Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Интерпретация метамоделей в metaCASE-системе QReal

Курсовая работа студентки 345 группы
Птахиной Алины Ивановны

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

Санкт-Петербург
2012

Содержание

Введение

Программная инженерия заинтересована в средствах, позволяющих сделать процесс разработки более простым и удобным. Все большее внимание уделяется визуальному программированию, в котором программа представляется в виде набора диаграмм. Визуальное программирование позволяет моделировать ПО с разных точек зрения. Визуальные модели описывают отдельные аспекты ПО, что позволяет не рассматривать все многообразие предметной области, а сосредоточиться лишь на некоторых ее свойствах. В целом это упрощает процесс разработки и делает его более наглядным.

Для реализации такого подхода применяются CASE-системы[1]. Термин CASE (Computer Aided Software Engineering) появился в индустрии разработки ПО в начале 1980-х годов. CASE-системы предоставляют средства генерации кода и используются в качестве инструментов для разработки, анализа и проектирования программного обеспечения.

Для упрощения создания CASE-систем были разработаны metaCASE-системы, позволяющие автоматически генерировать CASE-системы по описанию языка. MetaCase-системы применяются для создания предметно-ориентированных языков, с помощью которых можно решать конкретные задачи.

Язык, используемый для создания других языков моделирования, называется метаязыком. Метаязык строится на основе некоторой модели, содержащей описание всех абстракций, которые нужны при моделировании. Модель языка моделирования называется метамоделью.

Одним из инструментов, предназначенных для создания специализированных сред визуального программирования, является QReal - проект научно-исследовательской группы кафедры системного программирования СПбГУ. QReal является metaCase-системой, которая позволяет автоматически генерировать код произвольных визуальных редакторов по описаниям их метамоделей.

На практике часто возникают задачи, при автоматизации решений которых естественным образом возникает необходимость описывать сложные, нетривиальные структуры. Для этого довольно часто возникает потребность видоизменения существующих или добавления новых сущностей языка. Однако визуальные редакторы в основном либо предоставляют пользователю фиксированный набор диаграмм и сущностей, либо добавление новых и изменение существующих затруднительно.

QReal в настоящий момент использует генеративный подход к созданию редакторов. Это неудобно, поскольку затрудняет процесс изменения метамодели: при любом изменении метамодели требуется перегенерация кода и пересборка редактора, на что уходит много времени. Поскольку хотим менять метамодель быстро, например, если хотим попробовать несколько вариантов синтаксиса языка, то мы хотим избежать этих затрат и сделать так, чтобы метамодели интерпретировались, а не генерировались в код на C++. Кроме того, это позволит изменять и дополнять метамодель прямо в процессе моделирования.

Таким образом, задача заключается в следующем: реализовать интерпретатор метамодели, который предоставит возможность интерпретирования визуальных языков, соответствующий следующим требованиям:

- должен являться компонентой QReal
- на вход принимает сохраненную метамодель
- эмулирует функциональность сгенерированного редактора

Обзор существующих решений

Существует множество различных средств, позволяющих использовать в процессе разработки информационной системы языки моделирования, созданные специально для предметной области, в которой будет функционировать система. Такие языки называются предметно-ориентированными языками (DSL, Domain Specific Language), а их использование – предметно-ориентированным моделированием (DSM, Domain-Specific Modeling). Однако немногие среди них используют подход, основанный на интерпретации метамоделей, а не на генерации на их основе исходного кода системы. В качестве обзора рассмотрим именно такие DSM-средства.

MetaEdit+

MetaEdit+ является средой для создания и использования предметно-ориентированных языков, которая была разработана в рамках научно-исследовательского проекта MetaPHOR в University of Jyväskylä как расширение среды MetaEdit, созданной ранее проектом SYTI в сотрудничестве с компанией MetaCase в период с конца 1980-х – начала 1990-х годов. Среда MetaEdit ограничивалась созданием в текущий момент только одного модельного языка для одного пользователя и обеспечивала моделирование лишь диаграмм. MetaEdit+ поддерживает несколько встроенных языков моделирования, обеспечивает одновременный доступ к среде для нескольких пользователей и создание диаграмм, матриц и таблиц.

MetaEdit+ содержит встроенный MetaEdit+ Workbench, предоставляющий простой, но мощный язык метамоделирования и инструмент для проектирования модельного языка[2]. Используя данный инструмент, можно определять основные понятия языка, их атрибуты, взаимосвязи между понятиями и различные правила интеграции нескольких языков в одной системе.

При построении метамодели в MetaEdit+ используется язык метамоделирования GOPRR[3], получивший свое название от основных понятий, которыми он оперирует:

- граф (Graph) – создаваемый язык моделирования (совокупность объектов и связей);
- объект (Object) – сущность модельного языка;
- свойство (Property) – атрибут какого-либо объекта графа;
- связь (Relation) – задает отношение между объектами;
- роль(Role) – определяет, каким образом объект участвует в связи.

MetaEdit+ позволяет изменять метамодели, а, следовательно, и модели, созданные с их помощью, без перезапуска приложения. Таким образом, пользователь может внести изменения в описание метамодели, вновь импортировать ее в MetaEdit+ Workbench и продолжить работу над моделью, при этом данная DSM-платформа сама произведет все необходимые изменения в соответствующих моделях.

Однако существенным недостатком MetaEdit+ является то, что эта DSM-платформа для экспорта моделей использует свой собственный формат файлов (MXT), который отличается от общепринятого стандарта – XML, а это сказывается на открытости данной технологии.

MetaLanguage

В 2008 году в межвузовском сборнике научных статей “Математика программных систем” была опубликована статья Л.Н.Лядовой и А.О.Сухова “Языковой инструментарий системы MetaLanguage”[4]. В статье описывались базовые элементы метаязыка. В этом же сборнике была и другая статья А.О.Сухова под названием “ Среда разработки визуальных предметно-ориентированных языков моделирования”[5]. К сожалению, в открытом доступе средство MetaLanguage отсутствует, однако данные статьи заслуживают внимания, поскольку свидетельствуют о том, что исследовании в данном направлении велись и были получены результаты. На основе указанных статей и рассмотрим данную систему.

Среда разработки MetaLanguage включает следующие компоненты:

- графический редактор – используется для создания, изменения, удаления моделей, а также установления взаимосвязей между различными моделями, описанными с помощью определенных графических нотаций;
- браузер объектов – инструментальное средство, предназначенное для просмотра и редактирования информации, хранящейся в репозитарии;
- репозитарий – хранилище, содержащее информацию о метамоделях, моделях, сущностях, отношениях, атрибутах, ограничениях, пиктограммах, используемых для отображения сущностей и отношений; фактически представляет собой реляционную БД, может работать в многопользовательском режиме, для согласованности действий пользователей использует механизм транзакций;
- валидатор – проверяет соответствие модели ограничениям, заданным пользователем;
- генератор – на основе имеющихся моделей генерирует XML-файл, содержащий информацию о модели, и документацию, включающую название модели,

информацию о разработчиках, которые создавали ее, графическое представление модели со ссылками на описание отдельных частей.

При открытии метамодели из репозитария загружаются все модели, созданные с помощью этой метамодели, все сущности, отношения метамодели и соответствующих моделей. При удалении метамодели удаляются все модели, созданные на ее основе. При внесении изменений в метамодель для поддержания системы в согласованном состоянии изменения вносятся и в зависимые от нее модели.

Для работы, как с моделями, так и с метамоделями, используется один и тот же инструментарий. Процесс создания модели является итеративным: создав некоторый язык, можно использовать его как метаязык для создания другого языка, который в свою очередь также может быть использован как метаязык и т.п. Таким образом, MetaLanguage является инструментом для создания визуальных динамически настраиваемых предметно-ориентированных языков моделирования.

REAL и QReal

Данная тема вызывала ранее интерес и была затронута в дипломных работах нашей кафедры системного программирования Санкт-Петербургского Государственного Университета, а именно в работах:

- М.А.Бакалов “Эффективная реализация расширяемой метамодели CASE-средства на основе UML 2.0”[6];
- А.Косякин “Реализация UML 2 на основе технологии REAL”[7];
- Е.И.Такун “Реализация режима быстрого прототипирования в CASE-системе QReal”[8].

Рассмотрим подробнее данные работы.

CASE-пакет, поддерживающий стандарт UML 2.0

Задачей, которой занимались М.А.Бакалов и А.Косякин, являлось создание новой версии CASE-пакета REAL - разработки компании ЗАО “Ланит-Терком”[9]. Новая функциональность включала в себя интеграцию и поддержку UML 2.0, многопользовательский репозиторий, многоплатформенность и более совершенный интерфейс. Однако требовалось произвести слишком много изменений, несовместимых со старой версией, вследствие чего было принято решение написать новое CASE-средство, по возможности переиспользуя существующую систему. Остановимся подробнее на реализации поддержки UML 2.0. Стандарт UML описывает 13 типов диаграмм, на каждой из которых в среднем содержится от 5 до 10 элементов. В сумме получается более 100 похожих объектов. Каждый из этих элементов, помимо реализации специфичной логики,

требовалось нарисовать средствами выбранного языка программирования. Реализовав несколько редакторов диаграмм этого стандарта, разработчики пришли к выводу, что дальнейшее движение в данном направлении является трудоемкой работой по кодированию однотипных задач. Была предложена идея описания и интерпретации метамодели стандарта, что позволило бы свести реализацию языка к созданию его графической метамодели.

В работе был использован подход, применяющийся самими авторами UML 2.0: модели – конкретные диаграммы, описывались на визуальном языке UML, который был описан на визуальном языке MOF (Meta Object Facility). Таким образом, получается, что метамодель метаязыка MOF являлась метаметамоделью языка UML.

Более детально подход состоял в следующем:

- на языке программирования C++ было реализовано небольшое подмножество MOF, которого было достаточно для описания большинства метамоделей;
- на основе подмножества MOF был реализован графический редактор диаграмм, в котором пользователь мог изменять и создавать собственные метамодели. Таким способом можно было реализовать и метамодель UML 2.0;
- созданные метамодели сохранялись в репозиторий, после чего на их основе пользователь мог создавать собственные модели: графический редактор пользовательских диаграмм интерпретировал метаинформацию из репозитория.

Для описания графического представления элементов на диаграмме использовался специально разработанный в рамках данной работы формат, основанный на стандарте W3C SVG (Scalable Vector Graphics, XML – представление векторной графики). Описания фигур создавались пользователем в любом графическом редакторе, который поддерживал сохранение рисунков в формате SVG, после чего доводились вручную в текстовом редакторе для задания связей с моделью и сохранялись в репозиторий. В процессе отрисовки текст такого расширенного SVG-формата интерпретировался встроенным в CASE-средство модулем, также реализованным в рамках данной работы.

К сожалению, после написания диплома работа над данным CASE-средством прекратилась и продолжена не была.

QReal

В системе QReal созданием режима быстрого прототипирования и переходом от концепции генерируемых метамоделей к концепции интерпретируемых метамоделей в рамках дипломной работы занималась Е.И.Такун.

На момент написания диплома в системе QReal уже был реализован встроенный метаредактор, с помощью которого можно было изменять и создавать новые метамодели. Созданные в нем метамодели генерировались в код на C++ - плагин, который в дальнейшем использовался при загрузке и в процессе работы редактора. При внесении изменений в метамодель требовалась перегенерация редактора. Такой подход удобен, когда сущности и их свойства, которые будут использоваться при создании языка, заранее известны. Если же при решении той или иной задачи пользователь захочет изменить существующие элементы или добавить новые, то ему потребуется изменить метаязык, а значит переключаться между редактором и метаредактором и перегенерировать метамодели. При внесении большого количества мелких изменений усилия, затраченные при этом, будут не пропорциональны полученному результату. В таких случаях используется метамоделирование «на лету» и режим быстрого прототипирования, которые были поддержаны в системе QReal и апробированы на реальной задаче в результате дипломной работы. Все это стало возможным благодаря реализации концепции интерпретируемых метамоделей.

Однако в то время полученные результаты встроены не были, а в данный момент встраивание невозможно, поскольку в QReal произошло слишком много изменений. Поэтому было принято решение реализовать данную концепцию с нуля.

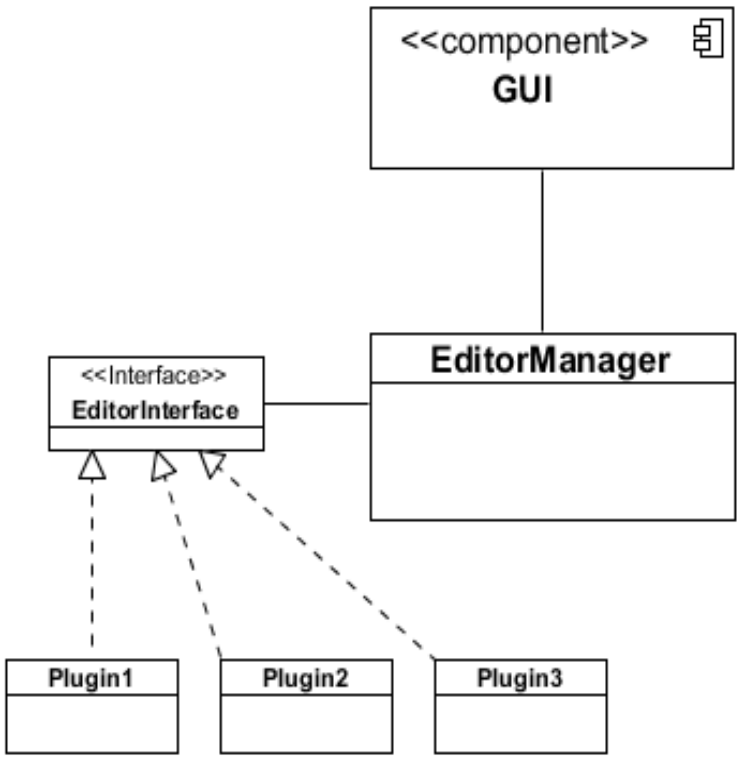
Реализация

Метамодель хранится в формате сохранений QReal, который представляет собой сжатые наборы XML-файлов. Он содержит всю необходимую информацию о метамодели, которая в дальнейшем генерируется в XML-файл, который, в свою очередь, используется для генерации редактора. Для изменения метамодели можно, как уже было сказано ранее, использовать встроенный метаредактор, а можно и напрямую вносить изменения в соответствующий xml-файл метамодели.

Рассмотрим как реализуются генерация и поддержанная в результате этой курсовой работы интерпретация метамодели в metaCASE-системе QReal.

Генеративный подход

На момент написания курсовой работы в системе QReal был реализован подход для работы с метамоделями, принцип работы которого поясняет диаграмма классов, представленная на рисунке:

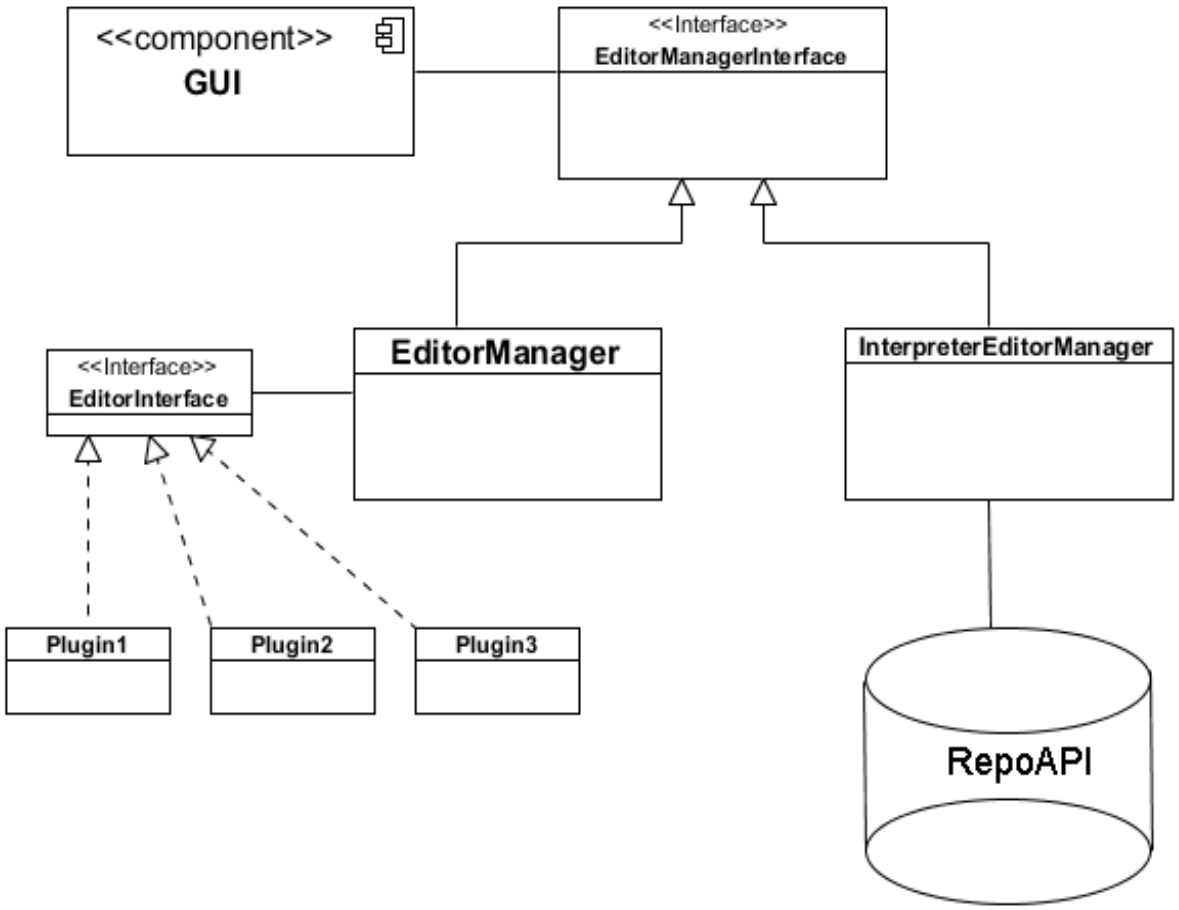


Данный подход заключается в следующем: графический пользовательский интерфейс (GUI) для отображения информации, касающейся языка, такой как: имеющиеся сущности, их свойства и графическое представление, использует класс EditorManager. EditorManager содержит в себе набор редакторов, каждый из которых реализует интерфейс EditorInterface. Реализация редакторов генерируется во время компиляции приложения с помощью xml-парсера из файла метамодели в код на C++ - плагины по

описанию метамоделей. Во время запуска приложения библиотечные файлы плагинов используются для загрузки редакторов.

Интерпретативный подход

Интерпретация моделей происходит непосредственно во время работы приложения, в то время как генерация обязана завершиться до начала работы. По этой причине при разработке интерпретатора потребовалось несколько видоизменить существующую архитектуру. Рассмотрим произведенную модификацию на диаграмме:



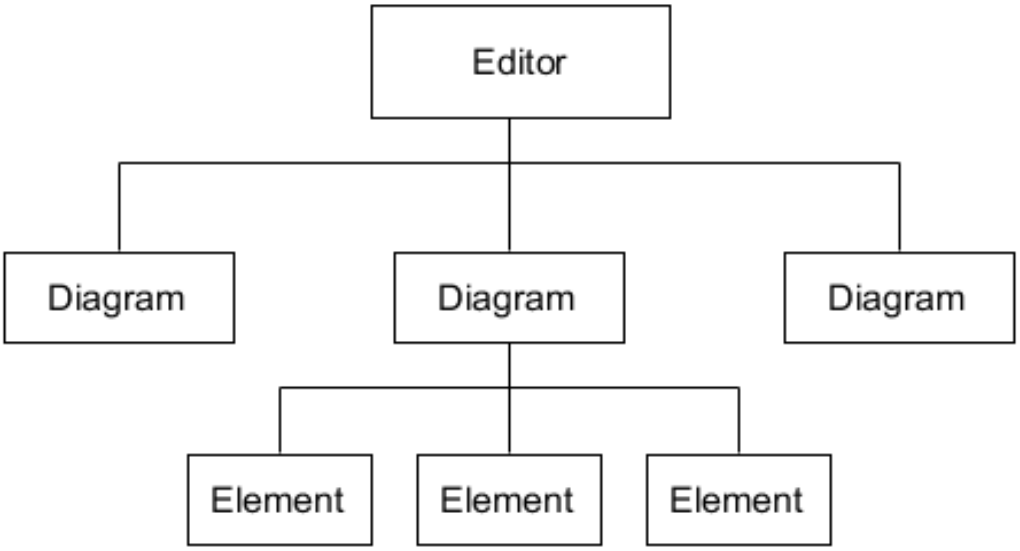
Из класса EditorManager был выделен интерфейс EditorManagerInterface, за реализацию которого отвечают наш старый класс EditorManager и класс интерпретатора, который был реализован в течение данной курсовой работы. Он работает по несколько иному принципу. Рассмотрим отличия на примере одного из конкретных методов: “получение списка элементов диаграммы, имеющейся в данном языке”.

При использовании концепции генерируемых метамоделей данный метод реализуется следующим образом: генератор по метамодели генерирует методы, каждый из которых содержит жёстко зашитые в код перечисления каких-то свойств, например, “список всех элементов диаграммы”. В связи с этим, чтобы получить этот список,

достаточно просто найти нужный плагин по имени (а генератор ещё и генерирует плагину метод, возвращающий его имя), и вызвать этот самый сгенерированный метод у него.

Класс интерпретатора же для получения идентичной информации имеет доступ к репозиторию с сохраненной метамоделью. По имеющемуся названию диаграммы в нем находится элемент метамодели, соответствующий данной диаграмме, и берутся имена всех сыновей для получения списка элементов.

Для большей наглядности реализации данного подхода приведем иерархическую структуру объектов QReal:



Как видно из диаграммы выше, редактор состоит из набора диаграмм, каждая из которых в качестве сыновей содержит элементы диаграммы, список которых и необходимо было получить в приведенном выше примере.

Задание графического представления элемента

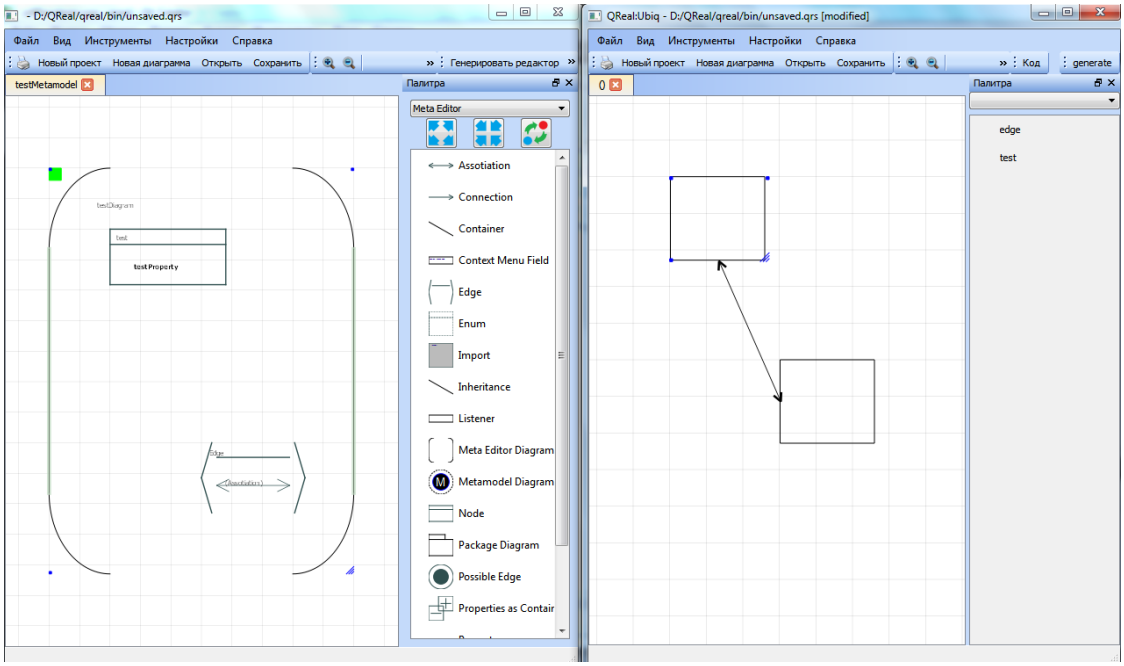
Метамодель задаёт абстрактный синтаксис элементов создаваемого языка. Однако для изображения объектов языка и получения их графического представления данной информации недостаточно, нужен конкретный синтаксис.

Графические элементы бывают двух типов: “узлы” и “связи”. Все они используют для отрисовки интерфейс ElementImpl. В генеративном подходе данный интерфейс реализовывался каждым элементом языка в сгенерированном коде. В интерпретативном подходе сгенерированный код отсутствует, поэтому пришлось реализовать класс, который бы рисовал элемент по его xml-описанию, которое он берет из метамодели. В этом описании содержится информация о графическом представлении элемента, которая представляется в формате SDF. Этот формат был создан в рамках одной из курсовых

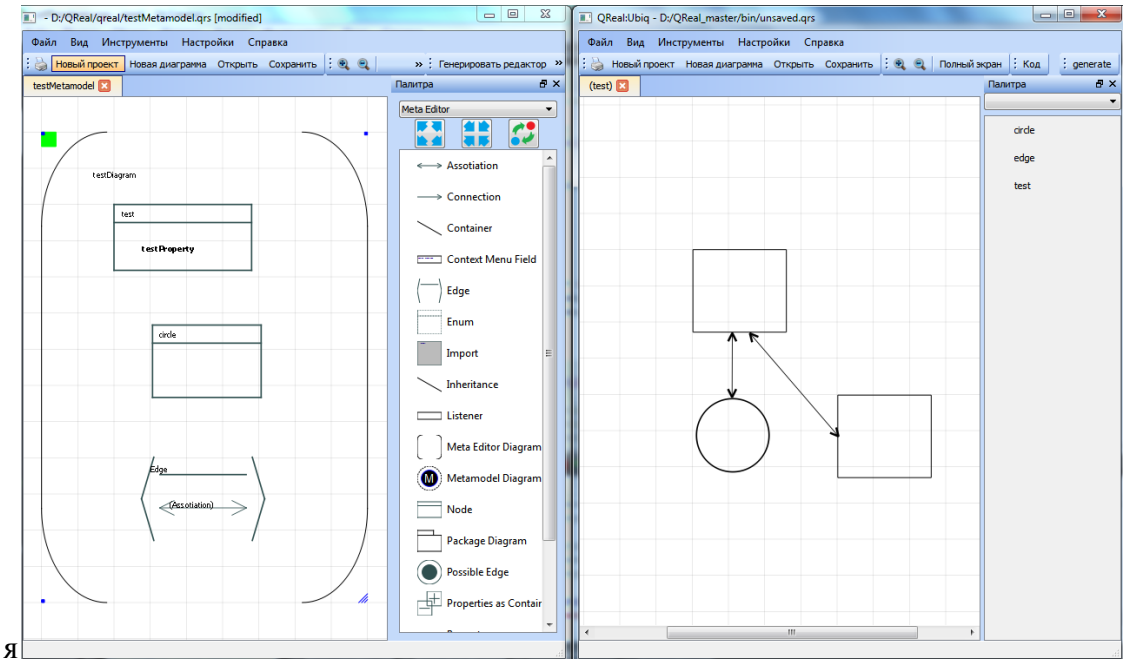
работ в качестве расширения имеющегося формата SVG. Он позволяет задавать дополнительные графические свойства элемента, такие как масштабируемость, список доступных для редактирования текстовых элементов и многие другие. Для задания самого изображения объекта используется встроенный в QReal редактор фигур, который позволяет как загружать имеющиеся рисунки в формате SVG, так и создавать новые.

Апробация интерпретативного подхода

Для апробации метода интерпретации метамодели была создана в метаредакторе тестовая метамодель, состоящая из двух элементов: узла под названием “test” и ассоциации “edge”. После она была проинтерпретирована, и в результате получили модельный язык, который предоставляется на рисунке ниже. Слева изображена метамодель, открытая в метаредакторе, а справа - нарисованная с её помощью модель.



Далее метамодель изменили, добавили новый узел “circle”, задали с помощью редактора фигур ему соответствующее изображение и измененную метамодель передали интерпретатору. В результате интерпретации все изменения были корректно отображены, и новый объект модели был добавлен в палитру:



Я

Результаты

В рамках курсовой работы были получены следующие результаты:

- был реализован класс интерпретатора InterpreterEditorManager, который для входной метамодели, сохраненной в формате XML, производил ее интерпретацию и тем самым эмулировал функциональность сгенерированного редактора;
- для получения графического объекта была произведена реализация интерфейса ElementImpl, который интерпретирует описание графических свойств объектов metaCASE-системы QReal;
- была произведена апробация интерпретатора на тестовой метамодели.

Список литературы

1) Д.Кознов, Основы визуального моделирования, Бином. Лаборатория Знаний, Интернет-Университет Информационных Технологий, 2008

2) Domain-Specific Modeling with MetaEdit+. URL: http://www.metacase.com/ (дата обращения: 16.05.2012)

3) А.О.Сухов, А.А.Кандакова, Е.С,Попова, Обучающий видео-курс по работе с инструментальным средством MetaEdit+, рекламно-техническое описание. URL: https://docs.google.com/viewer?a=v&q=cache:HsTIW78qt9MJ:ofernio.ru/rto_files_ofernio/17010.doc+&hl=ru&gl=ru&pid=bl&srcid=ADGEEShfmUwzo4iK3wPit4NuNMq-dZr9PBiwP5CY3SU01B3dSIQt1ody3O55vRlFGDGKLXRKBx8PXWfWeasotq_UJYesnxb5fWv3ufBgbArWP6E-rqVsrERz1loI6d4zX1Yv_6Dte_YJ&sig=AHIEtbR1MqYHW-gesYxX8QL0XeGkx1PKHw

(дата обращения: 16.05.2012)

4) Л.Н. Лядова, А.О. Сухов, Языковой инструментарий системы MetaLanguage, Математика программных систем: межвуз. сб. науч. ст. / М34 Перм. гос. ун-т. – Пермь, 2008.

5) А.О. Сухов, Среда разработки визуальных предметно-ориентированных языков моделирования, Математика программных систем: межвуз. сб. науч. ст. / М34 Перм. гос. ун-т. – Пермь, 2008. С. 84-94

6) М.А.Бакалов, Эффективная реализация расширяемой метамодели CASE-средства на основе UML 2.0, дипломная работа, СПбГУ, кафедра системного программирования, 2007 http://se.math.spbu.ru/SE/diploma/2007/index_html (дата обращения: 16.05.2012)

7) А.Косякин, Реализация UML 2 на основе технологии REAL, дипломная работа, СПбГУ, кафедра системного программирования, 2007 http://se.math.spbu.ru/SE/diploma/2007/index_html (дата обращения: 16.05.2012)

8) Е.И.Такун, Реализация режима быстрого прототипирования в CASE-системе QReal, дипломная работа, СПбГУ, кафедра системного программирования, 2011 http://se.math.spbu.ru/SE/diploma/2011 (дата обращения: 16.05.2012)

9) Отдел CASE – технологий. Технология Real. URL: http://real.tepkom.ru/REAL.php (дата обращения: 16.05.2012)