ДИАГРАММЫ СОСТОЯНИЯ UML КАК СПОСОБ ПРЕДСТАВЛЕНИЯ ГРАФА СОБЫТИЙ ИМИТАЦИОННОЙ МОДЕЛИ ДИСКРЕТНОЙ СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ

О.А. ЗМЕЕВ, А.В. ПРИСТУПА

Введение

Наиболее естественным способом представления дискретной имитационной модели являются графы событий, которые задают логику наступления событий в модели. Данный граф состоит из вершин и ребер: вершины представляют собой события, а ребра – отношения между ними. С одной стороны, задача построения модели от начала до конца с помощью графа событий весьма трудоемка, поскольку имитационная модель может содержать большое число элементов. С другой стороны, можно выделить некоторый базовый набор элементов, которые будут являться общими для широкого класса моделей. При отображении графа состояний средствами UML (унифицированный язык моделирования) приходится использовать так называемые диаграммы состояний [1]. При этом возникает необходимость сопоставить стандартные элементы стандарта проектирования информационных систем с контекстом предметной области. Один из возможных способов построения такого соответствия приведен в первой части настоящей статьи.

Как уже было отмечено выше, имитационные модели дискретных СМО (систем массового обслуживания) обычно состоят из некоторого набора стандартных элементов. Таким образом, при проектировании информационных систем, реализующих имитационное моделирование, одним из ключевых вопросов становится так называемое повторное использование ранее разработанных элементов подобной системы. Это приводит к проектированию и разработке отдельных настраиваемых компонентов, которые могут использоваться при построении различных

имитационных моделей. При этом у разработчика модели должна быть возможность установления связей и способов взаимодействия между зарегистрированными в системе компонентами. Во второй части данной работы обсуждается применение паттерна поведения «Observer» («Наблюдатель») [2], который позволяет определить зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом и автоматически обновляются.

Определение графов событий с помощью диаграмм состояний UML

Обычно каждое событие является отдельной вершиной в графе, а отношения порядка (или планирования) между событиями представляются направленными дугами. При этом дуга может начинаться и заканчиваться в одной и той же вершине, а также между двумя событиями может быть более одной дуги (как это будет показано ниже). Каждой дуге ставится в соответствие некоторая функция (иногда просто число), задающая разницу во времени наступления двух событий, связанных этой дугой. Также каждой дуге ставится в соответствие некоторое условие, которое является булевой функцией состояния системы (если условие не задано, то считаем его всегда истинным). Планируемое событие наступает тогда и только тогда, когда условие истинно.

Любое событие влечет за собой смену состояния модели (по определению). Состояние модели не есть нечто абстрактное, а с математической точки зрения состоит из конкретных величин, характеризующих модель в любой момент времени. На графе событий изменение этих величин (т.е. смена состояния) обозначается выражением, заключенным в фигурные скобки. Поскольку смена состояния происходит в момент наступления события, то такие выражения записываются на графе под событиями.

Проиллюстрируем описанное выше представление логики имитационной модели, используя стандартные элементы диаграммы состояний UML (рис. 1). Его можно интерпретировать следующим образом: «Когда наступает событие A,

выполняются все изменения состояния модели, связанные с этим состоянием. Далее, если условие c истинно, планируется наступление события B через t единиц времени».

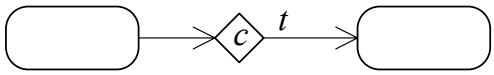

Рис. 1. Стандартные элементы структуры графа событий

В случаях, когда некоторое событие имеет собственные параметры, все дуги графа, ведущие в соответствующую этому событию вершину, имеют список аргументов, которые должны быть при вычислении подставлены вместо параметров. На рис. 2 представлен c пример соответствующей диаграммы состояний. Ее интерпретация в этом случае выглядит следующим образом: «Когда наступает событие A, выполняются все изменения состояния модели, связанные с A. Далее, если условие c истинно, планируется наступление события B через t единиц времени. При этом параметр p события B принимает значение выражения e».

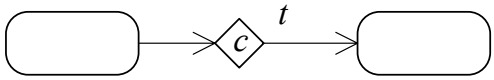

Рис. 2. Элементы диаграммы состояний с параметрами

Граф событий, который представляет некоторую законченную (замкнутую) логику, может выступать в моделях и как неделимый компонент. Это позволяет для часто используемых структур не создавать каждый раз заново подобные графы, а использовать однажды разработанные [4]. В терминах объектно-ориентированной методологии проектирования это приводит к повторному использованию соответствующего класса, изменение состояний которого описывается с помощью графа событий и представляется соответствующей диаграммой состояний. В качестве примера этой идеологии построим диаграммы состояний для типичных компонентов имитационной модели дискретной СМО.

Поступление заявок

Простейшим примером может служить обыкновенное поступление заявок в систему. Оно может быть представлено в виде диаграммы состояний следующим

образом: событие A – «Поступление заявки» – инициирует наступление аналогичного события через t_A единиц времени (рис. 3). Состояние соответствующего класса описывается атрибутом n, в котором хранится число сгенерированных заявок. При поступлении новой заявки значение n увеличивается на единицу. Причем служебное состояние H в этом случае иллюстрирует необходимость сбора статистики по этому событию.

Однажды разработанная подобная конструкция впоследствии может применяться многократно: разработчик модели сможет инстанцировать требуемое количество таких объектов, каждый со своим собственным состоянием и значениями t_A, которые определяются законом распределения.

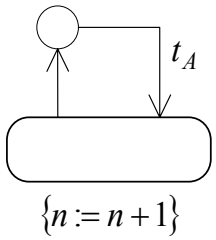

Рис. 3. Источник заявок

Очередь к группе устройств

Другим примером может служить очередь к группе устройств (рис. 4). Событие «Поступление заявки» изменяет состояние модели путем увеличения счетчика ожидающих заявок q на единицу. Если хотя бы одно из устройств свободно, мгновенно инициируется событие «Начало обслуживания». При этом опять же изменяется состояние модели: счетчик ожидающих заявок q и счетчик свободных устройств s уменьшаются на единицу. В этот же момент планируется наступление события «Конец обслуживания», которое произойдет через t_s единиц времени. Когда оно наступит, счетчик свободных устройств s увеличится на единицу. Если в очереди есть еще заявки, то снова инициируется событие «Начало обслуживания» и т.п.

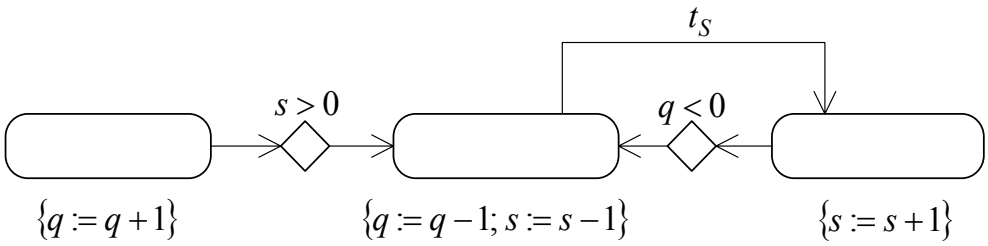

Рис. 4. Очередь к группе устройств

Две рассмотренные конструкции встречаются практически в каждой имитационной модели, поэтому они (и им подобные) должны быть разработаны самими создателями системы имитационного моделирования. Однако у конечного пользователя такой системы могут возникать и свои собственные конструкции, которые он часто использует в своей предметной области. И весьма желательно предоставлять пользователю возможность не только пользоваться готовыми компонентами, но и создавать собственные.

Применение паттерна «Observer» («Наблюдатель»)

При проектировании объектно-ориентированных приложений более гибкий и Поступление заявки
смысле повторного использования дизайн очень часто достигается за счет применения паттернов проектирования. В этой части статьи рассмотрим применение паттерна «Observer» («Наблюдатель») в контексте рассматриваемой проблемы. В качестве обсуждения полезности использования этого приема рассмотрим следующий пример.

Событие «Поступление заявки», изображенное на рис. 4, на самом деле может не являться самостоятельным событием. Очень часто такое событие служит лишь для обеспечения взаимодействия между объектом, который будет предшествовать очереди к группе устройств, и самой очередью. В самом деле, представленной на рис. 4 очереди могут предшествовать как, например, изображенный на рис. 3 источник заявок, так и другая такая же очередь.

Таким образом, необходим какой-то механизм, позволяющий событиям, возникающим в одном объекте, инициировать события другого объекта, не нарушая при этом принципа инкапсуляции. Именно эту проблему проектирования можно преодолеть, используя паттерн «Observer» («Наблюдатель») [1].

Добавим в граф событий дугу с соответствующим помеченным значением, которая представляет собой отношение наблюдения (рис. 5).

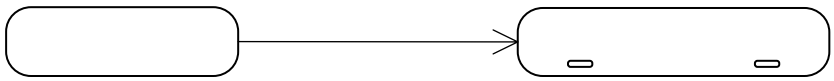

Рис. 5. Отношение наблюдения

Отношение наблюдения означает, что каждое событие «Поступление заявки» объекта, реализующего «Источник заявок», будет инициировать наступление одноименного события для объекта, реализующего «Очередь к группе устройств». Обратим внимание на то, что использование механизма композитных состояний UML позволяет представить «Очередь к группе устройств» в виде одного состояния.

Паттерн «Observer», таким образом, обеспечивает согласованное состояние взаимосвязанных объектов. Но субъектам при этом неизвестны конкретные классы наблюдателей, т.е. связи между субъектами и наблюдателями носят абстрактный характер и сведены к минимуму [2]. Следовательно, нам удалось избежать жесткой связанности классов, которая бы уменьшала возможности повторного использования.

Поступлени

Заключение

Использование принципа инкапсуляции и паттерна «Observer» («Наблюдатель») в системах имитационного моделирования позволяет быстро разрабатывать сложные модели и повышает возможности повторного использования. Согласованность взаимосвязанных объектов достигается не за счет жесткой связанности классов, а путем наблюдения за объектами.

ЛИТЕРАТУРА

1. Буч Г., Рамбо Д., Джекобсон А. Язык UML: Руководство пользователя. М.: ДМК, 2000. 432 с.
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001. 368 с.
3. Buss A. Component Based Simulation Modeling // Proceedings of the 2000 Winter Simulation Conference. Fishwick. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, 2000, U.S.A.
4. Buss A., Sanchez H. Building complex models with LEGOs // Proceedings of the 2002 Winter Simulation Conference. Naval Postgraduate School, Monterey, CA, 2002, U.S.A.