УДК 002.53; 002.53:004.65; 002.53:004.62/.63

МЕТОДЫ И СРЕДСТВА UML КАК ИНСТРУМЕНТЫ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

А.А. Вичугова, В.Н. Вичугов, Г.П. Цапко

Томский политехнический университет
E-mail: anya@aics.ru

Рассмотрен процесс разработки программного обеспечения на этапе его проектирования с помощью инструментов графического моделирования в соответствии с объектно-ориентированным подходом. Показано использование инструментария UML на практическом примере. Описаны средства графического моделирования с помощью UML.

Ключевые слова:
Информационные системы, моделирование, проектирование, разработка.

Вичугова Анна Александровна, ассистент кафедры автоматики и компьютерных систем Института кибернетики ТПУ.
E-mail: anya@aics.ru
Область научных интересов: бизнес-моделирование, структурный анализ, базы данных, информационные системы электронного документооборота, информационно-управляющие системы.

Вичугов Владимир Николаевич, канд. техн. наук, доцент кафедры автоматики и компьютерных систем Института кибернетики ТПУ.
E-mail: vlad@aics.ru
Область научных интересов: программирование, базы данных, автоматизированные системы управления.

Цапко Геннадий Павлович, д-р техн. наук, профессор, заведующий кафедрой автоматики и компьютерных систем Института кибернетики ТПУ.
E-mail: tsapko@aics.ru
Область научных интересов: системный анализ, информационные технологии управления жизненным циклом продукции.

Введение

В связи с возрастающей сложностью программных систем в 80-х гг. XX в. появились специальные методы и средства проектирования программного обеспечения (ПО), называемые CASE (от англ. Computer Aided Software Engineering). Современный рынок информационных технологий предоставляет множество программных продуктов категории CASE, которые поддерживают существующие подходы к разработке ПО. В настоящее время принято выделять следующие основные подходы к проектированию ПО:

- структурный, в основу которого положен принцип алгоритмической декомпозиции – структура системы описывается в терминах иерархии ее функций и передачи информации между отдельными функциональными элементами (модулями);
- объектно-ориентированный, который использует объектную декомпозицию – структура системы определяется множеством объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами.

Структурный подход имеет более длительную историю по сравнению с объектно-ориентированным, поэтому CASE-средства изначально поддерживали методы структурного подхода, например IDEF0, IDEF1, IDEF3, DFD. Наиболее распространенными в России CASE-средствами можно назвать такие информационные системы, как All Fusion Process Modeller, ARIS, Ramus и т. д.

В связи с активным развитием объектно-ориентированного подхода появляется все больше программных средств, реализующих методы объектно-ориентированного описания информационных систем, например Rational Rose, StarUML, Visual Paradigm UML, ArgoUML, BOUML, yEd и др. Кроме того, поддержка методов объектно-ориентированного подхода находит отражение и в новых версия некоторых CASE-систем, традиционно считающихся структурными, например ARIS [1].

Основным методом для описания информационной системы в соответствии с объектно-ориентированным подходом является универсальный язык моделирования UML (от англ. Unified Modeling Language), разработанный в 90-х гг. Г. Бучем, Д. Рамбо и И. Якобсоном. В современной версии языка UML 2.4 выделяют около 15 видов диаграмм для описания структуры и поведения проектируемого ПО [2]. Однако в связи с многообразием видов UML-диаграмм целесообразно определить типовую последовательность действий по применению этого инструмента на этапе проектирования ПО.

Проектирование ПО

Согласно существующим стандартам, которые регламентируют жизненный цикл ПО (ГОСТ 34.601-90, ISO/IEC 12207:1995 и его российский аналог – ГОСТ Р ИСО/МЭК 12207-99), а также моделям жизненного цикла ПО (водопадная, итерационная и спиральная) [3], неотъемлемой частью процесса разработки является этап проектирования, на котором определяется базовая структура информационной системы (ИС), ее компоненты, их назначение и взаимосвязь. Например, поставлена задача разработки информационной системы, которая включает базу данных (БД) и приложение для работы с ней. Начальным этапом проектирования ПО является определение его архитектуры и основных составляющих. Это можно сделать с помощью UML-диаграммы компонентов (рис. 1).

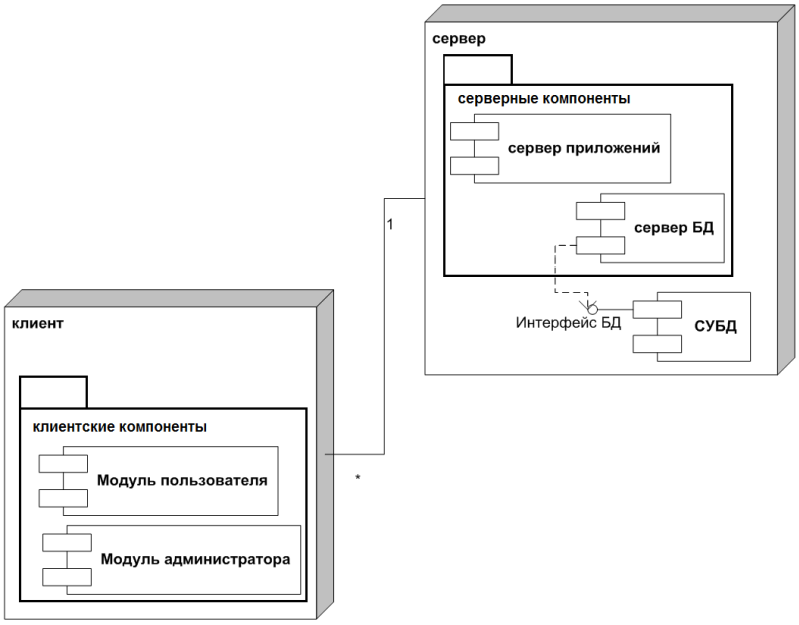

Рис. 1. UML-диаграмма компонентов, иллюстрирующая архитектуру проектируемого ПО

Представленное на рис. 1. проектируемое ПО описывается в терминах графических символов UML-диаграммы компонентов следующим образом:

- клиент-серверная архитектура показана с помощью двух узлов (клиент и сервер), связанных по типу «1 ко многим»;
- на клиентском узле «Модуль пользователя» и «Модуль администратора» объединены в программно-логический пакет «Клиентские компоненты»;
- аналогичным образом на узле сервера «Сервер приложений» и «Сервер БД» объединены в пакет «Серверные компоненты»;
- при этом компонент «Сервер БД» для связи с БД использует соответствующий интерфейс, предоставляемый системой управления базой данных (СУБД).

В соответствии с объектно-ориентированным подходом, необходимо выделить типы рассматриваемых сущностей, их характеристики (атрибуты) и операции, которые могут быть над ними произведены (методы). В UML это выполняется с помощью диаграммы классов

(рис. 2), на которой можно показать не только типы объектов в виде классов, но и различные связи между ними.

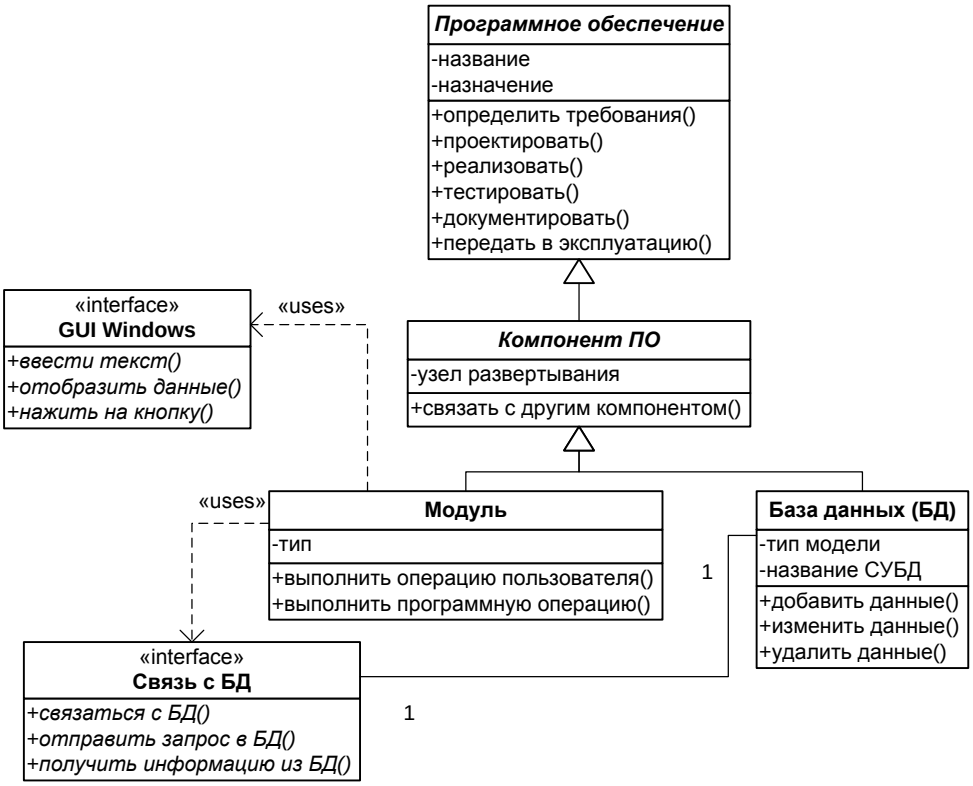

Рис. 2. Составные части проектируемого ПО и связи между ними на UML-диаграмме классов

В рассматриваемом примере определены следующие классы:

- *«Программное обеспечение»*, атрибутами которого являются «название» и «назначение», а операциями – действия, практически полностью соответствующие этапам проекта по водопадной модели жизненного цикла ПО («определить требования», «проектировать», «реализовать», «тестировать», «документировать», «передать в эксплуатацию»). Данный класс находится на самом верхнем уровне абстракции, не содержит конкретных объектов (экземпляров) и потому является абстрактным (название класса на диаграмме показано курсивом).
- *«Компонент ПО»*, наследник класса «ПО», также является абстрактным, характеризуется атрибутом «узел развертывания» и операцией «связать с другим компонентом»;
- *«Модуль»*, наследник класс «Компонент ПО», описывается атрибутом «тип», а также операциями «выполнить операцию пользователя» и «выполнить программную операцию». При этом класс «Модуль» связан с внешними интерфейсами, например графический интерфейс пользователя (GUI – от англ. Graphical User Interface) Windows и интерфейс связи с БД, которые предоставляют необходимые операции;
- *«БД»*, также наследник класса «Компонент ПО», содержит атрибуты «тип модели» (реляционная, иерархическая, сетевая и т. д.), название СУБД (Oracle, MSSQL и т. д.), а также операции типовые работы с данными (добавить, изменить, удалить). При этом класс «БД» связан с интерфейсом, обеспечивающим взаимодействие с БД связью типа «1 к 1».

Далее следует определить жизненный цикл каждого из вышеперечисленных классов, т. е. составить перечень состояний и переходов между ними. В UML это выполняется в виде диаграмм состояний, в основе которых лежит аппарат дискретной логики и конечных автоматов. На рис. 3 показан жизненный цикл класса «Программное обеспечение», который включает шесть типовых состояний.

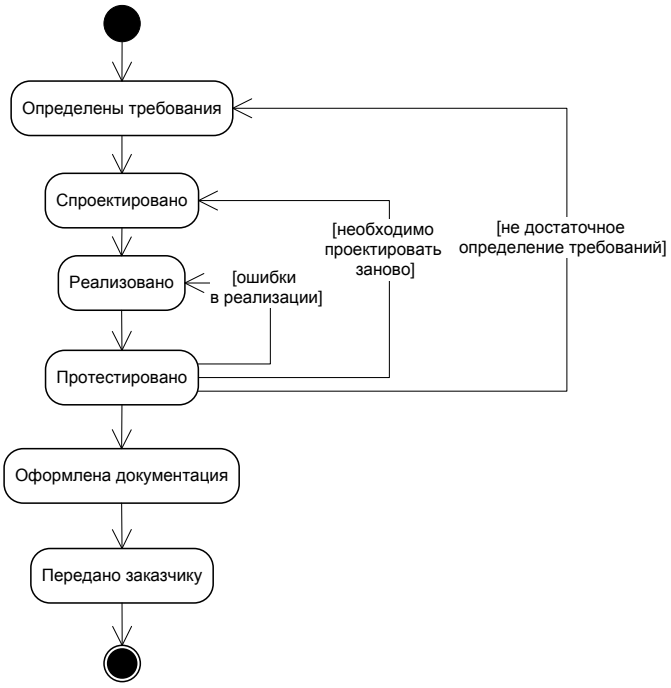

Рис. 3. UML-диаграмма состояний жизненного цикла у класса «Программное обеспечение»

После определения состояний жизненного цикла всех рассматриваемых классов целесообразно отобразить взаимодействие между их объектами, которые находятся в разных состояниях. Это можно выполнить в виде UML-диаграммы последовательностей, которая позволяет показать не только объекты разных классов в различных состояниях, но и обмен сообщениями между ними (рис. 4).

Рис. 4. Взаимодействие между объектами разных классов, показанное с помощью UML-диаграммы последовательностей

На рис. 4 показана следующая последовательность взаимодействий разных классов:

- как только определены требования к программному обеспечению, начинается этап проектирования БД;
- после проектирования БД идет этап ее реализации;
- после реализации БД начинается разработка модуля, обеспечивающего работу с ней.

Однако помимо технических аспектов проектирования и реализации ПО необходимо также описать технологию работы с ним, т. е. процессы разработки и эксплуатации. В UML это делается средствами диаграммы деятельности, которая позволяет описать логическую последовательность действий и их исполнителей, а также показать связанные с процессами объекты. В рамках рассматриваемого примера на рис. 5 показан процесс разработки ПО.

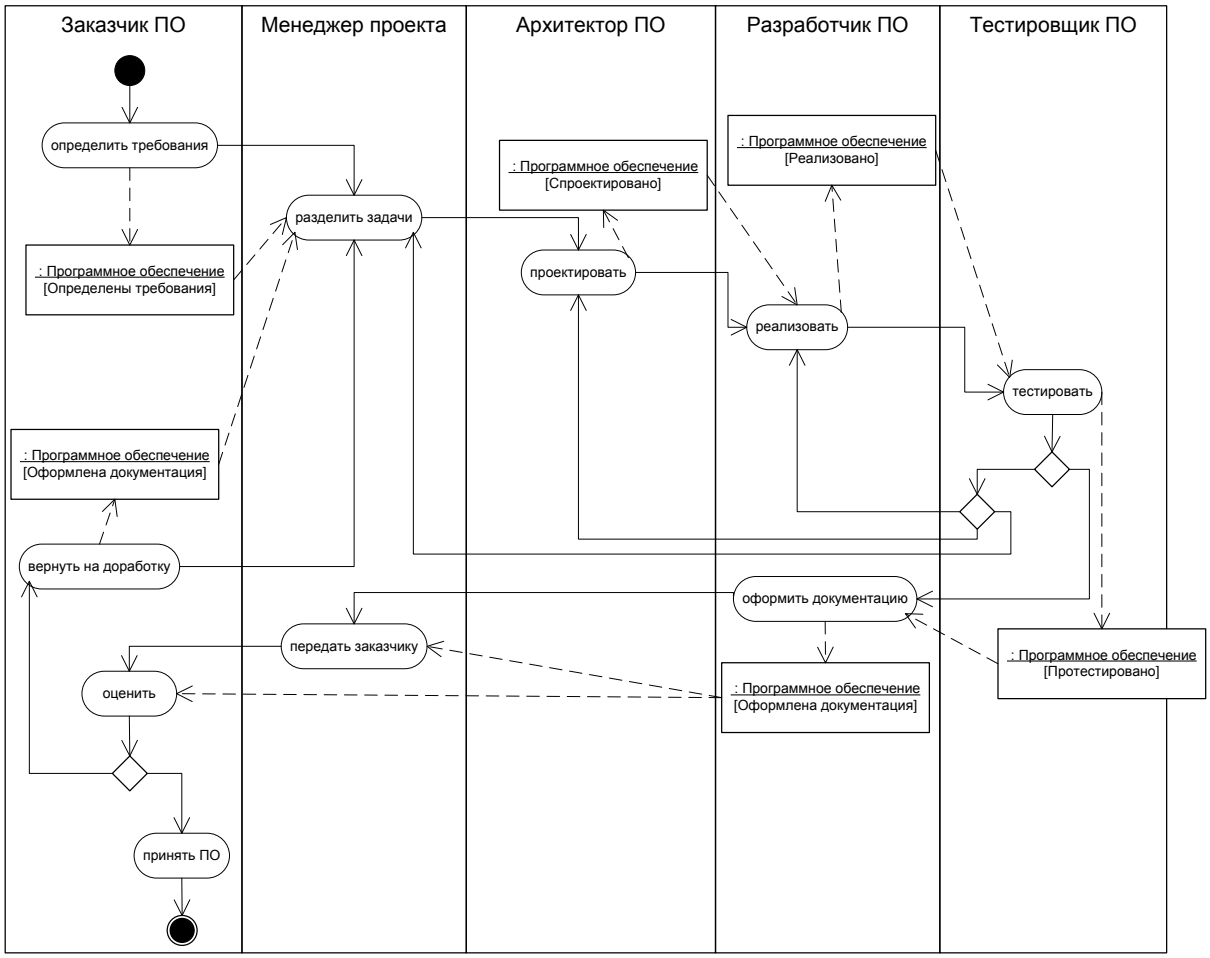

Рис. 5. UML-диаграмма деятельности по разработке ПО

В рассматриваемом процессе показаны его участники (Заказчик, Менеджер проекта, Архитектор, Разработчик и Тестировщик), исполняемые ими действия, а также входные/выходные данные для этих действий в виде объектов ранее указанных классов в соответствующих состояниях. Таким образом, UML-диаграммы деятельности подробно иллюстрируют алгоритмы выполнения процессов.

Выводы

Рассмотренный пример показывает некоторые приемы проектирования ПО с помощью методов UML. Полученные с их помощью диаграммы представляют собой основу проектируемого ПО в виде формальных информационных моделей и алгоритмического обеспечения. Объемы и сложность современных информационных систем привели к тому, что подобное проектирование ПО становится неотъемлемой частью процесса разработки и предваряет этап реализации. Говоря о методах UML, следует отметить также и средства их воплощения. В настоящее время многие среды разработки ПО включают инструменты визуального проектирования с возможностью последующей генерации программного кода из построенных UML-моделей, например Microsoft Visual Studio, NetBeans, Delphi XE3, TJI Java IDE, JBuilder и др. Целесообразно выделить универсальные графические редакторы, например yEd, Dia, MS Visio, которые позволяют графически изображать диаграммы как в структурных, так и в объектно-ориентированных нотациях.

СПИСОК ЛИТЕРАТУРЫ

1. Рожкова Е. CASE-средства. Сравнительный анализ // ARIS – Rational Rose. URL: http://ocnova.ru/?p=334 (дата обращения: 10.12.2012).
2. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS. 2-е изд. / Пер. с англ.; под общей ред. проф. С. Орлова. – СПб.: Питер, 2006. – 736 с.
3. Мирошниченко Е.А. Технологии программирования. 2-е изд., испр. и доп. – Томск: Изд-во ТПУ, 2008. – 128 с.

Поступила 10.12.2012 г.