УДК 004.942

В.И. ПАВЛОВСКИЙ, А.Л. ЗИНЧЕНКО

Черниговский государственный технологический университет

ПРОБЛЕМЫ ОБЪЕКТНОГО ПОДХОДА К МОДЕЛИРОВАНИЮ И ПРОТОТИПИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Рассмотрены проблемы объектного подхода к моделированию информационных систем. Проведен обзор современных средств прототипирования программного обеспечения. Показана их ограниченность в плане моделирования широкого спектра семантических отношений предметной области. Определен ряд требований, которым должна удовлетворять современная система моделирования и прототипирования. Показана возможность расширения современных систем, таких как Taylor, средствами, обеспечивающими приведенные требования.

Ключевые слова: *прототипирование, каркас, моделирование, модель, слабая типизация, иерархические данные, отношения, семантические связи.*

Введение

Создание практически любой современной сложной информационно-компьютерной системы (ИС) начинается с этапа моделирования и проектирования. На сегодня одним из наиболее популярных подходов к моделированию является объектный подход.

В соответствии с этим подходом в результате объектно-ориентированного анализа и объектно-ориентированного дизайна можно получить "хороший" проект информационной системы, прозрачный, удовлетворяющий требованиям заказчика, удобный для коллективной разработки, тестирования и отладки, развиваемый и допускающий повторное использование компонентов.

Основой такого проекта является модель предметной области (ПО) максимально приближенной к реальности.

Знание ПО проектировщиком существенно сказывается на результате моделирования. Чем более точно разработчик представляет ПО, среду познания, взаимосвязи объектов и процессов, законы и правила поведения объектов, тем более точно модель будет отражать действительность. Чаще всего ПО лучше представляет заказчик ИС, который не является одновременно проектировщиком, чем сам проектировщик. При передаче знаний о ПО от заказчика к проектировщику, как правило, происходит естественная потеря информации, связанная со многими субъективными факторами – различное мироощущение, разный опыт и т.п.

Каким бы опытным не был проектировщик, сколько бы он не знал видов моделирования, языков программирования, средств разработки и прочее, – все равно модель не будет "идеальной", не будет учитывать все особенности, нюансы, описывать все необходимые качества, свойства, характеристики реальных моделируемых объектов и процессов. Ему в любом случае требуется апробация заказчика. В этом случае очень хороши для применения методы постепенного и поэтапного описания модели, по мере накопления знаний, информации о сущностях и процессах. При этом очень важно как можно раньше начать эксплуатировать модель.

Одним из методов обеспечения этого является прототипирование.

Таким образом, в процессе разработки ИС возникают проблемы:

– поэтапного создания модели ПО, максимально приближенной к реальности;

– прототипирования ИС.

1. Прототипирование программного обеспечения

Прототипирование программного обеспечения проводят с целью проверки пригодности предлагаемых для применения концепций, архитектурных и технологических решений, а также для представления программы заказчику на ранних стадиях процесса разработки. На этапе прототипирования:

1) может быть обнаружено отсутствие важных пользовательских функций;

2) могут быть найдены и переопределены неточности или недопонимание в описании пользовательских функций;

3) могут быть обнаружены неполнота и/или противоречивость требований к системе;

4) могут быть обнаружены важные архитектурные просчеты и ошибки;

5) прототип служит основой для написания спецификации системных требований.

Прототипирование имеет множество вариантов, но выделяют два основных типа — быстрое прототипирование и эволюционное прототипирование.

Быстрое прототипирование (БП) предполагает создание макета, который не будет являться частью разрабатываемой системы и применяется в основном для систем с богатым пользовательским интерфейсом [1]. БП не обязательно проводится на платформе и с использованием технологий разрабатываемой системы.

Эволюционное прототипирование (ЭП) предполагает постепенное приближение макета системы к конечному продукту, на каждом витке разработки происходит уточнение модели ПО, детализация процессов, законов и отношений между сущностями [1].

ЭП позволяет определяться с требованиями к ИС, вносить поправки в интерфейсы модулей системы и перераспределять функциональность между модулями системы на каждом этапе по мере создания прототипа.

Для ускорения процесса разработки макета при ЭП широко используют методику повторного использования модулей, использование шаблонов, а также средства генерации каркасов приложений и пользовательских интерфейсов.

2. Проблемы объектного подхода к моделированию ПО

Объектный подход (ОП) к моделированию ПО является наиболее популярным, но в тоже время не является универсальным средством моделирования ПО.

С одной стороны, при разработке ИС для слабо регламентированных ПО, например для научных приложений, сложная природа системных исследований предъявляет серьезные требования к инфраструктуре программного обеспечения, необходимого для поддержки исследовательского процесса. В виду специфической области применения, возникают трудности проектирования и моделирования ПО таких систем средствами ОП.

Проблемными для моделирования с использованием ОП являются задачи: моделирование фазовых переходов, разделение и соединение объектов, сохранение истории изменений объектов, моделирование слабой и контекстно-зависимой типизации [2, 3].

С другой стороны, отношения между объектами реальных систем богаче, чем набор отношений, которые можно описать, используя распространенные средства моделирования. К сожалению, на сегодня не существует ни одной формальной системы описания данных, которая позволяет полностью описать реальную ПО на уровне схемы данных.

В результате, как правило, на этапе перехода от описания ПО к соответствующей модели данных происходит потеря ассоциативных связей между сущностями [3].

При разработке сложных ИС, для наиболее полного представления о ней, проектировщик должен иметь возможность рассмотреть ее с нескольких точек зрения – модульности, реализации бизнес и других процессов, объектов и отношений между ними и т.п. В процессе анализа системы создаются различные модели – модель ПО, модель бизнес-процессов и т.п. Для этой цели широко используются средства визуального моделирования.

3. Обзор средств визуального моделирования и прототипирования программного обеспечения

Современные средства визуального моделирования объединяют все аспекты проектирования и разработки программного обеспечения на основе моделей UML. Эти продукты предназначены для анализа, проектирования, сопровождения и развития корпоративных ИС и включают все возможности создания J2EE и веб-приложений, а так же веб-служб. Рассмотрим наиболее популярные средства разработки.

Enterprise Architect (EA) — CASE-инструмент для проектирования и конструирования программного обеспечения. EA является мощным инструментом, охватывающим все аспекты цикла разработки, обеспечивая возможность в полной мере отслеживать проект от начального этапа проектирования до развертывания и технического обслуживания.

Также в нем существует поддержка тестирования и контроля внесенных изменений. EA поддерживает все модели/диаграммы UML 2.0. С его помощью можно моделировать бизнес-процессы, веб-сайты, пользовательские интерфейсы, сети, конфигурации аппаратного обеспечения, сообщения, фиксировать и трассировать требования, ресурсы, тест-планы, дефекты и запросы на изменения. EA поставляется с рядом шаблонов моделирования, которые помогают создавать новые проекты и модели.

IBM Rational Software Architect (RSA) — CASE-инструмент для анализа, проектирования и развития корпоративных ИС. RSA построен на базе открытой и расширяемой платформы Eclipse, кото-

рая использует ряд открытых стандартов. RSA предоставляет большой выбор типов диаграмм, предоставляет возможности визуализации структуры и поведения приложений. RSA содержит функции структурного анализа и контроля, которые прямо направлены на поиск проблем в коде Java.

Шаблоны проектирования, реализованные в коде, можно визуализировать, вследствие чего возникают возможности для повторного использования компонентов. RSA также автоматически обнаруживает структурные «антишаблоны» − нежелательные, но распространенные проблемы программного обеспечения, осложняющие его поддержку и обновление.

EclipseUML — средство моделирования на базе IDE Eclipse. Архитектура EclipseUML схожа с архитектурами традиционных UML решений. EclipseUML поддерживает все виды диаграмм UML 2.0. В него встроен XMI редактор, который позволяет просматривать элементы модели в формате XMI.

Особенностью данного средства моделирования является то, что генерация программного кода классов сущностей происходит сразу после добавления соответствующих графических примитивов на диаграмму, и изменения в диаграмме автоматически отображаются в коде.

В зависимости от типа семантической связи, которая устанавливается между классами, происходит генерация соответствующих методов в исходном коде классов. Это говорит о том, что осуществляется семантическая обработка данной связи.

Все рассмотренные выше средства визуального моделирования поддерживают следующие типы семантических связей: ассоциации, агрегации, композиции, использования и наследования. Для этих связей они предоставляют возможность генерации программного кода, реализующего эти отношения. Также эти средства позволяют описывать новые типы отношений между сущностями, но не предоставляют возможности наделить эти связи соответствующей семантикой с последующей обработкой.

Далее рассмотрим средства, позволяющие быстро разрабатывать приложения за счет использования подхода разработки основанного на соглашениях (convention-based).

Spring Roo — средство быстрой разработки корпоративных приложений на Java. Обеспечивает 100% совместимость с JPA, автоматическую генерацию архитектуры системы с использованием лучших практик Spring Framework 3, прозрачную поддержку внедрения зависимостей (dependency injection) и методов сохранения для всех сущностей, включая полученные через JPA, динамическое создание finder-методов на JPA QL для сущностей безо всякого кодирования. Roo обеспечивает поддержку полного цикла жизни приложения с сохранением высокой продуктивности, обеспечивает экстремально эффективную производительность во время выполнения, безопасность типов и отсутствие зависимостей на Roo во время выполнения.

Grails — высоко-продуктивный программный каркас для создания J2EE-приложений. Мощная и прозрачная для пользователя поддержка персистенции (GORM), использование шаблонов страниц, динамическая библиотека тегов, хорошая поддержка Ajax. В Grails используется основанный на Java скриптовый язык Groovy и встроенный сервлет-контейнер.

Taylor — программный каркас с открытым кодом на базе IDE Eclipse для быстрой разработки корпоративных приложений. Taylor использует подход основанный на соглашениях (convention-based) и стереотипы с аннотациями для генерации кода по UML модели. Taylor отвечает концепции MDA. Разработка на Taylor сфокусирована на итеративном моделировании. Модель – это первичная форма документации. С каждым циклом модель эволюционирует, постепенно превращаясь в спецификацию разрабатываемого приложения. В каждом цикле модель генерируется в рабочее программное обеспечение, которое используется для пользовательского тестирования. Кроме генерации кода программного продукта, также возможна генерация документации. В модели Taylor широко распространена практика тестов – для тестирования модели. Сам процесс разработки максимально унифицирован.

4. Система визуального моделирования и прототипирования сложных ИС

Все рассмотренные системы моделирования, характеризуясь большим количеством преимуществ, не предоставляют разработчику той функциональности, которая позволила бы выполнять расширенное описание ПО – воспроизводить все виды отношений между сущностями и ограничения, которые на них накладывает семантика ПО.

Одним из существенных недостатков таких систем является то, что для описания отношений они предоставляют возможность использовать лишь базовые типы семантических связей. Очень часто их не хватает для выполнения расширенного описания семантики моделируемой ПО, поскольку в реальной жизни отношения между объектами намного разнообразнее. Как результат, в процессе

генерирования по построенной модели программного кода средствами таких систем, происходит потеря большого объема информации о ПО.

Для решения указанной задачи на инструменты моделирования данных необходимо взглянуть с точки зрения полноты описания типов отношений между данными. Тип отношения – это данные (метаданные, метаинформация) о семантических связях между данными.

Рассматривая тип отношения как специфические данные, можно достичь расширяемости множества отношений [3].

Таким образом, задача разработки средства моделирования и прототипирования сложных ИС является актуальной. Программное средство моделирования и прототипирования сложных ИС должно обеспечивать:

1) базироваться на уже существующих технологиях UML (Unified Modeling Language), MOF (Meta Object Facility) и XMI (XML Meta-data Interchange);

2) возможность описания и визуального представления семантических связей между сущностями ПО;

3) возможность расширения типов семантических связей между сущностями, т.е. возможность описания новых типов семантических связей, с помощью которых можно будет моделировать различные ПО;

4) средства контроля полноты и непротиворечивости семантических связей между сущностями ИС и сущностями моделируемой ПО;

5) возможность расширения средств валидации модели ПО;

6) возможность проверки семантической релевантности запросов к ХД на этапе проектирования программы;

7) возможность генерации метафайла (XMI) на основе модели, который должен содержать:

8) описание классов сущностей моделируемой ПО, их атрибутов и методов;

9) описание ограничений, накладываемых на классы сущностей и их методы;

10) описание моделей семантически-релевантных ПО SQL-запросов;

11) описание семантических связей между классами моделируемой ПО.

12) возможность на основе метафайла генерировать поведенческие классы, классы семантических связей, код обработки ограничений разрабатываемой ИС.

Из рассмотренных выше средств визуального моделирования и прототипирования можно выделить систему Taylor, которая объединяет функции моделирования и прототипирования, а ее архитектура предусматривает расширение функционала как модуля моделирования (внесение новых семантических отношений между сущностями ПО) так и модуля генерации, использующего технологию JET (Java Emitter Templates).

Расширение производится путем разработки дополнений реализующих Taylor Plugin API. На рис. 1 приведена архитектура системы моделирования и прототипирования, базирующаяся на системе Taylor и удовлетворяющая перечисленным выше требованиям. Система состоит из двух подсистем. Системы визуального моделирования и системы генерации кода.

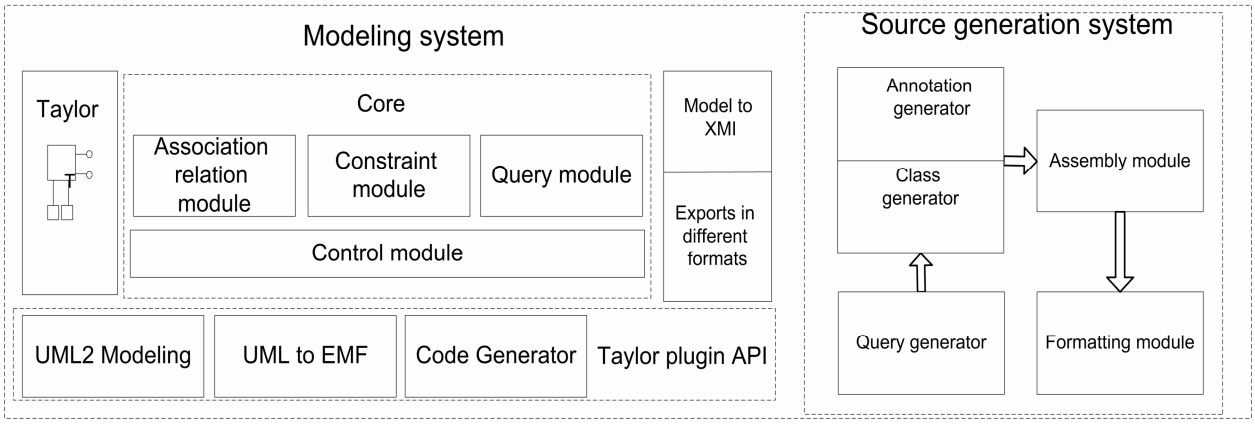

Рис. 1. Архитектура программного средства моделирования

Система визуального моделирования использует возможности визуальных редакторов, предоставляемых системой Taylor и базирующихся на технологии GMF (Eclipse Graphical Modeling Framework).

С помощью этих редакторов и помощников разработчик может использовать новые ассоциативные (семантические) отношения (например, отношение *same* [3, 4]), новые стереотипы для классов и атрибутов (например, *observed*, *history*

[4]) для создания модели ПО, максимально приближенной к реальности. Система генерации кода использует технологию JET для генерации кода аннотированных классов, поведенческих классов и вспомогательных классов.

Заключение

В статье проведен обзор современных средств визуального моделирования и прототипирования сложных ИС.

Показана их ограниченность в плане моделирования широкого спектра семантических отношений предметной области. Определен ряд требований, которым должна удовлетворять современная система моделирования и прототипирования.

Показана возможность расширения современных систем, таких как Taylor, средствами, обеспечивающими приведенные требования. Приведена архитектура расширенной системы моделирования и прототипирования.

Литература

1. Stacy D.A. Lecture notes on Rapid Prototyping [Электронный ресурс] / D.A. Stacy // 1997. – Режим доступа http://hebb.cis.uoguelph.ca/~dave/343/Lectures/prototype.html.

2. Владмиров С. И-Поиск. Информация. Знание. Моделирование в информационных системах. / С. Владимиров // 2008 [Электронный ресурс]. – Режим доступа http://subscribe.ru/archive/science.model.ipoisk/200811/26124927.html.

3. Павловский В.И. Система моделирования слаботипизированных иерархических данных. / В.И. Павловский, А.Л. Зинченко // Вестник Черниговского госуд. технологического университета. – Чернигов: ЧГТУ, 2009. – Вып. 40. – С. 162–169.

4. Pavlovsky V, Zincenko A. To the question about the problem of object modeling weak-typed hierarchical data / V. Pavlovsky, A. Zincenko // Сучасні комп'ютерні системи та мережі розробка та використання: Матеріали 4-ї Міжнародної НПК ACSN-2009. – Львів: НВФ "Українські технології", 2009. – С. 127–129.

Поступила в редакцию 28.01.2010

Рецензент: д-р техн. наук, проф. Г.Н. Жолткевич, Харьковский национальный университет им. В.Н. Каразина, Харьков, Украина

ПРОБЛЕМИ ОБ'ЄКТНОГО ПІДХОДА ДО МОДЕЛЮВАННЯ ТА ПРОТОТИПУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

В.І. Павловський, А.Л. Зінченко

Розглянуто проблеми об'єктного підходу до моделювання інформаційних систем. Проведено огляд сучасних засобів прототипування програмного забезпечення. Наведена їх обмеженість в плані моделювання широкого спектру семантичних відносин предметної області. Визначено ряд вимог до сучасної системи моделювання та прототипування. Наведено можливість розширення сучасних систем, таких як Taylor, засобами які забезпечують наведені вимоги.

Ключові слова: прототипування, каркас, моделювання, модель, слабка типізація, ієрархічні дані, відношення, семантичні зв'язки.

PROBLEMS OF THE OBJECTIVE APPROACH TO MODELING AND PROTOTYPING OF INFORMATION SYSTEMS

V.I. Pavlovsky, A.L. Zinchenko

The problems of objective approach to modeling informational systems are considered. The modern facilities of software prototyping are analytically reviewed. Limited application modeling possibility of domain's semantic wide spectrum relations is shown. A number of requirements to modern prototyping and modeling system is given. The possibility of modern systems expansion, such as Taylor, with facilities that satisfy defined requirements, are shown.

Key words: prototyping, framework, modeling, model, weak typification, hierarchical data, relations, semantic links.

Павловский Владимир Ильич – канд. техн. наук, доцент, заведующий кафедрой информационных и компьютерных систем, Черниговский государственный технологический университет, Чернигов, Украина, e-mail: Pavlovsky@stu.cn.ua.

Зинченко Андрей Леонидович – старший преподаватель кафедры информационных и компьютерных систем, Черниговский государственный технологический университет, Чернигов, Украина, e-mail: a.zinchenko@stu.cn.ua.