Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Системное программирование

Зайнуллин Егор Евгеньевич

Разработка редактора REAL.NET

Курсовая работа

Научный руководитель:
к. т. н., доцент Литвинов Ю.В

Санкт-Петербург
2019

Оглавление

Введение

Графические языки применимы во многих сферах, начиная от робототехники и обучения школьников программированию и заканчивая промышленностью. При этом языки такого плана изучаются достаточно быстро из-за их наглядности. А значит программирование с их использованием может быть гораздо проще и эффективнее обычных текстовых языков общего назначения в некоторых случаях, например, при работе с конкретной предметной областью [4]. Но, с другой стороны, создание языка с нуля для каждой конкретной области может быть неоправданно дорого. Для того чтобы упростить его разработку, существуют DSM-платформы[1]. На данный момент существует множество инструментов для создания графических языков, таких как: MetaEdit+ [3], Eclipse MP [6], а также DSM-платформа, разработанная в СПбГУ, QReal [8]. Несмотря на указанные преимущества, визуальные языки встречаются не часто. Таким образом, получается, что достаточно удобных инструментов для их создания нет [7].

По этой причине было принято решение создать DSM-платформу REAL.NET, с помощью которой разработка нового графического языка не требовала бы много времени и затрат.

Постановка задачи

Целью работы является реализация части основных функций редактора моделей, в частности для этого необходимо:

- реализовать «умную» панель свойств, то есть отображение свойств элементов модели, изменения их значений, а также валидацию этих значений;

[1]Инструмент для быстрого создания визуальных языков

- реализовать возможность создавать ребра, которые не будут соединены концом с вершиной, откреплять их от вершин, перетаскивать;
- поддержать возможность сериализации и десериализации графических составляющих элементов модели: формы, цвета, вида, отображения свойств.

1. Обзор

1.1. Существующие решения

В настоящее время существует несколько инструментов для создания графических языков. Рассмотрим самые популярные их них.

1.1.1. MetaEdit+

MetaEdit+ [3] — это среда для создания и использования предметно-ориентированных языков, разработанная учеными из Финляндии.

1. Состоит из двух компонентов: первая отвечает за метамодель[2], вторая поддерживает работу с моделью.

2. Имеет свой собственный язык для построения метамодели. Содержит редактор для форм элементов для модели. Поддерживает метамоделирование «на лету», то есть можно изменять язык прямо в процессе его использования.

3. Язык, созданный с помощью MetaEdit+, будет интерпретируемым.

Основные недостатки MetaEdit+ заключаются в том, что он является коммерческим проектом, не имеет открытого исходного кода и не поддерживает расширения.

1.1.2. Eclipse Modeling Project (EMP)

EMP [6]— фреймворк, основанный на Eclipse, для генерации кода на Java, создания и отображения моделей.

1. Состоит из множества проектов, основным из которых является Eclipse Modeling Framework (EMF). С помощью EMF происходит разработка метамоделей для новых визуальных языков.

[2]Модель, которая описывает структуру и принципы действия другой модели

2. Метамодели задаются благодаря использованию специального языка Ecore. Элементами этого языка являются по сути Java классы.

3. Сама модель может быть задана множеством способов, например: Java код, UML диаграммы, XML и так далее.

Громоздкость EMP, его привязка к Eclipse и Java — его основные недостатки.

1.1.3. QReal

QReal [8]— инструмент с открытым исходным кодом, разработанный кафедрой системного программирования СПбГУ, предназначенный для быстрого создания графических языков.

1. С помощью специального редактора разрабатывается метамодель нового языка. Поддерживается метамоделирование «на лету».

2. Расширение функциональности осуществляется посредством добавления новых плагинов

3. QReal использует генеративный подход.

4. На его основе была создана платформа для программирования роботов TRIK Studio.

QReal тяжеловесен — написано большое количество строк кода, и является основой TRIK Studio, которая, в свою очередь, имеет десятки тысяч пользователей по всему миру. Следовательно, кардинально менять ядро платформы не предоставляется возможным. Таким образом, QReal стал неудобен для экспериментов, для научного применения [8].

1.1.4. Microsoft Modeling SDK

Microsoft Modeling SDK [1] — компонент Visual Studio, позволяющий создавать простые графовые языки. Метамодель создается с помощью специального языка. После генерации можно вручную изменить код

на C#, чтобы добавить дополнительную функциональность. Работать с созданным языками необходимо прямо в Visual Studio.

Таким образом, создать независимый редактор от Visual Studio и C# невозможно. При этом стоит отметить, что поддержка этого проекта Microsoft практически прекратилась.

1.2. Что уже реализовано

На данный момент коллегами было сделано некоторое количество шагов для создания редактора:

- реализован простой редактор моделей, отображение моделей осуществляется посредством библиотеки GraphX. Она является достаточно функциональной и использует графическую подсистему .NET Framework Windows Presentation Foundation (WPF), которая, в свою очередь, умеет взаимодействовать с технологией DirectX[3],что позволяет красиво рисовать графы и использовать новый графический интерфейс;

- была создана панель свойств, отображающая их как строки в таблице (рис. 1);

- изначально ребра можно было создавать только поочередно указывая две вершины на сцене[4], первая из них становилась началом ребра, вторая – концом.

Name	Type	Value
delay	Int	3000
instanceMetatype	String	Metatype.Node
isAbstract	Boolean	false

Рис. 1: Первоначальная панель свойств

[3]Набор API для работы с графикой и мультимедиа

[4]Элемент редактора, на котором отображается модель

2. Описание решения

2.1. Панель свойств

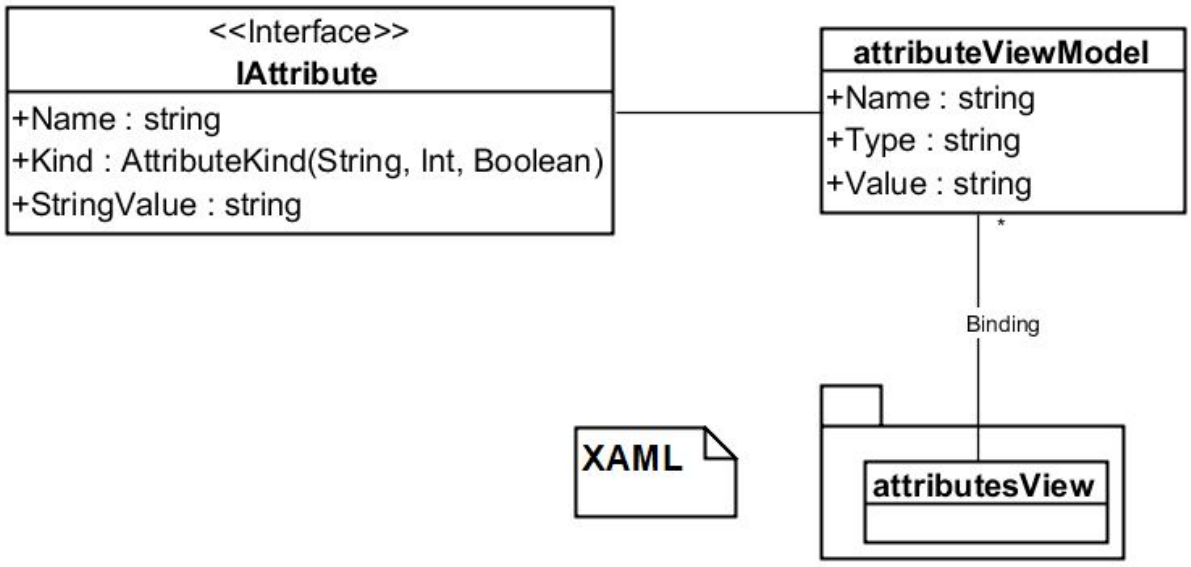

Рис. 2: Атрибуты на панели свойств

Для построения приложения с графическим интерфейсом Microsoft рекомендует применять паттерн[5] Model-View-ViewModel [9]. Описание графических элементов в WPF пишется на языке разметки XAML, основанном на XML – эта часть называется View. Бизнес-логика приложения, то есть Model, содержится в коде на C#. Для того, чтобы связать эти два компонента используется промежуточная составляющая ViewModel – класс, предоставляющий данные View для отображения и обновляющий модель в случае необходимости.

Панель привязана к списку свойств через класс AttributeViewModel (рис. 2). Панель представляет собой контрол[6] DataGrid, который представляет из себя таблицу. Отсюда, для того чтобы значение выглядело надлежащим образом,в XAML описывается специальный шаблон DataGridTemplateColumn, который отвечает за вид ячейки, который меняется при срабатывании триггера в зависимости от столбца Type.

[5]Подход к решению задачи по проектированию

[6]Примитив графического интерфейса пользователя, имеющий стандартный внешний вид и выполняющий стандартные действия

Также происходит проверка данных, чтобы они соответствовали типу с помощью определения валидатора, в частности, проверяется, что строка не содержит запрещенных символов, а число содержит только цифры – в View указывается имя класса, специальный метод которого вызывается после заполнения значения(рис. 3).

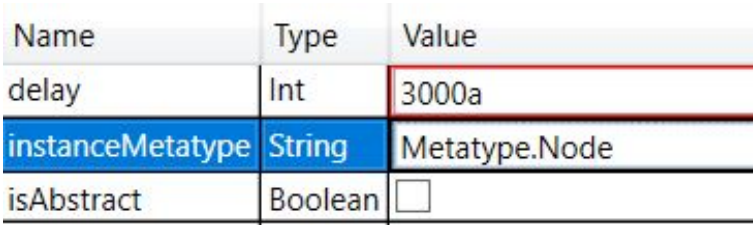

Рис. 3: Полученная панель свойств

2.2. Перетаскивание ребер

Чтобы программу в редакторе можно было легко изменять, необходимо было создать поддержку перетаскивания ребер, но в связи с особенностями библиотеки GraphX, а также репозитория[7], создавать ребра можно только указав вершины – концы. По этой причине было принято решение добавить возможность создавать временные «виртуальные» вершины. Для этого в классе NodeViewModel, отвечающем за хранение данных о вершинах, появилось булево свойство IsVirtual. Если при нажатии на сцену мышь не указывает ни на вершину, ни на ребро и на панели элементов[8] выбрано ребро, то вызывается метод сцены, который создает контрол, содержащей в себе NodeViewModel с истинным значением IsVirtual.

Чтобы присоединить ребро, надо перетащить его конец, то есть «виртуальную» вершину, на другую. При отпускании кнопки мыши вызывается обработчик, который с помощью VisualTreeHelper.HitTest [2] проверяет то, что центр первой вершины попадает на контрол второй(рис. 4). Для отсоединения ребра вызывается метод UnpinEdgeFromVertex.

[7]Хранилище моделей и метамоделей

[8]Список элементов метамодели, которые могут быть добавлены на сцену

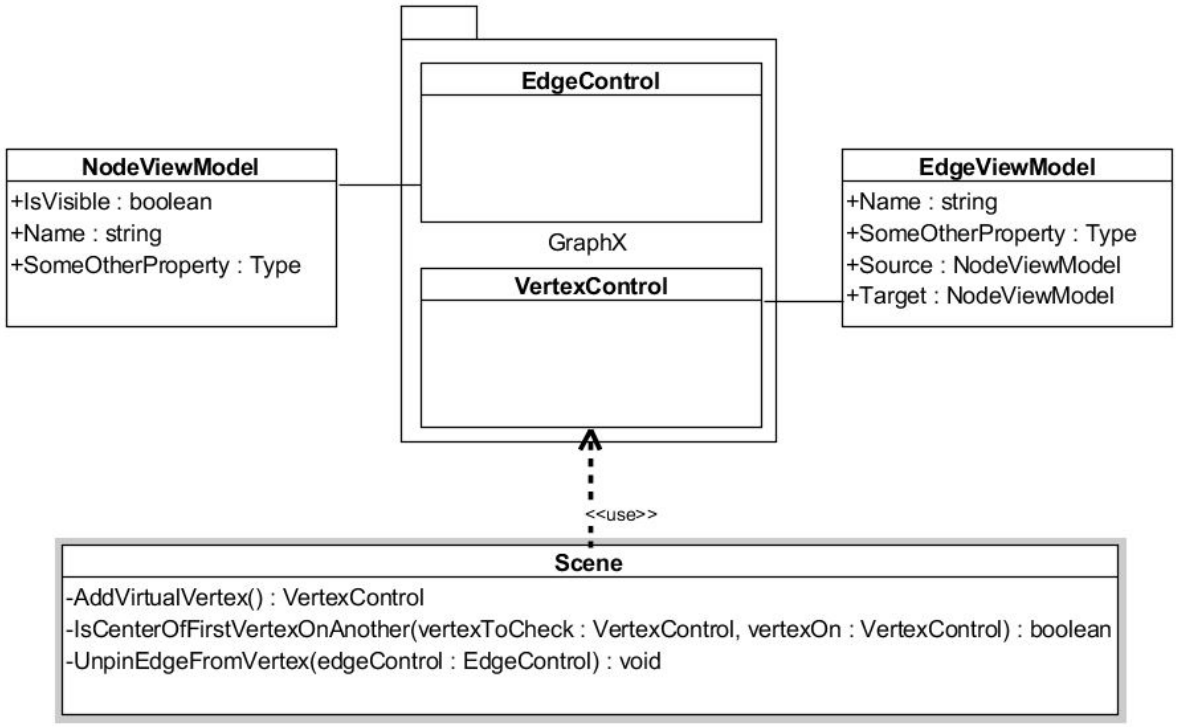

Рис. 4: Новые методы и свойства для перетаскивания ребер

2.3. Сериализация и десериализация графической составляющей элементов

Для того, чтобы пользователь платформы мог легко редактировать форму, размер, цвет, расположение свойств на экране, было принято решение хранить эту графическую информацию в виде файлов простого формата, например, XML. По этой причине необходимо было поддержать возможность сохранения в этот формат и чтения из него, то есть сделать сериализацию и десериализацию данных об отображении(рис. 5).

В репозиторий были добавлены абстракции, хранящие соответствующую информацию: IVisualInfo запоминает ссылку на файл и его тип, IAttributeView и IElementView содержат в себе данные о виде атрибута и элемента на сцене соответственно(рис. 6). Класс ViewFactory отвечает за извлечение данных из файла: он создает объект, реализующий интерфейс IElementView, то есть, например, он использует XMLParser для чтения, и обработки XML файла и извлечения из него информа-

ции(рис. 7). Класс ViewFileGenerator по IElementView генерирует соответствующий код(рис. 8).

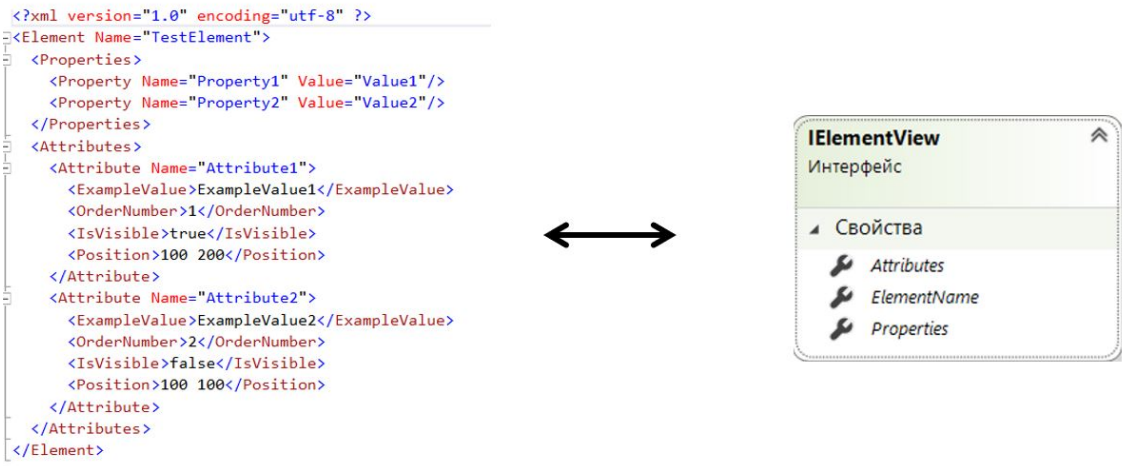

Рис. 5: Иллюстрация сериализации и десериализации

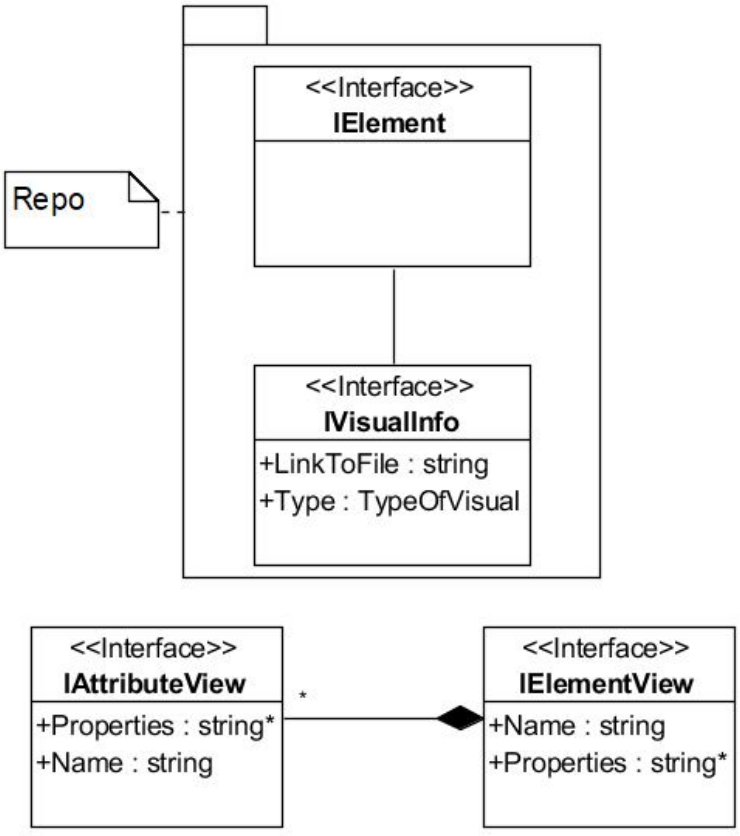

Рис. 6: Репозиторий

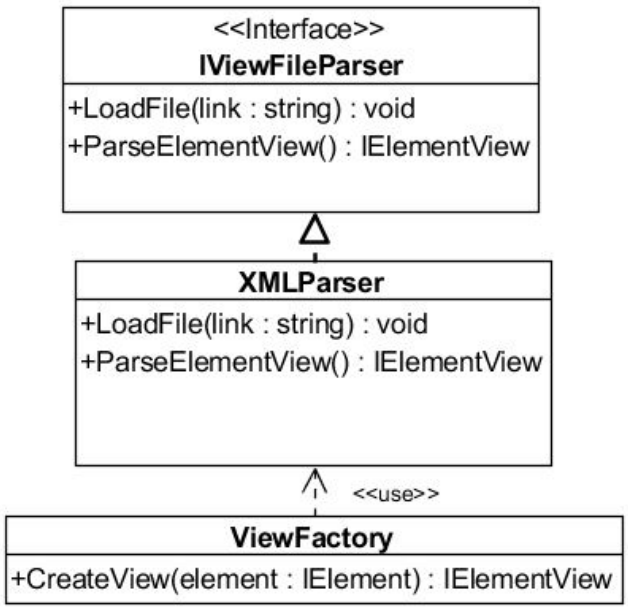

Рис. 7: Архитектура работы с файлами, десериализация

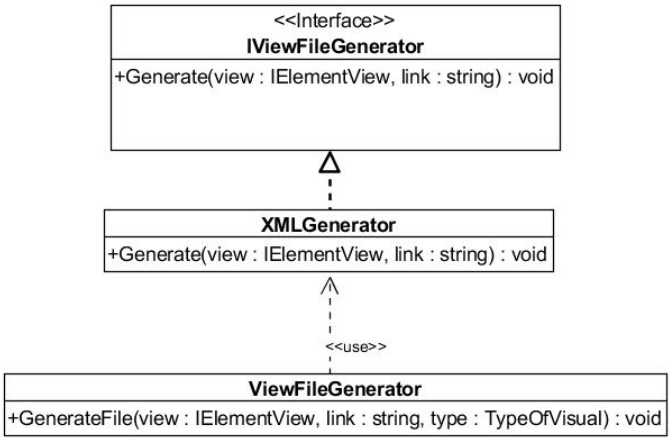

Рис. 8: Архитектура работы с файлами, сериализация

3. Апробация

I. Для тестирования панели свойств были выполнены следующие тестовые сценарии:

   1. Выбран некоторый элемент на сцене, изменены его свойства разных типов. Далее был выделен другой элемент, после этого – изначальный. Отмечено, что свойства сохранили свои значения после изменения;
   2. Проверена валидация целочисленного и строкового типа – на данный момент в репозитории поддерживаются только простейшие типы.

II. Для тестирования перетаскивания ребер были выполнены следующие действия:

   1. Создано ребро без привязки к вершине;
   2. Перетащен конец к другой вершине, произошло присоединение;
   3. Ребро было отсоединено и удалено со сцены;
   4. Было создано множество ребер так, чтобы из одной вершины шло несколько;
   5. Была удалена вершина, было отмечено, что ребра были удалены с ней;
   6. Было создано ребро от одной вершины к другой;
   7. Была добавлена новая вершина;
   8. Был перетащен конец созданного ребра к добавленной вершине.

III. Для проверки корректной работы сериализации и десериализации файлов было сделано:

   1. Было создано вручную несколько XML файлов;

2. Произведен их разбор;
3. Результат этого разбора был записан и проверен на соответствие полученных данных с информацией, заключенной в XML файл;
4. Далее было создано несколько объектов, реализующих интерфейс IElementView;
5. Произведена их сериализация;
6. Потом была произведена их десериализация;
7. Было проверено, что изначальные и полученные объекты эквиваленты.

Заключение

Были сделаны следующие новые функции редактора:

- реализована «умная» панель инструментов, изменяющая свое отображение в зависимости от типа и поддерживающая валидацию;
- реализована поддержка перетаскивания ребер;
- добавлена возможность сериализации и десериализации графических данных в XML формате.

Ссылка на репозиторий проекта в GitHub [5].

Список литературы

[1] Domain-specific development with Visual Studio DSL Tools / S. Cook, G. Jones, S. Kent, A.C. Wills. — Crawfordsville, Indiana, USA : Addison-Wesley, 2007. — 576 p.

[2] Hit Testing in the Visual Layer. — URL: `https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/graphics-multimedia/hit-testing-in-the-visual-layer` (дата обращения: 25.12.2018).

[3] Kelly S., Tolvanen J.-P. Advanced Tooling for Domain-Specific Modeling: MetaEdit+ // Internet. — URL: `http://www.dsmforum.org/events/DSM07/papers/tolvanen.pdf` (online; accessed: 15.06.2018).

[4] Kelly S., Tolvanen J.-P. Visual domain-specific modeling: Benefits and experiences of using metaCASE tools // International Workshop on Model Engineering. — 2000.

[5] Домашняя страница REAL.NET. — URL: `https://github.com/yurii-litvinov/REAL.NET` (дата обращения: 13.12.2018).

[6] Сорокин А., Кознов Д. Обзор проекта Eclipse Modeling Project // Системное программирование. — 2010. — С. 6–31.

[7] Литвинов Ю.В., Кузьмина Е.В., Небогатиков И.Ю., Алымова Д.А. Среда предметно-ориентированного программирования REAL.NET // Список. — 2017. — URL: `https://github.com/yurii-litvinov/articles/blob/master/2017-realNet/realNet.pdf` (дата обращения: 09.12.2018).

[8] Терехов А.Н., Брыксин Т.А., Литвинов Ю.В. QReal: платформа визуального предметно-ориентированного моделирования // Программная инженерия. — 2013. — № 6. — С. 11–19.

[9] Шаблон Model-View-ViewModel. — URL: `https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/`

enterprise-application-patterns/mvvm (дата обращения: 25.12.2018).