Пушкина Н.В., Богомолов Д.Р.

Создание программного приложения с использованием UML

(СПбГУЭФ, Санкт-Петербург)

Универсальный язык моделирования – Unified Modeling Language (UML), прежде всего, является средством для моделирования программных продуктов, основанных на ОО (объектно-ориентированном) подходе к программированию. В сравнении с алгоритмическим объектно-ориентированный подход увеличивает масштабируемость программной системы за счет более высокого уровня абстракции кода. Данная концепция продемонстрировала свою полезность при построении очень сложных и больших по размерам программных продуктов, обеспечивая как высокую скорость разработки, так и относительно простую поддержку готового приложения. В связи с этим UML необходимо рассматривать в неразрывной связи с ОО подходом.

Разработчиками языка Гради Буч, Джеймс Рамбо, Ивар Якобсон UML определяется в качестве языка визуализации, специфицирования, конструирования и документирования программных систем. При этом если в общем плане язык рассматривается как словарь и правила комбинирования входящих в него слов в целях коммуникации, то язык моделирования – как язык, словарь и правила которого сосредоточены на концептуальном и физическом представлении системы, а соответственно UML – как стандартное средство представления «чертежей» программного обеспечения.

Язык UML в полной мере удовлетворяет четырём принципам моделирования. С точки зрения визуализации модели, он позволяет графически отобразить поведение системы. Состоя из набора графических символов, данный язык полностью удовлетворяет условию однозначной интерпретации его элементов, что даёт возможность кооперированной работы над проектом. Конструирование приложения обеспечивается с помощью средств прямого (forward engineering) и обратного (reverse engineering) проектирования, благодаря чему появляется возможность как отображения готовой UML модели на исходный код объектно-ориентированных языков, так и получения модели из готового исходного кода. Причем возможна работа, как с графическим, так и с текстовым представлениями. UML содержит в себе и средства обеспечения документации проекта, что удовлетворяет четвёртому принципу моделирования систем.

Язык моделирования UML содержит полный набор инструментальных средств для разработки архитектуры приложения. Являясь, прежде всего, абстрактным средством моделирования, он позволяет абстрагироваться от конкретной платформы и конкретного языка программирования. В то же время предоставляется инструментарий для создания исходного кода на основе предложенной модели.

Для иллюстрации шаблонов проектирования приложений, основных инструментов языка UML и анализа средств для работы с данным языком в

качестве условного эксперимента автором была выбрана разработка программного приложения, в котором с одного персонального компьютера выполняются HTTP DoS воздействия на вычислительную систему с целью изменить ее состояние или затруднить доступ к ресурсу для пользователей.

В приложении используются расхожие шаблоны проектирования, такие как коллекция объектов (indentity map), фабрика исполнителей (executor factory) и синхронизация с использованием критических секций, интерфейсы (и работа с интерфейсными ссылками), подключение сторонних компонентов (java runtime environment) и реализуются следующие функции:

1) осуществляется многопоточный вызов GET запросами содержимого удалённых веб-приложений;
2) предоставляется статистика по HTTP ответам от целевого узла:

- ввод url адресов происходит через текстовый файл в кодировке UTF-8;
- ввод параметров происходит через стандартную консоль;
- предусматривается выход до завершения цикла DoS'a;
- вывод статистики осуществляется без прерывания основной логики.

Определяются классы приложения, представленные файлами исходного кода:

- Main.java – главный класс, в нём задаются параметры, запускаются необходимые для функционирования нити
- Tinput.java – класс для считывания вводимых символов в консоль и на основе ввода выполнения определённых действий (остановка программы и вывод статистики)
- CollectionI.java – интерфейс, определяющий методы для получения произвольного url адреса
- CurlExecutor.java – класс, являющийся непосредственно генератором Get-запросов
- FludExecutorFactory.java – класс, реализующий шаблон программирования фабрики исполнителей, создаёт и работает с потоками по интерфейсным ссылкам типа ExecutorInterface.java
- FileToArrayList.java – класс, загружающий файл со списком url'ов в коллекцию java.util.ArrayList и предоставляющий набор методов для получения доступа к произвольному (рандомному) элементу коллекции. Реализует интерфейс CollectionI.java
- Statistics.java – класс с критической секцией, обладающий набором методов для ведения статистики

С подробным описание разработанного приложения можно ознакомиться по адресу: http://sourceforge.net/projects/funny-bunny/files/.

Создание UML диаграммы классов

Процесс моделирования логики программного приложения рассматривается в программном продукте Oracle Jdeveloper и начинается с составления диаграммы классов. Главным классом (инициализирующимся по умолчанию) является класс Main. Соответственно, в конструкторе Main обрабатываются параметры программы и создаются экземпляры необходимых для рабо-

ты классов. В нём, к примеру, специализируется экземпляр класса FileToArrayList и передаётся далее по интерфейсной ссылке в качестве экземпляра объекта CollectioI. Диаграмму вы можете найти по адресу: http://sourceforge.net/projects/funny-bunny/files/article.pdf.

Написание исходного кода приложения «вручную»

Обычно исходный код создается при помощи средств IDE. При этом автоподсветка синтаксиса, автодополнения, и прочие функции позволяют программисту избежать ошибок. Дизайнеры графического интерфейса пользователя позволяют отойти от кода, описывающего функциональные графические элементы. Справка по языкам программирования даёт исчерпывающие данные о применении того или иного пакета, так что, основная задача программиста избежать логических ошибок, избыточного функционала и, желательно, избыточного кода.

Можно выделить некоторые требования к коду:

- название переменных и методов должны носить осмысленные имена;
- необходимо придерживаться общепринятых правил форматирования кода выбранного языка;
- желательно оставлять комментарии, как для разработчиков, так и для пользователей библиотеки компонентов приложения.

Многие языки программирования имеют набор мета-тегов для использования в комментариях, что даёт возможность автоматической генерации технической документации. Например, комментарии, начинающиеся с «/**», обрабатываются утилитой javadoc, что позволяет создавать техническую документацию автоматически. Существует набор тегов, которые описывают основные параметры методов, с помощью которых можно определить поля получаемого документа. Комментарии, полученные с помощью javadoc, агрегируются и собираются в HTML документ.

Создание модели UML из исходного кода приложения

Процесс трансформации исходного кода приложения в модель посредством отображения из определённого языка реализации называется обратным проектированием. Данный процесс, несомненно, сложен, так как использует алгоритмы синтаксического поиска и обработки, кроме того не всегда просто на основе лексем языка однозначно определить модель как UML модель. Тем не менее, средства обратного проектирования являются важными составляющими для нормального хода процесса RUP, на данный момент являющегося стандартом де факто для фирм – разработчиков ПО, таких как Oracle, SAP и IBM. Ключевыми фазами для использования средств обратного проектирования в итеративном процессе RUP являются фаза разработки и конструирования. Существует множество причин, из-за которых сложно отказаться от средств обратного проектирования при создании программного приложения (проекты среднего и большого масштаба). Выделим следующие:

- Благодаря использованию шаблонов проектирования, отпадает необходимость описывать каждый компонент системы досконально, так как использование расхожих методик при создании приложения уже включает в себя часть проектирования.

- Как бы хорошо не был разработан проект, какие бы требования не были поставлены, по ходу создания проекта возникает необходимость внести правки в модель. Приложение разрабатывается программистами, требования к нему выдвигаются проектировщиками, которые не всегда знают язык разработки, в результате постоянные совещания представителей этих двух уровней приводят к потере времени. В связи с этим, для выполнения требований RUP необходим инструментарий, позволяющий получать модель из исходного кода.
- Многие разработчики предпочитают старые методы. В связи с этим, вместо того, чтобы терять время (а значит и деньги) на переучивание кадров, которые неохотно воспринимают данную технологию, лучше иметь инструментарий для изменения модели на «лету».

Рассмотрим Oracle Jdeveloper и его средства для разработки UML модели. Этот продукт используется при разработке ПО на технологической платформе Java. Полученная с его помощью диаграмма практически полностью соответствовала диаграмме, которая была смоделирована в самом начале, за исключением связей, ведущих от класса Main к классу создания коллекции на основе текстового файла и самого интерфейса, который реализуется данным классом. Корректность полученной модели достигается за счёт следующего:

- Oracle Jdeveloper изначально разрабатывался для работы с Java, поэтому идеально работает с UML средствами по отношению к Java.
- Oracle Jdeveloper имеет собственный «диалект» UML, точнее набор сущностей, отражающих некоторые ключевые аспекты Java, для реализации которых в UML требуются свои способы.

Даже на таком простом примере, как разработка вышеназванного приложения, видны ограничения для отражения UML диаграммы в код приложения, и обратно: для сохранения совместимости иногда необходимо придерживаться излишних синтаксических конструкций языка. В целом, Oracle Jdeveloper хорошо справился с задачей обратного проектирования.

Рассмотрим продукт Umbrello, который также предназначен для обратного проектирования. Этот продукт распространяется по GPL v2 лицензии и является продуктом с полностью открытым кодом. Как оказалось, данный программный продукт не достаточно хорошо справляется с преобразованием исходного кода на Java. Следует отметить два важных момента: Umbrello по умолчанию не распознает основных пакетов (библиотек) Java, и не совсем корректно работает с javadoc стилем комментирования исходного кода, благодаря чему приложение лишается комментариев, которые являются важной составляющей при создании технической документации. Кроме того, Umbrello не смог взять из кода приложения реализации импортированных функций. Можно заметить следующие ошибки генерации UML диаграммы на основе исходного кода:

- В первую очередь: неправильные связи между классами, что является наиболее критическим.

- Umbrello некорректно обработал следующие нативные типы языка Java: Runnable, File, ArrayList, Random, в результате чего были созданы «несуществующие» элементы модели.

Генерация исходного кода

В первую очередь рассмотрим, что было получено на выходе генератора исходного кода Oracle Jdeveloper. Полученный набор (пакет) исходных кодов рассматриваемой программы оказался очень хорошего качества, а именно:

- Не было создано ничего лишнего в классах, пропущенные стандартные аннотации были созданы автоматически, также были проанализированы комментарии javadoc.
- Реализации функций не претерпели каких-либо изменений.
- Не было замечено никаких проблем с некорректным форматированием кода, более того, все недочёты в плане форматирования исходного кода были автоматически исправлены.

Получился полностью рабочий код, отформатированный по всем правилам Java, причем он не был изменён структурно. Более того, полученный код соответствовал стандартам корпоративной и командной разработки программного обеспечения на рассматриваемом языке.

Теперь перейдём к анализу кода, полученного с помощью Umbrello. Исходный код Java был создан без каких-либо настроек, кроме определения языка для генерации. Код получился не самого лучшего качества. Перечислим причины:

1) Ввиду того, что при генерации модели из исходного кода путём обратного проектирования Umbrello не взяла реализацию функций, полученный код не имеет реализованных методов.

2) Документация, как было замечено ранее, определилась некорректно, в связи с чем полученные комментарии javadoc не соответствуют стандартам. Более того, были созданы избыточные теги @return и @param и полностью потеряна javadoc документация классов и внутренних переменных.

3) Неправильно определились конструкторы всех классов.

4) Созданы избыточные методы получения и задания внутренних переменных класса (так называемые setter и getter методы).

5) Замечено неправильное форматирование исходного кода для Java.

6) Замечены излишние комментарии для разделения исходного кода класса на смысловые блоки.

В результате можно сделать вывод, что Umbrello целесообразно использовать только для непосредственной разработки моделей.

Сравнение и анализ методов получения исходного кода

Сравнивая написание кода вручную и создание кода из UML модели посредством средств генерации кода, можно отметить следующее:

- Несомненно, моделирование композиции классов и других UML композиций позволяет избежать большого количества ошибок на стадии написания приложения и, благодаря графическому отображению модели, помогает лучше смоделировать и понять разрабатываемое приложение.

- Необходимо использовать UML моделирование с некоторой осторожностью: масштаб создаваемой модели очень сильно влияет на количество работы. К примеру, рассматриваемая в статье программа не нуждалась в средствах моделирования с самого начала: хватило бы блок-схемы и структуры пакета.
- UML удобен в качестве связующего звена между проектировщиками и программистами, он позволяет работать на уровне архитектуры, абстрагируясь от конкретной реализации.
- Создание проекта полностью в UML редакторах – достаточно неудобное занятие. Постоянная работа с пользовательским интерфейсом занимает много времени, при написании исходного кода в редакторе текстов IDE документацию можно создавать, не отрываясь от процесса кодирования, часто ради стандартизации кода, текст программы обрастает излишними инструкциями. К примеру, был проведён эксперимент: набор кода представленной в этой статье программы вместе с документацией занял около 30 минут работы в редакторе NetBeans. Тот же самый код через средства работы с UML создавался около 2-х часов.
- На данный момент UML технологии предоставляются в основном на коммерческих основаниях. В то же время такие распространяющиеся бесплатно продукты, как Oracle Jdeveloper и Eclipse также обладают неплохим инструментарием для работы с UML case средствами.

Щугорева В.А.

Технологии применения ППП «Business Studio» в качестве платформы для проектирования системы управления

(СПбГУЭФ, Санкт-Петербург)

Введение

Деятельность любой компании направлена на достижение поставленных целей, задаваемых собственниками или топ-менеджерами, причем на достижение этих целей с минимальными затратами времени и ресурсов. Добиться этого компания может, если она обладает эффективной, грамотно построенной системой управления. Система управления организацией – совокупность взаимосвязанных элементов, из которых основными являются система целей и показателей, модель бизнес-процессов и организационная структура управления. Система целей и показателей отвечает на вопрос «Чего?» необходимо достигнуть организации и как будет определяться достижение целей, модель бизнес-процессов отвечает на вопросы ***что?***, ***когда?*** (в некоторых случаях и ***как?***) необходимо для этого делать, организационная структура отвечает на вопрос ***кто?*** это будет делать.

Управление предприятием в современных условиях требует все большей оперативности. Поэтому использование информационных систем управ-