УДК 598.3

О.А. Змеев, А.В. Лезарев

ШАБЛОН ОБЪЕКТНОГО ПРОЕКТИРОВАНИЯ ДЛЯ РЕАЛИЗАЦИИ ФУНКЦИОНАЛЬНОСТИ ПРОЦЕССА МОДЕЛИРОВАНИЯ В ИМИТАЦИОННЫХ МОДЕЛЯХ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Рассмотрены типичные задачи, возникающие при создании программного обеспечения для имитационного моделирования систем массового обслуживания. Предложен шаблон, обеспечивающий функциональность модели, необходимую для реализации непосредственного процесса моделирования. Описание шаблона выполнено средствами унифицированного языка моделирования (UML).

В современной действительности существует достаточно широкий круг систем, для исследования которых применяются методы теории массового обслуживания. Основным методом изучения таких систем является математическое моделирование, которое приводит к построению и анализу систем массового обслуживания (СМО). Но не менее актуальной является и задача построения имитационных моделей СМО. Результаты имитационного моделирования используются для анализа полученных на этапе математического моделирования аналитических результатов, а в случае, когда получить точное решение для СМО не представляется возможным, являются основным источником для дальнейших исследований. Для создания имитационных моделей СМО и их изучения могут быть использованы специальные системы имитационного моделирования. На сегодняшний день самой распространенной и общепринятой является GPSS. Другие системы малоизвестны или узко специализированы. Но и GPSS обладает рядом ограничений, самое известное из которых – требование дискретного времени. К тому же эта система была создана в 60-х годах. Поэтому желательна разработка новой системы имитационного моделирования на основе новых технологий программирования, тем более, что некоторые особенности в реализации систем имитационного моделирования СМО позволяют достаточно успешно применять для их разработки методы объектно-ориентированного анализа и проектирования. В настоящей работе предлагается ряд шаблонов, обеспечивающих реализацию функциональной составляющей имитационной модели СМО, разработанных с применением унифицированного языка моделирования (UML).

Анализ и проектирование

Типичные задачи имитационного моделирования

Основная идея объектно-ориентированного анализа и проектирования (ООАП) состоит в рассмотрении предметной области и логического решения задачи с позиций объектов (понятий или сущностей). В процессе ООАП основное внимание уделяется определению и описанию объектов в терминах предметной области. В результате этого процесса выявляются логические программные объекты, которые в последующем будут реализованы средствами объектно-ориентированного языка программирования.

Для представления результатов ООАП существуют порядка десяти различных систем обозначений, хотя в последние годы группа OMG сертифицировала в качестве производственного стандарта для объектно-ориентированного моделирования унифицированный язык моделирования UML, спецификации которого мы и будем придерживаться при дальнейшем изложении.

При создании программного обеспечения необходимо описать проблему и требования к системе. В нашем случае необходимо описать процессы, протекающие при функционировании имитационных моделей СМО. Несмотря на значительное разнообразие моделей, которые приходится строить, для их программной реализации практически всегда приходится решать ряд задач, которые являются абсолютно типичными для данного контекста. Проанализируем эти задачи.

Во-первых, имитационная модель обычно содержит некоторый источник поступления требований в систему. Кроме непосредственного моделирования поступления требований в систему, иногда на этом уровне требуется организовать дополнительную функциональность, связанную с обработкой потока входящих требований. Например, формирование суммарного потока требований из разных источников.

Во-вторых, необходимо обеспечить выполнение обязанностей, связанных с правилами поступления и формирования очереди или очередей требований, ожидающих обслуживания. Хотя для некоторых классов СМО, например для систем с бесконечным числом обслуживающих приборов, такая функциональность не является обязательной.

В-третьих, в модели должна существовать подсистема обслуживания требований с естественной для нее функциональностью.

И, наконец, на протяжении всей работы системы необходимо обеспечить средства, позволяющие накапливать статистический материал о ходе работы модели, для его последующего анализа и обработки.

Такой типичный набор проблем, возникающих в типичном контексте, позволяет применять для их решения образцы, механизмы и каркасы.

Образцы и каркасы

Согласно [1] образец, или паттерн (pattern), – это типичное решение типичной проблемы в данном контексте. Механизм (mechanism) – это образец проектирования, применимый к сообществу классов. Каркас (framework) – это архитектурный образец, предлагающий расширяемый шаблон для приложений в одной конкретной области. Прежде чем приступить к разработке каркаса и механизмов для приложений, связанных с имитационным моделированием СМО, дадим краткую характеристику перечисленным понятиям, при этом будем акцентировать внимание на средствах UML, предназначенных для их реализации.

Любая хорошо структурированная система содержит образцы на различных уровнях абстракции.

Образцы проектирования описывают структуру и поведение сообщества классов, а каркасы – структуру и поведение системы в целом. В UML есть средства для моделирования и тех, и других. При этом обычно любой образец оказывается автономным в контексте некоторого большого пакета, если не учитывать отношений зависимости, связывающих его с другими частями системы.

Механизм – это другое название образца проектирования, когда он применяется к сообществу классов. При моделировании механизмов следует принимать во внимание его внутренний и внешний вид.

При взгляде снаружи механизм изображается в виде параметризированной кооперации. Являясь кооперацией, механизм представляет собой набор абстракций, поведение и структура которых обязаны в ходе совместной работы выполнить некоторую полезную функцию. Параметры кооперации именуют те элементы, которые пользователь образца должен либо с чем-то связать, либо переопределить. Это и обеспечивает превращение образца проектирования в шаблон, который используется в конкретном контексте путем подстановки элементов, соответствующих параметрам шаблона.

При взгляде изнутри образец проектирования представляется простой кооперацией и изображается с определением ее структурной и поведенческой составляющих. Обычно для моделирования кооперации используются диаграммы классов и диаграммы взаимодействия. Диаграммы классов специфицируют структурную составляющую кооперации, а диаграммы взаимодействия – поведенческую. При этом параметры кооперации именуют некоторые из структурных элементов, которые при использовании с каким-то определенным контекстом конкретизируются абстракциями из этого контекста.

Для моделирования образца проектирования нам предстоит решить следующие задачи: идентифицировать типичное решение типичной проблемы и материализовать его в виде механизма, смоделировать механизм в виде кооперации, описав ее структурный и поведенческий аспекты, идентифицировать те элементы образца проектирования, которые должны быть связаны с элементами в конкретном контексте, и изобразить их в виде параметров кооперации.

Каркас – это более широкое понятие, чем механизм. Фактически каркас – это род микроархитектуры, включающий в себя множество механизмов, совместно работающих для решения типичной для данной предметной области проблемы. Определяя каркас, мы создаем скелет архитектуры со всеми управляющими органами, которые определяются пользователями, желающими применять этот каркас для адаптации к нужному контексту. В UML каркас моделируется в виде стереотипного пакета. Задача создания каркаса является достаточно сложной, и в рамках настоящей работы ее решение полностью не рассматривается. В последнем разделе приведены некоторые соображения, связанные с решением этой задачи.

Образец проектирования для имитационного моделирования СМО

Образец «Источник событий»

Сформулируем простой образец, позволяющий организовать очередь событий имитационной модели. Функциональность имитационной модели определяется тремя компонентами: подсистемой поступления требований, подсистемой функционирования очередей и, наконец, подсистемой обслуживания требований. Несмотря на то, что эти подсистемы реализуют различную прикладную функциональность, механизмы, с помощью которых она реализуется, являются общими для всех подсистем. Такая общность позволяет реализовать шаблон для проектирования этой типичной для имитационных моделей проблемы, а затем при помощи наследования и полиморфизма проектировать необходимый в рассматриваемом контексте вариант. Диаграмма классов, описывающая структурный аспект шаблона, приведена на рис. 1.

Рис. 1. Диаграмма класса

Поведенческий аспект шаблона раскрывается диаграммой последовательностей, которая приведена на рис. 2.

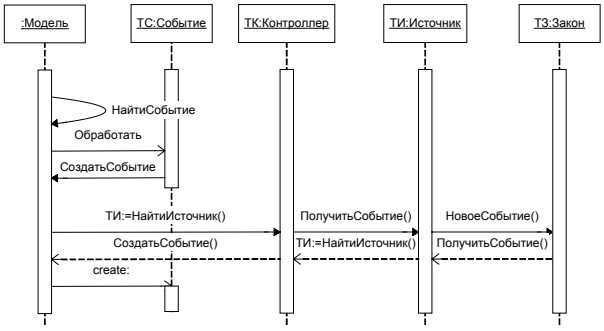

Рис. 2. Диаграмма последовательностей

Окончательный вид в виде параметризованной кооперации приведен на рис. 3.

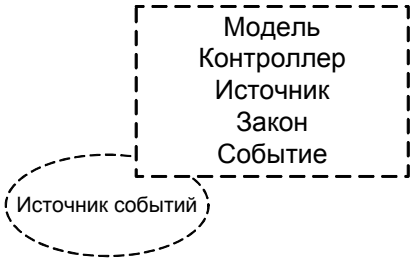

Рис. 3. Параметризованная кооперация

Перейдем к последовательному анализу предлагаемого образца, рассмотрев более подробно его структурный и поведенческий аспекты.

Структурный аспект – диаграмма классов

Рассмотрим множество классов, представленных на рис. 1.

Наиболее важным и сложным моментом объектно-ориентированного анализа и проектирования является квалифицированное распределение обязанностей между компонентами программной системы [2]. Разберем обязанности классов в предлагаемом множестве. Прежде всего, обратим внимание на то, что все предлагаемые классы абстрактные, так как это шаблон проектирования. А это значит, что непосредственная реализация этих классов зависит от вида моделируемой системы.

Класс «Закон» предназначен для моделирования законов поступления требований, их обслуживания, а также логики, связанной с очередями. Моделирование законов в виде класса, а не соответствующих функций у объектов модели не случайно. Известно, что в большинстве случаев в качестве закона будут фигурировать различного рода вероятностные распределения, число параметров в которых для различных распределений различно. Поэтому необходимо либо использовать концепцию перегрузки функций, что является достаточно сложным для не искушенного в тонкостях ООП читателя, либо использовать классы и механизм наследования, который мы и применили. Обязательной и единственной операцией данного класса является *НовоеСобытие* ( ), с его помощью связанный с классом «Закон» элемент получает время наступления следующего события.

Класс «Источник» предназначен для реализации компонентов модели, генерирующих системные события, связанные с изменениями состояния модели. В реальной имитационной модели данный класс будет иметь от одного до трех потомков, предназначенных для моделирования источников подобной функциональности: вход, очередь, обслуживание. Потомки в свою очередь могут образовывать собственное дерево наследования, например, в системах с функционально различными типами входа. Операция *ПолучитьСобытие*( ) предназначена для генерации нового события: поступления требования в систему или очередь, начала нового обслуживания и т.д.

Класс «Контроллер» является, прежде всего, классом-контейнером, который содержит соответствующие коллекции элементов-источников. Кроме того, на этом уровне реализуется функциональность, связанная с реализацией различного рода сетевых протоколов, правил постановки требований в очередь и т.д. Данный класс должен удовлетворять требованиям одноименного шаблона GRASP [2], согласно которому контроллер – это, объект, не относящийся к интерфейсу пользователя и отвечающий за обработку системных операций. У этого класса в реальной системе могут быть определены три потомка: контроллер множества источников поступающих в систему требований, контролер множества очередей и контролер множества обслуживающих приборов. Операция контроллера предназначена для поиска и передачи управления текущему активному «Источнику».

Класс «Модель» представляет собой всю систему в целом. В обязанности данного класса входят: создание очереди системных событий, организация их обработки, инициализация системы сбора статистики и ее включение.

Класс «Событие» является абстрактным классом для хранения информации о текущей системной операции. Кроме того, на этапе реализации сбора статистической информации этому классу можно поручить сбор соответствующей статистики, так он удовлетворяет шаблону «эксперт», согласно которому обязанности передаются классу, имеющему информацию, необходимую для их выполнения.

Динамический аспект – диаграмма последовательностей

Для моделирования динамических аспектов системы в UML применяются диаграммы последовательностей и кооперации. И те, и другие называются еще диаграммами взаимодействий. На диаграммах взаимодействий изображаются связи, включающие множество объектов и отношений между ними, в том числе сообщения, которыми эти объекты обмениваются. При этом диаграмма последовательности акцентирует внимание на групповой упорядоченности сообщений, а диаграмма кооперацией – на структурной организации посылающих и принимающих сообщения объектов. Эти диаграммы являются изоморфными, то есть могут быть преобразованы друг в друга без потери информации. Для специфицирования шаблонов более удобно использовать диаграмму последовательностей, которая для нашего шаблона приведена на рис. 2.

Прокомментируем представленную диаграмму. Естественно, что для начала работы системы необходимо провести инициализацию объектов, составляющих модель. Кроме этого, необходимо провести начальное заполнение очереди системных событий, которое заключается в формировании нового события для каждого потомка класса «Источник». Для краткости изложения соответствующий блок на диаграмме последовательностей опущен.

Далее процесс моделирования осуществляется по следующему циклу: «Модель» находит очередное системное событие, которое необходимо обработать (событие с минимальным значением атрибута *ВремяНаступления*). Для этого события вызывается операция *Обработать*( ), затем по информации, сопровождающей событие (ассоциация «Принадлежит на диаграмме классов»), определяется потомок класса «Источник», который с помощью ассоциированного с ним «Закона» генерирует следующее «Событие» и передает его по цепочке «Источник»–«Контроллер»–«Модель»–«Событие» в очередь системных событий. Далее цикл обработки повторяется. Условием завершения цикла является превышение атрибутом «Время» заданного времени моделирования, из аналогичных соображений включается механизм сбора статистики.

Диаграмма классов для каркаса «Имитационная модель»

Как уже было указано выше, в настоящей работе не ставится задача построения каркаса моделирования для приложений, реализующих имитационное моделирование СМО. В качестве стартовой точки для разработки такого каркаса может быть рассмотрен предлагаемый шаблон и диаграмма классов, представленная на рис. 4.

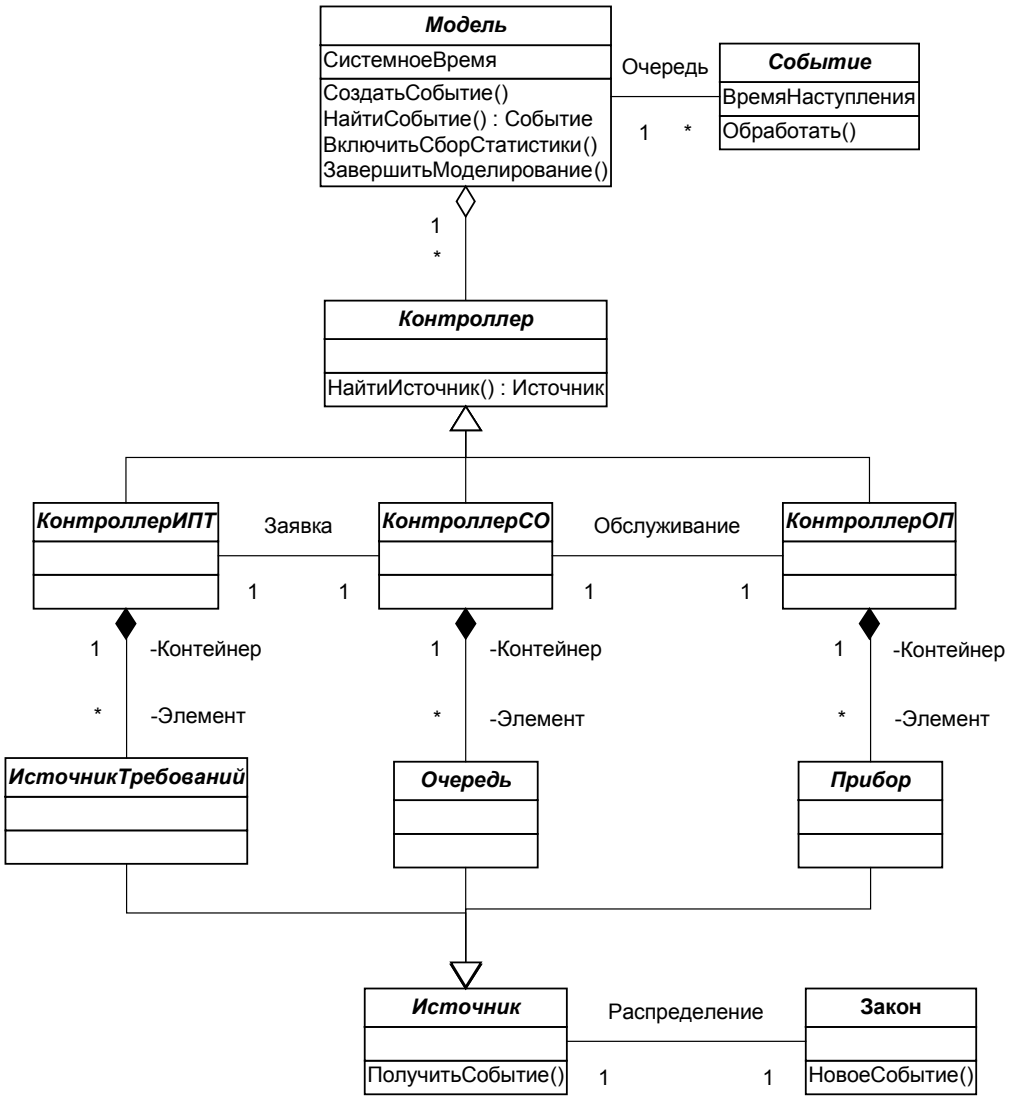

Рис. 4. Диаграмма классов каркаса

Визуализация, специфицирование, конструирование и документирование каркаса являются задачей, которую в настоящий момент рассматривают авторы. О ее проектном решении речь пойдет в наших последующих работах.

ЛИТЕРАТУРА

1. *Буч Г., Рамбо Д., Джекобсон А.* Язык UML: Руководство пользователя. М.: ДМК, 2000. 432 с.
2. *Ларман К.* Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование. М.: Вильямс, 2001. 496 с.

Статья представлена кафедрой прикладной информатики факультета информатики Томского государственного университета, поступила в научную редакцию номера 3 декабря 2001 г.