УДК 004.4:004.415(045)

Мацуева К. А.
Национальный авиационный университет, Киев

ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ ТЕХНОЛОГИЯ OMT ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ СИСТЕМ

В данной статье рассматривается объектно-ориентированное проектирование программных систем, в частности методология OMT проектирования прикладных программных систем. Проектирование программных систем при помощи данной методологии охватывает стадию анализа требований и предварительной разработки, где программная система рассматривается только в аспекте ее прагматики, и стадию проектирования, на которой принимаются основные решения по реализации проектируемой программной системы, а ее структура, разработанная на первой стадии, изменяется с учетом требований ее эффективности.

Объектно-ориентированное проектирование программного обеспечения связано с применением объектно-ориентированных моделей при разработке программных систем и их компонентов.

Объектно-ориентированное проектирование может начаться на самом первом этапе жизненного цикла; оно не связано с языком программирования, на котором предполагается реализовать разрабатываемую программную систему: этот язык может и не быть объектно-ориентированным. На этапе проектирования объекты – это некоторые формальные конструкции (например, прямоугольники с закругленными углами, с помощью которых они изображаются на схемах), никак пока не связанные с их будущей реализацией на языках программирования [2].

Объектно-ориентированное проектирование программного обеспечения связано с применением объектно-ориентированных методологий (технологий). Обычно эти объектно-ориентированные методологии поддерживаются инструментальными программными средствами, но и без таких средств они полезны, так как позволяют хорошо понять различные аспекты и свойства разрабатываемой программной системы, что в последующем существенно облегчает ее реализацию, тестирование, сопровождение, разработку новых версий и более существенную модификацию.

Проектирование прикладной программной системы начинается с анализа требований, которым она должна будет удовлетворять. Такой анализ проводится с целью понять назначение и условия эксплуатации системы настолько, чтобы суметь составить ее предварительный проект.

При объектно-ориентированном подходе анализ требований к системе сводится к разработке моделей этой системы. Модель системы (или какого-либо другого объекта или явления) это формальное описание системы, в котором выделены основные объекты, составляющие системы, и отношения между этими объектами. В модели опущены многочисленные детали, усложняющие понимание. Моделирование широко распространено и в науке, и в технике [3].

Модели позволяют:

- проверить работоспособность разрабатываемой системы на ранних этапах ее разработки;
- формировать и уточнять заказчиком системы его требования к системе;
- вносить изменения в проект системы, как в начале ее проектирования, так и на других фазах ее жизненного цикла.

В настоящее время существует несколько технологий объектно-ориентированного проектирования прикладных программных систем, в основе которых лежит построение и интерпретация на компьютере моделей этих систем. Одной из таких технологий является OMT (Object Modeling Techniques). Методология OMT поддерживается системой Paradigm+, одной из наиболее известных инструментальных систем объектно-ориентированного проектирования.

Проектирование программных систем при помощи данной методологии охватывает первые две стадии их жизненного цикла: стадию анализа требований и предварительной разработки, на которой будущая программная система рассматривается только в аспекте ее прагматики, и стадию проектирования (конструирования), на которой принимаются основные решения, связанные с реализацией проектируемой программной системы, а ее структура, разработанная на первой стадии, изменяется с учетом требований ее эффективности.

В технологии OMT на первом этапе жизненного цикла проектируемая программная система, обычно представляется в виде трех взаимосвязанных моделей:

- объектной модели, которая представляет статические, структурные аспекты системы, в основном связанные с данными;
- динамической модели, которая описывает работу отдельных частей системы;
- функциональной модели, в которой рассматривается взаимодействие отдельных частей системы (как по данным, так и по управлению) в процессе ее работы.

Объектная модель описывает структуру объектов, составляющих систему, их атрибуты, операции, взаимосвязи с другими объектами. Сначала определяются классы объектов, затем зависимости между объектами, включая агрегацию. Для упрощения структуры классов используется наследование. Объектная модель должна содержать краткие комментарии на естественном языке. Должны быть отражены те понятия и объекты реального мира, которые важны для разрабатываемой системы. В объектной модели отражается прежде всего прагматика проектируемой системы, что выражается в использовании терминологии прикладной области, связанной с использованием разрабатываемой системы [1].

Динамическая модель показывает поведение системы, в особенности последовательность взаимодействий. Сначала готовятся сценарии типичных сеансов взаимодействия с системой, затем определяются внешние события, отражающие взаимодействие системы с внешним миром; после этого строится диаграмма состояний для каждого активного объекта, на которой представлены образцы событий, получаемых системой и порождаемых ею, а также действий, выполняемых системой. Построенные диаграммы состояний сравниваются между собой, чтобы убедиться в их непротиворечивости. На этом построение динамической модели заканчивается [1].

Функциональная модель показывает функциональный вывод значений безотносительно к тому, когда они вычисляются. Сначала определяются входные и выходные значения системы как параметры внешних событий. Затем строятся диаграммы потоков данных, показывающие как вычисляется каждое выходное значение по входным и промежуточным значениям. Диаграммы потоков данных выявляют взаимодействие с внутренними объектами системы, которые служат хранилищами данных в периоды между сеансами работы системы. В заключение определяются ограничения и критерии оптимизации [1,6].

Эти три вида моделей позволяют получить три взаимно-ортогональных представления системы в одной системе обозначений. Совокупность моделей системы может быть проинтерпретирована на компьютере (с помощью инструментального программного обеспечения), что существенно упрощает согласование предварительного проекта системы.

Модели, разработанные и отлаженные на первой фазе жизненного цикла системы, продолжают использоваться на всех последующих его фазах, облегчая программирование системы, ее отладку и тестирование, сопровождение и дальнейшую модификацию.

Модели системы не связаны с языком программирования, на котором будет реализована система.

На втором этапе жизненного цикла, после того как прикладная задача исследована и результаты ее исследования зафиксированы в виде объектной, динамической и функциональной моделей, необходимо перейти к конструированию системы. На этапе конструирования системы принимаются решения о распределении подсистем по процессорам и другим аппаратным устройствам и устанавливаются основные принципы и концепции, которые формируют основу последующей детальной разработки программного обеспечения системы.

Внешняя организация системы называется архитектурой системы. Выбор архитектуры системы является еще одной задачей, решаемой на этапе ее конструирования. Архитектура системы определяет ее разбиение на модули, задает контекст, в рамках которого принимаются проектные решения на следующих этапах разработки.

На этапе проектирования системы необходимы действия по:

- определению разбиения системы на модули;
- выялению асинхронного параллелизма в системе;
- определлению состава вычислительного комплекса, на котором будет работать система;
- распределению компонент системы по процессорам вычислительного комплекса и независимым задачам;
- организации управления хранилищами данных;
- организации управления глобальными ресурсами;
- выбору принципов реализации управления программным обеспечением;
- организации управления пограничными ситуациями.

Начиная этап проектирования системы, необходимо определить ее разбиение на некоторое

количество компонентов – модулей. Модуль не является ни объектом, ни функцией; модуль - это набор (пакет) классов и отдельных объектов, подсистем, зависимостей, операций, событий и ограничений, которые взаимосвязаны и имеют достаточно хорошо определенный и по возможности небольшой интерфейс с другими модулями. Зачастую модуль включает одну подсистему, являясь ее реализацией. Модуль – подсистема обычно определяется через службы, которые он обеспечивает.

Каждый асинхронный (независимый) модуль (объект или подсистема) должен быть приписан к одному из устройств аппаратуры: универсальному процессору или специализированному функциональному устройству.

Решение использовать несколько процессоров обычно бывает связано с потребностью иметь более высокую производительность, чем производительность одного процессора. Количество требуемых процессоров зависит от объема вычислений и производительности компьютера.

Аппаратуру следует рассматривать как неизменяемое тщательно оптимизированное программное обеспечение. Каждое устройство может рассматриваться как объект, который работает параллельно с другими объектами. Проектант может принять решение о замене некоторых объектов подходящими аппаратными устройствами. Обычно такое решение принимается по следующим причинам:

- требуемые устройства должны быть легко доступными;
- требуется более высокая производительность специализированных устройств выше, чем производительность имеющихся процессоров.

При распределении модулей и подсистем по процессорам следует иметь в виду следующее:

- некоторые задачи нужно выполнять на определенных устройствах;
- время ответа или скорость информационного потока превышает пропускную способность канала между процессором и программой; например, высокоскоростные графические устройства требуют спаренных контроллеров;
- скорости вычислений слишком высоки для одного процессора, и задачи нужно разместить на нескольких процессорах; подсистемы, которые часто взаимодействуют, нужно поместить на одном процессоре.

В качестве хранилищ данных могут использоваться базы данных, файлы и другие структуры данных, размещенные во внешней или основной памяти. Выбор вида хранилища данных зависит от ситуации [2].

В базах данных обычно размещают данные, удовлетворяющие одному из следующих условий:

- данные, для которых требуется доступ на высоком уровне детализации со стороны многих пользователей;
- данные, которые могут эффективно управляться командами СУБД;
- данные, которые должны переноситься на многие платформы;
- данные, для которых требуется доступ со стороны нескольких прикладных программ.

Глобальными ресурсами могут быть: процессоры, устройства внешней памяти, экран рабочей станции, логические имена (идентификаторы объектов, имена файлов, имена классов), доступ к базам данных и т.п.

Во время анализа все взаимодействия представляются в виде событий. Управление аппаратурой соответствует этой модели, но необходимо выбрать метод управления программным обеспечением системы. Существует два класса методов управления программным обеспечением: методы внешнего управления и методы внутреннего управления.

Методы внешнего управления:

- последовательное управление процедурами,
- последовательное управление событиями,
- параллельное асинхронное управление.

При последовательном управлении процедурами в каждый момент времени действует одна из процедур. При последовательном управлении событиями управлением занимается монитор (диспетчер). При параллельном асинхронном управлении этим заведует несколько управляющих объектов (мониторов).

Внутреннее управление связано с потоками управления в процессах. Оно существует только в реализации и потому не является только последовательным или параллельным. В отличие от внешних событий, внутренние передачи управления, как например, вызовы процедур или обращения к параллельным задачам контролируются программой и могут быть структурированы в случае необходимости.

Последующим шагом является предусмотрение поведения каждого объекта и всей системы в пограничных ситуациях: инициализации, терминации и обвале.

Перед тем, как начать работать, система (объект) должна быть приведена в фиксированное

начальное состояние: должны быть проинициализированы все константы, начальные значения глобальных переменных и параметров, задачи и, возможно, сама иерархия классов. Во время инициализации, как правило, бывает доступна лишь часть возможностей системы.

Терминация состоит в освобождении всех внешних ресурсов, занятых задачами системы.

Обвал – это незапланированная терминация системы. Обвал может возникнуть в результате ошибок пользователя, нехватки ресурсов, или внешней аварии. Причиной обвала могут быть и ошибки в программном обеспечении системы.

Последующим этапом является разработка объектов (классов), составляющих систему. Часть объектов, которые были выявлены на этапе анализа системы, могут рассматриваться как основа системы. На рассматриваемом этапе разработки системы необходимо выбрать способ их реализации, стремясь минимизировать количество потребляемых ими ресурсов (времени их выполнения, используемой памяти и др.). При реализации объектов классы, атрибуты и зависимости должны быть реализованы в виде соответствующих структур данных, операции – в виде функций. При этом может возникнуть необходимость введения новых классов (объектов) для промежуточных данных.

Разработка объектов предполагает выполнение следующих шагов:

- рассматривая совместно три модели, получение операции над классами;
- разрабатка алгоритмов, реализующие полученные операции;
- оптимизация пути доступа к данным;
- реализация управления взаимодействиями с внешними объектами;
- уточнение структуру классов, чтобы повысить степень наследования;
- разрабатка зависимостей;
- определение представления объектов.

Методология OMT опирается на программный продукт OMTTool, который позволяет разрабатывать модели проектируемой программной системы в интерактивном режиме с использованием многооконного графического редактора и интерпретатора наборов диаграмм, составляемых при анализе требований к системе и ее проектировании с использованием методологии OMT. OMTTool входит в состав системы Paradigm+.

Список литературы

1. *Йордон Э., Аргила К.* Объектно-ориентированный анализ и проектирование систем – М.: Лори, 2010. – 264 с.

2. *Ларман К.* Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование. – М.: Вильямс, 2008. – 496 с.

3. *Гамма Э.* Приемы объектно-ориентированного проектирования. Паттерны проектирования. – М.: Питер, 2007. – 366 с.

4. *Маклаков С.В.* BPwin ERwin CASE-средства разработки информационных систем. М.: Диалог-МИФИ, 2001, 304 с.

5.*Корнеев В.В., Гареев А.Ф,, Васютин С.В., Райх В.В.* Базы данных: интеллектуальная обработка информации.-М.: Нолидж, 2000. – 352 с.

6.*Вендров А.М.* CASE – технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998. – 176 с.

Научный руководитель – Труш А. И., канд. техн. наук, доц.