Обзор Eclipse Modeling Project*

А. В. Сорокин
Anton.Sorokin@lanit-tercom.com

Д. В. Кознов
dkoznov@yandex.ru

В данной статье приводится обзор одной из наиболее бурно развивающихся и перспективных платформ для разработки средств визуального моделирования — Eclipse Modeling Project. Этот проект разрабатывается открытым (open-source) сообществом, объединяя людей из университетов и индустрии вокруг различных задач по созданию программных средств поддержки модельно-ориентированной разработки ПО. Ядром проекта является технология EMF (Eclipse Modeling Framework), с которой в той или иной степени связано несколько десятков других инструментов, входящих в состав проекта. Эти инструменты имеют различную степень зрелости — от широко известных и активно используемых в индустрии технологий, таких как EMF, GEF и GMF, до пробных университетских разработок. На настоящий момент не существует краткого обзора всего проекта в целом. Следует также отметить, что проект интенсивно развивается, поэтому востребована актуальная информация о нем. Наконец, по данной тематике имеется недостаток сведений на русском языке. В связи со всем этим статья является востребованной. Представленная в ней информация описывает состояние Eclipse Modeling Project на сентябрь 2010 года.

Введение

За последние годы инструменты визуального моделирования значительно эволюционировали. Ещё недавно на рынке доминировали средства, поддерживающие стандарт UML [15], однако сей-

*Работа выполнена при финансовой поддержке РФФИ (грант 11-01-00622а).

час становится очевидным смещение интереса в сторону разработки и использования DSM-решений. Предметно-ориентированное моделирование (Domain-Specific Modeling, DSM) является подходом к разработке ПО, который повышает уровень абстракции используемых концепций. Этот подход позволяет создавать ПО в терминах целевой предметной области, при этом программный код приложения генерируется автоматически по таким высокоуровневым спецификациям. Последнее оказывается возможным, поскольку и язык моделирования, и кодогенератор созданы специально для данной области — часто для специальной компании, для одного крупного проекта или линейки продуктов [14, 25]. Этот подход активно распространяется в индустрии в силу следующих причин: из-за трудностей практического использования стандартных средств визуального моделирования — UML и соответствующих программных пакетов [6]; благодаря развитию парадигмы разработки линейки продуктов (software product lines), что требует в компаниях создания специальной инфраструктуры [9]; вследствие активного развития средств спецификации и реализации визуальных языков — Eclipse Modeling Project [8], Miscrosoft DSL Tools [9, 19], MetaEdit+ [16]. Все три упомянутые технологии поддержки DSM-подхода предназначены для разработки полноценных систем визуального моделирования и позволяют определять абстрактный и конкретный синтаксисы новых метамоделей, автоматически создавать для этих метамоделей редакторы и другие инструменты поддержки.

В статье приводится обзор одной из наиболее мощных и перспективных платформ для разработки DSM-решений — Eclipse Modeling Project [8]. Данная платформа предлагает целое семейство различных инструментов для разработки и визуализации моделей, средства для сравнения и слияния моделей, утилиты для генерации по модели исходного кода на различных языках программирования и многое другое. Стоит отметить, что уже существуют обзоры и описания этой платформы (см., например, обзор всего проекта в книге [10], его отдельных частей — EMF [24], GEF [11] и др.). Однако на данный момент нет краткого (в рамках одной статьи) и полного обзора всего проекта. Кроме того, проект бурно развивается, и требуются обновленные обзоры — данная статья отражает положение вещей в Eclipse Modeling Project на сентябрь 2010 года.

Наконец, существующая литература по этому вопросу полностью англоязычна. Все это делает актуальным материал, представленный в статье.

1. Основные концепции и понятия модельно-ориентированной разработки

1.1. Предметная область, модель, метамодель и метаметамодель

При визуальном моделировании программного обеспечения используются следующие уровни абстракции — предметная область, модель, метамодель и метаметамодель.

Предметная область (domain) — это тот фрагмент реальной действительности, который подвергается моделированию.

Модель (model) — это упрощенное описание предметной области. Более простая модель дает возможность не рассматривать все бесконечное многообразие вариаций и свойств объектов предметной области, а сосредоточиться лишь на некоторых.

Метамодель (metamodel) — это язык для задания моделей, т. е. язык моделирования. Формальное описание UML является примером метамодели.

Метаметамодель (metametamodel) — это язык описания различных языков моделирования. Метаметамодели оказываются востребованы в силу разработки различных визуальных языков и необходимости стандартизации средств описания метамоделей. Так, комитет OMG, ведущая в мире организация по стандартизации моделирования в сфере информационных технологий[1], кроме UML развивает и другие языки моделирования. Вот некоторые из них:

- BPMN — язык для моделирования бизнес-процессов [7];
- IMM — язык для информационного менеджмента [13];
- SBVR — язык для создания бизнес-словаря и бизнес-правил [23];

[1] www.omg.org

- SysML — язык для анализа и проектирования ПО; он является более компактным, чем UML, не содержит многих программно-ориентированных и объектно-ориентированных черт UML, дополнительно поддерживает такие понятия, как модели, представления и точки зрения [22].

OMG поддерживает стандарт MOF (Meta Object Facility) — метаметамодель для задания метамоделей [18]. С помощью MOF специфицирован язык UML. MOF, фактически, является «вырезкой» из UML в части диаграмм классов. Метаметамодели оказываются также востребованными в виду развития DSM-подхода: различные DSM-платформы по разработке специфических предметно-ориентированных языков и соответствующих программных средств поддержки включают в себя средства для задания новых языков с возможностью дальнейшей более-менее автоматической генерации графических редакторов для этих языков. Таким образом, DSM-платформы реализуют различные метаметамодели: язык Goffer в MetaEdit+ [14], модели GMF, язык в Microsoft DSL Tools [19] и т. д. Одной из таких метаметамоделей является также формат Ecore, который поддерживается в проекте EMF и будет подробно рассмотрен ниже. Обзор подобных средств и более подробное обсуждение всех этих понятий можно найти в работах [2, 3].

1.2. Абстрактный, конкретный и служебный синтаксис визуальных языков

При изучении различных языков — в языковедении и лингвистике, искусствоведении, кинематографе, математике, программировании и т. д. — принято выделять синтаксис, семантику и прагматику языка.

Синтаксис — это взаимосвязи между знаками языка, которые определяют правила построения из этих знаков корректных текстов на данном языке.

Семантика — это значения знаков языка, т. е. их проекции в предметную область.

Прагматика — это способы применения языка пользователями. Детали этих определений и дальнейшие ссылки можно найти, например, в работе [2].

В области разработки и использования визуальных языков выделяют следующие виды синтаксиса — абстрактный, конкретный и служебный [9, 2].

Абстрактный синтаксис (abstract syntax) — это определение структуры текста (визуальной модели). Абстрактный синтаксис задает все типы конструкций, их возможные атрибуты и связи, т. е. задает структуру графа модели. Это наиболее формализуемая часть языка, и здесь широко используются различные формальные подходы.

Конкретный синтаксис (concrete syntax) является графической нотацией языка или просто нотацией, т. е. он задает правила изображения конструкций языка на диаграммах.

Служебный синтаксис (serialization syntax) — это формат хранения визуальных спецификаций, выполненных с помощью данного языка. Одним из самых распространенных средств служебного синтаксиса является XMI (XML Metadata Interchange) — стандарт OMG, описывающий единый XML-формат для обмена визуальными моделями (в частности, UML-моделями) между различными визуальными средствами, в том числе и от разных производителей [20].

1.3. Трансформации моделей

Разработка сложного программного обеспечения может вестись в рамках концепции модельно-ориентированной разработки, когда именно модели ПО становятся основными артефактами разработки, и уже на их основе генерируются код, структура базы данных и т. д. Консорциум OMG предоставляет стандарт QVT (Query/View/Transformation) [17], описывающий средства для спецификации трансформации моделей.

Несколько упрощенно можно сказать, что перед QVT стояли следующие задачи: пусть у нас есть две метамодели, описанные с помощью MOF. Требуется автоматически определять по двум моделям, соответствуют ли они друг другу, а кроме этого при необходимости автоматически трансформировать одну в другую. Также уделялось особое внимание необходимости поддержки итеративных изменений моделей и их синхронизации (при изменении одной мо-

дели вторая не перегенерируется заново, а лишь соответственно изменяется).

Стандарт QVT имеет двухуровневую архитектуру.

1. На верхнем уровне находится удобный для использования декларативный язык Relational, который позволяет описывать отношения между MOF-метамоделями. Он поддерживает сложное сравнение по шаблонам и неявно сохраняет информацию о процессе выполнения трансформации. Язык позволяет также проверять выполнение отношений между определенными элементами модели, выбираемыми по их паттернам. Кроме текстового формата, QVT описывает графический формат записи на Relational.
2. На нижнем уровне находится простой язык Core. Он поддерживает только сравнение плоских множеств переменных по шаблонам, вычисляя условия относительно этих переменных для заданного множества моделей. Этот язык по мощности равнозначен Relational, его семантика определяется гораздо проще, однако из-за его простоты описания трансформаций на нем оказываются чрезмерно длинными и сложными для чтения.

Сами авторы стандарта говорят о том, что если Relational можно сравнить с языком Java, то Core аналогичен Java байт-коду. Действительно, в одной из частей спецификации описан процесс трансформации из языка Relational в Core.

В дополнение к декларативным языкам Relational и Core, QVT определяет возможность императивного описания трансформаций (Procedural QVT). Он является простым Java-подобным языком с набором специальных конструкций.

Еще одним распространенным языком задания трансформации является ATL (ATLAS Transformation Language) [30], активно развиваемый в Eclipse Modeling Project.

2. Ядро Eclipse Modeling Project

2.1. Обзор проекта

Eclipse Modeling Project (EMP) состоит более чем из десятка проектов, реализующих различные идеи в области визуального моделирования. На рис. 1 представлена обзорная диаграмма компонентов Eclipse Modeling Project. Как видно из данной диаграммы, ключевым проектом в Eclipse Modeling Project является EMF. Он предназначен для создания метамоделей новых визуальных языков. Остальные проекты, входящие в состав Eclipse Modeling Project, в той или иной степени используют возможности EMF, предоставляя упрощённые механизмы работы с метамоделями. Ниже перечислены проекты, входящие в состав Eclipse Modeling Project.

- Проект GMF позволяет создавать для нового языка готовый графический редактор без написания дополнительного кода «вручную», используя EMF как средство для работы и хранения моделей.
- Проект MDT посвящен реализации промышленных стандартов визуального моделирования средствами Eclipse (UML и др.).
- Проект M2M реализует поддержку трансформаций моделей, заданных средствами EMF.
- Проект M2T обеспечивает трансформацию моделей в текстовые описания (документацию).
- Проект AMP на основе EMF организует работу с моделями агентов (задание агентных сетей, симуляцию, генерацию целевого кода).
- Проект XText посвящен работе с моделями, заданными в текстовой нотации, и может как использовать EMF, так и обходиться без него (на рис. 1 это отмечено стереотипом <optional> на связи с EMF).
- Аналогично, проект GEF, предназначенный для разработки графической составляющей визуальных средств, может как интегрироваться с EMF, так и обходиться без него (на рис. 1 это отмечено стереотипом <optional> на связи с EMF). Более того, данный проект формально даже не входит в Eclipse

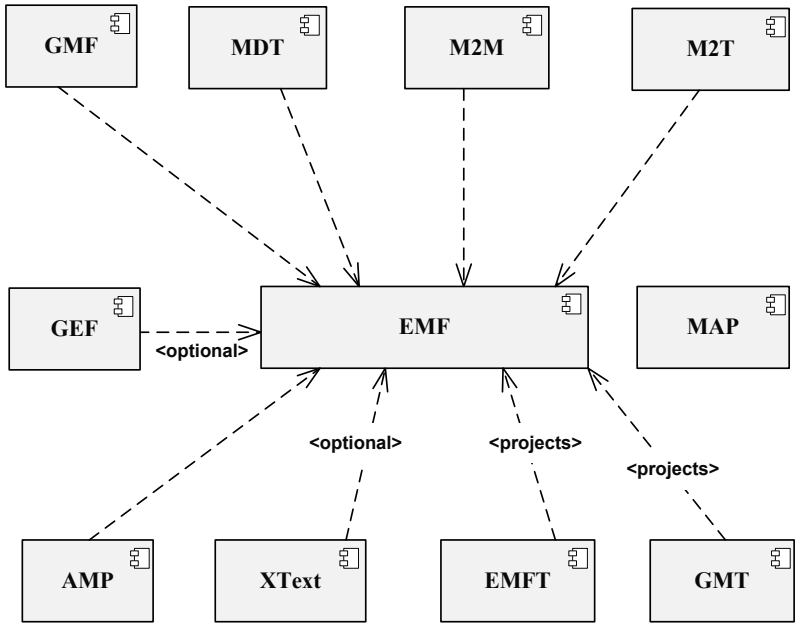

Рис. 1. Схема Eclipse Modeling Project

Modeling Project, но мы включили его в наш обзор в виду его большой важности, а также потому, что некоторые проекты из Eclipse Modeling Project его используют.

- EMFT и GMT являются служебными проектами. EMFT содержит проекты, которые претендуют быть включенными в EMF по достижении зрелости и стабильности. GMT содержит различные «начинающие» проекты, которые посвящены модельно-ориентированной разработке вообще. Они связаны с EMF в том смысле, что входящие в них проекты его используют, что отмечено на рис. 1 стереотипом <projects>.
- Проект MAP является служебным и предназначен для более удобного развертывания компонент проектов, входящих в состав Eclipse Modeling Project.

2.2. Eclipse Modeling Framework (EMF)

Проект EMF предлагает полноценный инструментарий для разработки метамоделей и поддерживает работу с моделями, соответствующими этим метамоделям. Данная функциональность оказывается востребованной для графических редакторов. Визуальные спецификации обычно делятся на диаграммы (views) и модели. На диаграммах представляются элементы модели, на них же можно добавлять и удалять эти элементы. А модель, как правило, имеет два представления. Во-первых, она как-то хранится — в базе данных, в XML-файле и т. д. Во-вторых, она имеет представление в виде классов/объектов, с которыми удобно работать приложению, реализующему редактор, и эти объекты умеют себя сохранять и восстанавливать (serialization/deserialization). Кроме того, различные другие приложения, помимо графических редакторов, используют возможность задать схему классов, экземплярами которых являются сохраняемые объекты, а также сгенерировать соответствующие классы по этой схеме и работать с ними программным способом.

Метамодели в EMF задаются с помощью специального языка, называемого Ecore. Упрощённая метамодель этого языка (Ecore-метаметамодель) представлена на рис. 2. Каждый элемент нового языка, для которого строится метамодель с помощью Ecore, является экземпляром класса `EClass`. Этот класс характеризуется именем, а также списком атрибутов (класс `EAttribute`), каждый из которых имеет тип (класс `EDataType`). Кроме того, классы в создаваемой новой метамодели могут содержать ссылки на классы-предки (связь `eSuperTypes`), а также иметь ассоциации друг с другом. Средством для задания ассоциаций в Ecore является класс `EReference`, представляющий конец ассоциации. Конец агрегируется классом (агрегирование `eReference`) и имеет следующие атрибуты: имя (`name`), признак того, является ли этот конец агрегированием (булевский атрибут `containment`), верхнюю и нижнюю границы множественности (атрибуты `lowerBound` и `upperBound`). Конец ассоциации также имеет ссылку `eOpposite` на другой (парный) конец (ассоциация имеет два конца — по одному с каждой стороны соединяемых классов). Эта ссылка не нулевая лишь в том

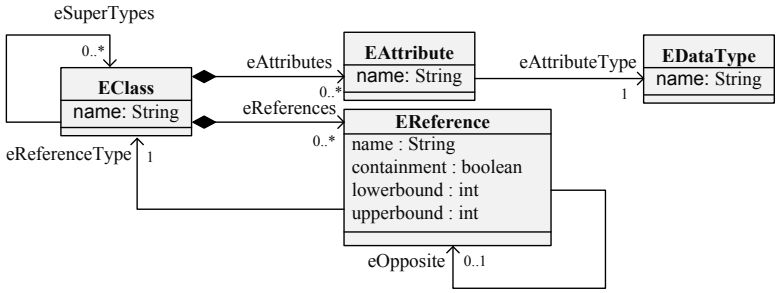

Рис. 2. Ecore метаметамодель

случае, когда у задаваемой ассоциации существует направление от этой роли к парной. В противном случае (т. е. задается однонаправленная ассоциация, и направления от данной роли в ней нет) для доступа к связываемому ассоциацией классу используется только ссылка `eReferenceType` (даже в случае однонаправленной ассоциации нужно знать, какие классы соединяются ассоциацией). Ecore-метамодели хранятся в XMI-формате.

Любому элементу Ecore-метаметамодели в EMF соответствует Java-класс, реализующий соответствующий интерфейс, задающий методы для установки и чтения свойств элементов. Сохранение объекта этого класса — то есть элемента новой метамодели — происходит путём применения механизма сериализации (serialization), позволяющего получить XMI-представление элемента.

EMF позволяет задавать Ecore-метамодель множеством различных способов, включая XMI-описание, XML-схему, аннотированный Java-код, схему базы данных, UML-модели и т. д.

Ключевым компонентом технологии EMF является EMF Core, состоящая из проектов EMF, EMF.Edit и EMF.Codegen. Проект EMF содержит описание Ecore-метаметамодели, среду для работы с созданной метамоделью, включая поддержку уведомлений об изменениях метамодели, а также программный интерфейс для работы с сущностями, описанными в метамодели. Проект EMF.Edit предоставляет набор базовых классов для реализации возможностей редакторов метамоделей. Кроме того, он обеспечивает автоматиче-

скую поддержку команд Undo/Redo. Проект EMF.Codegen предоставляет функционал (включая пользовательский интерфейс) для генерации кода по Ecore-метамодели[2], а также поддерживает перегенерацию кода с сохранением пользовательских модификаций. На основе метамодели EMF.Codegen генерирует заготовку программного средства для работы с соответствующими моделями:

- интерфейсы и классы для каждого метакласса, заданного в метамодели Ecore, а также соответствующие классы-фабрики[3] для генерации элементов модели;
- классы-адаптеры (ItemProviders), предназначенные для редактирования и отображения классов модели (например, в древовидной структуре);
- редактор для работы с моделями, соответствующими данной метамодели. Этот редактор является очень простым, не графическим (графические редакторы генерирует GMF) и поддерживает создание и редактирование моделей через специальное окно вида TreeViewControl, где редактируемая модель представляется в виде дерева объектов, поддерживаются также операции Undo/Redo и др. Кроме того, EMF предоставляет богатые возможности по дополнительной настройке полученного редактора через использование механизма оповещений (Notification). Данный механизм позволяет создавать собственные обработчики для различных действий, производимых над моделью. Предполагается, что эта заготовка может быть доделана и расширена «вручную» или с помощью других технологий, в которые EMF может быть встроен.

Рассмотрим остальные проекты, входящие в состав EMF. Проекты CDO, Net4j и Teneo предназначены для предоставления программных средств распределенной работы с Ecore-моделями, хранящимися в базе данных, а также для сохранения моделей в базе данных и их обратной загрузки (serialization/deserialization).

[2]При этом используется так называемая «модель генератора» (generator model), в которой нужно задать параметры генерации кода по исходной Ecore-метамодели.

[3]В данном контексте класс-фабрика — это класс, созданный в соответствии с шаблоном проектирования Abstract Factory. Последний определяет интерфейс для создания семейств взаимосвязанных объектов, не определяя их конкретные свойства [1].

Поддерживаются следующие базы данных: Derby, H2, HSQLDB, MySQL, Oracle (TBD). Проект Teneo позволяет задавать соответствие между элементами Ecore-метамодели и схемы реляционной базы данных, а также получать элементы соответствующей модели с помощью специального языка запросов HQL (Hibernate Query Language). Проект CDO (Connected Data Objects) обеспечивает распределенный программный доступ к моделям — поддержку бизнес-транзакций, версионирование моделей, механизм уведомлений об изменениях и пр., — а также репозиторий для хранения моделей на сервере. Протокол работы с репозиторием реализуется в проекте Net4j. Сравнение и слияние моделей реализовано в рамках проекта Compare. Использование языков запросов с SQL-подобным синтаксисом для работы с моделями предоставляется библиотеками Model Query и Model Query 2. Проект Validation Framework реализует программный интерфейс для задания ограничений к Ecore-метамоделям на языках OCL или Java с последующей валидацией этих ограничений для их моделей, а также средства для обхода модели и информирования о событиях, произошедших в процессе валидации. Задачей проекта Model Transactions является разработка средств для упрощения работы с моделями — обеспечение многопоточного чтения и записи моделей, автоматическая проверка семантической корректности модели, пакетная обработка событий, а также поддержка команд Undo/Redo и истории изменений.

Проект EMF является ключевым в Eclipse Modeling Project, на нем основывается большое количество других проектов в рамках платформы Eclipse, в том числе в областях, не связанных с моделированием напрямую. Примером такого использования является проект e4 (следующее поколение Eclipse Rich Client Platform), где при помощи EMF описывается модель распределенного приложения. Среди наиболее известных промышленных применений EMF можно выделить следующие продукты: IBM Rational Software Modeler (компания IBM), Apollo for Eclipse (компания Gentleware AG), Blueprint (компания @-portunity).

Первый выпуск EMF состоялся 26 марта 2003 года (версия 1.0.2). Следующими заметными вехами в проекте стал выход версии 2.0 в июне 2004 года, а также присоединение к проекту компаний JBoss (подразделение известной компании RedHat) и IBM соответ-

ственно в декабре 2004 и июле 2005. С тех пор проект возглавляет Эд Меркс (Ed Merks) из университета им. Саймона Фрейзера (Канада). Согласно проектному плану, очередной выпуск EMF должен был состояться 30 июня 2010 года с выходом версии 2.6.0, однако на момент написания статьи версия 2.6.0 была ещё недоступна.

2.3. Modeling Amalgamation Project (MAP)

Целью проекта MAP является создание решения для более удобного развёртывания и использования других компонентов из состава Eclipse Modeling Project на целевой платформе. В рамках проекта созданы примеры использования различных технологий из Eclipse Modeling Project, предоставлен интерфейс, позволяющий с бóльшей эффективностью работать с данным проектом, созданы удобные мастера для создания собственных решений, основанных на одном из проектов Eclipse Modeling Project. Проект MAP может поставляться в различных комплектациях.

Проект стартовал в октябре 2008 года и до июля 2009 развивался исключительно силами сообщества независимых разработчиков. В декабре 2009 года к проекту присоединилась компания Obeo, а ее специалист Седрик Брун (Cedric Brun) стал лидером проекта. Последняя версия проекта вышла 23 июня 2010 года.

3. Средства разработки визуальных пакетов

3.1. Graphical Modeling Framework Project (GMF)

Задачей проекта GMF[4] является предоставление разработчикам визуальных средств высокоуровневых возможностей для автоматической генерации графических редакторов на основе заданного множества их моделей. Речь идет именно о реализации DSM-средств, а не столько о профессиональной разработке сложных визуальных систем. Чтобы обеспечить генерацию полноценного графического редактора, GMF предлагает создать несколько метамо-

[4]Необходимо отметить, что GMF — это устаревшее название данного проекта. Его текущее его название — GMP (Graphical Modeling Project). Однако в статье мы используем старое название, так как оно широко распространено.

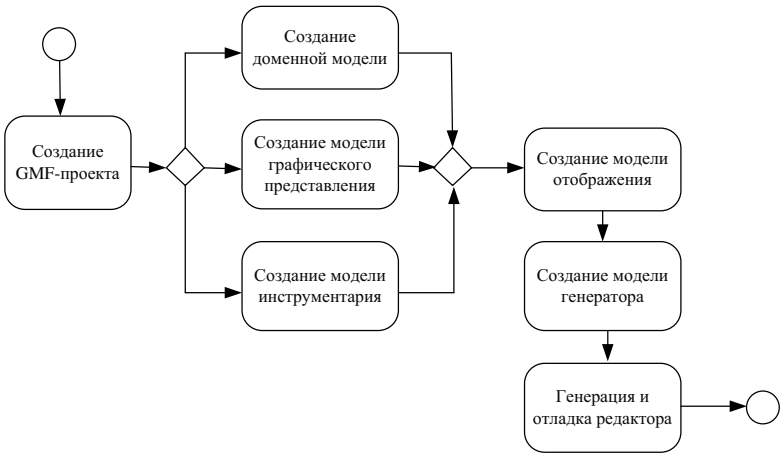

Рис. 3. Процесс разработки графического редактора с помощью технологии GMF

делей (далее, следуя терминологии GMF, будем называть их в этом разделе «моделями»). Процесс разработки графического редактора на основе GMF изображен на рис. 3 и состоит из следующих шагов.

1. Разработка описания доменной модели (domain model), например, с помощью графического редактора GMF Ecore (выходной файл *.ecore).
2. Разработка описания графического редактора — модели графического представления (graphical model, выходной файл *.gmfgraph).
3. Разработка описания вспомогательных средств — палитры объектов, списка действий, меню графических объектов, то есть модели инструментария (tooling model, выходной файл *.gmftool).
4. Разработка описания связки доменной модели и графических объектов, а также вспомогательных средств, описание ограничений на OCL или Java, т. е. создание модели отображения (mapping model, выходной файл *.gmfmap).

5. Создание описания генератора, графического редактора по модели отображения с доработками — модель генератора (выходной файл *.gmfgen).
6. Генерация кода целевого графического редактора, запуск отладочного экземпляра Eclipse Application и отладка.

GMF основан на Eclipse-технологиях EMF и GEF.

Лидером проекта является Артём Тихомиров из компании Borland. Первый выпуск проекта состоялся 30 июня 2006 года. Существенное развитие проекта произошло 27 июня 2007 года с выходом версии GMF 2.0. Были добавлены новые мастера для работы с моделями, улучшена навигация по элементам моделей, для задания модели отображения был предоставлен специальный графический редактор и т. д. Выход очередной версии 2.3.0 был намечен на 23 июня 2010 года, однако на момент написания статьи новая версия так и не вышла. В качестве примера промышленного применения GMF можно упомянуть продукт Apollo for Eclipse (компания Gentleware AG), который был создан на основе этой технологии.

3.2. Graphical Editor Framework (GEF)

Проект GEF служит для создания графических редакторов произвольного назначения. GEF основывается на концепции модель-представление-контроллер (MVC)[5] и состоит из двух основных проектов: org.eclipse.draw2d и org.eclipse.gef. При этом модель создаётся вне GEF самим разработчиком редактора, например, с помощью EMF. В качестве контроллера выступают объекты, реализующие интерфейс EditPart, в частности, GraphicalEditPart, который отвечает за отображение элемента модели. В качестве представления используются объекты, реализующие интерфейс IView.

Проект org.eclipse.draw2d предоставляет поддержку расположения графических элементов и их отрисовку, также реализацию фигур, границ, соединений элементов, указателей и всплывающих

[5]MVC (Model View Controller, модель-представление-контроллер) — один из наиболее распространённых шаблонов проектирования пользовательского интерфейса. Суть его сводится к разделению модели, её представления и контроллера, задающего правила взаимодействия модели и её представления [5].

подсказок, слоёв (в том числе прозрачных), гибкой системы координат, реализует окно обзора диаграммы и печать.

Проект org.eclipse.gef предоставляет поддержку следующего функционала: создание, выбор, соединение графических элементов на диаграмме; панель инструментов; изменение размеров элементов и реализация изгибов соединений; графическое и древовидное представление модели; контроллеры для отображения и изменения модели через представление; операции Undo/Redo через стек команд.

Проектом руководит Энтони Хантер (Anthony Hunter) — менеджер из компании IBM Rational Software (Канада). Проект развивается компаниями IBM и Itemis (Германия). Проект стартовал в июне 2002 года. Последний выпуск состоялся 30 июня 2010 года.

4. Model Development Tools (MDT)

Данный проект предназначен для реализации поддержки в Eclipse существующих индустриальных стандартов моделирования, например, UML. Кроме UML (версия 2.x) реализована также поддержка следующих языков.

1. BPMN2 (Business Process Model and Notation 2.0) — реализация стандарта OMG BPMN [7].
2. IMM (Information Management Metamodel) — реализация стандарта OMG IMM [13].
3. OCL (Object Constraint Language) — реализация стандарта OMG OCL [21] для EMF-моделей.
4. SBVR (Semantics of Business Vocabulary and Business Rules) — реализация стандарта OMG SBVR [23].

Также в рамках проекта MDT разрабатываются следующие компоненты.

1. Papyrus — компонента, предоставляющая среду для редактирования произвольных метамоделей, созданных с использованием EMF. На данный момент, в частности, поддерживается работа со следующими метамоделями: UML, SysML [22] и MARTE [31]. При этом задачей проекта является создание максимально гибкого (с точки зрения числа настроек) интерфейса для редактирования произвольных метамоделей.

2. MST (Metamodel Specification Tools) — компонента, основной задачей которой является предоставление инструментария для создания MOF-совместимых метамоделей, а также для генерации соответствующих спецификаций. На момент написания статьи компонента была еще недоступна для скачивания.
3. XSD (XML Schema Document) — компонента, которая обеспечивает поддержку схем XML (в формате XSD), а также предоставляет интерфейс для работы с документами XML с использованием модели DOM[6].

Проект MDT активно развивается с 2003 года. В 2004-м году к развитию проекта присоединилась компания IBM, чьё участие, впрочем, оказалось довольно незначительным. В разные годы проект поддерживался также компаниями Intalio, Montages AG, Obeo, Open Canarias S.L. и др. Наиболее существенный вклад в развитие проекта внесли представители компаний Borland Software и CEALIST, а также компания Thales, поддерживающая проект в настоящее время.

Проект возглавляет Кенн Хасси (Kenn Hussey), являющийся заместителем руководителя по разработке программного обеспечения в компании Cloudsmith, Inc. (США). Проект развивается компанией Thales. Регулярные выпуски новых версий происходят с июня 2007 года. Последний выпуск состоялся 23 июня 2010 года. На момент написания статьи доступны для загрузки лишь компоненты, реализующие UML, XSD и OCL.

5. Трансформации моделей

5.1. Model-to-Model Transformation (M2M)

Данный проект предназначен для реализации трансформаций моделей. M2M состоит из трех проектов.

1. ATL — среда для описания трансформаций исходных моделей в целевые на основе языка ATL. Редактор поддерживает

[6]DOM (Document Object Model) — программный интерфейс для доступа к содержимому файлов в форматах HTML, XHTML и XML.

подсветку синтаксиса, имеет встроенный отладчик. Сам язык включает в себя как декларативные конструкции, позволяющие задавать соотношения между моделями и проверять их выполнение, так и императивные, с помощью которых можно реализовывать более сложную логику. Технология разрабатывалась как альтернатива стандарту QVT, разработанному консорциумом OMG.

2. Procedural QVT — реализация языка Procedural QVT.
3. Declarative QVT — реализация Core и Relational частей стандарта QVT.

Проект ATL активно и успешно развивается. Попытки использовать его «движок» для реализации QVT предпринимались, начиная с 2008 года. Однако возникли проблемы в связи с тем, что ATL ориентирован на Ecore и EMF, а QVT предполагает использовать MOF в качестве средства задания метамоделей.

Лидером проекта является Фредерик Юалт (Frederic Jouault), который работает в институте INRIA и в университете Алабамы (США). Проект поддерживается институтом INRIA и компаниями Obeo, Borland. Первый выпуск проекта состоялся 25 июня 2008 года, хотя стартовал проект в ноябре 2004. Последний выпуск — 23 июня 2010 года.

5.2. Model-to-Text Transformation (M2T)

Данный проект предназначен для генерации текстов (документации, кода) по существующим моделям. Он состоит из трех проектов.

1. JET — обеспечивает генерацию исходного кода по заданной модели. Компонент активно используется в EMF для генерации кода на языке Java по заданной Ecore-метамодели.
2. Acceleo — реализует стандарт OMG MOFM2T [26] для преобразования моделей в текстовое представление.
3. Xpand — реализует одноименный статически типизированный язык, используемый для генерации исходного кода по Ecore-метамоделям. Особенностью Xpand является возможность задания параметров генерации исходного кода с помощью файлов с описанием шаблона генерации. Также Xpand позволя-

ет задавать правила для проверки метамодели, по которой будет осуществлена генерация кода. Таким правилом, например, может быть ограничение на минимальную длину имени элемента метамодели.

Лидером проекта является Пол Элдер (Paul Elder) — менеджер компании IBM Rational Software (Канада). Проект развивают компании Obeo, Itemis, IBM. Первый выпуск проекта состоялся в 2007 году, последний — 23 июня 2010 года. К сожалению, документации по этому проекту почти нет.

6. Специализированные проекты

6.1. Agent Modeling Platform (AMP)

Задачей данного проекта является создание расширяемого инструментария (framework) для поддержки моделирования интеллектуальных агентов (Agent-Based modeling, ABM).

Проект состоит из трёх частей.

1. Agent Modeling Framework (AMF) содержит описание метамодели агентных сетей (на основе Ecore), редактор моделей этих сетей, генератор исполняемого кода (для эмуляции сетей агентов и создания целевого кода на Java), а также тестовых сценариев[7].
2. Agent Execution Framework (AXF) предоставляет службы и интерфейс для исполнения, представления и управления моделями агентов. Этот проект не связан только с AMP и ABM и может использоваться вообще для любых коллекций объектов, для которых реализованы соответствующие интерфейсы.
3. Agent Graphics Framework (AGF) — это графическая среда, предназначенная для визуализации процесса работы агентов и их взаимодействия друг с другом. Среди прочих библиотек использует GEF.

Проект AMP развивается под руководством Майлса Паркера (Miles Parker), основателя компании Metascape LLC. Последняя

[7]Тестовые сценарии создаются с использованием библиотеки JUnit, предназначенной для модульного тестирования Java-программ [27].

версия данного компонента вышла 8 января 2010 года (версия 0.8.0). Проект очень молодой — ему фактически ещё только год, но развивается очень бурно, можно видеть большое число commits исходного кода проекта, а также на сайте проекта в последнее время опубликовано большое количество видео выступлений его участников.

6.2. Textual Modeling Framework (XText)

Данный проект посвящен поддержке метамоделей, заданных в текстовой нотации. В настоящее время он активно развивается в сторону от моделирования, реализуя многочисленные возможность по поддержке текстовых языков, в частности, подсветку синтаксиса, автоматическое редактирование текстов с использованием всплывающих подсказок, проверку корректности конструкций языка непосредственно в процессе их ввода. XText также предоставляет богатый набор интерфейсов, позволяющий расширять и изменять существующую функциональность.

Руководителем проекта XText является Свэн Эффтиндж (Sven Efftinge), работающий консультантом в компании itemis (Киль, Германия), а также возглавляющий open-source направление в этой компании. 23 июня 2010 года состоялся выпуск первой версии.

7. Незрелые проекты

7.1. Eclipse Model Framework Technology (EMFT)

Проект EMFT является, в терминах Eclipse, «инкубационным» и служит для хранения проектов, которые не достигли еще достаточной стабильности или уровня зрелости, необходимых для того, чтобы быть включенными в проект EMF. Для данного проекта не предусмотрено выпусков — каждый отдельно взятый проект в EMFT существует относительно независимо и по достижении определённого уровня развития переходит в состав проекта EMF. Как и в случае с EMF, лидером EMFT является Эд Меркс.

Ниже перечислены наиболее заметные проекты, входящие в состав EMFT на момент написания статьи.

1. Ecore Tools — среда для разработки Ecore-моделей. Предоставляет графический редактор, а также интерфейс для до-

ступа к другим утилитам (компонентам) проекта EMF, позволяющим проверять корректность модели, осуществлять поиск по модели, сравнение и слияние моделей, генерировать код из модели и т. п. Также поддерживается одновременное редактирование набора диаграмм. В дальнейшем планируется предоставлять возможности для рефакторинга[8].

2. Mint расширяет возможности JDT[9], предоставляет улучшенные средства для работы со сгенерированным по модели Java-кодом.
3. Search предоставляет инфраструктуру для задания и исполнения поисковых запросов над Ecore-моделями.
4. EMFatic — текстовый редактор Ecore-метамоделей с Java-подобным синтаксисом (текстовое представление модели может быть преобразовано в Ecore-метамодель).
5. EEF предоставляет расширенные (и потенциально более удобные, чем EMF) средства для редактирования метамоделей. Заменяет традиционные элементы в виде списка свойств и значений на диалоги с различными элементами интерфейса, зависящими от типа редактируемого объекта.
6. Texo — компонента для генерации кода серверной части Web-приложений по аннотированной Ecore-метамодели в виде POJO-объектов[10].
7. EMF4Net — компонента для генерации C# кода по Ecore-метамодели. Проект находится на очень ранней стадии — готовой к использованию компоненты для загрузки ещё не предоставлено.
8. Temporality — компонента для реализации поддержки версионирования в существующих Ecore-моделелях. Предполагается, что наследование каждого элемента модели от базового класса, предоставляемого данной компонентой, позволит вести историю изменений для этого объекта.

[8] Рефакторинг — это процесс изменения программной системы, при котором не меняется её внешнее поведение, но улучшается её внутренняя структура [4].

[9] JDT (Java Development Tools) — это проект Eclipse, задачей которого является создание интегрированной среды разработки на языке Java [29].

[10] POJO (Plain Old Java Objects) — концепция бизнес-объектов Мартина Фаулера. С деталями этого подхода можно познакомиться на сайте http://en.wikipedia.org/wiki/Plain_Old_Java_Object.

7.2. Generative Modeling Technologies (GMT)

Данный проект служит для временного хранения различных прототипов проектов. В отличие от EMFT, задачей которого является хранение проектов, предназначенных для расширения функциональности EMF, в GMT входят разнородные проекты. Постулируется, что они должны быть связаны с модельно-ориентированной разработкой ПО (Model Driven Engineering MDE). Проекты из GMT реализуют трансформации, композиции, и верификацию моделей и т. п.

Ниже представлен список проектов, которые на данный момент включены в GMT.

1. VIATRA2 (VIsual Automated model TRAnsformations) — проект, предоставляющий инструментарий для создания метамоделей с использованием специального языка; включает также язык для описания трансформаций моделей (в том числе в текстовое представление, например, в код на языке программирования) и высокопроизводительную среду для исполнения трансформаций. Проект активно развивается и используется в целом ряде исследовательских разработок. Стоит отметить, что применяемые в проекте форматы хранения метамоделей, а также язык описания трансформаций, не являются стандартом. Это, в частности, делает невозможным интеграцию данного проекта с другими проектами платформы Eclipse.
2. AM3 (AtlanMod MegaModel Management) — проект, задачей которого является управление метамоделями. Создание новых метамоделей в рамках проекта не предполагается (хотя расширение множества изначально доступных метамоделей с помощью других средств допускается). Проект не предоставляет возможностей для реализации трансформаций или композиций моделей — предполагается, что данные возможности уже доступны. Проект предназначен для создания инструментария по управлению различными артефактами разработки — исходными кодами программ, моделями, исполняемыми файлами и т. д., а также для управления соответствующими расширенными метамоделями.

3. AMW (ATLAS Model Weaver) — проект, позволяющий создавать связи между моделями. Данные связи хранятся в специальной модели, называемой Weaving Metamodel. В качестве внешнего ключа для элементов моделей по умолчанию используются ID элементов (XMI ID), однако допускается и переопределение данного поведения.
4. Epsilon — реализация семейства языков, предназначенных для выполнения различных операций над Ecore-метамоделями, в частности, генерация кода. Поддерживается также проверка корректности, сравнение, слияние, миграция и рефакторинг, M2M-трансформация моделей.
5. EuGENia — проект, который позволяет генерировать следующие модели GMF по аннотированной EMF-модели: модель инструментария, модель графического представления и отображения. Таким образом, оказывается возможным создать готовый графический редактор с использованием GMF только на основе EMF-модели с аннотацией.
6. HUTN (Human Usable Textual Notation) — реализация OMG-стандарта Human-Usable Textual Notation [12][11]. Для описания моделей используется C-подобный синтаксис, поддерживаются прямое и обратное преобразования в XMI.
7. The EMF EPackage Registry View — проект для просмотра зарегистрированных пакетов, из которых состоит EMF.
8. Exeed — усовершенствованная версия редактора метамоделей для EMF. Позволяет устанавливать для каждого узла дерева в EMF-модели свою иконку и подпись. Соответствующие данные хранятся непосредственно в метамодели в виде аннотаций.
9. ModeLink — редактор связей между Ecore-метамоделями.
10. GEMS (Generic Eclipse Modeling System) предназначен для реализации «мостов», т. е. средств для переноса метамоделей, определённых в других средах, на технологии платформы Eclipse.

[11] HUTN — это текстовый аналог MOF, предназначенный для создания текстовых спецификаций метамоделей с целью облегчения восприятия человеком таких спецификаций и для удобства формальной обработки.

11. MoDisco является инструментарием для создания средств моделирования, нацеленных на поддержку и развитие устаревшего (legacy) программного обеспечения.
12. MOFScript — проект по созданию инструментов для преобразования моделей в текстовое представление (например, в исходный код на языке программирования). Язык, лежащий в основе технологии, в дальнейшем планируется использовать в качестве стандарта для описания трансформаций моделей в текстовое представление.
13. OMCW (Open Model CourseWare) — набор инструментов и учебных материалов для ознакомления с технологиями моделирования.
14. TCS (Textual Concrete Syntax) — компонента, позволяющая задавать конкретный синтаксис для текстовых метамоделей. При этом поддерживается генерация текстового представления метамодели по соответствующей Ecore-метамодели и наоборот.
15. UMLX — экспериментальный проект, в рамках которого предложен графический конкретный синтаксис для языка трансформаций QVT.

Заключение

Подводя итог, хотелось бы сказать, что на данный момент средства моделирования, предоставляемые платформой Eclipse, являются одними из наиболее развитых, что подтверждается большим числом коммерческих и исследовательских проектов, созданных на их основе. В этой статье была предпринята попытка дать общее представление о доступных в рамках EMP-платформы инструментах и технологиях моделирования. Мы считаем, что такая информация полезна разработчикам и исследователям, использующим или планирующим использовать в своей работе средства моделирования, предоставляемые данной платформой. Эта информация полезна также всем специалистам в области разработки средств визуального моделирования, хотя бы только для того, чтобы не изобретать велосипед, а использовать необходимые готовые компоненты из этого проекта в своей работе.

Список литературы

[1] *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д.* Приёмы объектно-ориентированного проектирования, Питер. 2007. 266 с.

[2] *Кознов Д. В.* Основы визуального моделирования. Учебное пособие. М: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. 248 с.

[3] *Павлинов А., Кознов Д., Перегудов А., Бугайченко Д., Казакова А., Чернятчик Р., Фесенко Т., Иванов А.* О средствах разработки проблемно-ориентированных визуальных языков // Системное программирование / Вып. 2, под ред. А. Н. Терехова и Д. Ю. Булычева. СПб.: Изд. СПбГУ, 2006. С. 121–147.

[4] *Фаулер М.* Рефакторинг улучшение существующего кода. Символ. 2005. 432 с.

[5] *Фаулер М.* Архитектура корпоративных приложений. Вильямс. 2008. 544 с.

[6] *Dalgarno M., Fowler M.* UML vs. Domain-Specific Languages. Models and Tools. Vol. 16, N 2. 2008. P. 2–8.

[7] Business Process Model and Notation (BPMN). Version 1.2. OMG. 2009. 316 p.

[8] Eclipse Modeling Project, http://www.eclipse.org/modeling/

[9] *Greenfield J., Short K., Cook S., Kent S.* Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley. 2004. 500 p.

[10] *Gronback R.C.* Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit, Addison-Wesley Professional. 2009. 736 p.

[11] *Hudson R., Shah P., Winchester J.* The Graphing Editing Framework (GEF). Addison Wesley. 2008. 320 p.

[12] Human-Usable Textual Notation (HUTN) Specification. OMG. Version 1.0. 2004. 74 p.

[13] Information Management Metamodel (IMM). Request for proposal. OMG. 2006. 44 p.

[14] *Kelly S., Tolvanen J.* Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society Press. 2008. 448 p.

[15] OMG Unified Modeling Language, Superstructure. Version 2.3. OMG. 2010. 758 p.

[16] MetaCase MetaEdit+, http://www.metacase.com/

[17] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Version 1.0. OMG. 2008. 240 p.

[18] Meta Object Facility (MOF) Core Specification OMG Available Specification Version 2.0. 2006. 88 p.

[19] Microsoft DSL Tools,
http://msdn.microsoft.com/en-us/library/bb126235.aspx

[20] MOF 2.0/XMI Mapping, Version 2.1.1. OMG. 2007. 120 p.

[21] Object Constraint Language (OCL). Version 2.2. OMG. 238 p.

[22] OMG Systems Modeling Language. Version 1.2. OMG. 2010. 260 p.

[23] Semantics of Business Vocabulary and Business Rules (SBVR). Version 1.0. OMG. 2008. 434 p.

[24] *Steinberg D., Budinsky F., Paternostro M., Merks E.* EMF Eclipse Modeling Framework (2nd Edition). Addison-Wesley. 2009. 744 p.

[25] *Tolvanen J., Kelly S.* Defining Domain-Specific Modeling Languages to Automate Product Derivation: Collected Experiences. LNCS, Vol. 3714, Springer Berlin / Heidelberg. 2005. P. 198–209.

[26] http://www.omg.org/spec/MOFM2T/1.0/

[27] http://www.junit.org/

[28] http://www.openarchitectureware.org/

[29] http://www.eclipse.org/jdt/

[30] http://www.eclipse.org/atl/

[31] http://www.omgmarte.org/