

ИНФОРМАТИКА

УДК 681.3

*А. А. Павлинов, Д. В. Кознов, А. Ф. Перегудов, Д. Ю. Бугайченко, А. С. Казакова, Р. И. Чернятчик, Т. А. Фесенко, А. Н. Иванов*

**КОМПЛЕКС СРЕДСТВ РАЗРАБОТКИ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ВИЗУАЛЬНЫХ ЯЗЫКОВ \***

**Введение.** Фактически уже давно многие компании создают свои собственные решения, основанные на визуальном моделировании и поддерживающие эффективную генерацию кода, сборку целевых систем на основе готовых компонент и т. д. При этом разработка таких решений требует от компаний значительных затрат, но приносит им большие выгоды, поскольку максимально соответствует их задачам.

Однако такие решения не создаются «с нуля» — компании разработчики часто не имеют соответствующих ресурсов, квалификации и пр. В настоящее время активно развиваются так называемые платформы предметно-ориентированного моделирования (Domain Specific Modeling, DSM), которые позволяют легко реализовывать средства графического проектирования, ориентированные на специфику той или иной компании. Эти средства успешно конкурируют с универсальными CASE-пакетами типа IBM Rational Rose, Borland Together Control Center и др. в случае, когда нужно существенно модернизировать UML, встраивать целевые средства работы с визуальным моделированием в определенные среды разработки, бизнес-пакеты и пр. Оказывается, что сделать на их основе собственный графический редактор намного проще, чем настраивать и «доводить» большой универсальный CASE-пакет, а во многих случаях из-за проблем с открытыми программными интерфейсами, лицензионными соглашениями, интеграционными проблемами это оказывается и невозможно.

Однако, несмотря на большой выбор, стандартные DSM-платформы часто нуждаются в доработке, так как не обеспечивают нужной функциональности. Так, пакет Microsoft Visio 2003, обладая привлекательными возможностями по созданию независимых и легких графических редакторов, не поддерживает, например, средств разработки репозитория.

В данной работе приводится краткий обзор самых «зрелых» на сегодняшний день DSM-платформ, делаются выводы о целесообразности использования той или иной в различных условиях. Предлагаются методика разработки DSM-средств на основе платформы Microsoft Visio/.Net, а также комплекс дополнительных средств, созданных нами специально для этой платформы: архитектурный шаблон DSM-пакета, средство создания репозитория. Методика и комплекс средств хорошо зарекомендовали себя для разработки относительно простых инструментов и была неоднократно успешно применена на практике в исследовательских проектах кафедры системного програм-

---

\*) Работа выполнена при финансовой поддержке гранта МинРосНауки, № РИ-19.0/002/273.

©А. А. Павлинов, Д. В. Кознов, А. Ф. Перегудов, Д. Ю. Бугайченко, А. С. Казакова, Р. И. Чернятчик, Т. А. Фесенко, А. Н. Иванов, 2007

мирования СПбГУ. В этой статье будет кратко описан созданный на их основе набор DSM-инструментов, предназначенный для графического проектирования семейства телевещательных систем, разрабатываемых в Санкт-Петербургской компании «ДИП».

## 1. О DSM-подходе.

**1.1. Мотивация.** Визуальное моделирование — это методы, языки и программные средства для использования графовых моделей в проектировании и визуализации программного обеспечения [1]. Фактически, предполагается перенести подходы к проектированию сложных инженерных объектов на основе чертежей, используемых в строительстве, машиностроении, энергетике и т.д. на индустрию производства программного обеспечения. Исходно эта идея появилась в рамках структурного анализа (конец 60-х – 80-е года XX в.), была развита в рамках методов проектирования телекоммуникационных систем (1970-е – 1990-е годы)\*, расширена методологиями объектно-ориентированного анализа и проектирования (1990-е годы), стандартизована и широко распространена в индустрии в рамках инициатив комитета OMG\*\* (с конца 1990-х годов).

Тем не менее до сих пор не предложены универсальные методы визуального моделирования программного обеспечения, что связано с невидимостью программных продуктов [2] и отсутствием естественной и общепринятой метафоры для визуализации ПО [1], а также с тем, что на данный момент вообще не существует универсального процесса разработки ПО [3]. Как правило, любое средство визуального моделирования требует адаптации и настройки для удовлетворения нужд конкретного процесса, особенно, если есть потребность в автоматизации использования визуальных моделей — генерации целевого кода системы, валидации моделей, отладка исполняемых спецификаций в терминах модели и пр. [4]. В частности, язык UML [5] хорошо зарекомендовал себя как средство моделирования, документирования и коммуникаций, которое используется разработчиками на разных стадиях создания ПО. Однако, вопреки постоянному развитию UML, уточнению его исполняемой семантики и метамодели, введению средств расширений языка (профайлов), машинная обработка UML-моделей в общем случае является непростой и дорогостоящей задачей [6]. Анализируя отчеты успешного использования средств поддержки визуального моделирования компании Telelogic AB — одного из мировых лидеров по созданию программных средств в этой области можно обратить внимание, что во многих случаях, когда применялась кодогенерация на основе продуктов этой компании, сами продукты дорабатывались под нужды заказчика [7]. В настоящее время многие компании, например Моторола (Санкт-Петербургское отделение), имеют собственные средства визуального моделирования, основанные на стандартных [8]. В качестве еще одного примера можно привести технологию RTST [9], созданную и используемую для разработки телекоммуникационных систем в компании ЗАО «ЛАНИТ-ТЕРКОМ», реализующую вариант языка SDL.

Все приведенные примеры свидетельствуют о некоторой общей ситуации с инстру-

---

\*) Стандарты европейского комитета ССИТТ, ныне ITU <http://www.itu.int> — языки моделирования SDL (Specification and Description Language) и MSC (Message Sequence Chart).

\*\*) Международный комитет OMG [www.omg.org](http://www.omg.org) с конца 1990-х годов занялся стандартизацией и дальнейшим развитием языков, методов и концепций в области визуального моделирования: универсальный язык моделирования UML (Unified Modeling Language), метод проектирования многоплатформенных систем с помощью UML — MDA (Model-Driven Architecture), визуальный язык спецификации бизнес-процессов BPMN (Business Process Management Notation).

ментами визуального моделирования, которая обобщается в рамках подхода DSM. Суть его – в сужении целевой области применения визуального моделирования (вплоть до отдельных проектов) и достижении за счет этого большей эффективности автоматизированных решений, созданных на его основе [10]. Ключевым аспектом DSM-подхода являются создание и наладка (за приемлемые ресурсы) в компании инструментальных средств, поддерживающих тот вариант использования визуального моделирования, который оптимален для компании. Выработка самого способа бессмысленна вне проектирования и реализации соответствующих инструментов.

При этом можно применять уже имеющиеся на рынке средства типа пакетов IBM Rational Rose, Borland Together Control Center и пр., создавая на их основе технологические решения [4]: все подобные средства имеют открытый программный интерфейс, модульную архитектуру, более того, язык UML, который они реализуют, поддерживает механизм расширения (extension-mechanism), позволяющий создавать на базе UML различные языки-диалекты. Однако, как отмечалось в [4], наладка таких пакетов на практике часто оказывается весьма трудоемким процессом – начиная от «доделки» языка UML (не просто даже разобраться в механизме расширения UML) и заканчивая использованием программного интерфейса стандартных пакетов, который часто плохо документирован, содержит ошибки, наконец, попросту неполон, т. е. не предоставляет всех необходимых возможностей.

В сложившейся ситуации во многих случаях оказывается проще создать собственный небольшой визуальный язык и реализовать свой графический редактор, генератор кода и т. д. (о функциональном составе целевых DSM-средств смотри ниже). Тем более, что на рынке появились для этого специальные средства (мы их называем DSM-платформами). Выбор между настройкой стандартных средств визуального моделирования или разработкой собственных, рекомендации и шаблоны при создании последних, анализ достоинств DSM-платформ и многое другое – все это должно стать содержанием DSM-методологии.

**1.2. Обзор DSM-платформ.** Приведем сравнительный анализ основных на DSM-платформ – Eclipse/GMF, Microsoft DSL Tools, MetaEdit+, Microsoft Visio 2003 (сводная информация об их возможностях представлена в таблице).

Обычно реальная практическая задача по созданию DSM-средств диктует достаточно жесткие условия и ограничения на их средства разработки и среду эксплуатации. Например, если целевой проект (т. е. те программные системы, для создания которых предназначаются данные DSM-средства) разрабатывается на Java, то предпочтительнее использовать средства Eclipse/GMF, если в Microsoft Developer Studio – то Microsoft DSL Tools, если нужно средство графического проектирования вне контекста разработки ПО или «на стыке» (например, легкий графический пакет для инженеров, участвующих в создании программно-аппаратной системы), то здесь удобен пакет Microsoft Visio, позволяющий создавать DSM-средства, минимально «привязанные» к исполняемой платформе (.Net или Java) и, значит, имеющие простую схему инсталляционного процесса, низкие требования по вычислительным ресурсам. В этой ситуации можно воспользоваться также MetaEdit+.

Нужно отметить, что технологии Eclipse/GMF и Microsoft DSL Tools очень сильно связаны с платформами разработки – Eclipse и Microsoft Visual Studio соответственно, особенно технология Microsoft DSL Tools, которая требует для работы не только многочисленных библиотек периода исполнения, подобно Eclipse/GMF, но и среды

разработки (Microsoft Visual Studio), а также является составной частью концепции разработки ПО под названием Software Factory [11]. Обе технологии – Eclipse/GMF и Microsoft DSL Tools - предоставляют богатые возможности по доводке программного кода DSM-средств «в ручную», с сохранением результатов при повторной генерации. Характеризуя рассмотренные выше DSM-платформы, можно выделить следующие случаи их использования, помимо платформенных предпочтений. Если требуется разработать многофункциональные средства визуального моделирования, то лучше применять Eclipse/GMF. Например, с помощью этой технологии создаются отдельные части известного пакета Borland Together, который, конечно, не попадает в разряд DSM-средств, являясь мощным промышленным пакетом визуального моделирования (точнее, целым семейством таких продуктов). Но часто DSM-средства оказываются весьма трудоемки по разработке, так как требуют богатой функциональности и могут заказываться большими компаниями типа Siemens, Motorola и т. д. для внутреннего использования.

Если создается целое семейство визуальных языков и средств их поддержки, то лучше использовать технологию MetaEdit+. Кроме того, она является наиболее зрелой в плане создания DSM-средств без программирования.

Если же нужен «легкий» графический редактор, имеющий специфическую нотацию и несложные правила поведения фигур, то лучше применять пакет Microsoft Visio. В этом случае часто оказывается, что можно обойтись только средствами Microsoft Visio и избежать использования библиотеки Visio SDK, а также программирования в среде .Net. Например, таким образом был создан редактор MSC-диаграмм достаточно полно реализующий эту графическую нотацию \*

Наконец, скажем несколько слов о степени готовности и перспективности DSM-платформ. Сейчас наиболее зрелыми являются платформы Eclipse GMF и MetaEdit+, которые активно развиваются. Платформа Microsoft DSL Tools еще совсем молодая и годится лишь для экспериментов. Пакет Microsoft Visio фактически законсервирован как DSM-платформа, и поддерживается только как графический пакет. В таблице приведены основные свойства описанных выше DSM-платформ.

**2 Описание комплекса.** То, что был выбран пакет Microsoft Visio в качестве DSM-платформы, несмотря на отсутствие в нем таких важных компонент как автоматические средства генерации контроллеров, репозитория, можно объяснить следующими причинами:

1. Этот пакет широко распространен в академической и научной среде, так что решения на его основе легче понимаются и принимаются.

2. Нам нужны «легкие» DSM-пакеты, не при применении приобретения и установки таких продуктов как Microsoft Visual Studio или базовых библиотек технологии Eclipse. Правда, используя .Net платформу для создания дополнительных средств поддержки разработки DSL-пакетов, будущим пользователям целевых DSM-средств, созданных на этой платформе, необходимо иметь среду исполнения .Net Framework. Ее установочный пакет компактен и его можно получить бесплатно по Интернету с сайта компании Microsoft. Кроме того, данная среда исполнения будет входить в следующее поколение операционной системы Windows – Windows Vista.

---

\*)MSC (Message Sequence Chard) – стандарт комитета ITU для визуального моделирования сценариев работы телекоммуникационных систем. Редактор MSC-диаграмм был создан А. Н. Замышляевым в его дипломной работе «Визуализация недетерминированных сценариев ПО при возвратном проектировании»(СПбГУ, 2005. 34 с.)

## Сравнение DSM-платформ.

DSM-платформа	Eclipse GMF	Microsoft DSL Tools	MetaEdit+	MS Visio	
Разработка целевого DSM-пакета	Средства описания DSL-метамодели	EMF-модель, обычно сериализованный XML, возможен импорт аннотированных Java интерфейсов, XML-Schema, модели классов Eclipse UML2	Диаграмма классов специального вида, создаваемая с помощью редактора VisualStudio 2005	Создание GOPPRR-описания DSL-метамодели с помощью встроенного набора средств. Грань между процессом описания DSL-метамодели и соответствующего редактора прослеживается не четко	Нет
	Средства описания графического DSL-редактора	Описание графического редактора в специальном формате. Описание связей элементов метамодели с графическими объектами. Описание ограничений и методов валидации на языках OCL и Java	XML-описание графического редактора, генерируемое по диаграмме классов		Средства создания графических объектов, использование встроенного языка для описания ограничений и модели поведения графических объектов
	Средства описания дополнительной функциональности и DSM-пакета	Описание вспомогательных средств (палитры объектов, список действий, меню графических объектов)	Средства описания поведения навигатора модели; средства описания палитры визуальных объектов	Средства описания палитры визуальных объектов, средства проектирования диалоговых форм для ввода и редактирования свойств модельных объектов, средства описания декомпозиции и разбиения	Нет
	Средства автоматизации разработки целевого DSM-пакета	Автоматическая генерация графического редактора средствами EMF; автоматическая генерация редактора диаграмм и палитры средствами GMF	Автоматическая генерация реализационных классов метамодели языка, автоматическая генерация XML-описания графического редактора, «ручная» реализация процедур валидации моделей на основе предлагаемой архитектуры	Автоматическая генерация метеописания языка и графического редактора для работы с ним. Эти описания хранятся в репозитории. Процесс создания DSM-пакета производится "на лету". MetaEdit+ динамически настраивает среду на поддержку конкретного DSL, описание которого выбрано из репозитория	Нет
Характеристики целевого DSM-пакета	Среда целевого DSM-пакета	Eclipse IDE	Microsoft Visual Studio 2005	MetaEdit+	MS Visio
	Графический DSL-редактор.	Диаграммный редактор основывается на GMF Runtime	Встроенный редактор Microsoft Visual Studio 2005	Встроенный редактор MetaEdit+	MS Visio
	Готовые решения для генерации артефактов по моделям.	Нет	Механизм генерации артефактов по шаблонам на ASP-подобном языке	Механизм генерации артефактов по шаблонам на собственном языке Report Definition Language	Создание отчетов в форматах Excel, HTML или XML по свойствам фигур
	Репозиторий	Нет	Нет	Многопользовательский репозиторий среды MetaEdit+	Нет
	Отчуждаемость целевого DSM-пакета	Нет	Нет	Нет	Нет

3. Создаваемые на основе Microsoft Visio DSM-пакеты обладают почти полной функциональностью исходного пакета Visio (главное меню, многостраничные диаграммы, возможности работы с базовыми свойствами геометрических фигур, многостраничная печать и т. д.), что очень удобно.

4. Возможность быстро разрабатывать демонстрационные версии и обсуждать их с заказчиком.

5. Возможность создавать (при необходимости) средствами базового Visio графические элементы, которых недостает в разработанном графическом редакторе, естественно, с потерей сервисов автоматической обработки, но зато с сохранением и восстановлением при открытии/закрытии диаграмм, т.е. с так называемой диаграммной информацией, не попадающей в модель. Но она часто необходима при практическом использовании DSM-пакета, так как оказывается, что какие-то свойства не успели реализовать и можно поступиться свойствами автоматической обработки.

6. Наличием «задела» для недостающих компонент данной DSM-платформы, созданной в рамках проекта REAL [12].

**2.1 Общее описание методик.** Ниже будет предложен подход к реализации DSM-пакетов на базе пакета Microsoft Visio и платформы разработки .Net. В пакет Microsoft Visio встраивается графический редактор нового DSL, автоматически генерируется репозиторий нового редактора по метаописанию DSL. Вся разработка DSM-пакета происходит в рамках специального архитектурного шаблона, основанного на MVC-шаблоне. Реализация уровня контроллеров производится «вручную» на основе набора базовых классов и архитектуры, общих для данного подхода.

Процесс создания DSM-пакета имеет циклический вид и включает в себя следующие этапы (рис. 1):

1. Проектирование метамодели создаваемого DSL в виде диаграммы классов. Создание диаграммы классов осуществляется в специализированном редакторе классов на базе пакета Microsoft Visio, созданном целиком «вручную», но следующем тем же архитектурным принципам, что предлагаются в подходе.

2. Использование генератора для создания исходного кода реализации объектно-ориентированного репозитория. Генерация осуществляется по модели классов, описанной выше.

3. Разработка визуальной составляющей (создание графических объектов, реализация простейших ограничений на поведение графических объектов) с помощью средств пакета Microsoft Visio.

4. Разработка уровня контроллеров для связи сгенерированного кода и графических элементов пакета Microsoft Visio.

5. Компиляция и отладка.

Рассмотрим подробнее архитектуру и средства предлагаемого подхода.

**2.2 Архитектурный шаблон.** Основная задача, которую решает данный шаблон, — связывание графических объектов, созданных в среде Microsoft Visio, и модельных объектов, хранящихся в репозитории. Для решения этой задачи была выбрана классическая трехуровневая модель MVC. Общая схема шаблона представлена на рис. 2.

Роль модели (Model) играют объекты репозитория, которые являются теми объектами, которые визуализирует графический редактор данного DSL. Эти объекты разбиты на классы, которые наследуют от абстрактного класса ModelEntity. Репозиторий

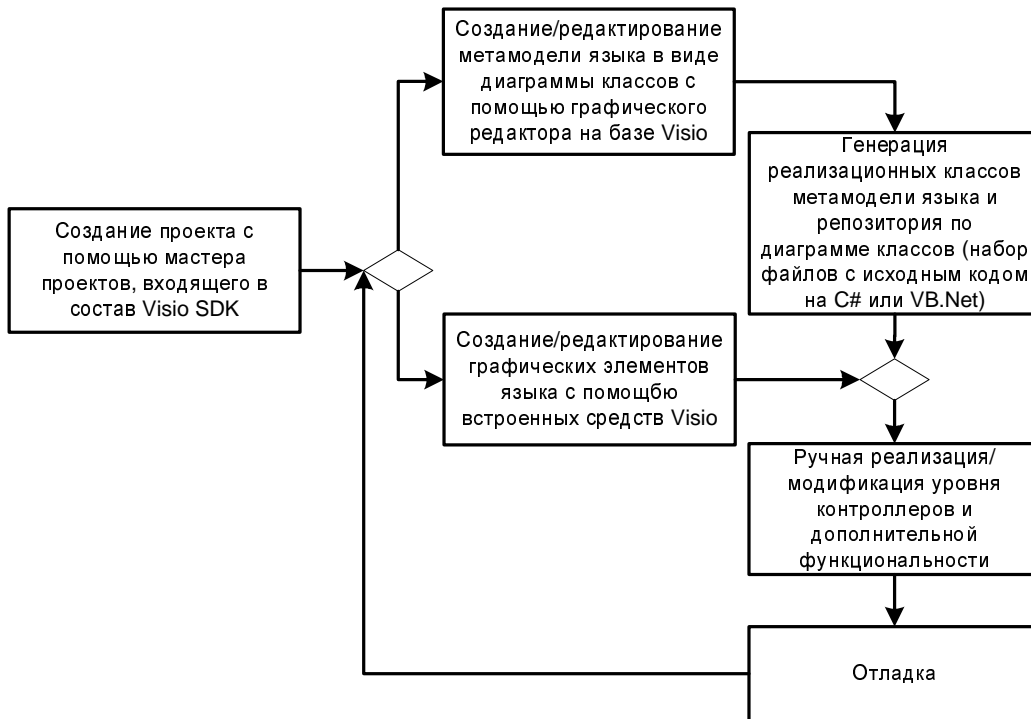


Рис. 1: Предложенный процесс разработки DSM-пакета.

представляет собой обособленную компоненту, которая отвечает за создание модельных объектов, их хранение, программный доступ к ним. Репозиторий также осуществляет процедуру сериализации графа объектов. Ему недоступна информация о других частях системы, в частности, о представлении его объектов на диаграммах. Он может быть использован вспомогательными инструментальными средствами, такими как генераторы кода, документации или компонентами проверки корректности модели, отдельно от редакторов диаграмм.

Представлением (View) являются графические объекты пакета Microsoft Visio. Графические объекты также не имеют информации о других частях DSM-пакета, однако содержат идентификаторы модельных классов, сущности которых они отображают на диаграммах, а также уникальные идентификаторы связанных модельных объектов, хранящихся в репозитории. Такая информация необходима для динамического создания репозиторных сущностей в момент размещения нового графического элемента на диаграмме, а также для воссоздания связей между графическими и модельными объектами после загрузки ранее сохраненных диаграмм и репозитория проекта.

Роль контроллеров (Controller) в системе играют потомки абстрактного базового класса Controller. Экземпляр контроллера создается для каждого графического объекта на диаграмме. Контроллер является связующим звеном между репозиторным объектом и его графическим представлением на диаграмме. В задачи контроллера входят подписка и реакция на события, как происходящие с модельной сущностью, так соответствующего графического объекта. В зависимости от количества различных типов

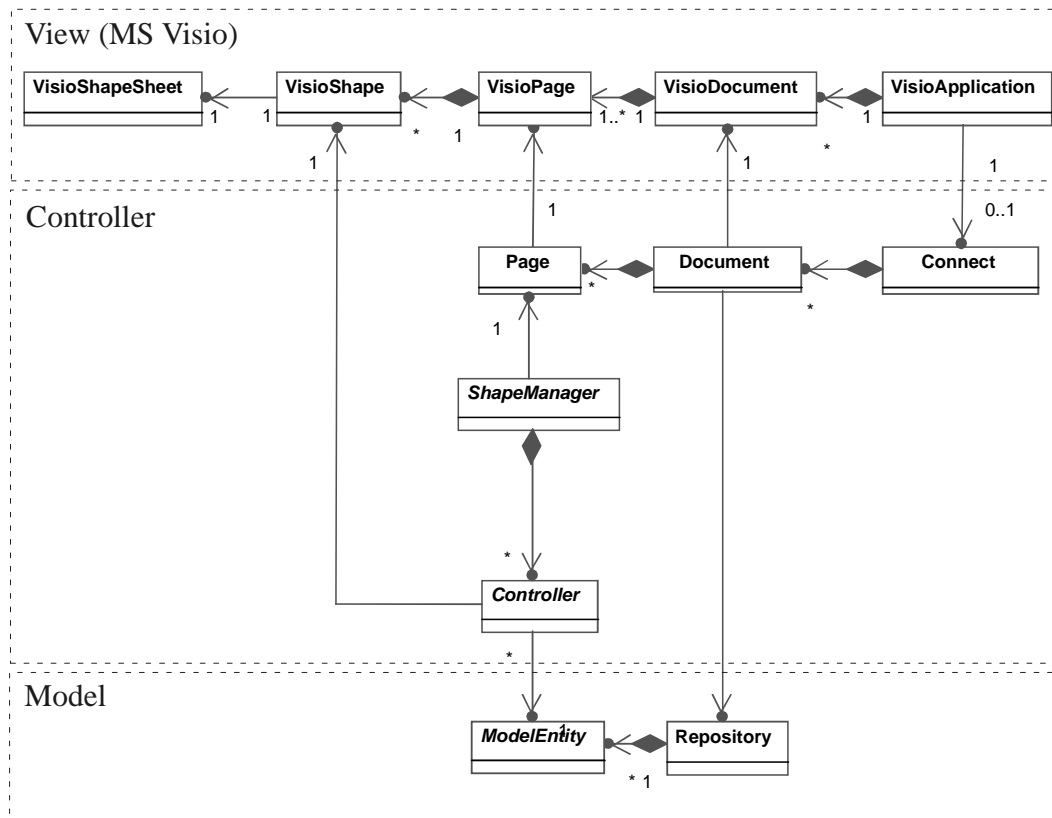


Рис. 2: MVC-шаблон архитектуры DSM-пакета.

диаграмм и возможности отображения различных модельных объектов на них существует возможность того, что одному классу метамодели соответствует несколько классов контроллеров, так как контроллер является тем элементом, который содержит в себе информацию о контексте использования модельной сущности. Фактически, контроллер осуществляет контроль за соблюдением правил, определяемых синтаксисом DSL, при создании и модификации диаграмм пользователем.

Этот шаблон использовался нами во многих проектах, по реализации различных диаграммных средств на основе Microsoft Visio и зарекомендовал себя как хороший архитектурный базис. То есть он не изменялся в разных проектах, а, наоборот, структурировал разработку, позволяя «наращивать мясо» на предлагаемую схему.

**2.3 Генератор репозиториев.** В дополнение к предложенному выше архитектурному шаблону был разработан генератор репозиториев, который автоматически создает исходный код репозитория по спецификации метамодели DSL. Здесь использовались рекомендации стандарта ODMG [13], а также идеи, реализованные в рамках проекта REAL [14]. Для создания метамодели реализован специализированный редактор классов на базе пакета Microsoft Visio.

Для генерации исходного кода была использована технология CodeDOM (Code Document

Object Model), входящая в состав .Net Framework 1.1 [15]. Она позволяет разработчикам генерировать исходный код программ на различных языках программирования платформы .Net по единой объектной модели кода \*.

Генератор кода репозитория способен обрабатывать следующие элементы диаграммы классов: определения классов, определения свойств, отношения наследования, бинарные ассоциации, различные типы композитного агрегирования.

В результате генерации порождается исходный код классов, типизированных коллекций и делегатов. Далее опишем функциональность, которую поддерживает сгенерированный репозиторий.

**2.3.1. Сериализация графа объектов.** Сериализация (serialization) — это сохранение/восстановление долгоживущих объектов относительно некоторого внешнего по отношению к программному приложению потока (чаще всего им является файл). Репозиторий DSL, который генерируется в рамках нашего метода, содержит необходимые конструкторы и методы для реализации расширенного механизма сериализации графа графических объектов, основывающегося на общем механизме платформы Microsoft .Net. Такая реализация позволяет управлять порядком сохранения и восстановления графа объектов, дает возможность осуществлять выборочную сериализацию, а также избегать ошибок времени исполнения при различиях в версиях сохраненных объектов и измененных классах метамодели репозитория.

**2.3.2. Унифицированная система событий.** Сгенерированный разработанными средствами для какого-либо DSL репозиторий имеет унифицированный набор событий, покрывающий различные случаи использования репозитория (например, для корректного отображения репозиторных объектов в браузере проекта). Унифицированность заключается не только в схожести сигнатуры делегатов, имен событий, но и в определенности времени порождения событий в системе.

**2.3.3. Поддержка ссылочной целостности.** Сгенерированный репозиторий берет на себя ответственность за поддержание ссылочной целостности отношений, описанных в спецификации с помощью отношений ассоциаций и композиции. Данный механизм был предложен в стандарте объектно-ориентированных баз данных ODMG [13]. Рассмотрим его на простом примере.

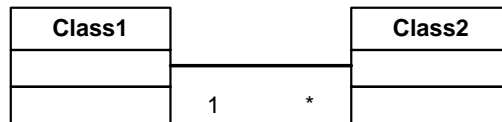


Рис. 3: Пример части спецификации репозитория.

На рис. 3 представлена часть спецификации репозитория – классы ClassA, ClassB, связанные ассоциацией один ко многим.

Для того, чтобы создать отношение между двумя объектами этих классов, прикладному приложению нужно лишь присвоить одному из них ссылку на другой. После

---

\*) Примечателен также тот факт, что благодаря использованию технологии CodeDOM в результате генерации может быть получен код на различных языках платформы .Net, для которых существует реализация абстрактного класса CodeDomProvider из пространства имен System.CodeDom.Compiler. В состав .Net Framework version 1.1 входят реализации данного класса для языков C# и VB.Net.

этого репозиторий автоматически присвоит обратную ссылку. Аналогичный механизм работает и для отношений с концами, чья мощность отлична от единицы.

**2.3.4. Каскадное удаление объектов.** Данный механизм во многом похож на поддержку ссылочной целостности, но существенно шире по функциональности. Речь идет о корректных удалениях репозиторных объектов. В частности, в случае наличия композитной агрегации перед удалением репозиторного объекта, не агрегирующего другие модельные объекты, будут корректно разрушены отношения, связывающие удаляемый и связанные с ним объекты. В том случае, если удаляемый объект агрегирует другие репозиторные объекты, сначала будут удалены агрегируемые объекты.

**3. Опыт использования предложенных средств.** Описанный выше комплекс средств был применен при создании DSM-пакета, предназначенного для автоматизации проектирования аппаратуры семейства систем телевидения (ТВ). На основе пакета Microsoft Visio были разработаны графический редактор принципиальных схем для таких систем, реализован генератор Excel-отчетов по диаграммам, а также генератор загрузочной конфигурации ПО целевой системы. Репозиторием редактора активно пользовался не только он сам, но также и другая программная компонента – модуль описания и учета аппаратуры семейства ТВ систем. Этот проект подробно описан в работе [16].

Комплекс представленных в данной статье средств позволил эффективно использовать пакет Microsoft Visio как основу DSM-средства, существенно уменьшив трудоемкость разработки. Отметим некоторые ограничения, с которыми мы при этом столкнулись:

- невозможность применять стандартный для Microsoft Visual Studio путь создания инсталляционных пакетов;
- слабая объектная ориентация модели графических фигур, в силу чего ее почти невозможно надстраивать и расширять, а также затруднительно активно использовать через открытый API (мало операций над данными);
- трудности с заданием сложных фигур: сложные фигуры создаются только через группы, у которых оказываются проблемы с производительностью при перерисовке, а также с автоматическим выравниваем в соответствии со специфическими требованиями графических нотаций;
- трудности с созданием и изменением диаграмм из кода вне Visio, через API;
- трудности с заданием корректного поведения вложенных фигур — например, сложное состояние в диаграммах состояний и переходах UML, включающее в себя несколько простых, к которым из вне, через границу сложного состояния, идут линии-переходы.

**4. Заключение.** Предложенный в работе комплекс средств хорошо зарекомендовал себя на практике. С учетом выявленных ограничений по области его применения (небольшие, легко отчуждаемые DSM-решения, использование в образовании) перспективным становится его доработка в направлении улучшения средств разработки репозитория (поддержка транзакций, большего разнообразия связей между метаклассами и пр.). Также перспективна разработка других архитектурных шаблонов, например с «облегченными» репозиториями или совсем без них.

## Список литературы

1. Кознов Д. В. Языки визуального моделирования: проектирование и визуализация программного обеспечения. Учебное пособие. СПб.: Изд-во С.-Петерб. ун.-та, 2004, 143 с.
2. Brooks F. No Silver Bullet // Information Proceeding of the IFIP 10th World Computing Conference. 1986. P. 1069–1076.
3. Sommerville I. Software Engineering. 6th ed. Boston, USA: Addison-Wesley, 2001, 693 p.
4. Кознов Д.В. Визуальное моделирование компонентного ПО. Канд. дис. СПб.: 2000, 82 с.
5. OMG. UML 2.0 Infrastructure Specification, September, 2004 // <http://www.omg.org/>
6. France R. B., Ghosh S., Dinh-Trong S. et al. Model-driven development using UML 2.0: promises and pitfalls. //Computer, 2006 Vol. 39, issue 2, p. 59–66.
7. Примеры успешного внедрения продуктов Telelogic // <http://www.telelogic.com/customers/success-stories.cfm>
8. Дробинцев П. Д. Интегрированная технология обеспечения качества программных продуктов с помощью верификации и тестирования. Канд. дис. СПб.: 2006. 238 с.
9. Парфенов В. В. Проектирование и реализация программного обеспечения встроенных систем с использованием объектно-базируемого подхода. Канд. дис. СПб.: 1995. 113 с.
10. Grundy, J. C., Hosting, J. G., Amor R.W., et al. Domain-Specific Visual Languages for Specifying and Generating Data Mapping Systems. Journal of Visual Languages and Computing. 2004. Vol. 15. P. 207–209.
11. Greenfield J., Short K. Cook S. et al. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Indianapolis, USA: Wiley Publishing, 2004. 666 p.
12. Терехов А. Н, Романовский К. Ю., Кознов Д. В. и др. Real: методология и CASE-средство для разработки систем реального времени и информационных систем // Программирование. 1999 №.5. С. 44–51.
13. Cattell R. G., Barry D., Bartels D. et. al. The Object Database Standard: ODMG 2.0. San Mateo, USA: Morgan Kaufmann Publ. 1997. 270 p.
14. Кондратьев А. М. CASE-средство и объектные базы данных // Объектно-ориентированное визуальное моделирование / Под ред. Терехова А. Н. СПб.: Изд-во С.-Петерб. ун.-та, 1999. С. 57–78.
15. .Net Framework version 1.1. Documentation // <http://msdn.microsoft.com/>.

16. Кознов Д. В, Перегудов А. Ф., Бугайченко Д. Ю. и др. Опыт создания визуальных средств проектирования для систем телевизионного вещания // «Системное программирование» Под ред. А. Н. Терехов, Д. Ю. Булычев, СПб.: Изд-во С.-Петерб. ун.-та, 2006, Вып. 2, С 54–62.

Статья принята к печати 20 ноября 2006 г.

## Summary

*Pavlinov A. A., Koznov D. V., Peregodov. A. F., Bugaychenko D. Y., Kazakova A. S., Chernyatchik R. I., Fesenko T. A., Ivanov A. N.* An Approach for Development of Domain-Specific Visual Languages.

Domain specific modeling (DSM) is a software engineering approach that supposes to develop domain-oriented visual languages, methods and tools. Some approach to support DSM based on using Microsoft Visio 2003/.Net is presented is presented. This DSM-platform is chosen because of independence, small size and easiness of distribution and dissemination of target DSM-tools. Our approach provides repository development (modeling and generation) and suggests some architectural pattern for MVC-based development of graphical tools using Microsoft Visio 2003. We also describe some restrictions of our approach and Microsoft Visio as DSM-platform. Finally, we present a toolkit that has been developed on the base of our DSM-approach for visual modeling of software/hardware broadcasting product line in Saint-Petersburg software company «DIP».

Предметно-ориентированное моделирование (Domain Specific Modeling, DSM) является подходом к разработке программного обеспечения с помощью создания специфических визуальных языков, методов и программных средств, нацеленных на задачи определенной предметной области. Представляется комплекс средств, реализующий DSM-подход на базе Microsoft Visio 2003/.Net, включающий в себя средство создания репозитория, а также архитектурный MVC-шаблон для упрощения разработки графических редакторов на базе продукта Microsoft Visio. Приведены результаты апробации комплекса средств для создания средств проектирования в рамках семейства телеведущих систем Санкт-Петербургской компании «ДИП». *Библиогр. 15 назв. Ил. 3. Табл. 1.*