СРАВНИТЕЛЬНЫЙ АНАЛИЗ РЕДАКТОРОВ ВИЗУАЛЬНЫХ МОДЕЛЕЙ

Золотарева М.А.

Национальный исследовательский университет «Высшая школа экономики», Пермский филиал, факультет бизнес-информатики (614060, Пермский край, г. Пермь, бульвар Гагарина, 37а), e-mail: hayumi@rambler.ru

Процесс разработки программного обеспечения всегда требовал больших затрат, в частности, на глубинное изучение предметной области. При проектировании часто возникает потребность в создании множества моделей предметной области. Визуальное программирование, которому сейчас уделяется большое внимание, позволяет упростить создание программы путем представления отдельных ее аспектов в виде диаграмм. Использование при создании DSL языкового инструментария значительно упрощает процесс создания языков. Существуют DSM-платформы (также называемые metaCASE-системами), которые позволяют создавать визуальные языки при помощи парадигмы визуального моделирования. В работе формулируются требования к графическим редакторам, используемым для разработки визуальных моделей и метамоделей (например, MetaEdit+, MS DSL Tools, Eclipse GMF, QReal и др). Приводится сравнительный анализ основных существующих metaCASE-систем (DSM-платформ) по выделенным критериям.

Ключевые слова: визуальное моделирование, графический редактор, metaCASE-система, создание программы.

Margarita Zolotareva

National Research University Higher School of Economics, Perm Branch, Department of Business Informatics (614060, Perm, Gagarina Boulevard, 37a), e-mail: hayumi@rambler.ru

Software development process has always required high costs, particularly for deep study of the subject area. A need to create a plurality of domain models often arises at designing. Visual programming, which now received more attention, allows to simplify the creation of the program by providing some of its aspects in the form of diagrams. The use of a language tools in creating DSL toolkit greatly simplifies the process of creating languages. There are DSM-platforms (also called metaCASE-systems), which allow you to create visual languages using the paradigm of visual modeling. In this paper we formulate the requirements for the graphic editor for creating visual models and metamodels (eg, MetaEdit +, MS DSL Tools, Eclipse GMF, QReal etc.). The comparative analysis of the main existing meta-CASE-systems (DSMplatform) via dedicated criteria.
Keywords: visual modeling, graphics editor, metaCASE-system, development of program.

Введение

Тенденции развития современных информационных технологий приводят к постоянному усложнению автоматизированных систем. Для снижения трудоёмкости и повышения качества сложных проектов в настоящее время созданы системы автоматизированного проектирования самих программных продуктов.

Модельно-ориентированная разработка является бурно развивающейся областью программной инженерии. Использование моделей и предметно-ориентированных языков в процессе создания информационных систем позволяет разрабатывать гибкие, настраиваемые системы, а также привлекать к процессу их построения не только профессиональных разработчиков, но и экспертов в предметных областях, конечных пользователей.

Важную роль играют исследования различных методов упрощения анализа предметной области, формализации. Одним из таких методов может служить визуальное моделирование [2].

Визуальное моделирование

Визуальным моделированием называют процесс графического представления модели с помощью некоторого набора графических элементов (рис. 1) [15].

Визуальное моделирование обычно ассоциируется с использованием UML. В UML (Unified Modeling Language) определяют графические нотации для различных диаграмм (диаграммы классов, диаграмм кооперации и диаграммы состояний и т. д.). Комбинируя различные типы диаграмм UML можно определить полную модель приложения.

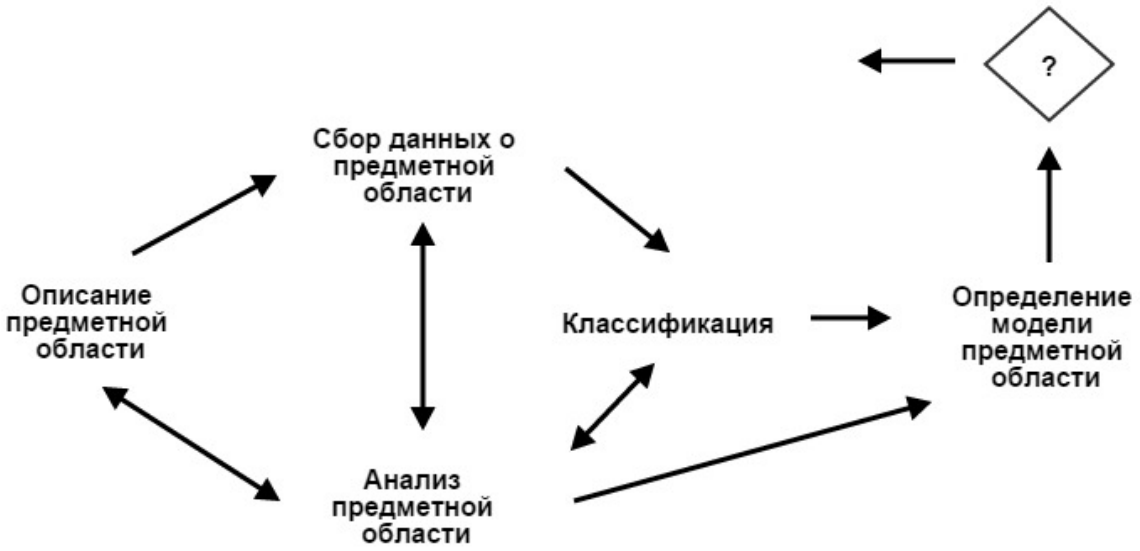

Рис. 1. Схема общего получения модели предметной области

Модели служат для создания программных продуктов (ПП) через разработку модели, обмена информацией между всеми заинтересованными сторонами (пользователями, специалистами в предметной области, аналитиками, проектировщиками и т.д.) в качестве языка, проектирования программного обеспечения (ПО), а также подготовки документации. Моделирование способствует более полному пониманию требований.

Существуют специальные среды разработки ПО, а именно CASE-системы (Computer Aided Software Engineering), определяемые в широком смысле как ”инструменты и методы для поддержки инженерного подхода к разработке программного обеспечения на всех стадиях процесса” [14]. CASE-системы включают в себя один или несколько редакторов визуальных языков, средства генерации целевого кода или интерпретации моделей по описанным диаграммам и другие инструменты разработки ПО [2].

CASE-системы ориентированы на два вида визуальных языков:

1) общецелевые (General-Purpose Language, GPL);
2) предметно-ориентированные (Domain-Specific Language, DSL).

Общецелевые визуальные языки позволяют описать любую предметную область (или, по крайней мере, очень широкий класс областей). Предметно-ориентированные языки же, в свою очередь, ориентированы на конкретный круг задач в некоторой области, т. е. DSL содержат только необходимые для конкретной области элементы и связи, что позволяет создавать более компактные и наглядные программы, более приближенные к решаемой задаче, а значит более понятные специалистам в предметной области, не обладающим навыками программирования. Следовательно, данный подход значительно упрощает создание программ по сравнению с GPL. Минусом DSL является то, что для каждой новой предметной области или задачи требуется новый визуальный язык, что может быть довольно трудозатратно при ручном кодировании ПП [3].

Для решения этой проблемы необходимы metaCASE-системы, или DSM-платформы, которые позволяют автоматизировать процесс разработки новых визуальных языков.

Язык, с помощью которого описывается предметно-ориентированный язык, называется *метаязыком*. С помощью метаязыка строятся различные модели, которые описывают создаваемый визуальный язык. Такие модели называются *метамоделями*. Для создания и описания метамоделей визуального языка необходимо разбираться в предметной области, в рамках которой он создается. Всегда возникают проблемы с пониманием некоторых терминов и связей в рассматриваемой предметной области, поэтому этап анализа предметной области является неотъемлемой частью всего процесса разработки. В настоящее время этот этап, как правило, происходит в неформальном виде, в результате чего все равно остается недопонимание между аналитиком и программистом. Из-за этого возникают неточные метамодели визуального языка, из которых возникают противоречия или выявляется некоторое недопонимание того, что в итоге нужно реализовать. Таким образом, это негативно сказывается на сроках выполнения проекта. Очевидной трудностью является сам процесс извлечения информации, необходимой для построения метамодели визуального языка.

В настоящее время в большинстве существующих DSM-платформ нет средств для автоматизации создания модели предметной области, а также средств построения по этой модели метамодели визуального языка [1].

Требования к «идеальному» редактору визуальных моделей

При работе с редакторами метамоделей выявляются некоторые возможности, которые могли бы ускорить создание ПО. Далее производится сравнение и оценка на соответствие требованиям идеального редактора.

Основные требования:

1. Возможность создавать редакторы диаграмм для специализированных визуальных языков, не прибегая к программированию.
2. Метамоделирование “на лету” (объединение редактирования модели и метамодели), позволяющее быстро и легко расширять систему прямо в процессе разработки, а именно: добавлять новые элементы языка, удалять ненужные и изменять существующие [8].
3. Возможность создания графического редактора, независимого от среды разработки (который может быть использован в сторонних приложениях) и внесения изменений в DSL без использования среды, в которой он был разработан.
4. Расширяемость – свойство программной системы, позволяющее с легкостью повысить возможности системы (расширить функциональность путём загрузки новых визуальных языков), не нарушая ее работоспособности [9].
5. Возможность работы с подмоделями – проведение операций над вложенной моделью (подмоделью) как над полной моделью.
6. Возможность пошаговой детализации – разбиения модели на подмодели.

Дополнительные требования (являются второстепенными, т.к. не относятся к процессу разработки графического редактора):

1. Многоплатформенность – возможность перекомпиляции исходного кода всей системы для разных целевых операционных систем.
2. Отлаженная система работы с данными, распределение прав доступа к данным и наличие удалённого доступа.
3. Невысокая стоимость пакета.

Обзор средств создания редакторов

Существует большое число средств для разработки графических редакторов DSL с возможностью определения собственных графических нотаций. Такими средствами являются MetaEdit+, MS DSL Tools, Eclipse GMF, State Machine Designer, REAL-IT, UFO-toolkit и пр. [7].

Eclipse modeling framework

Eclipse Modelling Framework – универсальная платформа для интеграции инструментов разработки. Имеет открытую архитектуру (рис. 2) [4], расширяемую за счёт подключаемых модулей (plug-ins).

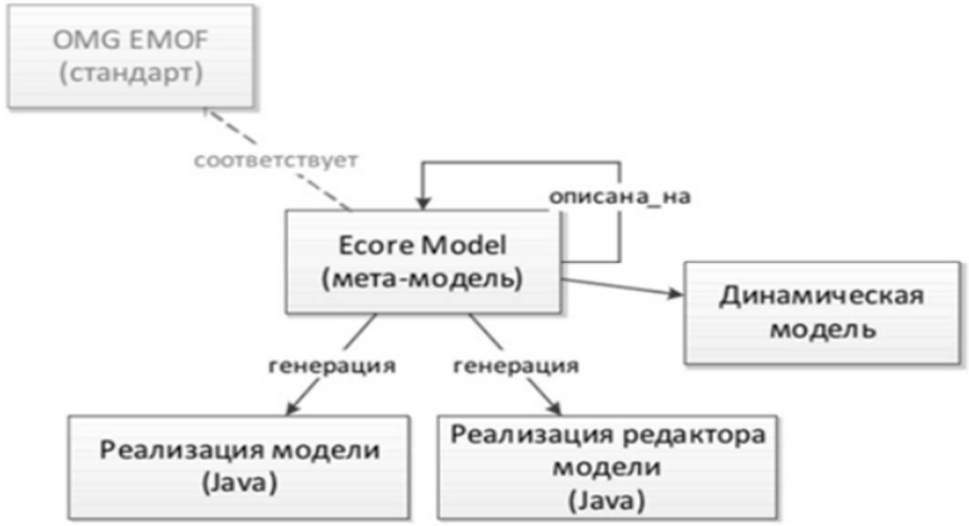

Рис. 2. Архитектура EMF

Реализованы три способа генерации кода:

1. *Модель* предоставляет Java интерфейсы и классы реализации для модели, а также классы-фабрики (класс, созданный в соответствии с шаблоном проектирования Abstract Factory) и пакет (метаданные) классов реализации.
2. *Адаптеры* генерируют классы реализации (называемые ItemProviders), предназначенные для редактирования и отображения классов модели.
3. *Редактор* создаёт правильно структурированный редактор, который соответствует рекомендуемому стилю Eclipse EMF и служит отправной точкой для начала настройки.

На базе среды разработки Eclipse создаются различные дополнительные технологии, одна из которых – Eclipse Graphical Modeling Framework (GMF), первая версия которой появилась в середине 2006 г.

Технология GMF предназначена для быстрой разработки графических средств, главным образом, интегрируемых в Eclipse. GMF интегрирует две широко используемые и известные Eclipse-библиотеки – Eclipse Modeling Framework (EMF) и Graphical Editing Framework (GEF).

Eclipse Graphical Modeling Project (GMP) предоставляет множество генеративных компонентов и среду выполнения для разработки графических редакторов на основе EMF и GEF.

GMP включает следующие подпроекты [19]:

1) GMF Notation;

2) GMF Runtime;

3) GMF Tooling;

4) Graphiti.

Архитектура DSL, созданного с применением GMF, строится на основе MVC-шаблона (Model/View/Controller). Для создания уровней представления и контроллеров используется технология GEF, для создания моделей – технология EMF.

Библиотека GEF состоит из двух частей: модуля OED, встроенного в Eclipse, и основной библиотеки GEF [12]. Визуальные DSL с помощью GEF можно создавать на базе любых моделей, однако максимальной эффективности можно достичь, интегрируя технологию GEF с EMF. Технология EMF предназначена для проектирования приложений, использующих модели бизнес-процессов, имеющих сложную структуру.

Технологии GEF и EMF создавались независимо друг от друга, поэтому существует несогласованность их интерфейсов. Кроме того, каждая из них предназначена для решения более широкого класса задач, чем поддержка создания DSL. Для преодоления этих проблем на основе двух технологий был реализован проект GMF.

Этапы процесса создания DSL с помощью технологии Eclipse GMF показаны на рис. 3 [10], а также описаны в [16].

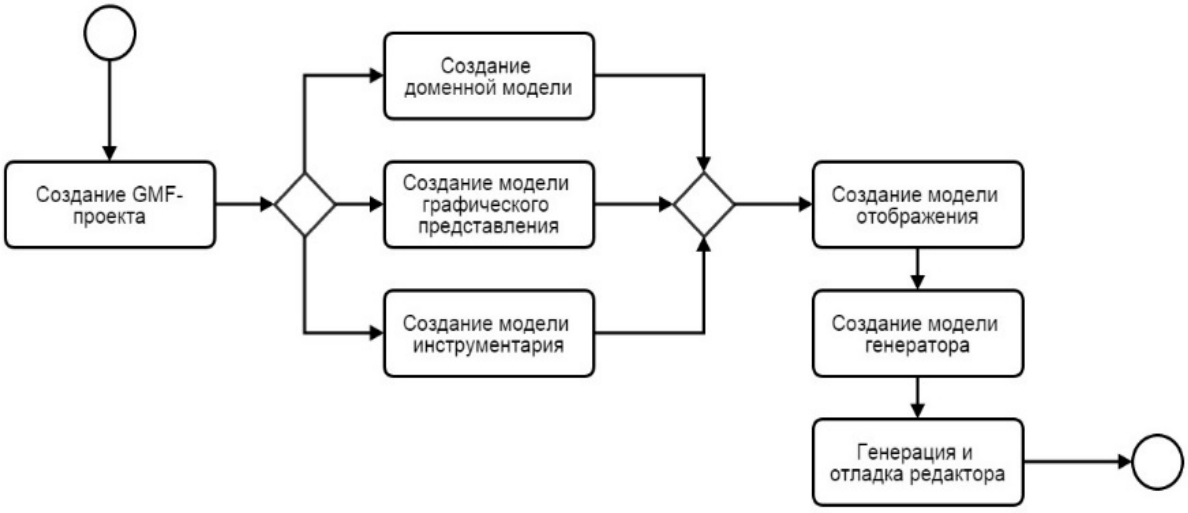

Рис. 3. Процесс разработки графического редактора с помощью технологии GMF

Система Qreal

Развитием системы Real-IT стала система QReal, разрабатываемая на кафедре системного программирования Санкт-Петербургского государственного университета под руководством профессора А.Н. Терехова [12].

QReal – сложная система с многоуровневой архитектурой. Создаваемые в QReal визуальные редакторы являются отдельными подключаемыми к системе модулями (набор графических редакторов QReal не фиксирован).

В Qreal реализованы два подхода к созданию редактора диаграмм:

1. Метамодель разрабатываемого языка описывается в виде XML-формата.
2. Метамодель языка задается графически в метаредакторе QReal посредством простого визуального языка.

В QReal уже существует ряд средств, ускоряющих и упрощающих работу проектировщика (редактор форм, распознавание жестов мыши, визуальный отладчик и т.д.). Метамодели различных диаграмм обрабатываются генератором редакторов, который затем генерирует по ним код на языке C++ (рис. 4) [5].

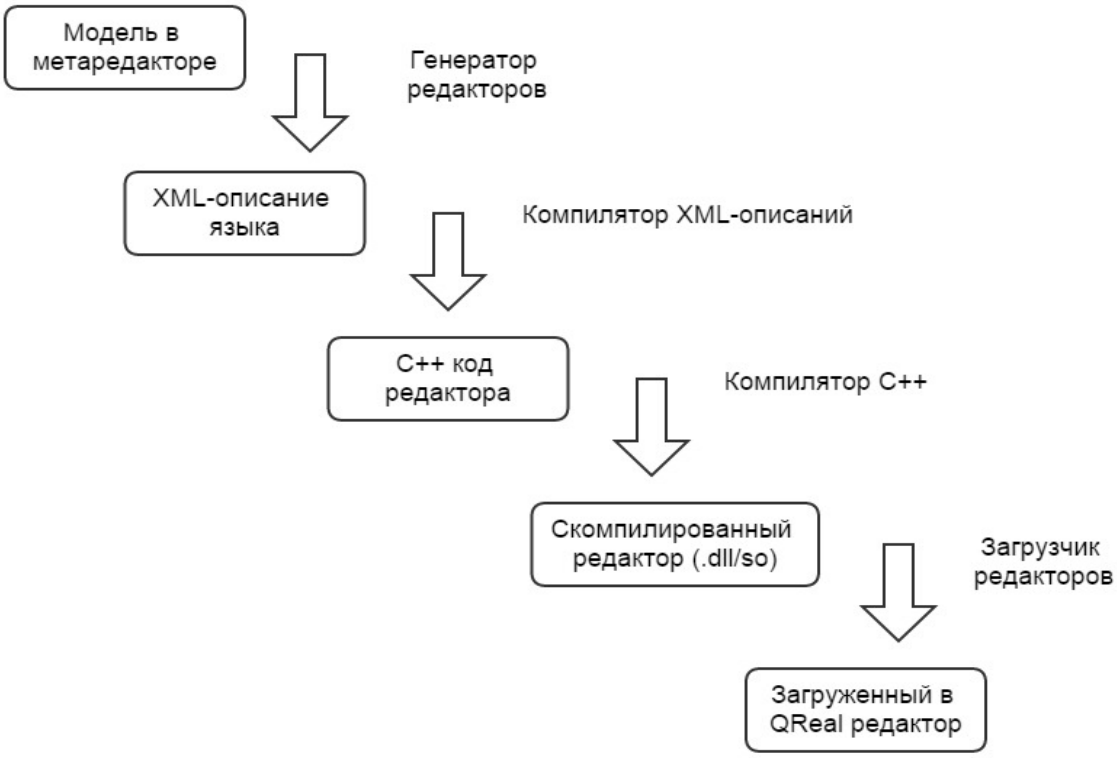

Рис. 4. Этапы генерации нового подключаемого модуля в QReal

При разработке нового приложения требуется работа с метаредактором среды QReal, который и позволяет создавать новые и/или изменять существующие визуальные языки. При этом, если впоследствии потребуется генерация готового приложения по этому формальному описанию, то при дополнении языка новыми блоками придется также каждому из них задавать большие куски кода, в которые этот блок должен раскрываться при генерации. Таким образом, для полноценной работы с данным визуальным языком, помимо знаний предметной

области, необходимо также базовое владение средой QReal как metaCASE-системой (рис. 5) [5] и знание целевого текстового языка, в который будет происходить генерация приложения. Такие требования от пользователя могут заметно сузить круг специалистов, способных эффективно создать своё приложение. Разработанный язык может быть применен в любой области, так как все необходимые, но отсутствующие блоки нужно будет каждый раз создавать под конкретную задачу.

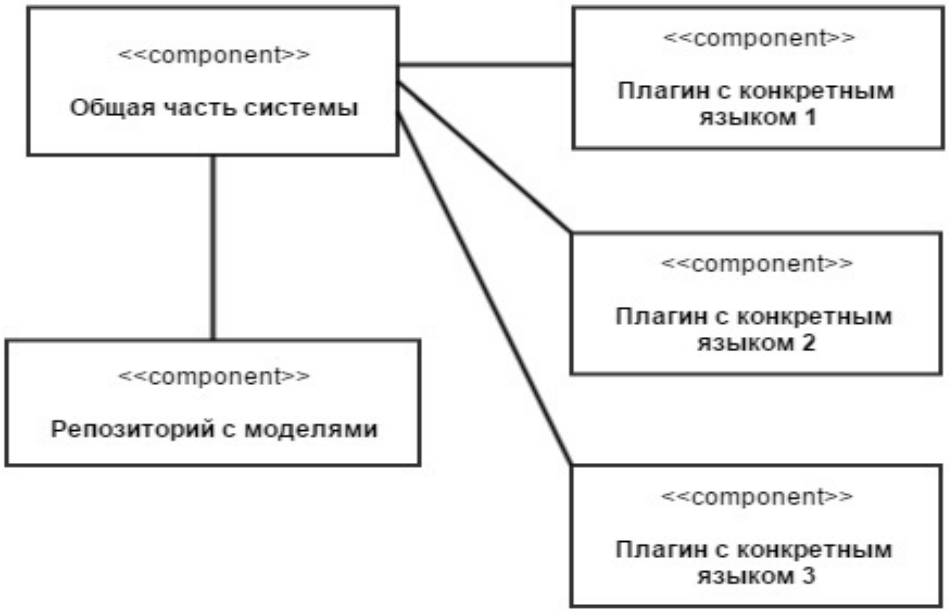

Рис. 5. Представление архитектуры QReal как metaCASE-системы

Microsoft DSL tools

Domain-Specific Language Tools (DSL Tools), которые встроены в Visual Studio, позволяют определить предметно-ориентированный язык и затем генерировать всё, что необходимо пользователю для создания основанных на языке моделей [18].

Процесс создания нового редактора в Microsoft DSL Tools:

1. Описание метамодели языка на специальном визуальном языке DSL Tools.
2. Генерация редактора.
3. Внесение в сгенерированный код ручных изменений для реализации дополнительной функциональности (валидация моделей).

В рамках DSL Tools можно по специальным правилам написать модуль генерации исходного кода, который будет выполнять генерацию кода на основе моделей, созданных в новом редакторе.

После определения метамодели DSL и свойств редактора необходимо провести валидацию. В случае успешной валидации на основе спроектированной модели производится генерация кода редактора в тексты на языке C#.

После компиляции кода получается редактор с палитрой графических элементов, браузером модели, редактором диаграмм, валидатором, генератором кода. Созданный редактор может быть открыт в новом экземпляре MS Visual Studio или для него может быть создан Installer. С помощью этого редактора пользователь может строить модели [12].

На рис. 6 показан внешний вид окна целевого редактора, сгенерированный для языка SCL с помощью Microsoft DSL Tools [6].

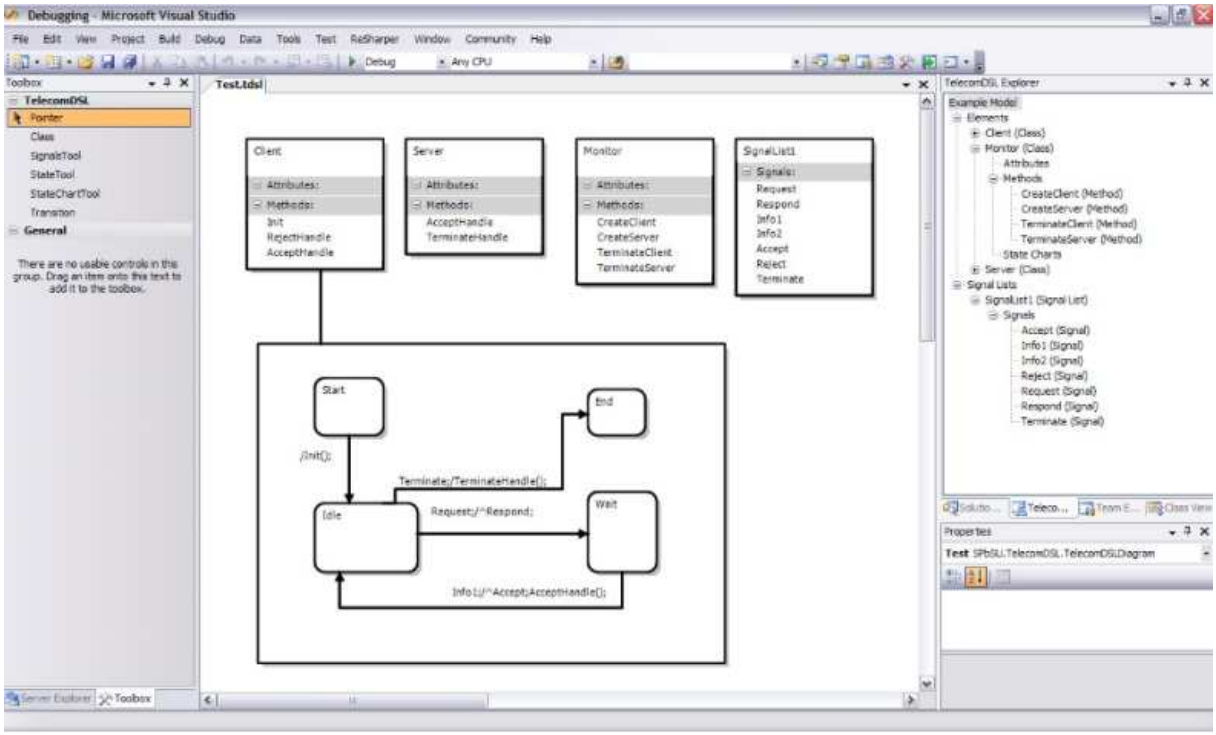

Рис. 6. SCL-редактор

Процедуры отрисовки недостающих графических элементов можно реализовать самостоятельно на C#. С помощью DSL Tools возможно создание несложных предметно-ориентированных языков, которые будут использоваться внутри среды Visual Studio, но независимый графический редактор с помощью DSL Tools создать нельзя. При реализации сложных визуальных редакторов с возможностями, не поддерживаемыми DSL Tools, требуется большой объём кода на C#.

Metaedit+

MetaEdit+ – среда для создания и использования предметно-ориентированных языков, разработанная в качестве расширения среды MetaEdit. Среда MetaEdit обеспечивала моделирование лишь диаграмм; MetaEdit+ позволяет создавать диаграммы, матрицы и таблицы [13].

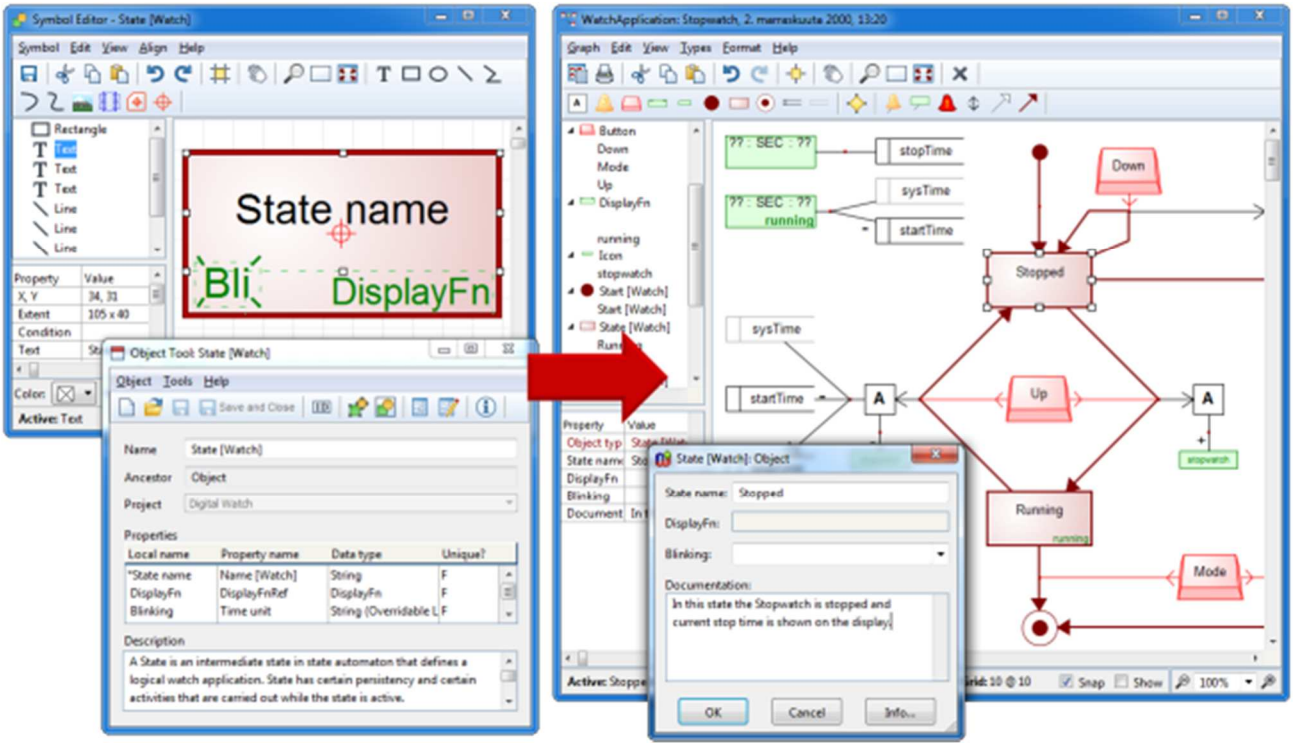

Рис. 7. Среда предметно-ориентированного моделирования MetaEdit+

Среда моделирования (рис. 7) состоит из двух основных компонентов:

1. MetaEdit+ Modeler обеспечивает настраиваемый функционал DSM одновременно для нескольких пользователей и проектов (среда MetaEdit поддерживала создание только одного модельного языка для одного пользователя в текущий момент).
2. MetaEdit + Workbench позволяет строить языки пользовательского моделирования (DSL) и текстовые генераторы и включает в себя функциональность MetaEdit + Modeler и MetaEdit + API.

При построении метамодели в MetaEdit+ используется язык мета- моделирования GOPRR, получивший свое название от основных понятий, которыми он оперирует: граф (Graph), объект (Object), свойство (Property), связь (Relation) и роль (Role) [17].

Создание метамодели начинается с построения GOPRR-модели, после чего она может быть открыта в MetaEdit+ Workbench для построения модели предметной области. При этом метамодель может быть загружена в систему и использоваться в качестве метаязыка.

MetaEdit+ использует подход, основанный на интерпретации метамоделей. При таком подходе пользователь вынужден работать отдельно с моделью и метамоделью: метамодель можно изменить во время работы системы без перезапуска приложения – достаточно внести изменения в описание метамодели, вновь импортировать ее в MetaEdit+ Workbench и продолжить работу над моделью. При этом все необходимые изменения в соответствующих моделях DSM-платформа произведет сама.

MetaLanguage

MetaLanguage – система для создания визуальных предметно-ориентированных языков, используемых для разработки моделей. Для описания метамоделей в системе MetaLanguage используется метазык, базовыми конструкциями которого являются сущность, отношения и ограничения.

На рис. 9 представлен пример диаграммы, созданной с использованием конструкций метаязыка системы MetaLanguage: метамодель описания диаграммы «Сущность-Связь» [11].

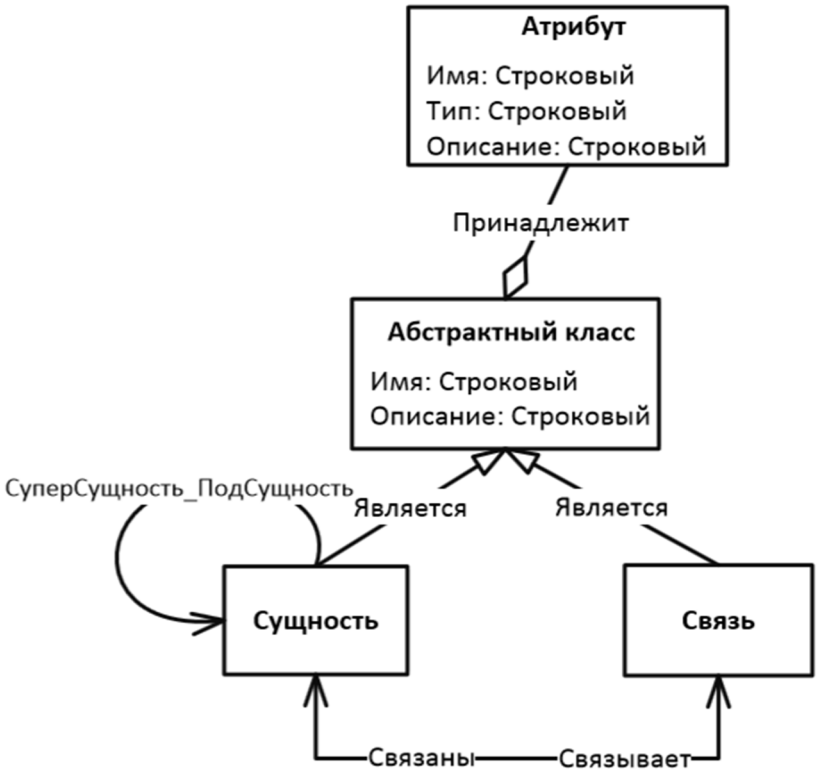

Рис. 9. Метамодель ER-диаграммы

Для работы с моделями среда разработки включает следующие компоненты графический редактор, браузер моделей, хранилище, валидатор и генератор-трансформатор (рис. 8) [11]. Среда интегрирует все компоненты в единое целое [7].

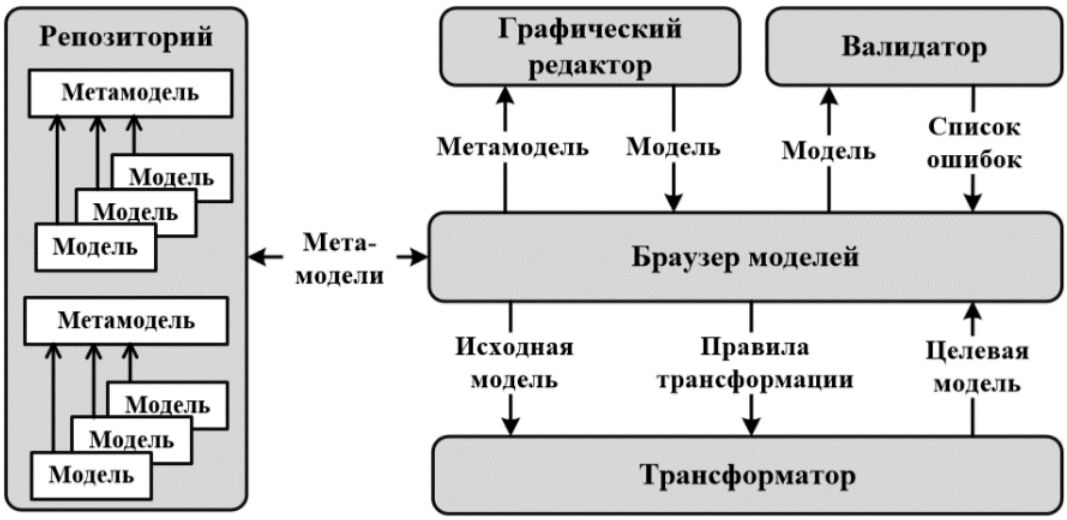

Рис. 8. Архитектура системы MetaLanguage

Назначение графического редактора: создание, изменение и удаление моделей, а также установление связей между ними. Каждая сущность модели представлена некоторым графическим символом, а связи сущностей представлены разными типами линий. Графический редактор позволяет выделить различные фигуры на листе (экземпляры сущностей и связей), а затем установить различные графические свойства.

В MetaLanguage реализована возможность проведения трансформаций визуальных моделей двух типов: «Модель-Модель», применяемо для трансформации модели из одной графической нотации в другую, и «Модель-Текст» для генерации кода по заданным пользовательским шаблонам.

Заключение

Результаты сравнения кратко представлены в следующей таблице (рис. 10):

	EMF	Qreal	DSL Tools	MetaEdit+	MetaLanguage
Возможность создавать редакторы диаграмм, не прибегая к программированию	+	+	+	+	+
Метамоделирование “на лету”	-	реализация метамоделирования "на лету" для данной системы описана в работе [8]	-	-	-
Независимость разработанного редактора от среды разработки	-	+	-	+	+
Расширяемость	+	+	-	+	-
Работа с подмоделями	+	-	-	+	-
Пошаговая детализация	-	-	-	+	-
Многоплатформенность	+	+	-	+	-
Распределенность	-	+	-	+	+
Невысокая стоимость	Бесплатно	Бесплатно	Бесплатно	Платно	Бесплатно

Рис. 10. Таблица сравнения редакторов

Из этой таблицы можно сделать вывод, что наиболее удовлетворяющим требованиям редактором является MetaEdit+, а среди бесплатных редакторов наилучшее решение – QReal. Эти системы также являются лучшими с точки зрения создания графического редактора: в них есть возможности создания независимого от среды разработки редактора, добавления в систему новых функций и возможность создания редактора диаграмм, не прибегая к программированию (для более удобной и быстрой разработки).

В статье были рассмотрены пять наиболее мощных и популярных DSL-инструментариев. По итогам анализа были выявлены главные проблемы: возможности в редакторах метамоделирования «на лету» и пошаговой детализации модели. Возможность метамоделирования на «лету» упрощает процесс разработки, например при модификации метамодели, а пошаговая детализация облегчает работу с данными модели.

Библиографический список

1. *Гудошникова А.А.* Генерация метамодели языка по модели предметной области в проекте QReal. // 2014. [Электронный ресурс] [Режим доступа: http://se.math.spbu.ru/SE/diploma/2014/b/Gudoshnikova_Anna_Andreevna-text.pdf] [Проверено: 20.11.2014].
2. *Дерипаска А.О.* Визуальный язык для платформы Ubiq Mobile в среде QReal. //2013. [Электронный ресурс] [Режим доступа: http://se.math.spbu.ru/SE/YearlyProjects/2013/YearlyProjects/2013/445/445-Deripaska-report.pdf] [Проверено: 20.11.2014]
3. *Дерипаска А.О.* Средства задания правил генерации в QReal. //2014. [Электронный ресурс] [Режим доступа: http://se.math.spbu.ru/SE/diploma/2014/s/DeripaskaAnna_Diploma.pdf] [Проверено: 20.11.2014].
4. *Иванов А.* Редактор ISO15926 на основе Eclipse EMF. // [Электронный ресурс] [http://www.slideshare.net/ailev/iso-15926-eclipse-emf] [Проверено: 20.11.2014].
5. *Колантаевская А.С.* Исследование удобства процесса моделирования на базе DSM-платформы QReal. // 2013. [Электронный ресурс] [Режим доступа: http://se.math.spbu.ru/SE/diploma/2013/b/KolantaevskayaAnna_text.pdf] [Проверено: 20.11.2014].
6. Лекция 13: Знакомство с DSM-платформой Microsoft DSL TOOLS. // [Электронный ресурс] [http://www.intuit.ru/studies/courses/1041/218/lecture/5639?page=3] [Проверено: 20.11.2014].
7. *Лядова Л.Н., Сухов А. О.* ВИЗУАЛЬНЫЕ ЯЗЫКИ И ЯЗЫКОВЫЕ ИНСТРУМЕНТАРИИ: МЕТОДЫ И СРЕДСТВА РЕАЛИЗАЦИИ. // 2010. [Электронный ресурс] [Режим доступа: http://www.hse.ru/pubs/share/direct/document/109214959] [Проверено: 20.11.2014].
8. *Птахина А.И.* Разработка метамоделирования “на лету” в metaCASE-системе QReal. // 2013. [Электронный ресурс] [Режим доступа: http://se.math.spbu.ru/SE/YearlyProjects/2013/YearlyProjects/2013/445/445-Ptakhina-report.pdf] [Проверено: 20.11.2014].
9. *Серый А.П.* О ПОДХОДЕ К ОБЕСПЕЧЕНИЮ РАСШИРЯЕМОСТИ, МОБИЛЬНОСТИ И ИНТЕРОПЕРАБИЛЬНОСТИ DSL-ИНСТРУМЕНТАРИЯ. // 2013. [Электронный ресурс] [Режим доступа: http://www.scienceforum.ru/2013/147/2500] [Проверено: 20.11.2014].

10. *Сорокин А.В., Кознов Д.В.* Обзор Eclipse Modeling Project. // 2010. [Электронный ресурс] [Режим доступа: http://www.math.spbu.ru/user/dkoznov/papers/SorokinKoznov.pdf] [Проверено: 20.11.2014].
11. *Сухов А.О.* РАЗРАБОТКА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ СОЗДАНИЯ ВИЗУАЛЬНЫХ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ: дис. канд. физ.-мат. наук. – Перм. гос. университет, Пермь, 2013.
12. *Сухов А.О.* СРАВНЕНИЕ СИСТЕМ РАЗРАБОТКИ ВИЗУАЛЬНЫХ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ. // 2012. [Электронный ресурс] [Режим доступа: http://www.hse.ru/pubs/share/direct/document/71628791] [Проверено: 20.11.2014].
13. Domain-Specific Modeling with MetaEdit+. [Электронный ресурс] [Режим доступа: www.metacase.com] [Проверено: 20.11.2014].
14. *Forte, G., Norman, R.J.* A self-assessment by the software engineering community // Communications of the ACM, Vol. 35, No. 4 (April), 1992.
15. *G.Arango*. Software Reusability, chapter 2. Domain analysis methods. P.17-49.Workshops M.E.Horwood, London 1994.
16. Gronback R.C. Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. – Reading : Addison-Wesley, 2009. – 706 p.
17. *Kelly S.* Comparison of Eclipse EMF/GEF and MetaEdit+ for DSM [Электронный ресурс]. [Режим доступа: http://www.softmetaware.com/oopsla2004/kelly.pdf] [Проверено: 20.11.2014].
18. Overview of Domain-Specific Language Tools. [Электронный ресурс] [Режим доступа: http://msdn.microsoft.com/en-us/library/bb126327.aspx] [Проверено: 20.11.2014].
19. What is the Graphical Modeling Project (GMP)? [Электронный ресурс] [Режим доступа: https://wiki.eclipse.org/GMP] [Проверено: 20.11.2014].