Санкт-Петербургский государственный университет
Математико-механический факультет

Кафедра системного программирования

Сравнение функциональности Eclipse GMF и MetaEdit+

Курсовая работа студентки 361 группы
Николаевой Дарьи Михайловны

Научный руководитель _____ Т.А.Брыксин
ст.преподаватель

Санкт-Петербург
2013

Введение

Одним из подходов к разработке программного обеспечения является предметно-ориентированное моделирование. Его суть в том, что программа представляется в виде набора моделей, описывающих её с разных точек зрения. При помощи специальных генераторов затем можно получить программный код. Таким образом разработчику теперь совсем не обязательно досконально знать языки программирования, достаточно лишь разбираться в предметной области и используемом языке моделирования.

Многообразие задач, решаемых программистами, невероятно велико, и не существует универсального предметно-ориентированного языка с инструментарием, позволяющим автоматически генерировать программный код. Поэтому, как правило, выбирается достаточно узкая предметная область, в которой есть некоторый класс схожих между собой задач. Создавая для неё предметно-ориентированный язык, или DSL (domain-specific language), разработчик выделяет её основные понятия (часто при помощи эксперта данной предметной области), которые и станут абстракциями нового языка. Имея теперь такой DSL, для решения задачи необходимо лишь описать её особенности, отличающие её от других задач того же класса, в терминах предметной области. Таким образом реализация значительно упрощается, и ей может заниматься человек, далёкий от программирования.

Ограничение класса решаемых задач позволяет также написать генератор, который будет создавать полностью работающий программный код, транслируя высокоуровневые абстракции предметной области в язык целевой платформы.

Чтобы решение поставленной задачи таким образом было более эффективным, чем традиционное написание программного кода вручную, необходимо автоматизировать процесс создания специализированных предметно-ориентированных языков, а также редакторов и генераторов. Для этого были разработаны DSM-платформы (domain-specific modeling). Помимо того, что они облегчают процесс создания языков моделирования, существенно упрощается процедура внесения изменений в уже имеющийся DSL: разработчику достаточно изменить его описание в платформе вместо того, чтобы изучать объёмный код парсера языка. Ещё одним достоинством DSM-платформ является

возможность интеграции в одной системе нескольких DSL.

В данной работе будут проанализированы 2 платформы — MetaEdit+ (как самая ранняя и самая популярная среди коммерческих) и Eclipse GMF (самая мощная и популярная среди исследовательских open-source-платформ).

1. Создание предметно-ориентированных языков

Прежде всего стоит отметить, что предметно-ориентированные языки бывают текстовыми и визуальными. Первые позволяют описывать модель в текстовом виде, а вторые — в графическом. При этом предпочтение отдаётся визуальным DSL, поскольку диаграммы более наглядны и понятны не только программистам, но и экспертам предметной области, и другим пользователям системы.

Для создания языка необходимо определить его абстрактный и конкретный синтаксис.

Абстрактный синтаксис описывает понятия, используемые в языке, т.е. элементы, их свойства и их взаимодействие друг с другом. Задать абстрактный синтаксис можно при помощи КС-грамматики или метамодели (в случае визуальных языков используется последний способ).

Конкретный синтаксис — это описание конструкций языка с помощью конкретных пиктограмм и текстового представления, т.е. визуальное представление языковых элементов на диаграммах.

DSL проектируется как средство быстрого и удобного создания моделей. Модель - это некое упрощённое описание чего-либо, в данном случае предметной области. Набор таких моделей становится предметной областью уже для DSL, т.е. происходит переход на более высокий уровень абстракции, поэтому предметно-ориентированный язык логично назвать метамоделью, а процесс его создания - метамоделированием. Соответственно, язык, на котором разрабатывается DSL, называется метаметамодель.

2. Критерии сравнения функциональности предметно-ориентированных платформ

Для сравнения DSM-платформ были разработаны следующие критерии:

- usability - комбинация следующих факторов, которые оказывают влияние на пользовательский опыт:
    - простота обучения: насколько быстро пользователь сможет освоить выполнение простых задач, если он впервые видит интерфейс системы;
    - эффективность использования: насколько быстро пользователь сможет выполнять задачи, научившись пользоваться ситемой;
    - запоминаемость: если пользователь ранее работал с системой, сколько он смог запомнить и не придётся ли ему изучать систему снова;
    - субъективное удовлетворение: насколько пользователю нравится работать с системой;
- наличие такой базовой функциональности, как:
    - feedback: информирование пользователя о действиях с важными последствиями;
    - undo: отмена действия пользователя для того, чтобы сэкономить время на исправлении ошибок и сделать работу с инструментом более приятной;
    - form/field validation: подтверждение правильности введённых данных в формы/поля;
    - shortcuts;
    - user expertise;
    - reuse information;
    - help;
- функционал платформы;
- алгоритм создания DSL и инструментария для него.

3. MetaEdit+

MetaEdit+ (URL https://www.metacase.com/) - это платформа для создания и использования предметно-ориентированных языков. Она разрабатывалась в University of Jyväskylä как часть научно-исследовательского проекта MetaPHOR. MetaEdit+ является расширением среды MetaEdit, созданной в конце 1980-х - начале 1990-х годов проектом SYTI в сотрудничестве с компанией MetaCase. Среда MetaEdit позволяла в текущий момент времени работать только над одним предметно-ориентированным языком и только одному пользователю, для отображения модели использовались только диаграммы. MetaEdit+ снимал эти ограничения, предоставляя одновременный доступ нескольким разработчикам для

работы с несколькими проектами, каждый из которых может содержать несколько моделей, поддерживая несколько встроенных языков моделирования, а также создание диаграмм, матриц и таблиц.

Платформа MetaEdit+ состоит из двух самостоятельных подсистем:

- MetaEdit+ Workbench предоставляет средства для создания модельных языков и генераторов.
- MetaEdit+ Modeler предназначен для использования этих языков.

Как правило, несколько ключевых разработчиков с помощью MetaEdit+ Workbench создают предметно-ориентированный язык для решения задач в рамках конкретного проекта. Затем этот язык моделирования применяется для разработки конечных продуктов с использованием MetaEdit+ Modeler.

3.1. Инструменты MetaEdit+

Платформа MetaEdit+ предоставляет разработчику внушительный набор инструментов для работы с метамоделями. Он включает в себя репозиторий, браузеры, редакторы, генераторы. Рассмотрим их подробнее.

Репозиторий

Репозиторий располагается, как правило, на сервере. В случае однопользовательской версии - на локальном диске компьютера пользователя. Кроме того, если у компьютера есть доступ к сетевому диску, репозиторий может располагаться там, позволяя работать удалённо.

В репозитории хранится вся информация MetaEdit+, в т.ч. описания языка, диаграммы, матрицы, объекты, свойства и т.д. Все внесённые пользователем изменения автоматически отображаются везде, где они встречаются. Таким образом разработчик всегда имеет перед собой актуальные данные. Не сохраняется только временная информация, например о том, какие окна открыты и где. Генераторы основаны непосредственно на информации, хранящейся в репозитории.

Одновременное изменение разными пользователями одних и тех же данных блокируется. Однако операция чтения данных доступна всегда, независимо от того, какие действия совершают другие пользователи.

Браузеры

Браузеры являются инструментом для просмотра и редактирования содержимого проектов в репозитории. При помощи браузеров пользователь обладает доступом ко всем имеющимся графам (под графом здесь понимается диаграмма - модель, созданная пользователем при помощи разработанного им предметно-ориентированного языка), а также к их загруженным в настоящий момент элементам, таким как объекты, отношения, роли и свойства. Менеджер графов позволяет просмотреть содержимое и связи между выбранными графами, а также экспортировать их в другой репозиторий MetaEdit+.

MetaEdit+ предоставляет в распоряжение пользователя следующие четыре вида браузеров:

- браузер графов позволяет просматривать иерархическую структуру графа и содержащиеся в нём элементы, а также управлять графом: открыть его в редакторе, увидеть его свойства, просмотреть информацию о графе, удалить его, создать новый граф и др.
- браузер типов позволяет просматривать и открывать редакторы элементов метамодели, в том числе графа (имеется в виду сама метамодель, см.ниже), объектов, отношений, ролей, типов, а также генератора; кроме того, он предоставляет возможность открывать, закрывать и создавать проекты;
- браузер объектов обеспечивает иерархическое представление объектов графа (это особенно удобно при наличии сложноструктурированных каталогов объектов, являющихся свойствами других объектов);
- браузер метамодели (отображаемый в Launcher только в MetaEdit+ Workbench) демонстрирует пользователю типы графа и типы, содержащиеся в них.

Редакторы

Редакторы предназначены для того, чтобы создавать, изменять и удалять модели, а также устанавливать взаимосвязи между различными моделями. В MetaEdit+ существуют редактор диаграмм, редактор матриц и редактор таблиц. Каждый из них в определённом виде отображает модель, оперируя при этом одними и теми же понятиями проекта.

Редактор диаграмм предоставляет графическое изображение модели, т.е. каждый элемент представлен некой пиктограммой, между собой они соединяются при помощи

разнообразных линий. Его применение при создании и изменении моделей наиболее удобно.

Редакторы матриц и таблиц представляют модель, используя матрицы и таблицы соответственно. Разница в том, что редактор матриц отображает сразу все объекты графа, а редактор таблиц в определённый момент времени позволяет работать только с одним объектом. При работе с этими редакторами пользователь меняет значение свойства объекта в соответствующей ячейке.

Генераторы

Генераторы MetaEdit+ позволяют проверить правильность созданных моделей и сгенерировать на их основе код или документацию. Разработчик может модифицировать имеющиеся генераторы или создавать новые.

Помимо возможности использования встроенных генераторов кода, поддерживающих языки C++, Smalltalk, CORBA IDL, Java, Delphi, Mobile Information Device Profile (MIDP) – набор Java API функций для мобильных телефонов, MetaEdit+ предоставляет разработчику возможность создавать генераторы для собственных языков программирования и моделирования. Таким образом, можно говорить о том, что поддерживается возможность определения правил трансформации моделей в код.

MetaEdit+ позволяет импортировать и экспортировать метамодели в формат MXT (файл MetaEdit+ XML Types), который базируется на общепринятом стандарте XML.

MetaEdit+ предоставляет возможность на основе существующих моделей сгенерировать документацию к проекту. Документация модели включает в себя название модели, информацию о разработчиках. Если модель имеет графическое представление, то в документацию будет включена и диаграмма в формате .png со ссылками на описание отдельных ее частей.

3.2. Создание предметно-ориентированного языка в MetaEdit+

При создании предметно-ориентированного языка в MetaEdit+ применяется язык метамоделирования GOPPRR (Graph-Object-Property-Port-Relationship-Role). Он оперирует следующими понятиями (которые и дали ему название):

- Граф (Graph) - сама метамодель, часто изображаемая в виде диаграммы. Граф

представляет собой набор объектов, отношений, ролей и их взаимосвязей (bindings), которые показывают, какие объекты и отношения соединяются через какие роли (рис.1, рис.2). Примером графа может служить диаграмма классов UML.

- Объект (Object) - элементы графа, выделенные абстракции языка. Пример - класс в диаграмме классов UML. Все экземпляры объектов могут переиспользоваться, т.е. существующий объект может быть добавлен в граф другого проекта при помощи функции Add Existing.
- Свойство (Property) - атрибут, характеризующий одну из описываемых здесь составляющих языка метамоделирования.
- Порт (Port) - особая область объекта, куда может быть подключена роль. Обычно роли соединяются непосредственно с объектами, семантика соединения определяется типом роли. Чтобы предоставить роли возможность подсоединяться с разной семантикой к различным частям объекта, можно добавить на символ объекта порты. Порты определяются для типа объекта, и все его экземпляры будут иметь те же порты.
- Отношение (Relationship) - это явная связь между двумя или более объектами. В MetaEdit+ поддерживаются следующие виды отношений:
    - ассоциация - соединяет два или более объектов графа;
    - наследование - позволяет создавать новый объект на основе уже существующего объекта;
    - агрегация - это частный случай ассоциации, при котором один из объектов является главным, а другой - зависимым;
    - декомпозиция - позволяет для любого объекта графа задать уточняющий граф.
- Роль (Role) - определяет, как объект участвует в отношении. Например, в отношении наследования существует два вида ролей: предки и потомки.

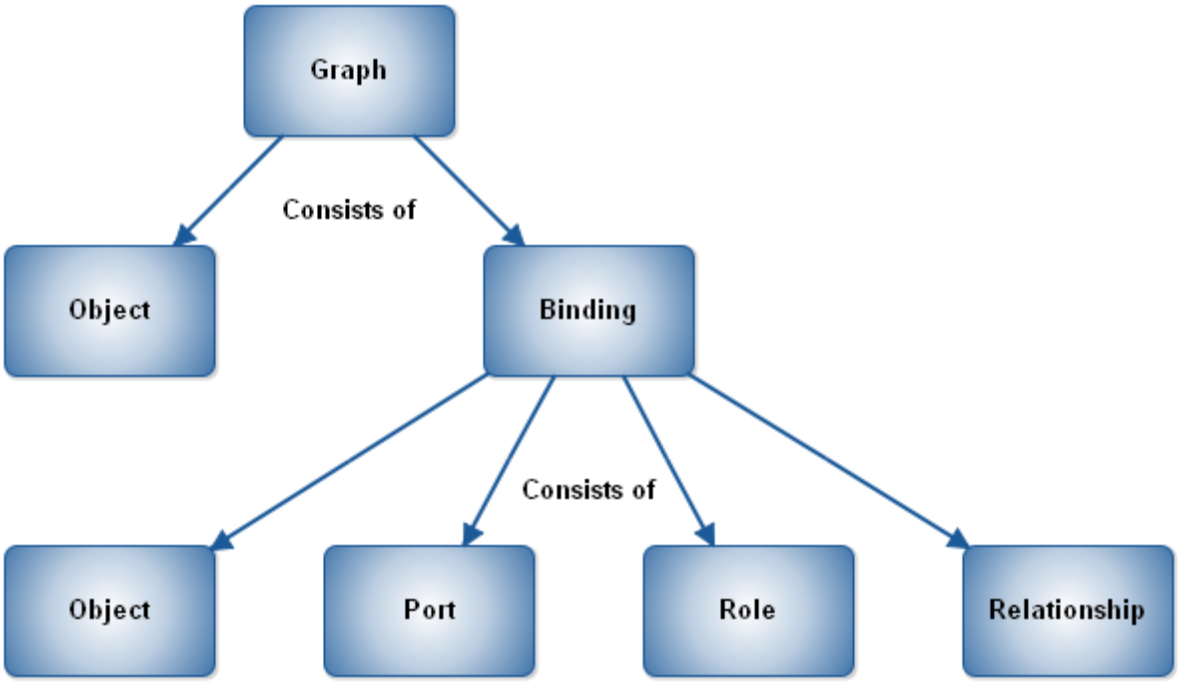

Рис.1. Компоненты графа

Рис.2. Компоненты взаимосвязи.

Для создания DSL необходимо построить GOPPRR-модель: определить вышеописанные понятия, их свойства, наложить на них правила и ограничения, задать для них графическую нотацию и описать генератор. Можно начинать как с создания графа (top-down fashion), так и с разработки отдельных фрагментов будущего языка (bottom-up approach). Рассмотрим первый вариант, так как он соответствует нашему порядку описания понятий языка GOPPRR.

Используя Graph Tool, создаём новый тип графа. Задаём для него имя, выбираем нужный проект и определяем для него свойства. После этого добавляем типы объектов, отношений, ролей и, при необходимости, портов. Так же задаём для них имена и свойства (можно переиспользовать уже существующие).

Следующий важный шаг - создание взаимосвязей между типами отношений, ролей, портов и объектов. Для этого сначала выбирается отношение, затем соответствующие ему роли. При необходимости для роли задаётся её мощность. Для каждой роли определяется порт и объекты, которые могут выступать в этой роли в этой связи.

Осталось задать графические нотации и описать генераторы. Первое выполняется при помощи инструмента Symbol Editor. Одной из особенностей MetaEdit+ является возможность широкого выбора пиктограмм для объектов (ролей и др.): их можно нарисовать в предоставляемом платформой графическом редакторе, либо импортировать из библиотеки имеющихся пиктограмм, либо загрузить свою картинку.

На данном этапе разработчик уже имеет описанный и готовый к использованию предметно-ориентированный язык. При помощи редактора диаграмм он может рисовать графы, для которых в дальнейшем будет создан программный код. Для этого необходимо дополнительно разработать генератор.

Описать генератор для предметно-ориентированного языка позволяет Generator Editor. Можно как редактировать уже имеющиеся шаблоны, так и создавать собственные генераторы с нуля. MetaEdit+ предоставляет для этих целей язык MERL (MetaEdit+ Reporting Language). Он содержит команды для навигации по конструкциям моделей, извлечения информации из элементов и вывода этой информации в окно или в файл. Кроме того, существует целый ряд дополнительных команд, реализующих вмешательство пользователя или запуск внешних программ. Встроенная проверка синтаксиса быстро находит ошибки в написании генератора, а с помощью дебаггера упрощается его отладка. Запустить генератор можно при помощи функции Generate for Graph, позволяющей выбрать граф, для которого будет создан программный код или документация.

4. Eclipse GMF

Eclipse - это кроссплатформенная интегрированная среда разработки программного обеспечения с открытыми исходными кодами, созданная компанией IBM. Основным языком разработки, который поддерживается этой средой, является Java, однако разработчики могут использовать также C++, Fortran, Perl, Haskell и др. Eclipse в первую очередь служит платформой для разработки расширений - этим он и завоевал популярность. Одним из таких расширений является Eclipse Graphical Modeling Framework (GMF). Оно впервые появилось как часть Eclipse 3.2 Callipso в июне 2006. Eclipse GMF является open source проектом и развивается, главным образом, специалистами компаний IBM и Borland.

Eclipse GMF предоставляет средства для создания графических редакторов на основе Eclipse Modeling Framework (EMF) и Graphical Editing Framework (GEF) - двух известных и широко используемых библиотек Eclipse, интегрируемых GMF.

Архитектура предметно-ориентированного языка, создаваемого с помощью этого фреймворка, строится на основе шаблона MVC (model - view - controller). Технология GEF используется для создания уровней представления и контроллера, модели разрабатываются при помощи технологии EMF.

4.1. Библиотеки GEF и EMF

Библиотека GEF состоит из трёх частей:

- Draw2d;
- GEF (MVC);
- Zest.

Draw2d - это инструментарий, позволяющий создавать и обрабатывать графические объекты. Он предоставляет такие средства, как менеджеры размещения графических объектов, палитра стандартных объектов, средства создания собственных графических объектов, средства создания связей между графическими объектами и др.

GEF (MVC) - это интерактивный MVC фреймворк, позволяющий создавать графические редакторы для Eclipse Workbench UI. Он облегчает отображение любой модели графически (используя элементы Draw2d), обрабатывает манипуляции пользователя с мышью

и клавиатурой, обеспечивает интеграцию с Eclipse Workbench UI. GEF (MVC) обеспечивает основу для создания практически любого редактора приложений.

Zest - инструментарий для визуализации на базе Draw2d, который позволяет реализовать графические представления для Eclipse Workbench.

При помощи Draw2d создаётся графический редактор, визуализирующий экземпляры Java-классов, которые описывают объекты предметной области. С использованием библиотеки GEF разрабатывается контроллер для синхронизации классов и визуальных диаграмм. Изменение модели производится не напрямую, а с помощью определенного набора команд, которые вносят изменения одновременно и в классы, и в визуальные диаграммы. Это позволяет поддерживать систему в синхронизированном состоянии и отменять все произведенные изменения.

Визуальные DSL с помощью GEF можно создавать на базе любых моделей, однако наибольшей эффективности можно достичь, интегрируя технологию GEF с EMF.

Проект EMF предназначен для построения инструментов и других приложений на основе структурированной модели данных. Создание таких приложений производится с помощью средств генерации EMF по созданной или импортированной модели. Разработчик имеет возможность расширить функциональность «вручную». Внесенные изменения будут сохранены и при последующих генерациях кода.

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.

4.2. Создание предметно-ориентированного языка в Eclipse GMF

Схема создания DSL при помощи Eclipse GMF, а также компоненты и модели, используемые в процессе разработки, изображены на рис.3:

- Модель предметной области (Domain Model, .ecore) разрабатывается путём создания модели Ecore в EMF Editor или импортирования из других источников. В первом случае Ecore tools создаёт 2 файла: собственно Domain Model и Diagram Definition (.ecore_diagram), который позволяет путём рисования диаграммы классов описать модель предметной области. Кроме того, необходимо создать EMF Generator Model (.genmodel) при помощи мастера EMF Model, который позволит генерировать

программный код модели предметной области.

- Графическое описание (Graphical Definition, .gmfgraph) создаётся с использованием мастера GMFGraph Model и служит для описания графических элементов предметной области.
- Инструментальное описание (Tooling Definition, .gmftool) разрабатывается при помощи мастера GMFTool Model для создания палитры инструментов, которая может быть использована в дальнейшем в графическом редакторе при создании моделей.
- Модель соответствия (Mapping Model, .gmfmap) связывает между собой модель предметной области, графическое и инструментальное описания. Таким образом, целью GMF является возможность переиспользования графического описания для нескольких моделей. Отдельная модель соответствия используется для связи графического и инструментального описаний с выбранной предметной областью (или несколькими областями).
- На основе модели соответствия создается модель генератора (Generator Model, .gmfgen). Она является аналогом .genmodel. С её помощью генерируется код графического редактора GMF.

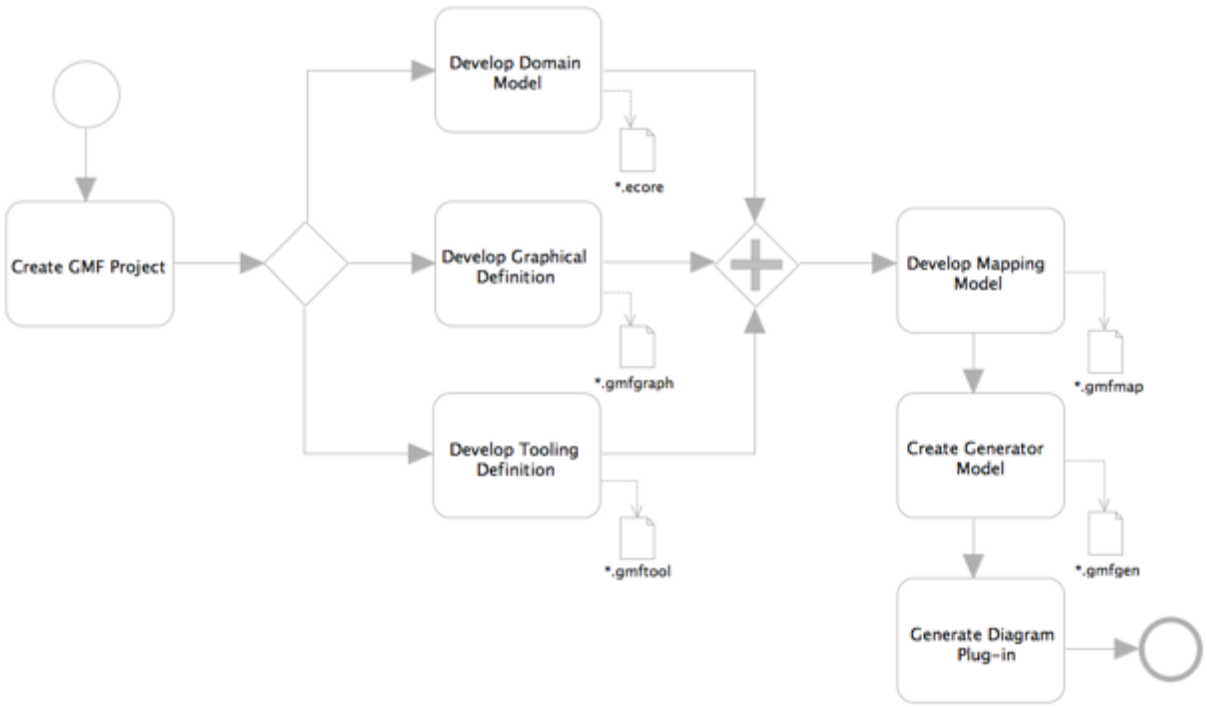

Рис.3. Обзор GMF

Список литературы

1. *Сухов А.О.* Сравнение систем разработки визуальных предметно-ориентированных языков
2. Eclipse Modeling Framework Project (EMF): http://www.eclipse.org/modeling/emf/
3. GEF (Graphical Editing Framework): http://www.eclipse.org/gef/
4. Graphical Modeling Framework: http://wiki.eclipse.org/GMF
5. *Сухов А.О.* Классификация предметно-ориентированных языков и языковых инструментариев
6. MetaEdit+ Version 5.0 Workbench User's Guide: https://www.metacase.com/support/50/manuals/mwb/Mw.html
7. MetaEdit+ Version 5.0 User's Guide: https://www.metacase.com/support/50/manuals/meplus/Mp.html
8. *Кознов Д.В.* Визуальное моделирование: теория и практика. http://www.intuit.ru/studies/courses/1041/218/info
9. http://www.i2r.ru/static/255/out_15530.shtml