте QReal была реализована поддержка системы непрерывной интеграции Travis. Как оказалось, для решения задач данной работы сервис Travis является подходящим вариантом: в нем существует возможность сборки проекта, запуска исполняемых файлов, реализована поддержка графического сервера Xvfb.

3.4 Научные работы на кафедре системного программирования СПбГУ

На кафедре системного программирования математико-механического факультета СПбГУ существует ряд работ, связанных с автоматизацией тестирования.

Самые близкие по тематике работы – это [2, 3]

3.5 Анализ в рамках поставленной цели

Важно отметить, что проект QReal – это проект с открытым исходным кодом, поэтому мы не можем в силу лицензионных ограничений использовать коммерческие решения.

Также важно отметить, что проект QReal разрабатывается с помощью инструментария Qt[11].

Каркас White framework был наиболее подходящим вариантом для использования в качестве решения поставленных задач, однако он не поддерживал тестирование программ, написанных при помощи инструментария Qt. Система Sikuli не подходила, так как в проекте QReal часто меняется графический интерфейс, создавая большую сложность поддержки тестов, написанных с помощью системы Sikuli.

[11] Инструментарий Qt, url: http://www.qt.io/developers/ (дата обращения 25.08.2015)

Необходимый инструмент должен иметь возможность находить нужные элементы не по картинке, не по координатам, а по каким-то параметрам, которые редко меняются с течением времени, например, по внутренним названиям объектов (object names), по названиям графических элементов, окон (window titles).

Работа «Система автоматизированного тестирования графического интерфейса в проекте QReal» оказалась неактуальной для продолжения в рамках данной работы, так как написанная библиотека позволяет писать тесты лишь на языке C++. Также многие методы работают неправильно в связи с накопленными изменениями в проекте QReal.

В работе «Разработка средств для создания обучающих демонстраций в среде QReal:Robots» был создан модуль ScriptAPI, позволяющий получать доступ к графическому интерфейсу среды. Изначально он задумывался для написания демонстраций в проекте QReal. Однако эти наработки можно применить в рамках данной работы для написания тестовых скриптов на языке QtScript.

4. Описание выбранного подхода и реализации

4.1 Создание тестирующей системы

Изначально было принято решение разработать систему, позволяющую писать тестовые сценарии на скриптовом языке. Такое решение было обусловленно рядом причин:

- Скриптовый язык позволяет программисту не отвлекаться на:
  - детали реализации методов;
  - очистку памяти;
- Синтаксис скриптового языка проще, чем в других языках;
- При использовании скриптов отпадает необходимость перекомпилировать в случае небольшого исправления в одном из скриптов;
- Упрощает дальнейший рефакторинг, так как если скриптовыми методами запрограммировать тестовые сценарии, то в случае изменений в графическом интерфейсе достаточно изменить только реализацию соответствующих вызываемых из скриптов методов;
- Позволяет легко подстраиваться даже под существенные изменения функциональности в проекте.

В проекте на момент написания данной работе существовал модуль ScriptAPI, предоставляющий доступ из скриптов, написанных на языке QtScript, к графическим элементам системы.

Таким образом, для решения поставленных задач было необходимо расширить функциональность модуля ScriptAPI и придумать, как использовать его для написания тестовых скриптов.

Для этого был использован каркас GTest[12], позволяющий организовать запуск тестовых сценариев в стиле юнит-тестов, предоставляя по результатам запуска тестовых наборов удобочитаемый отчет[13].

[12] Документация по Google's framework, url: https://code.google.com/p/googletest/ (дата обращения 25.08.2015)

[13] Документация юнит-тестирования QReal, url: https://github.com/qreal/qreal/wiki/Автоматическое-тестирование-в-QReal

4.2 Архитектура тестирующей системы

Для ознакомления с архитектурой QReal был выбран сайт с wiki-документацией проекта [4]. На этом сайте подробно рассказывается об устройстве различных частей, модулей системы. Информация о модуле ScriptAPI изложена в отчете к курсовой работе «Разработка средств для создания обучающих демонстраций в среде QReal:Robots» [3]. Подробней об устройстве системы можно ознакомиться в работе «Среда визуального программирования роботов QReal:Robots» [5].

Рассмотрим важные моменты, связанные с проектированием системы.

Класс QRealGuiTests наследуется от стороннего класса testing::Test, реализованного в каркасе GTest.

Класс ScriptAPIWrapper позволяет через основное окно приложения получать доступ к некоторым методам скриптового движка, реализованном в проекте.

Класс ScriptAPI регистрирует классы в скриптовом движке. И уже у них вызываются методы, помеченные макросом Q_INVOKABLE[14].

Класс Utils нужен для удобства, знает о большинстве классов модуля, таких как VirtualCursor, VirtualKeyboard, GuiFacade, HintAPI и т.д. Класс Utils содержит сложные действия, такие как: "кликнуть по определенной кнопке определенного типа определенного виджета", "произвести определенное действие в контекстном меню" и другие. Также он содержит различные методы, позволяющие работать в скриптах со сложными элементами, имея только указатель на них. Это необходимо, чтоб не усложнять модуль ScriptAPI и не регистрировать новые объекты в движке.

Для расширения функциональности модуля ScriptAPI для новых плагинов, можно использовать паттерн Facade [6]. Он используется для получения доступа к графическим элементам системы TRIK Studio.

Иногда для работы с некоторыми типами в скриптах необходимо их заранее "зарегистрировать". Для удобства регистрации новых типов был отдельно вынесен файл scriptRegisterMetaTypes.h с реализацией в .cpp.

---

[14] Документация по Qt, QObject Class, url: http://doc.qt.io/qt-5/qobject.html#Q_INVOKABLE (дата обращения 21.09.2015)

Класс TestAgent наследуется от класса QScriptEngineAgent. Его можно использовать для получения отладочной информации в ходе исполнения скрипта.

Диаграмма классов разработанной системы показана на рисунке 1.

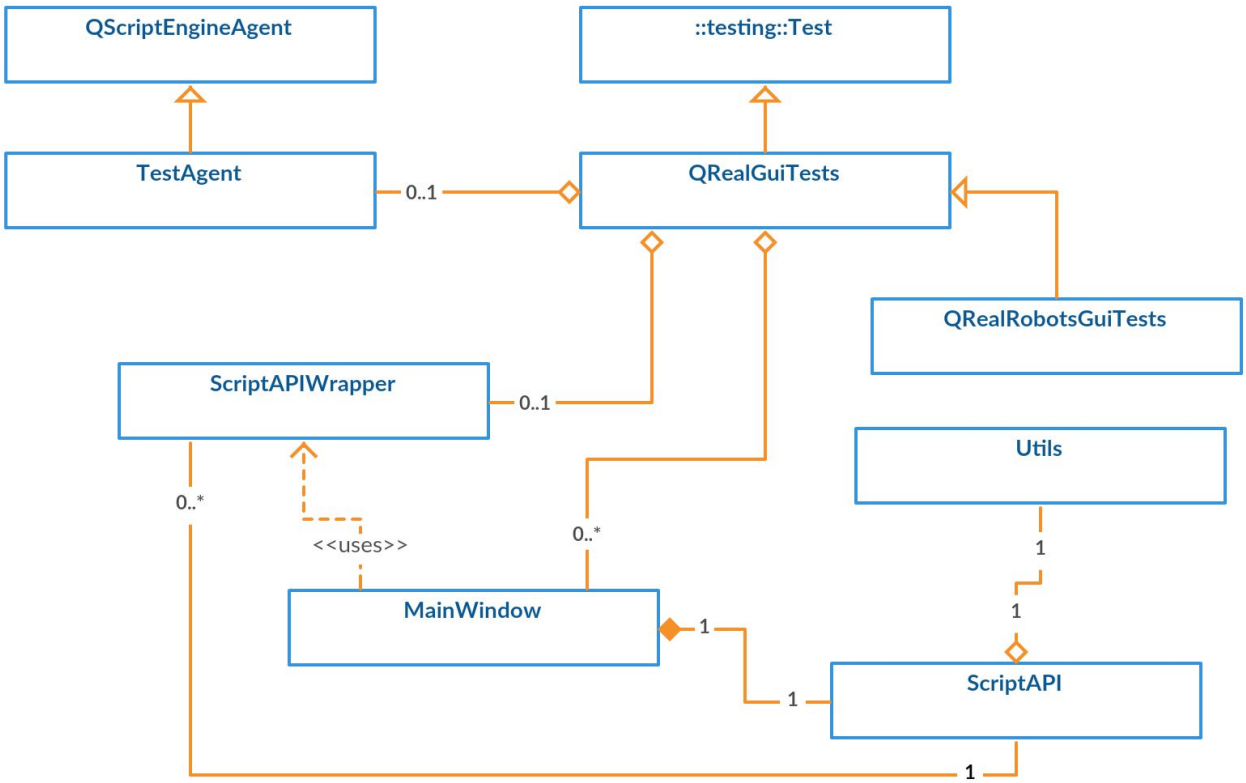


Диаграмма классов разработанной системы, рисунок 1.

4.3 Реализация функциональности тестирующей системы

В тестирующем классе QRealGuiTests реализована возможность запускать тестовые наборы. На каждый тест (тесты исполняются последовательно) запускается отдельно главное окно тестируемого проекта.

На все тесты устанавливается ограничение на максимальную длительность ожидания исполнения скрипта. После того, как время выйдет, происходит экстренное завершение выполнения скрипта с соответствующим уведомлением об ошибке.

Все тестовые скрипты из набора находятся в определенной папке, которую следует указать в тестирующем классе до запуска. Скрипты, которые являются общими, должны находиться на одну директорию выше.

Стоит отметить, что если тестовые скрипты не исполнят все инструкции в соответствующем файле .js, то тест будет считаться проваленным, и появится соответствующее уведомление об ошибке.

Реализованы проверки, которые можно писать в скриптах двух видов: критические и некритические, перечислим:

1. void assert(bool state);
2. void expect(bool state);
3. void fail(string text);
4. void add_failure(string text).

Все эти методы в случае (state == false) для первых двух и исполнения скриптовым движком последних двух помечают исполняемый тест проваленным. Первый и третий методы являются критическими и бросают скриптовое исключение. В text можно поместить сообщение, которое появится в выводе. Ниже небольшой пример использования перечисленных методов при написании скриптов:

```
openInterpreted = ui.getStartButton("Open interpreted diagram");

expect(openInterpreted != null);

expect(utils.isEnabledAndVisible(openInterpreted));

var createInterpreted = ui.getStartButton("Create interpreted diagram");

assert(createInterpreted != null);
```

Для добавления тестового скрита в исполняемый набор достаточно реализовать следующий метод в файле qRealGuiTests.cpp:

```
TEST_F(QRealGuiTests, newTest);
```

Можно подключать общие файлы до запуска основного тестового скрипта, располагающегося в определенной для каждого тестового набора папке.

Что касается написания и отладки написанных тестов, удобно пользоваться реализованным классом TestAgent, предоставляющим информацию по исполнению скрипта скриптовым движком, к которому экземпляр класса TestAgent был заранее присоединен.

Для реализации новых тестовых наборов, например, для плагинов, необходимо, как в случае с классом QRealRobotsGuiTests, наследоваться от класса QRealGuiTests.

Для доступа к скриптовому движку из тестирующего класса был использован класс ScriptAPIWrapper.

Была расширена функциональность модуля ScriptAPI, отметим важные изменения:

- добавлен отложенный вызов функций;
- добавлены многоступенчатые сложные действия для упрощения написания скриптов для однотипных частоиспользуемых действий;
- действия, описанные в предыдущем пункте, вынесены в отдельный класс Utils, так же включающий "одиночные" методы. Например, чтобы посчитать длину списка строк в скрипте, достаточно в классе Utils создать метод Q_INVOKABLE int length(list); и вызывать в скриптах этот метод следующим образом:
  utils = api.utils(); /* тут получаем список list*/ var len = utils.length(list).

Настроен автоматический запуск тестирования в Travis. Результат тестирования выводится в соответствующем отчете как для каждого набора, так и для конкретного теста. Для настройки графического сервера Xvfb была изучена соответствующая документация [8]. Графический сервер Xvfb необходимо было настраивать, так как в сервисе Travis сборка и запуск проекта QReal происходит на виртуальной машине. Тесты для проекта QReal построены на имитировании действий пользователя, соответственно нужно обрабатывать их, используя виртуальный дисплей.

4.4 Генерация html-отчета о функциях проекта

Необходимо было написать программу, которая по указанному ей проекту составляет статистику, содержащую количественную информацию о функциях проекта, юнит-тестах и документированных функциях.

Реализация такой программы основана на построении дерева директорий с определенными полями, содержащими статистику для данной директории. Статистика собирается рекурсивно, начиная с листьев. Подсчет нужной информации производится за счет регулярных выражений. Результат, (для каждой поддиректории проекта) включающий количество функций, количество документированных функций, количество

соответствующих тестов, выводится в генерируемый .html файл, содержащий гиперссылки для удобного перемещения по поддиректориям. Если программа для каких-то тестов не нашла тестируемого класса, она помещает эту информацию под таблицу.

Запуск программы происходит автоматически, каждый раз, когда в основную ветку проекта QReal вносятся изменения. При написании команд для соответствующего отчета в файл .travis.yml во многом помог сайт с документацией по сервису Travis [10].

Результат выводится в wiki-документацию проекта QReal: https://github.com/qreal/qreal/wiki/Statistics

При выводе вышеописанного результата для коммита в wiki-репозиторий проекта QReal используются зашифрованные данные одного из аккаунтов. Шифрование происходит с использованием симметричного алгоритма блочного шифрования AES. Об алгоритме шифрования AES, используемом в сервисе Travis, подробно написано в статье "The Advanced Encryption Standard" [9]. Для его использования было нужно с помощью специальной программы, указанной на официальном сайте сервиса Travis, создать приватные переменные для проекта, зашифровав при этом нужные данные, что и было сделано[15]:

Приватные переменные:
encrypted_2c91ba1b7c87_key

encrypted_2c91ba1b7c87_iv

Файл с зашифрованной информацией:

loginpass.enc

4.5 Написание тестовых скриптов

После ознакомления с материалом документа «Опыт автоматизации тестирования в компании Ланит-Терком» [7] и презентации «Тестирование графического интерфейса пользователя» [1], было принято решение разбить тестирование системы условно на три части:

1. Тестирование существования элементов и их начального состояния;

[15] url: https://travis-ci.org/qreal/qreal/settings (дата обращения 19.09.2015)

2. Правильность функционирования, работы отдельных элементов после некоторых подготовительных действий;
3. Корректная работа тестовых сценариев.

С помощью тестирования графического интерфейса можно находить ошибки в функциональности, такие как:

- Необработанные исключения при взаимодействии с интерфейсом;
- Потеря или искажение данных, передаваемых через элементы интерфейса.

Стоит отметить, что некоторые ошибки найти невозможно, или попытки их находить неоправданно затратны. Например, ошибки, связанные с правильностью отображения текста на графических элементах, а также связанные с юзабилити.

Для написания тестовых скриптов было сформулировано правило: писать тесты так, чтобы тесты написанные один раз, работали всегда. Такое правило мотивировано заинтересованностью в легкой и малозатратной поддержке тестовых сценариев в быстро меняющемся проекте. Понятно, что в связи с отсутствием заранее заявленной документации системы, это правило невозможно соблюдать всегда, однако мы старались придерживаться его при написании тестов в данной работе.

Отдельно стоит упомянуть про общие тестовые скрипты. Их можно подключать заранее в любом количестве до запуска основного скрипта. В них могут содержаться как часто используемые методы, так и инициализация каких-либо часто используемых переменных. Это упрощает написание скриптов и их поддержку.

В качестве примера, затрагивающего большинство описанных выше тем, можно рассмотреть тест, реализанный в классе QRealGuiTests[16]:

```
TEST_F(QRealGuiTests, createRootElementOnSceneAndQuit);
```

4.6 Проблемы

В ходе выполнения данной курсовой работы возникли следующие проблемы:

[16] url: https://github.com/Kirill-NiK/qreal (дата обращения: 19.09.2015)

- Невозможность отлавливать segmentation faults, из-за которых падает вся тестирующая система. То есть если какой-нибудь из тестов роняет тестируемое приложение, то прекращается исполнение всего набора тестов. И чтобы узнать результат тестов, следующих за «критическим», необходимо сперва поправить неисправную функциональность системы;
- Исключения, брошенные в коде C++ в методах, вызванных из скриптов, не ловятся никакими обработчиками. Поэтому при реализации методов, в которых нужны проверки, исключения бросаются скриптовые;
- Виртуальная клавиатура, использующая методы класса QTest (Qt 5.5), не может вводить русские буквы, для ввода доступны только символы из таблицы ascii;
- Некоторые виджеты невозможно прокликивать, изменять с помощью стандартных методов, описанных в модуле ScriptAPI по причине их платформозависимости или неприспособленности к искусственным способам управления (сложно получать данные на объекты внутри). Для таких случаев специально были созданы методы, которые позволяют использовать в тестовых скриптах не графический способ доступа (управления) к виджетам. Такие методы были помещены в файл workaroundTestFunctions.h с реализацией в .cpp;
- Одна проблема возникла при написании программы, генерирующей html-отчет по функциям проекта. Проблема заключается в отсутствии рекомендаций по расположению файлов с юнит-тестами относительно тестируемого класса. Таким образом, сложно сопоставить тестируемый класс и тестирующий, если в проекте несколько классов с одинаковыми именами. На данный момент проблема решена составлением списка нераспределенных по поддиректориям проекта тестирующих классов.

4.7 Апробация

Пример 1.

Создадим некоторое количество простых тестовых скриптов, которые проверяют, что отдельные элементы на главном окне видимы. Все тесты дают положительный результат, как и ожидалось. Часть из отчета сервиса Travis:

```
2664 MainWindow::~MainWindow()
2665 [       OK ] QRealGuiTests.linkNodeCooperationScript (7558 ms)
2666 [ RUN      ] QRealGuiTests.logicalGraphicalModelsAndSceneInteraction
2667 MainWindow::~MainWindow()
2668 [       OK ] QRealGuiTests.logicalGraphicalModelsAndSceneInteraction (6995 m
2669 [ RUN      ] QRealGuiTests.miniMapWorkScript
2670 MainWindow::~MainWindow()
2671 [       OK ] QRealGuiTests.miniMapWorkScript (7005 ms)
2672 [ RUN      ] QRealGuiTests.paletteAndEditorPropertiesConcordance
2673 MainWindow::~MainWindow()
2674 [       OK ] QRealGuiTests.paletteAndEditorPropertiesConcordance (6999 ms)
2675 [ RUN      ] QRealGuiTests.paletteSearchScript
2676 MainWindow::~MainWindow()
2677 [       OK ] QRealGuiTests.paletteSearchScript (7000 ms)
2678 [ RUN      ] QRealGuiTests.propertyEditorScript
2679 MainWindow::~MainWindow()
2680 [       OK ] QRealGuiTests.propertyEditorScript (6993 ms)
2681 [ RUN      ] QRealGuiTests.rightclickAllScreenScript
2682 MainWindow::~MainWindow()
2683 [       OK ] QRealGuiTests.rightclickAllScreenScript (7006 ms)
2684 [ RUN      ] QRealGuiTests.saveOpenScript
2685 MainWindow::~MainWindow()
2686 [       OK ] QRealGuiTests.saveOpenScript (7001 ms)
2687 [ RUN      ] QRealGuiTests.verySimpleShapeEditorTest
2688 MainWindow::~MainWindow()
2689 [       OK ] QRealGuiTests.verySimpleShapeEditorTest (7000 ms)
2690 [----------] 30 tests from QRealGuiTests (232679 ms total)
2691
2692 [----------] Global test environment tear-down
2693 [==========] 30 tests from 1 test case ran. (232679 ms total)
2694 [  PASSED  ] 30 tests.
```

Пример 2.

Теперь сознательно сделаем изменение в графическом интерфейсе, которое портит некоторую функциональность, скрыв некоторые доквиджеты, чтобы в отчете некоторые тесты, проверяющие эту функциональность, не прошли, ниже результат:

```
2799 [ RUN      ] QRealGuiTests.paletteSearchScript
2800 MainWindow::~MainWindow()
2801 [       OK ] QRealGuiTests.paletteSearchScript (6994 ms)
2802 [ RUN      ] QRealGuiTests.propertyEditorScript
2803 MainWindow::~MainWindow()
2804 [       OK ] QRealGuiTests.propertyEditorScript (7006 ms)
2805 [ RUN      ] QRealGuiTests.rightclickAllScreenScript
2806 MainWindow::~MainWindow()
2807 [       OK ] QRealGuiTests.rightclickAllScreenScript (7000 ms)
2808 [ RUN      ] QRealGuiTests.saveOpenScript
2809 MainWindow::~MainWindow()
2810 [       OK ] QRealGuiTests.saveOpenScript (7000 ms)
2811 [ RUN      ] QRealGuiTests.verySimpleShapeEditorTest
2812 MainWindow::~MainWindow()
2813 [       OK ] QRealGuiTests.verySimpleShapeEditorTest (7000 ms)
2814 [----------] 33 tests from QRealGuiTests (319067 ms total)
2815
2816 [----------] Global test environment tear-down
2817 [==========] 33 tests from 1 test case ran. (319067 ms total)
2818 [  PASSED  ] 31 tests.
2819 [  FAILED  ] 2 tests, listed below:
2820 [  FAILED  ] QRealGuiTests.dockWidgetsExistence
2821 [  FAILED  ] QRealGuiTests.fullscreenModeScript
2822
```

Как и планировалось, два соответствующих теста оказались проваленными.

Пример 3.

Результат работы программы, генерирующей html-отчет (часть файла):

	testing	doc	tests	testing in subfolders	doc in subfolders	tests in subfolders
D:/QReal/qreal/	0	0	0	3354	573	82
D:/QReal/qreal/docs/	0	0	0	0	0	0
D:/QReal/qreal/installer/	0	0	0	0	0	0
D:/QReal/qreal/plugins/	0	0	0	1125	257	1
D:/QReal/qreal/qrgui/	0	0	0	1531	200	0
D:/QReal/qreal/qrkernel/	40	31	13	2	2	1
D:/QReal/qreal/qrmc/	82	0	0	71	0	0
D:/QReal/qreal/qrrepo/	4	0	0	110	12	39
~D:/QReal/qreal/qrtest/	0	0	0	0	0	0
D:/QReal/qreal/qrutils/	25	14	11	208	54	17
D:/QReal/qreal/qrxc/	156	3	0	0	0	0
~D:/QReal/qreal/thirdparty/	0	0	0	0	0	0
D:/QReal/qreal/umlDocumentation/	0	0	0	0	0	0
D:/QReal/qreal/plugins/additionsEditor/	0	0	0	0	0	0
D:/QReal/qreal/plugins/ains/	0	0	0	11	0	0

Различные цвета обозначают различный уровень вложенности.

По поддиректориям можно перемещаться с помощью гиперссылок.

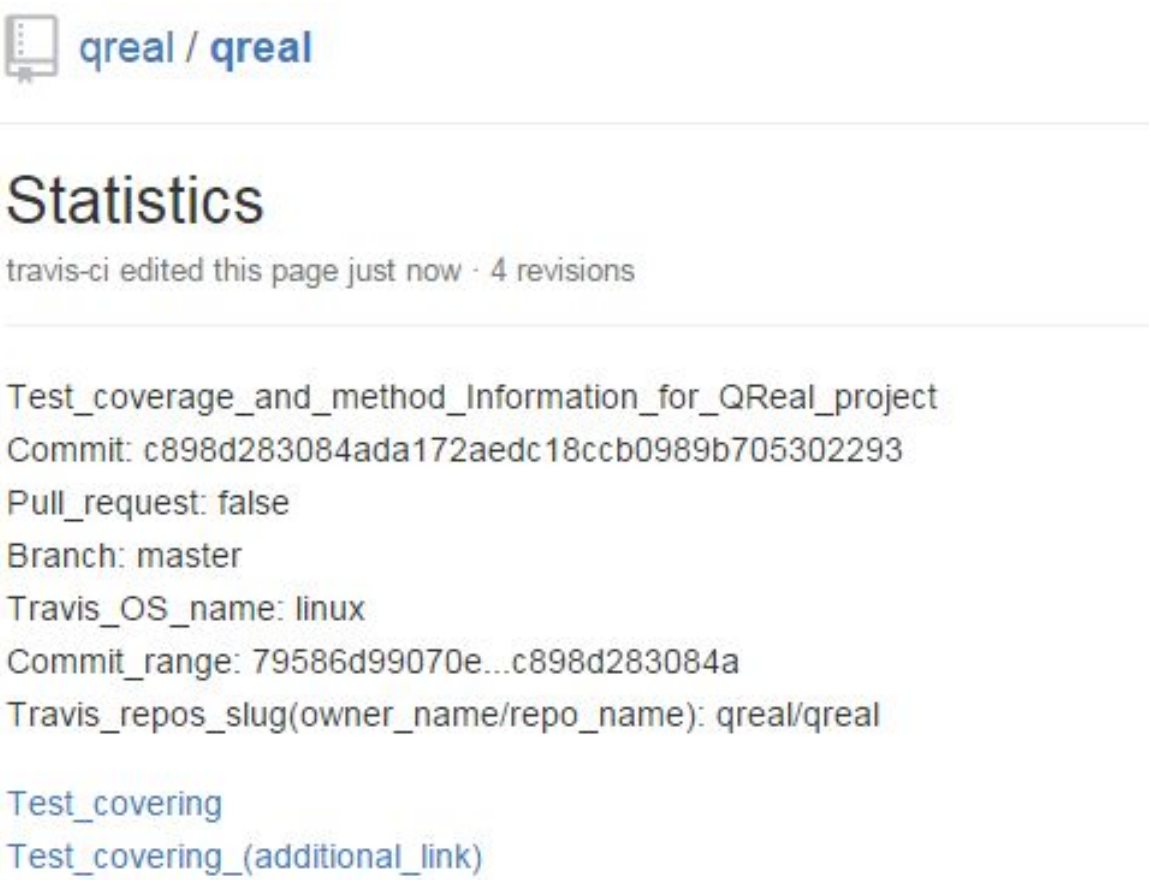
Первые три столбца:

- testing – количество функций, расположенных в файлах рассматриваемой директории (исключая поддиректории), которые могут быть протестированы (не считаются методы интерфейса, например);
- doc – количество функций, расположенных в файлах рассматриваемой директории (исключая поддиректории), которые имеют соответствующую документацию, описание;

- tests – количество тестов на функции, расположенных в файлах рассматриваемой директории (исключая поддиректории).

Следующие три столбца: то же, что и описано выше, только для суммы всех поддиректорий рассматриваемой директории.

Полученный отчет автоматически выкладывается в wiki-репозиторий проекта QReal вместе с метаинформацией:



5. Результаты

В рамках данной курсовой работы были получены следующие результаты:

- Создана тестирующая система, позволяющая писать тесты на скриптовом языке и запускать написанные, выдавая результатом запуска соответствующий отчет;
- Расширена функциональность модуля ScriptAPI, исправлены некоторые ошибки в работе этого модуля;
- Написан небольшой набор тестов, проверяющих существование, работоспособность некоторых частей проекта QReal, правильное взаимодействие различных сложных систем;
- Настроен автоматический запуск тестов для проекта QReal (TRIK Studio);
- Реализована программа, составляющая html-отчет по функциям проекта, предоставляя количественную информацию, включающую подсчет юнит-тестов и документированных методов;
- Настроена автоматическая генерация html-отчета по функциям последней версии проекта QReal после очередного коммита в основную ветку на GitHub. Файл автоматически выкладывается в документацию проекта на GitHub;
- Создан пулл-реквест на GitHub, содержащий всю новую функциональность, описанную в данной работе;
- Написаны рекомендации[17] на GitHub по написанию тестовых скриптов для запуска в реализованной тестирующей системе;

[17] url: https://github.com/qreal/qreal/wiki/Тестирование-GUI-в-QReal,-используя-ScriptAPI

- Обнаружен ряд ошибок и неточностей в проекте QReal (TRIK Studio) в процессе выполнения данной работы[18].

[18]QReal bugtracker, urls:
https://github.com/qreal/qreal/issues/1570
https://github.com/qreal/qreal/issues/1571
https://github.com/qreal/qreal/issues/1572
https://github.com/qreal/qreal/issues/1574
https://github.com/qreal/qreal/issues/1575
https://github.com/qreal/qreal/issues/1576
https://github.com/qreal/qreal/issues/1577
https://github.com/qreal/qreal/issues/1578
https://github.com/qreal/qreal/issues/1579
https://github.com/qreal/qreal/issues/1580
https://github.com/qreal/qreal/issues/1586
https://github.com/qreal/qreal/issues/1587
(дата обращения 25.08.2015)

6. Список литературы

1. Тестирование графического интерфейса пользователя, url: http://delta-course.org/docs/Delta2014S-T2-L7.pdf, дата обращения: 19.09.2015
2. Новожилов Е. А., Брыксин Т.А., Система автоматизированного тестирования графического интерфейса в проекте QReal // Курсовая работа кафедры системного программирования СПбГУ. Санкт-Петербург. 2014. url: http://se.math.spbu.ru/SE/YearlyProjects/2014/YearlyProjects/2014/444/444-Novozhilov-report.pdf, дата обращения: 19.09.2015
3. Чернов Д.В., Мордвинов Д.А., Разработка средств для создания обучающих демонстраций в среде QReal:Robots // Курсовая работа кафедры системного программирования СПбГУ. Санкт-Петербург. 2014. url: http://se.math.spbu.ru/SE/YearlyProjects/2014/371/371-Chernov-report.pdf, дата обращения: 19.09.2015
4. wiki-документация проекта QReal, url: https://github.com/qreal/qreal/wiki, дата обращения: 19.09.2015
5. Брыксин Т.А., Литвинов Ю.В., Среда визуального программирования роботов QReal:Robots // Материалы международной конференции "Информационные технологии в образовании и науке". Самара. 2011.
6. Gang of Four Design Patterns: Elements of Reusable Object Oriented Software. — Produced by KevinZhang — 208 с.
7. «Опыт автоматизации тестирования в компании Ланит-Терком»
8. Документация и описание Xvfb, url: http://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml, дата обращения: 19.09.2015

9. Christof Paar, Jan Pelzl, "The Advanced Encryption Standard", Chapter 4 of "Understanding Cryptography, A Textbook for Students and Practitioners". (companion web site contains online lectures on AES), Springer, 2009.
10. Документация сервиса Travis, url: http://docs.travis-ci.com/, дата обращения: 19.09.2015
11. Документация и описание X-сервера, url: http://www.x.org/wiki/, дата обращения: 19.09.2015