САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Нефёдов Ефим Андреевич

Технология разработки мобильных приложений

Дипломная работа

Допущена к защите.

Зав. Кафедрой

д.ф-м.н. проф. Терехов А.Н

Научный руководитель:

д.ф-м.н. проф. Терехов А.Н

Рецензент:

Мордвинов Д.А.

Санкт-Петербург

2014

SAINT-PETERSBURG STATE UNIVERSITY

Mathematics & Mechanics Faculty

Software Engineering Chair

Nefedov Efim

Mobile application development tools

Graduation Thesis

Admitted for defence.
Head of the chair:
Professor Terekhov A.N

Scientific supervisor:
Professor Terekhov A.N

Reviewer:
Mordvinov D.A.

Saint-Petersburg

2014

Оглавление

Введение

Мобильных платформ много и их количество только растёт. Например, в сегменте смартфонов помимо Apple с iOS, BlackBerry с BlackBerry OS и Google с Android в гонку уже давно включилась Microsoft с Windows Phone, включается Canonical с Ubuntu Phone. А также значительное количество телефонов работает на Java и Symbian.

При этом растёт и количество связанных с мобильными технологиями предприятий: рынок мобильных информационных услуг оценивается в 300 млрд долларов и его объем продолжает увеличиваться, [1] что делает его одним из немногих связанных с IT растущих рынков.

Развиваются различные облачные хранилища и появляется большое количество приложений, их использующих. Продолжаются тенденции по использованию мобильных онлайн-сервисов в менеджменте частных и государственных структур, в частной и государственнной медицине. Не исчерпаны возможности мобильных технологий и в сферах образования, личной жизни, развлечений.

Создание приложений для мобильных телефонов — сложная задача по многим причинам. Среди основных препятствий, сдерживающих распространение мобильных онлайн-сервисов, выделяют[1] следующие:

- Невозможность прямого переноса «компьютерных» сервисов на мобильные устройства, связанная с разницей в форм-факторе и вычислительных возможностях мобильных устройств по сравнению с «обычными» компьютерами.

- Более высокая сложность разработки мобильных приложений по сравнению с традиционным программированием.

- Разнообразие несовместимых между собой платформ.

На математико-механическом факультете СПбГУ преподавателями, аспирантами и студентами разрабатывается платформа Ubiq Mobile, основанная на идее терминальной архитектуры: приложения выполняются на серверах (которых может быть много), а мобильные устройства выступают в роли удалённых графических терминалов [2].

Другими важными особенностями платформы являются:

• Эффективность и надёжность выполнения мобильных сервисов на различных устройствах — клиенты поддерживают платформы от iOS и Android, до Symbian и Java ME.

• Устойчивый обмен информацией в различных условиях сети за счёт оригинального двоичного протокола, надстроенного над TCP\IP, в сочетании с математическими методами сжатия и упаковки информации.

• Удобное представление этого обмена с точки зрения разработчика посредством виртуального графического холста в памяти сервера.

• Серверы Ubiq Mobile работают в среде Microsoft.NET, соответственно разработчикам серверной части сервисов достаточно быть специалистами в популярных на сегодняшний день языках платформы .NET.

Всё это позволяет приблизить разработку мобильных сервисов к разработке приложений для персональных компьютеров, что привычно широкому кругу программистов и не требует более высокой квалификации по сравнению с программированием привычных веб-приложений.

Существует ещё один существенный резерв по сокращению рутинной работы при создании мобильных сервисов — это использование специализированных языков, ориентированных на конкретные предметные области (DSL – Domain Specific Languages). Они давно успешно применяются, например, для написания скриптов для UNIX shell — несмотря на то, что эти языки обычно Тьюринг полные, они значительно отличаются от

языков общего типа и хорошо справляются со своей основной задачей — объединение небольших UNIX программ (gawk, ls, sort или wc). В качестве классический DSL языков приводятся[2] TEX, Perl, SQL.

Язык TEX является DSL для компьютерной вёрстки текстов; Mathematica и MatLab – DSL для математических вычислений; SQL – DSL, созданный для написания запросов к базам данных.

Но даже при использовании DLS для создания мобильных сервисов, остаются некоторые важные проблемы. Логика смены экранов и использования функций часто запутана, её продумывание занимает много времени. Её текстовое представление не наглядно и поэтому часто при изменениях в различных местах появляются неожиданные побочные эффекты.

На наш взгляд, их решением является применение специализированных графических языков. Использование различных графических нотаций для описания мобильных приложений уже давно используется[3], например, при составлении шаблонов интерфейсов или диаграмм переходов между экранами, поскольку это позволяет ясно и однозначно описать структуру приложения.

Использование специализированных языков в противовес использованию языков общего назначения (например, SDL или UML) избавляет разработчиков от излишней громоздкости и позволяет легко переиспользовать часто появляющиеся абстракции. Более того, пространство различных мобильных сервисов более чётко разделено на узкоспециализированные предметные области, что ещё более способствует применению DSM-подхода (Domain Specific Modeling).

Препятствием использования DSL в различных областях является необходимость разрабатывать различные узкоспециализированные языки для

различных типов задач для того, чтобы их применение приносило выигрыш продуктивности (до 10 раз согласно [V. Onossovski, A.Terekhov, Modern Interactive Internet Services, Proceedings of 7th Conference of Open Innovation Framework Program FRUCT, 2010]).

Если для каждого приложения (типа приложения) применять свои языки, то требуется много ресурсов для реализации этих языков.

Таким образом, использование специализированных языков выгоднее использования языков общего типа, но необходимость затрат на создание редактора семантики и генерации смещает выбор в сторону традиционных методов разработки.

Поэтому появилось понятие мета-технология — технология, генерирующая специализированные технологии по некоторому формальному описанию. На кафедре системного программирования СПбГУ разрабатывается мета-технология QReal, позволяющая по описанию языка генерировать визуальные редакторы и писать к ним различные плагины, в том числе выполняющие генерацию кода по диаграммам. QReal уже успешно применяется в программировании роботов[4] (QReal:Robots для конструкторов Lego MindStorm и TRIK).

Постановка задачи

В задачу данной работывходит разработка DSL для создания мобильных приложений, реализация соответствующих редакторов с помощью мета-технологии QReal, а также создание плагинов, генерирующих по диаграммам, сделанным в этих редакторах, кода для платформы Ubiq Mobile.

Язык

Обзор существующих решений

Первый вариант подобного визуального языка[5] для платформы Ubiq Mobile был создан к конференции FRUCT в 2011 году Ю.В.Литвиновым.

Этот язык описывал программу с помощью следующих видов диаграмм:

- Диаграммы взаимодействия (Рис.1) содержали описание диспетчеров, отвечающих за сетевое взаимодействие клиента и сервера. Описание методов диспетчеров начинается с указания сигнала, получение которого должно инициировать этот метод, тело метода описывается с помощью элементов отправки сигнала, вызова подпрограммы или специального стереотипного элемента.

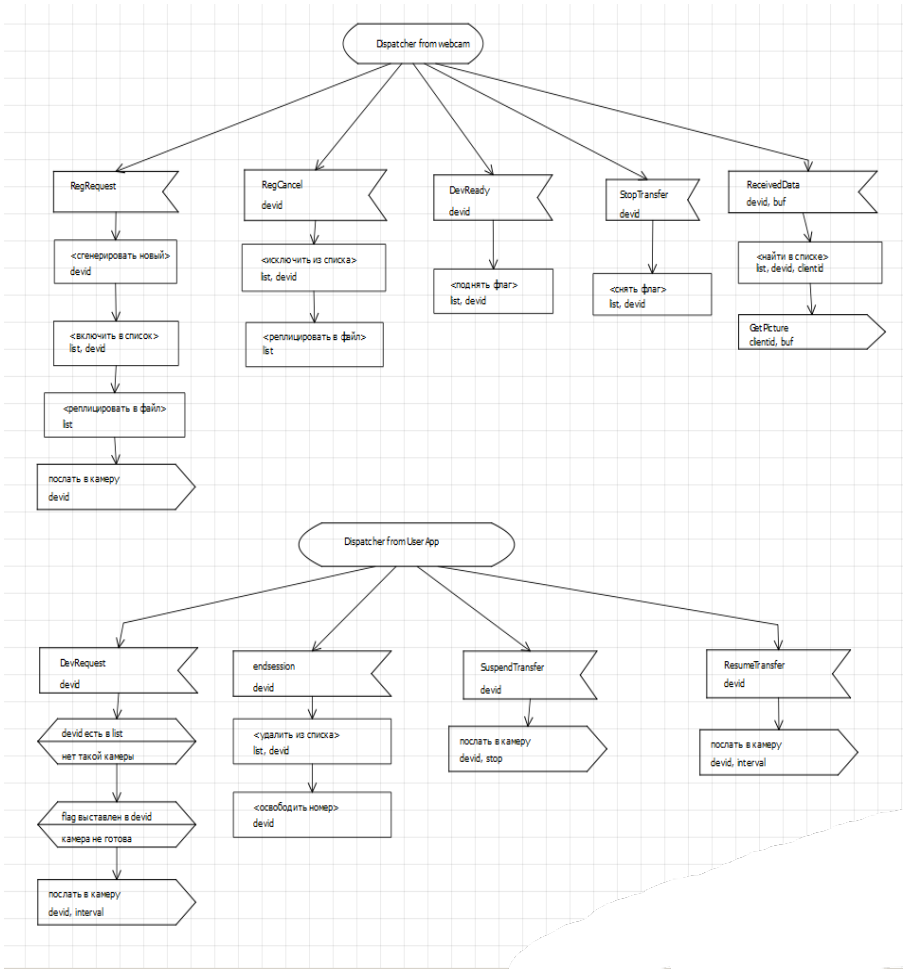

Рис. 1: Диспетчеры клиента и сервера

- Диаграммы подпрограмм (Рис.2 и 3) содержали SDL-подобное описание программ, вызов которых возможен из диаграмм описания взаимодействия и подпрограмм. Описание действий, которые должны произойти с системой при вызове подпрограммы, происходит с помощью элементов ожидания и отправки сигналов, элементов контроля потока управления и элементов, содержащих вставки C# кода.

Однако язык имел ряд недостатков, например, необходимость понимать общее внутреннее устройство библиотек Ubiq Mobile и необходимость написания в блоках большого количества текстовых кусков кода.

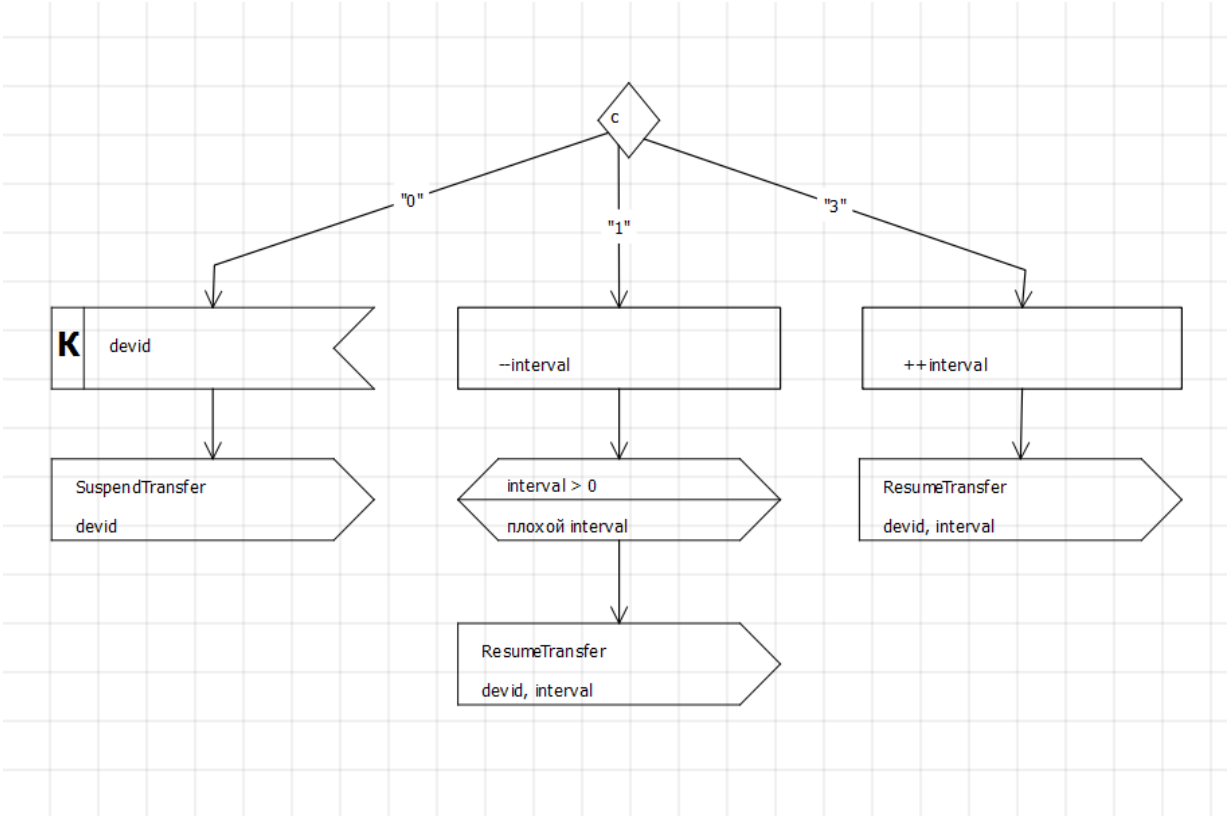

Рис. 2 Подпрограмма обработки ввода

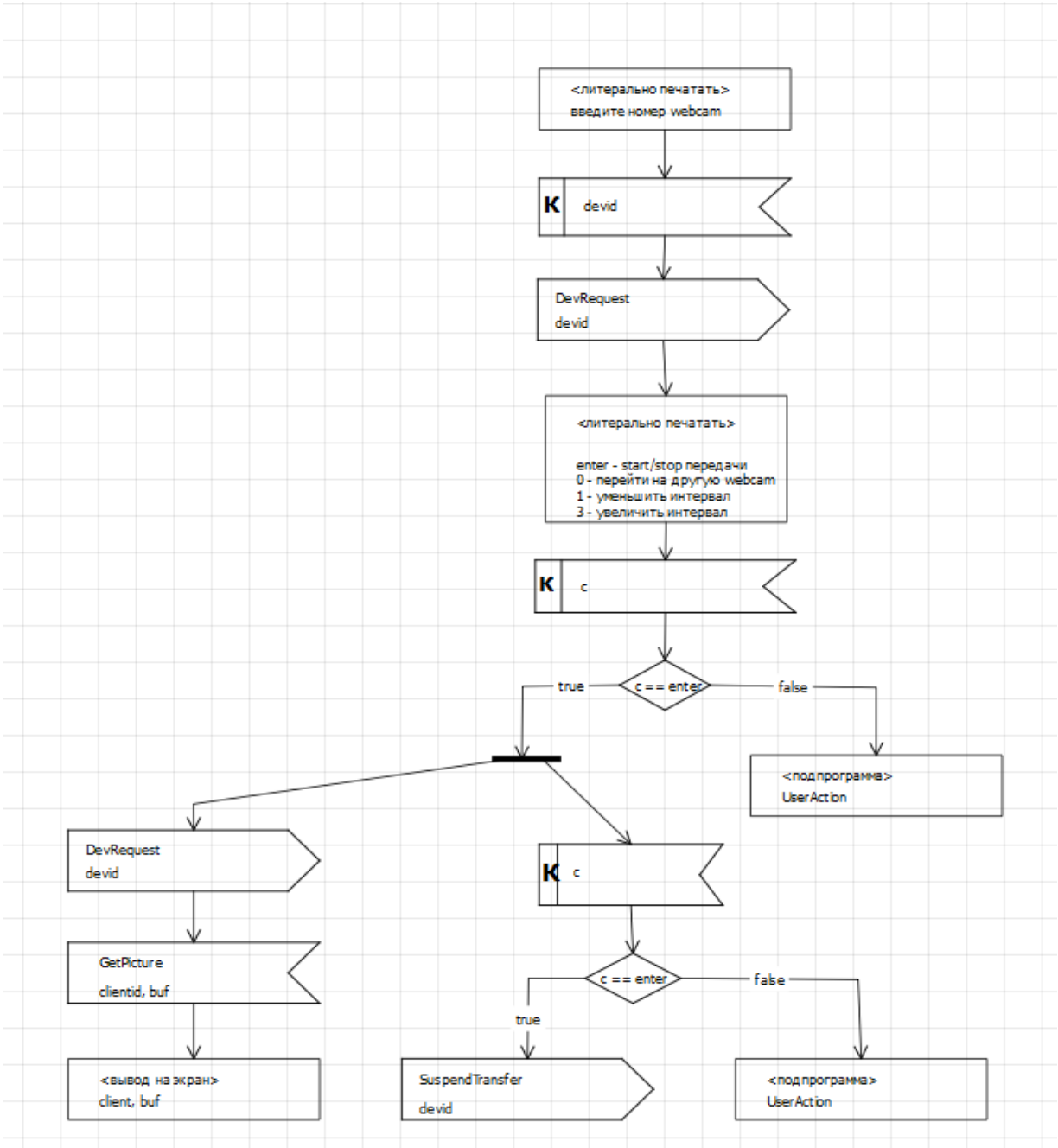

Рис. 3 Пример вызова подпрограммы

Устранением этих недостатков в рамках своей курсовой работы в 2013 году занималась[6] Дерипаска Анна. В результате её работы были разработаны и реализованы в QReal два языка для создания онлайн игр с игровым полем между двумя игроками.

Основной идеей первого языка было описание программы с помощью блоков, обозначающих куски конкретной готовой функциональности, часто специфичной для конкретной задачи. То есть, например, для описания программы «Морской бой» необходимо дополнить язык элементом «Размещение кораблей» и описать его генерацию. Тогда этот блок можно было бы легко переиспользовать в различных местах диаграммы, описывающей игру «Морской бой» (Рис.4)

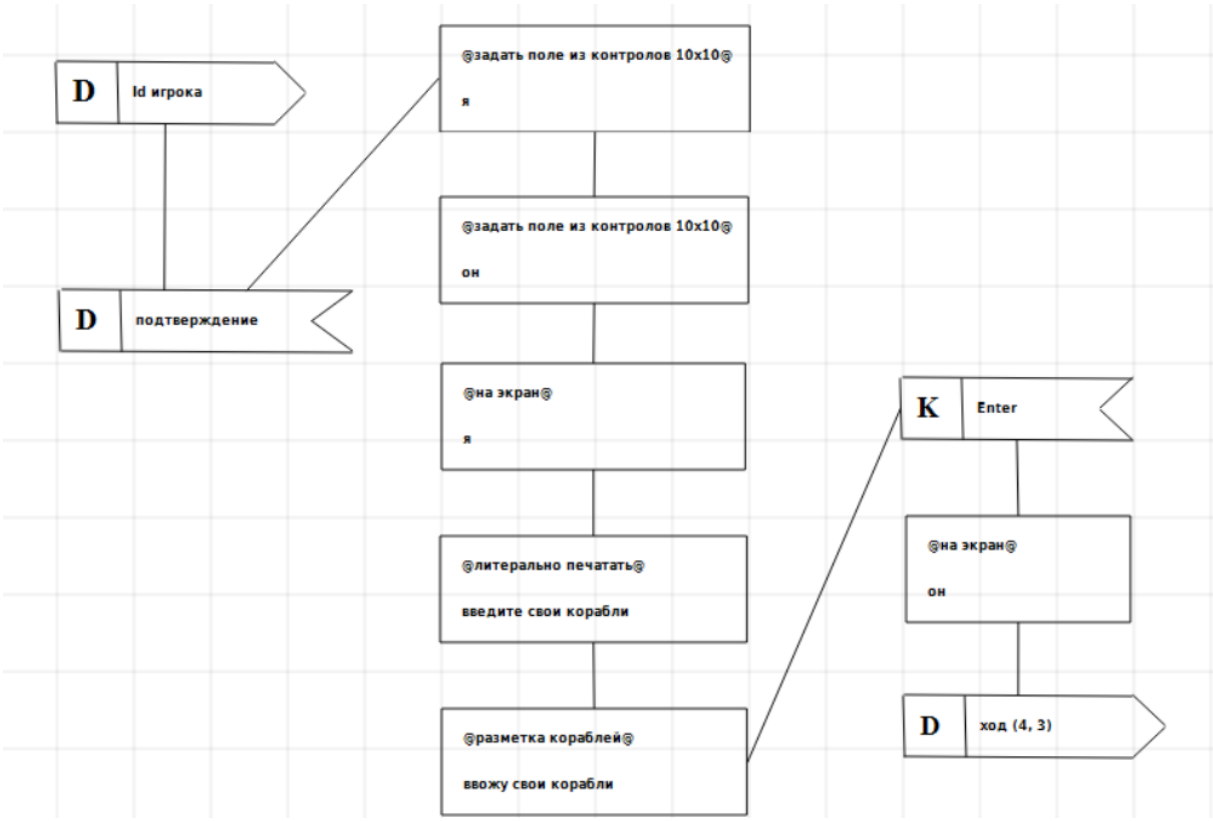

Рис. 4 Описание игры «Морской бой»

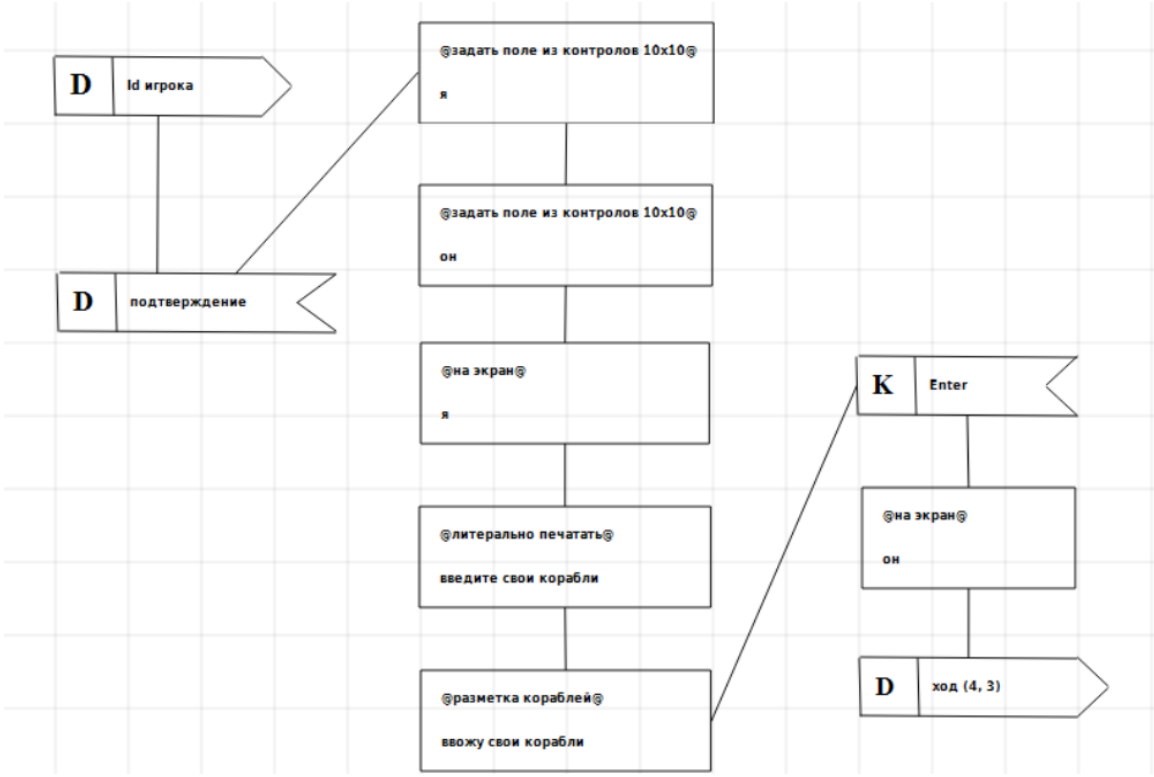
D
Id игрока
D
подтверждение
@задать поле из контролов 10x10@
я
@задать поле из контролов 10x10@
он
@на экран@
я
@литерально печатать@
введите свои корабли
@разметка кораблей@
ввожу свои корабли
K
Enter
@на экран@
он
D
ход (4, 3)

Рис. 5

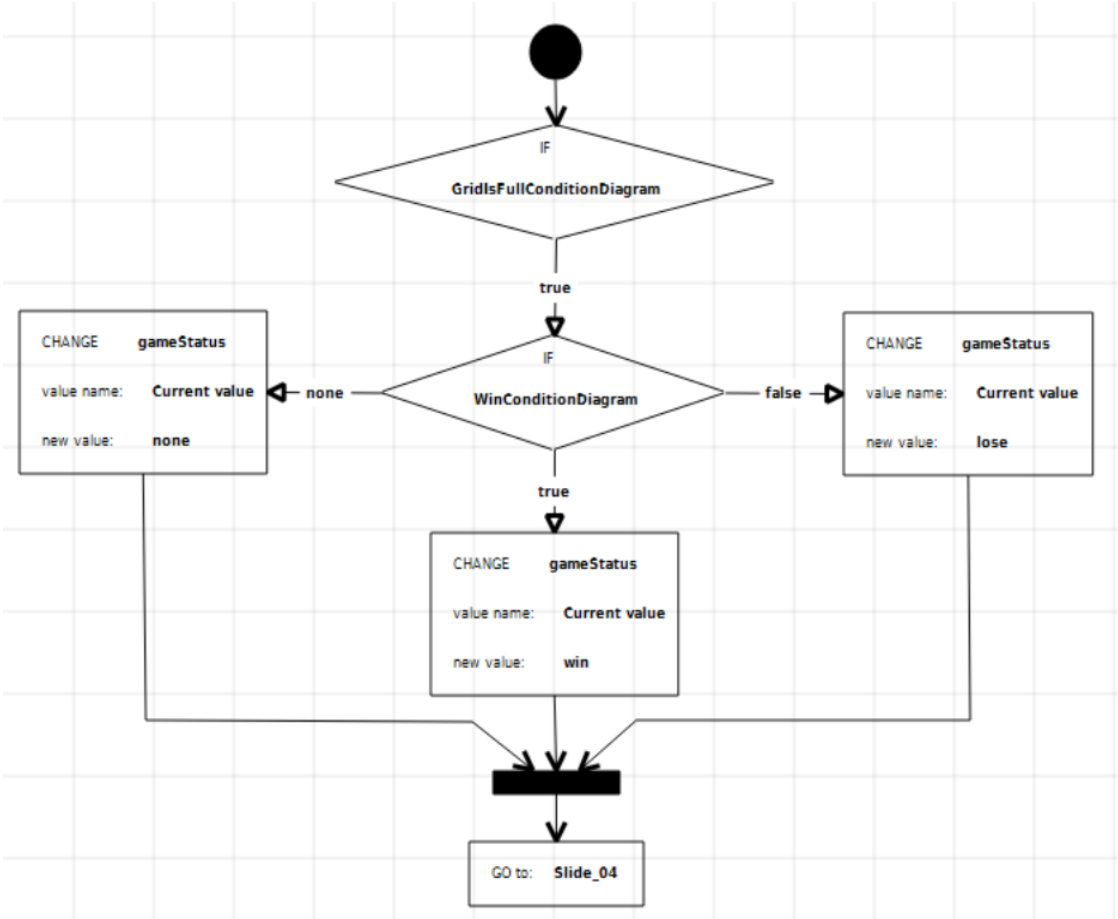
IF
GridIsFullConditionDiagram
true
CHANGE gameStatus
value name: Current value
new value: none
none
IF
WinConditionDiagram
false
CHANGE gameStatus
value name: Current value
new value: lose
true
CHANGE gameStatus
value name: Current value
new value: win
GO to: Slide_04

Рис. 6

На второй язык, описанный в работе Анны Дерипаски, были наложены следующие основные требования:

- сохранение наглядности разработки мобильных приложений
- отказ от требования к программисту обладать знаниями текстового языка, в который будет генерироваться приложение
- разделение задания логики поведения приложения и описания его внешнего вида

Описание программы состояло из нескольких диаграмм.

В этом наборе должна была быть одна диаграмма представления (Рис. 5), на которой схематично обозначены изображения различных экранов приложения (экран приветствия, экран выбора соперника) с располагающимися на них элементами упрвления и отображения информации: схематичные обозначения кнопок, игровых полей, полей ввода текста. Также на этой диаграмме можно было указать простейшие переходы между экранами.

Остальные диаграммы описывали логику обработчиков событий (Рис.6): логических действий, происходящих в системе при получении сообщения, например, о переходе на какую-то экранную форму или нажатии кнопки.

В работе Анны Дерипаски была реализована генерация прототипа приложения, состоящего из описанных экранных форм, простейших переходов между экранами и заглушек для обработчиков.

На разработанных языках не были реализованы детальные описания интересующих разработчиков Ubiq Mobile примеров, результаты было сложно оценить.

После изучения результатов предыдущих работ было замечено, что большое количество мобильных приложений хорошо представляется в виде автоматных переходов между состояниями. Основная идея, положившая основу создаваемому в этой работе языку, заключается в том, что этими

состояниями могут являться различные экраны приложения. Тогда графическое описание логики приложения будет заключаться в создании диаграмм активности пользователя — макетов экранов приложения и указанием переходов между ними.

Разработка языка

Основная идея языка - ограничить описываемые приложения теми, которые можно описать с помощью расширенной автоматной логики. Нетривиальным примером такого приложения является игра «Морской бой» для двоих игроков по сети, реализованная для платформы Ubiq Mobile.

Многие приложения для Ubiq Mobile уже написаны на таким образом. То есть основа приложения состоит из бесконечного цикла, в итерации которого происходит switch по состоянию приложения и, в зависимости от состояния отрисовывается соответствующий интерфейс. Смена состояния происходит в процедурах, вызываемых взаимодействием системы с окружающей средой. Эти же процедуры производят различные побочные действия, например формировка или отправка сообщений, обращения к серверу или базам данных.

Возможность описания такого приложения в графическом виде была основным критерием перспективности использования разработанного языка.

Первые итерации разработки языка были направлены на исследование кода игры «Морской бой» на предмет закономерностей и переиспользуемых абстракций. Изучение исходного кода показало, что любые изменения в системе происходят либо при получении сигнала: от интерфесов библиотек Ubiq Mobile или от элементов управления, либо через происходящий мгновенно по достижению состояния вызов процедур.

Описание языка

Первая версия языка совмещает два редактора — диаграмм активности

пользователя , содержащих изображения экранов с элементами управления и переходов между ними (один экран представляет одно состояние автомата), и редактора логики переходов: SDL-подобных диаграмм, позволяющих более подробно описывать изменения, происходящие при переходах, обозначаемых стрелками на диаграммах активности пользователя.

Одним из требований было обеспечение наглядного отображения макетов экранов (Рис.7). Для этого была предпринята попытка завершить проведенную Дмитрием Мордвиновым работу по включению поддержки виджетов сценой QReal. В результате стало возможным, например, использование кнопок с изменяемым названием и анимацией нажатия в качестве отображения элемента DSL.[7]

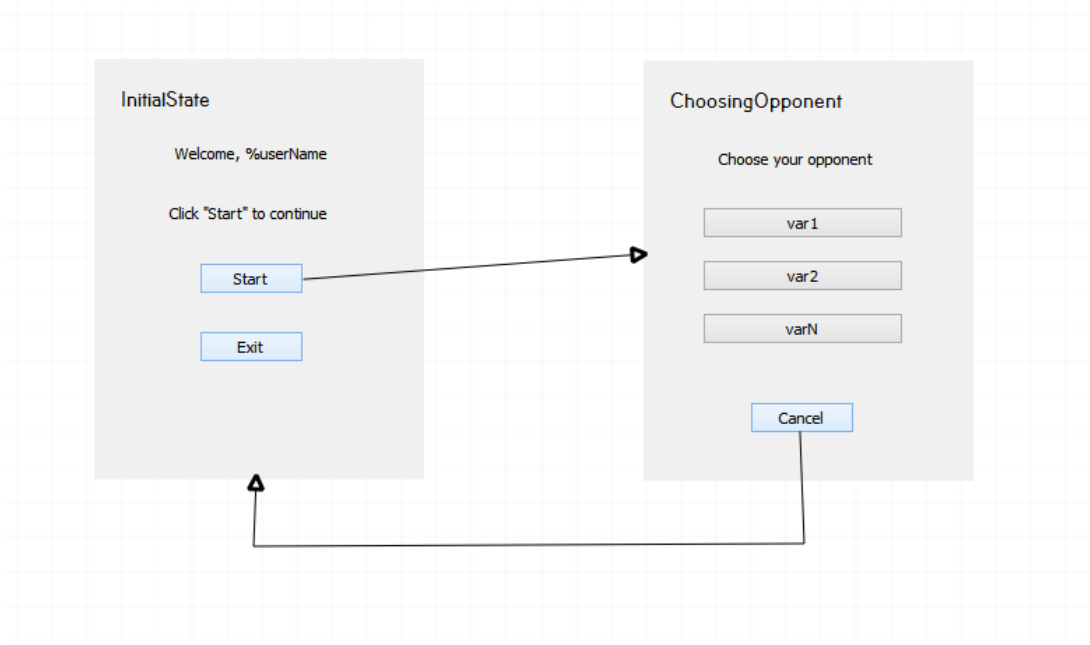

Рис. 7

Другим требованием к языку была реализации декомпозиции задачи через вложенность диаграмм, при этом, чтобы сохранялась понятность диаграмм и не пропадала возможность описывать систему на уровне, достаточном для генерации кода.

На диаграмме более высокого уровня поддиаграмма обозначается белым прямоугольником, в котором написано имя поддиаграммы (Рис.8). Все входящие в него стрелки обозначают переходы, концом которых являются состояния, содержащиеся в поддиаграмме, исходящие стрелки - переходы из состояний, описанных в поддиаграмме (Рис.8). Элемент, обозначающий поддиаграмму, через специальное контекстное меню связывается с другой диаграммой активности пользователя, на которой можно подробно описать инкапсулируемый участок программы (Рис. 9). На поддиаграмме стрелки, начало которых лежит снаружи, отображаются стрелками, начинающимися на специальном элементе «входной порт», отображаемом фигурной стрелкой «направо». Соответственно стрелки, начинающиеся в поддиаграмме, конец которых должен быть во внешней части системы — оканчиваются на элементе «выходной порт», изображаемым фигурной стрелкой «направо». Таким образом, при совпадении имён соответствующих стрелок во время генерации возможно проследить все необходимые связи.

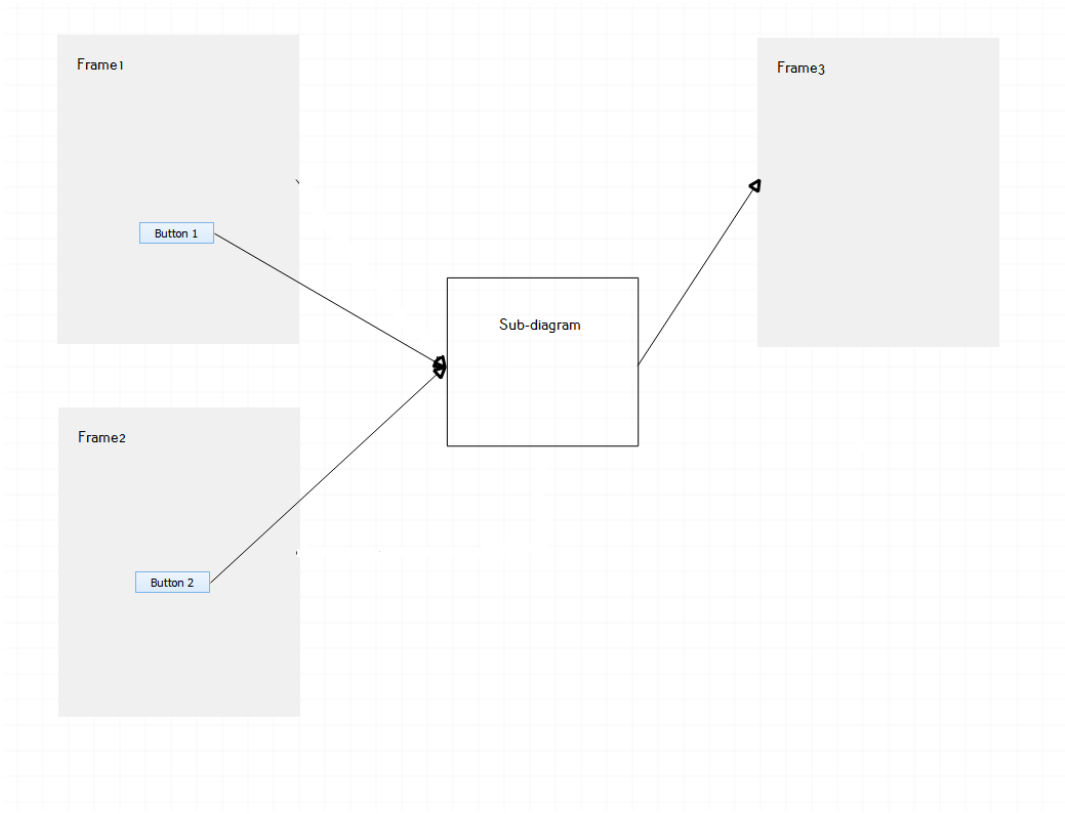

Рис. 8

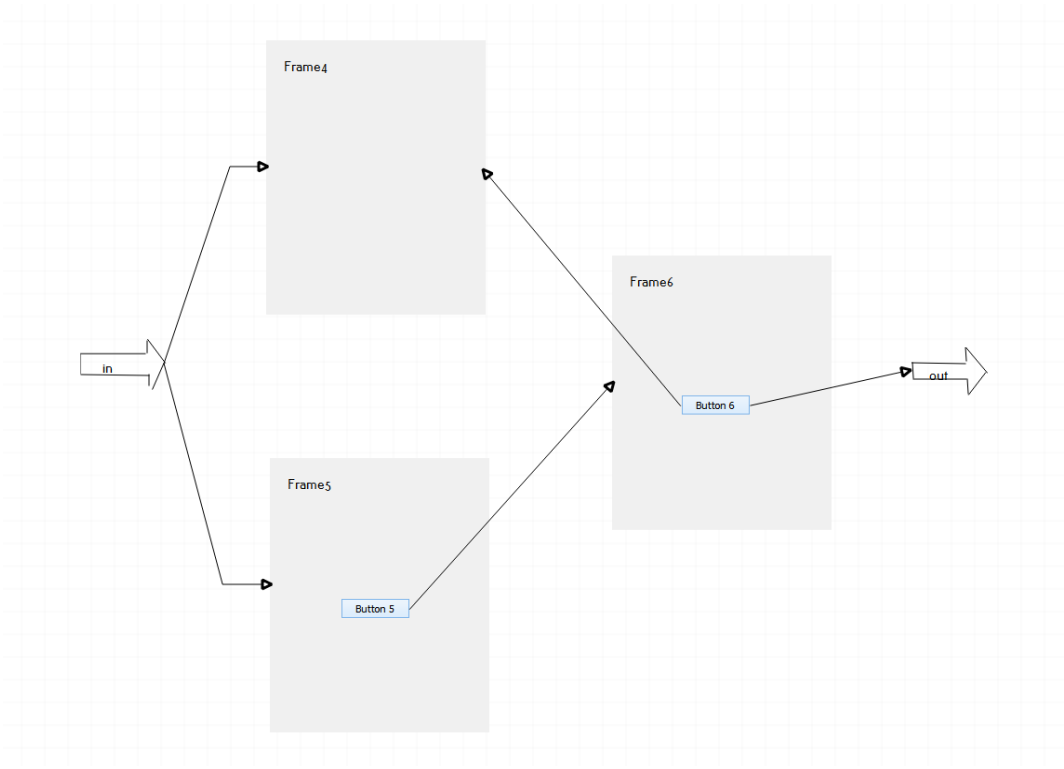
Frame4
Frame6
in
out
Button 6
Frame5
Button 5

Рис. 9

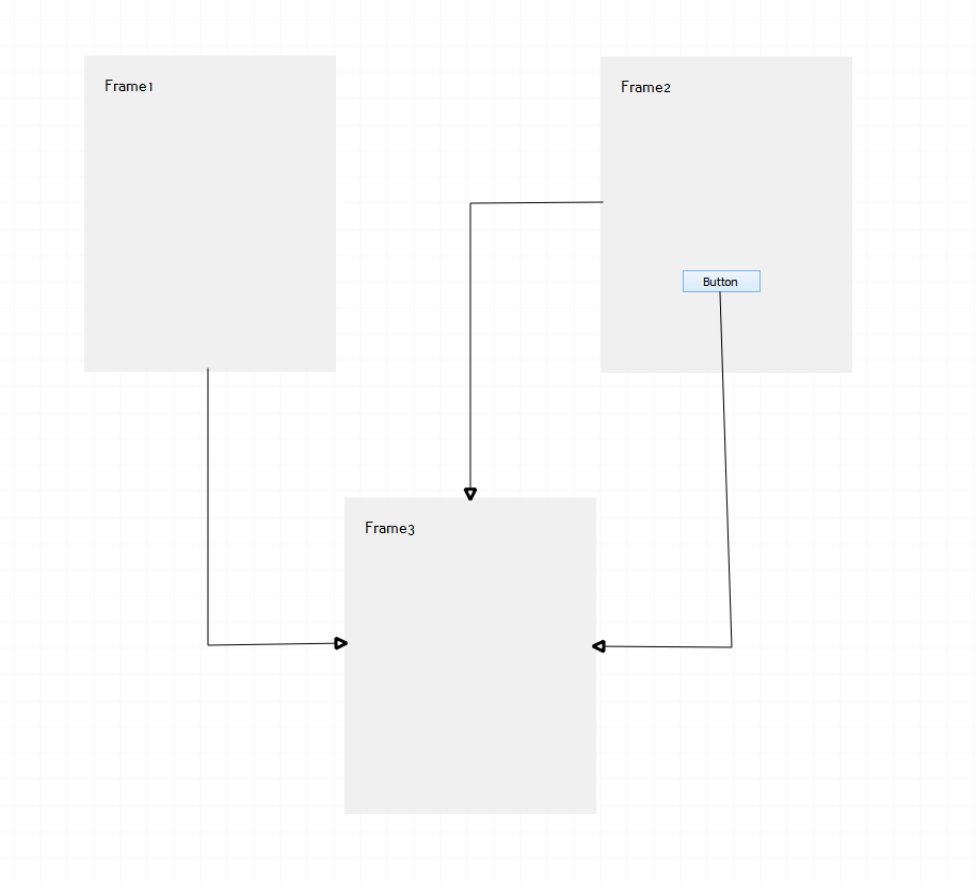
Frame1
Frame2
Button
Frame3

Рис. 10

Для выбора удобного графического отображения различных переходов были рассмотрены и классифицированы варианты инициации переходов. В предоставленном коде (gameTemplate, ship) все побочные эффекты, приводящие к смене автоматного состояния, можно было представить как функции, вызываемые либо по получении сигнала — от сетевого диспетчера или генерируемые нажатиями элементов управления, либо вызываемые мгновенно по переходу автомата в новое состояние. Была выбрана следующая графическая нотация (Рис.10), потому что она выглядит наиболее цельно и непротиворечиво. Стрелки, начинающиеся на элементах управления, обозначают переходы, инициация которых должна происходить по нажатию соответствующего элемента управления, стрелки прикреплённые к краю экрана обозначают переходы либо мгновенные (по входу в состояние), либо по сигналу, полученному системой, находящейся в данном состоянии. Различие между последними двумя проявляется на уровне диаграмм логики переходов, более подробно описывающих поведение системы.

Более подробное уточнение обозначаемых стрелкой переходов между экранами в графическом виде производится с помощью диаграмм описания логики переходов.

За основу диаграмм логики переходов брался язык SDL уровня описания процесса (Specification and Description Language) – это язык спецификаций, предназначенный для описания поведения реактивных и распределённых систем. На этом уровне иерархии языка SDL процесс описывается в форме, подобной блок-схемам, со специальными обозначениями отправки, ожидания сигналов.

Одна диаграмма описания логики содержит описание одного перехода и связывается с элементом «стрелкой», обозначающей этот переход на диаграмме активности пользователя, через контекстное меню.

На одной диаграмме размещается цепочка элементов, обозначающая

последовательность (с условными ветвлениями) действий, которые должны выполниться при совершении соответствующего перехода. Если стрелке не сопоставляется никакая диаграмма, то при переходе будет совершена только смена текущего состояния приложения и смена отображаемого экрана. Это соответствует диаграмме с единственным элементом «смена состояния».

Основная идея — разбиение логики на самые маленькие, относящиеся к смене исключительно одного состояния, части. Далее будут описаны элементы и соглашения, которые позволяют этого добиться.

Для наглядного разделения графического отображения сложной логики обработчиков элементов управления и сигналов на несколько более простых и понятных обработчиков, были введены специальные элементы. Они, в совокупности с местом начала описываемой стрелки на диаграмме активности пользователя, позволяют задать условие инициации выполнения перехода.

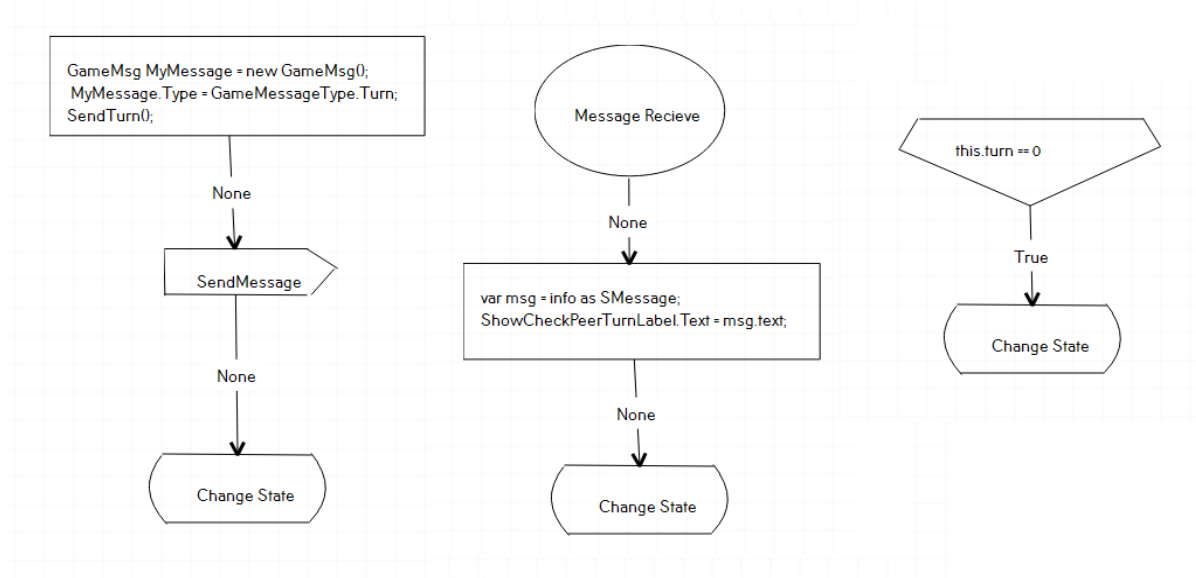

Рис. 11. Различные виды инициации перехода: (слева-направо) безусловный, по получению сигнала, по выполнению условия

Различными возможными вариантами инициации перехода могут быть:

- безусловный переход по нажатию элемента управления
- переход с условием по нажатию элемента управления

- переход из состояния по выполнении условия
- мгновенный переход из достигнутого состояния
- переход по получении сигнала.

Если первым элементом цепочки является охранник входа (Рис. 11, справа), то продолжение исполняется только при выполнении указанного логического условия.
Если первым элементом является указатель входящего сигнала (Рис. 10, посередине), то этот переход будет исполнятся только если система, находясь в необходимом состоянии, получит соответствующий сигнал от диспетчера.

Благодаря таким обозначениям инициации переходов возможно графически разделять сложные действия на несколько разных переходов, инициирующихся в зависимости от контекста одним и тем же элементом управления или сигналом.

Графическое описание мельчайших подробностей логики приводит к запутанным и громоздким диаграммам. Решение, применяемое в различных графических языках, состоит в том, чтобы добавить как один из элементов языка, блок, в который можно вписывать небольшие участки кода, которые удобнее представить в текстовом виде. Такой же элемент добавлен и в этот язык.

Обращения к API библиотек UbiqMobile выполнены в виде специальных блоков. Также есть блоки контроля управления — условный переход — ромб, который может иметь до двух выходов, с различными пометками 'True' и 'False', генерирует условный переход.

Добавление в начале цепочки специального элемента декларирует продолжение исполнения перехода исключительно по получении сигнала.

Редакторы

DSM-платформа QReal

QReal – платформа для создания графических DSL в течение нескольких лет разрабатывается на факультете математики и механики СПбГУ преподавателями и студентами.

Для разработки собственного графического языка с помощью платформы QReal необходимо описать метамодель желаемого языка. Описание метамодели происходит в графическом виде с помощью специального метаредактора.

Метаредактор позволяет описывать метамодель на высоком уровне абстракции. А именно на уровне «узел»-«связь», а также задавать различные синтаксические ограничения. Метаязык, позволяющий создавать диаграммы метамоделей, содержит следующие элементы:

- Metamodel Diagram – элемент, являющийся корневым. Язык может содержать несколько разных видов диаграмм, которые описываются элементами Metaeditor Diagram. Все эти элементы должны быть дочерними по отношению к Metamodel Diagram. Графически элемент метамодели отображается как пустая сцена, все элементы, относящиеся к описываемому языку, отображаются нарисованными на нём.
Также Metamodel Diagram содержит свойства, необходимые для генерации языка, например путь до подключённых внешних файлов или путь до места, в которое необходимо будет генерировать редакторы.

- Metaeditor Diagram — элемент, описывающий один редактор в языке, описываемом Metamodel Diagram. Графически отображается круглыми скобками. Всё, что имеет отношение к описываемому редактору, должно содержаться внутри этого элемента и быть по отношению к нему дочерним.

- Node – узел — элемент, описывающий класс элементов на целевой диаграмме некоторого редактора. Содержит в себе описание имени класса элементов, описание внешнего вида элементов этого класса, а также список свойств и их типов, которые они могут иметь.

- Enum – элемент, описывающий некоторый абстрактный тип перечисления, который можно использовать в качестве типа свойств классов элементов, описываемых Node. Содержит название перечисления, а также имена перечисляемых констант.

- Edge – элемент метадиаграммы, позволяющий описать связи целевой диаграммы, отображаемые стрелками. В свойствах возможно указать ограничения на классы элементов, которые можно соединить «стрелкой» описываемого типа, а также список свойств и их типов, которые связь может иметь.

Элементы метаязыка на диаграмме, описывающей требуемые редакторы, можно соединять следующими связями:

- Inheritance – связь, указывающая отношение наследования между элементами. Позволяет описывать общие свойства разных классов целевого редактора с помощью описания общего класса-предка.

- Container – связь, указывающая отношение вложения между элементами. Если один класс соединён этой связью с другим, то в целевой диаграмме при наложении элемента, принадлежащего к первому классу, на элемент, принадлежащий ко второму, он будет считаться вложенным. Графическое представление первого элемента станет дочерним по отношению к графическому отношению второго элемента, и при перемещении «контейнера» вложенные в него элементы будут перемещаться вместе с ним.

- Explodes to – связь, обозначающая раскрываемость элемента в диаграмму. Связь должна начинаться, на элементе Node или Edge,

описывающих тот или иной класс элементов в целевом редакторе, и заканчиваться на элементе Node, обозначающем редактор, в диаграмму которого первый элемент должен раскрываться.

При этом стоит отметить, что метаязык - это обычный графический язык в системе QReal со своей метамоделью, а, значит, он не использует каких-либо уникальных возможностей среды. Весь используемый им функционал QReal может быть использован и остальными разрабатываемыми в этой среде языками.

Для каждого элемента Node необходимо задать внешний вид, которым будут отображаться элементы этого типа на диаграммах целевого редактора. Для этого в редакторе форм для каждого элемента есть возможность нарисовать с помощью простейших инструментов (прямая, ломаная линия, овал, карандаш) внешний вид или использовать подгруженное изображение.

Ранее Дмитрием Мордвиновым была разработана реализация поддержки сценой QReal виджетов в качестве графического отображения элементов, а также расширение редактора форм, позволяющее задавать элементам целевого редактора вида с использованием виджетов. Эта разработка была интегрирована с новой версией QReal и использована для получения менее схематичных изображений элементов редактора активности пользователя.

Метамодель редакторов

Метамодели редакторов активности пользователя и логики переходов описывают следующие элементы языка:

№	Имя элемента	Описание элемента
Редактор активности пользователя		
1	ScreenFlowDiagram	Корневой элемент диаграммы редактора активности пользователя.
2	SubDiagam	Элемент, представляющий поддиаграмму редактора активности пользователя на диаграмме более высокого уровня вложенности. Раскрывается в отдельную диаграмму редактора активности пользователя.
3	InwardTransitions	Элемент, обозначающий входной порт на поддиаграмме, на котором начинаются стрелки, обозначающие переходы, начало которых находится вовне диаграммы.
4	OutwardTransitions	Элемент, обозначающий выходной порт на поддиаграмме на котором начинаются стрелки, обозначающие переходы, выходящие за пределы диаграммы.
5	Frame	Элемент, представляющий собой основу макета экрана. Каждому элементу этого типа ставится в соответствие состояние приложения. Может служить началом связи смены состояния.
6	StartNode	Маркер, обозначающий нулевое состояние системы.
7	ScreenTransition	Связь, обозначающая смену состояния. Изображается однонаправленной стрелкой. Обозначает переход системы из состояния, к которому прикреплено начало стрелки, в состояние, к

		которому прикреплён конец. Раскрывается в диаграмму редактора логики перехода
8	Button	Обозначает элемент управления «кнопка». Вкладывается в Frame, может служить началом смены состояния.
9	TextInput	Обозначает элемент интерфейса для ввода текста.
10	Label	Обозначает элемент интерфейса, предназначенный для вывода текста на экран.
11	List	Обозначает view на специальный элемент управления, показывающий список доступных игроков.
Редактор логики переходов		
1	LogicEditorDiagram	Корневой элемент диаграммы редактора логики переходов.
2	Edge	Связь, обозначающая последовательность выполнения действий, обозначаемых остальными элементами диаграммы.
3	EntryGuard	Один из возможных начальных элементов, обозначающий выполнение перехода в зависимости от выполнения заданного логического условия.
4	DialogInvite	Возможные начальные элементы, обозначающие выполнение перехода при получении конкретного сигнала диспетчера
5	DialogInviteReject	
6	DialogUserRemove	
7	DialogUserAdd	
8	MessageRecieved	
9	Guard	Обозначает логическое условие с возможностью выбора исполняемой ветви диаграммы в зависимости от его истинности или ложности.
10	CodeArea	Элемент, который может содержать код на C#.

11	EndState	Элемент, обозначающий конец выполнения перехода со сменой текущего состояния.
12	Terminate	Элемент, обозначающий обрыв выполнения перехода без смены текущего состояния.
13-23	Login, Register, SendMessage, ...	Элементы обозначающие вызовы интерфейсов библиотек Ubiq Mobile.

Генератор

Главной целью работы было создать прототип графической технологии разработки приложений для платформы Ubiq Mobile. После разработки графического языка и создания редакторов осталось сделать генератор кода для платформы Ubiq Mobile. Успех при генерации кода по графическим диаграммам обозначил бы перспективность графического подхода в разработке приложений.

Внутреннее представление диаграмм

Диаграммы, создающиеся в редакторах, разработанных на основе QReal, представлены в виде набора элементов, хранящихся в репозитории.

В QReal реализовано разделение элементов на логическую и графическую части. Логическая часть должна быть у любого элемента и содержит его свойства. Графическая часть — конкретное отображение элемента на диаграмму, при этом у элемента может не быть графической части или их может быть несколько.

И графические элементы, и логические образуют отдельные иерархии. Иерархия логических элементов может задаваться в окне обозревателя логической модели диаграммы, иерархия графических элементов соответствует вложенности элементов на диаграмме.

Различные предоставленные интерфейсы репозитория элементов позволяют перемещаться по деревьям элементов, а также проводить по ним поиск.

Плагины в QReal

Meta-CASE технология QReal позволяет создавать генераторы кода для разработанных с его помощью редакторов. Также содержит широкий спектр возможностей по созданию других расширений к редакторам.

Разрабатываемым плагинам QReal предоставляет специальный интерфейс — ToolPluginInterface. При создании плагина, унаследованного от ToolPluginInterface, разработчику предоставляется доступ к следующим интерфейсам:

- ProjectManagerInterface – функции работы с проектом, например диалоги открытия и сохранения проектов.
- SystemEventsInterfave – возможность подключиться к различным сигналам системы, например, смена открытой закладки или закрытие диаграммы.
- RepoControlInterface – интерфейс репозитория элементов диаграмм. Позволяет, например, сохранять отдельные диаграммы или проводить поиск по всем элементам.
- LogicalModelApi – работа с логическими элементами модели: создание элементов, а также получение и изменение их свойств.
- GraphicalModelApi – работа с графическими элементами модели.
- MainWindowInterpretersInterface – интерфейс, предоставляющий функционал для создания интерпретаторов диаграмм, например, выделение элементов.

Генераторы в QReal

Для поддержки создания генераторов кода к редакторам, созданным в meta-CASE системе QReal, есть базовый класс генератора abstractGenerator, предоставляющий основную функциональность и инфраструктуру для генерации одного файла.

Позволяет загрузить и использовать файл с набором шаблонов, соответствующих различным элементам языка. Предоставляет интерфейс для создания и сохранения сгенерированных файлов.

Предлагаемый процесс генерации кода заключается в изменении загруженных шаблонов в зависимости от элементов, находящихся на диаграмме, и составлении из параметризованных шаблонов результирующего файла.

Созданный генератор для разработанного языка описания мобильных приложений позволяет генерировать автоматную логику описанного приложения. Генерируются объявления состояний и основное тело программы, состоящее из переключения по этим состояниям в бесконечном цикле. Также по всем диаграммам логики перехода генерируются функции, совершающие соответствующие переходы. Генерируются объявления описанных элементов управления, вызовы функций подключаются к соответствующим сигналам.

Апробация

В разработанных редакторах для полученного графического языка были реализованы диаграммы, описывающие логику приложения «Морской бой». Приложение «Морской бой», как и многие другие пошаговые игры для платформы Ubiq Mobile, является расширением игрового шаблона.

Игровой шаблон gameTemplate позволяет переиспользовать в играх код модулей, отвечающих за сетевое общение игроков, основу логики смены ходов, регистрации и идентификации. При использовании игрового шаблона разработка игры в основном направлена на создание логики игрового поля, условий победы и логической составляющей хода игрока.

В разработанных редакторах были сделаны диаграммы, описывающие логику игры «Морской бой». Разработанный генератор позволил получить по диаграммам значительную часть автоматной логики приложения.

После обсуждения полученного результата с разработчиками Ubiq Mobile ими был предложен другой, более интересный им и более простой для проверки, содержательный пример — описать в графическом виде игровой шаблон gameTemplate.

Разработанного языка было достаточно для получения диаграмм, описывающих игровой шаблон в деталях, позволивших с помощью разработанного генератора получить значительную часть автоматной логики приложения.

Полученная диаграмма активности пользователя изображена на рисунке 12. При этом логическая нагрузка переходов в полученном языке изображается короткими и простыми диаграммами (Рис. 13)

Рис. 12 Диаграмма активности пользователя игрового шаблона

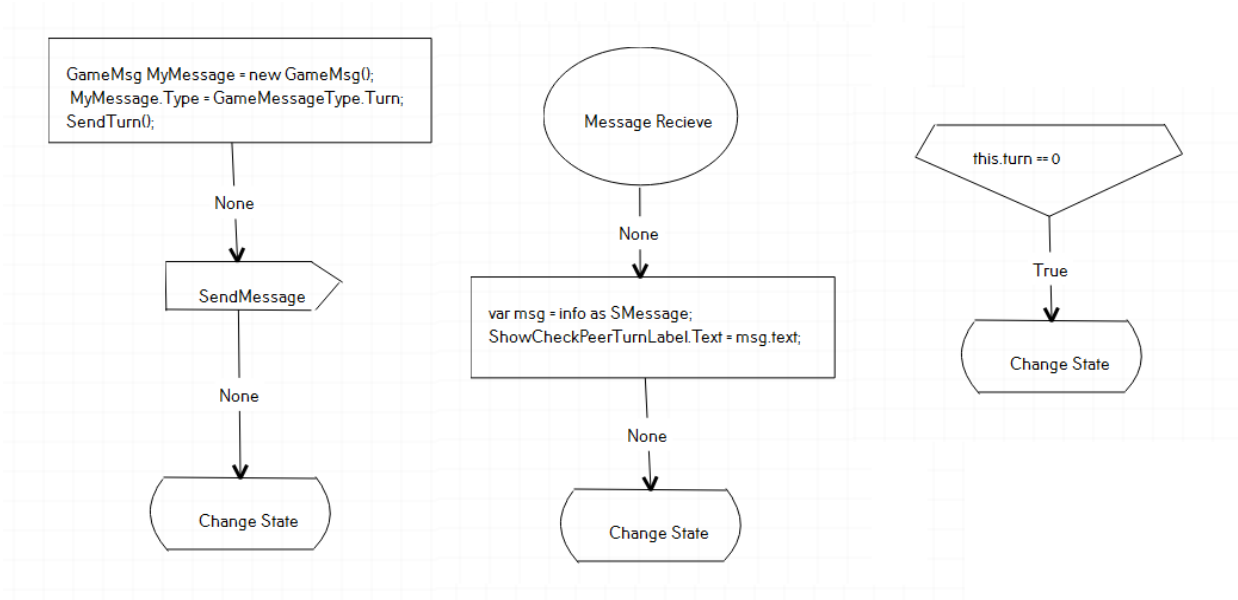

Рис. 13 Примеры диаграмм логики переходов игрового шаблона

Заключение

В результате выполнения данной работы был разработан графический DSL для описания мобильных приложений. Реализованы два графических редактора для диаграмм этого языка.

Реализован генератор кода для платформы Ubiq Mobile по графическим диаграммам.

Выполнены диаграммы, содержащие графические описания логики игры «Морской бой» и игрового шаблона, что подтверждает перспективность применения подобного подхода.

Дальнейшее развитие

Взаимодействие с разработчиками Ubiq Mobile помогло выявить приоритетные направления развития. Были сформулированы дальнейшие требования к языку для решения более узких задач. А также осознана важность для возможности промышленного использования решения наличия инструментов, включающих в себя графическую отладку и круговую разработку.

Список литературы

1. А.Н.Терехов, В.В.Оносовский. Технология разработки мобильных онлайн сервисов. // Конференция CEE-SECR 2011. С. 1-2
2. Martin Ward. Language Oriented Programming. // Computer Science Department, Science Labs, 1994. С. 14-16
3. Искандер Гиниятуллин. Создаем быстрый прототип мобильного приложения, http://habrahabr.ru/post/189524/
4. Терехов А.Н., Литвинов Ю.В., Брыксин Т.А. Среда для обучения информатике и робототехнике QReal:Robots // Девятая независимая научно-практическая конференция «Разработка ПО 2013» (CEE SEC(R)-2013), Москва, 24 октября 2013 года.
5. Timofey Bryksin, Yuri Litvinov, Valentin Onossovski, Andrey N. Terekhov. Ubiq Mobile + QReal a Technology for Development of Distributed Mobile Services // 10th Conference of Open Innovations Association FRUCT and the 2nd Finnish-Russian Mobile Linux Summit: Proceedings, printed by State University of Aerospace Instrumentation (SUAI). 2011. 232 p. Pp 27-35.
6. Дерипаска А.О. Визуальный язык для платформы Ubiq Mobile в среде QReal, http://se.math.spbu.ru/SE/YearlyProjects/2013/YearlyProjects/2013/445/445-Deripaska-report.pdf
7. Мордвинов Д.А. Средства разработки пользовательских интерфейсов в DSM-платорфме QReal, http://se.math.spbu.ru/SE/diploma/2013/s/MordvinovDmitry_thesis.pdf