Известия Гомельского государственного университета
имени Ф. Скорины, № 6 (93), 2015

УДК 519.876.5

Архитектура инструментария имитационного моделирования сетей нового поколения

А.И. Хобня

Представлены результаты разработки оптимальной архитектуры системы имитационного моделирования сетей нового поколения. Рассматриваются основные аспекты моделирования сетей и их влияние на проектирование. Данная архитектура использована для построения проблемно-ориентированного инструментария автоматизации имитационного моделирования сетей нового поколения.
Ключевые слова: моделирование, сети нового поколения, NGN.

The results of optimal architecture design for simulation system of the next generation networks are presented. The main aspects of networks simulation and their influence on design are considered. Presented architecture design is used for building problem-oriented toolkit for automation of next generation networks simulation.
Keywords: simulation, next generation networks, NGN.

Введение. Существует ряд методов формализации функционирования моделируемой системы. Одним из них является *транзактный* метод, при котором объект моделирования представляется как сеть массового обслуживания (СМО), обрабатывающая поток входных заявок (транзактов). Для СМО характерны следующие статические элементы:

– источники заявок на обслуживание – транзактов;

– обслуживающие устройства;

– очереди для сохранения транзактов перед занятыми устройствами;

– поглотители заявок [1], [2].

Однако при моделировании сетей существует ряд определенных аспектов, затрудняющих представление моделируемой системы в виде графа основных статических элементов СМО. Например, механизмы обеспечения качества обслуживания маршрутизаторов и программируемых коммутаторов реализуют сложные алгоритмы и не могут быть представлены примитивными очередями транзактов [3]. Генерация входящего трафика может зависеть от множества факторов [4]. С другой стороны, моделируемая сеть может содержать большое количество однотипных узлов. Поэтому для имитационного моделирования сетей нового поколения архитектура системы моделирования, с одной стороны, должна позволять описывать типовые элементы моделируемой системы на высоком уровне абстракции и, с другой стороны, позволять описывать нетривиальные аспекты работы системы на низком уровне.

При моделировании сетей в качестве транзактов представляются сетевые пакеты. Отправка, обработка и уничтожение сетевых пакетов моделируется как отдельные дискретные события. Поэтому в архитектуре системы моделирования должна быть заложена возможность оптимизации работы с большим количеством событий и транзактов.

Архитектура системы моделирования. Компоненты инфраструктуры имитационного моделирования можно отнести к одному из трех уровней абстракции модели: уровню дискретных событий, уровню связанных активностей, уровню устройств. Система допускает описание различных составных частей модели на различных уровнях абстракции. Это позволяет компактно описывать типовые алгоритмы поведения модели, используя API высокого уровня и, с другой стороны, программировать нетривиальные аспекты работы модели, используя более низкие уровни абстракции системы.

На самом низком уровне абстракции любая имитационная модель в динамике представляется в качестве цепи последовательных дискретных событий. Этот уровень описывается четырьмя абстрактными концепциями: событие, активность, объект назначения и задание. Активность инкапсулирует в себе определенное действие, как правило связанное с изменением состояния модели, которое может неоднократно выполнятся в ходе работы моде-

ли. Активность выполняется в контексте объекта назначения, который передается коду активности в качестве параметра. Объект назначения может рассматриваться активностью и как источник входных параметров, определяющих ход выполнения активности, и как хранитель состояния модели, которое должно быть модифицировано в результате выполнения активности. Активности параметризируются классами объектов назначения, т. е. определенные активности могут принимать на вход только те объекты назначения, которые поддерживают определенный интерфейс. Совокупность активности и конкретного объекта назначения, в контексте которого данная активность должна быть выполнена, образует задание. Задание, выполнение которого установлено на определенное модельное время, образует событие. Модельное время представляет собой виртуальное время внутри модели, которое изменяется при переходе между событиями во время прогона модели. Наступление события приводит к выполнению задания, что в свою очередь означает выполнение активности в контексте объекта назначения. Выполнение активности может порождать новые события. Таким образом, моделируемая система представляется в виде цепи последовательных дискретных событий.

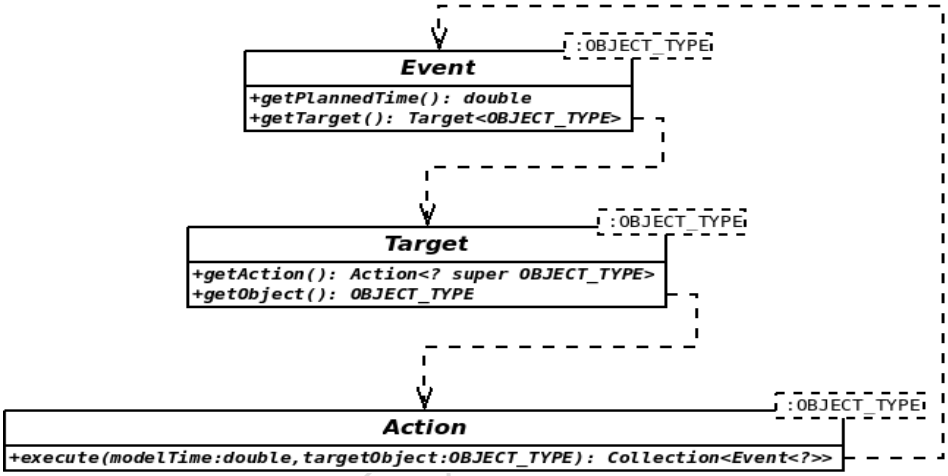

Рисунок 1 – Зависимости между основными интерфейсами

С точки зрения объектно-ориентированного программирования событие представляет собой объект, содержащий время и объект задания. Задание представляет собой объект, содержащий объект назначения и активность. Активность представляет собой объект, содержащий метод, который отвечает за ее выполнение и возвращает коллекцию событий. В описываемой инфраструктуре моделирования данные концепции представлены в виде интерфейсов. Зависимости между данными интерфейсами показаны на диаграмме классов, изображенной на рисунке 1. Интерфейс Event описывает событие и содержит два метода, возвращающих время, на которое данное событие запланировано, и задание (Target). Интерфейс Target в свою очередь содержит методы возвращающие активность (Action) и объект назначения параметризируемого типа. Интерфейс Action содержит единственный метод execute(), который принимает модельное время и объект назначения в качестве параметров. Реализации этого метода в конкретных активностях должны выполнять определенные действия в контексте объекта назначения. Метод excecute() возвращает коллекцию новых объектов событий, реализующих интерфейс Event.

Часть ядра системы моделирования, отвечающее за прогон модели, использует описанное выше API низкого уровня абстракции. На данном уровне работает механизм поиска активностей. Он реализуется при помощи управления списком будущих событий. Данный список содержит объекты будущих событий, реализующие интерфейс Event, и является всегда отсортированным по планируемому времени событий getPlannedTime(). На каждой итерации прогона модели система совершает следующие шаги:

– выбирается событие из вершины списка, новой вершиной становится следующее за ним событие;

– модельное время изменяется на планируемое время полученного события;

– выполняется задание полученного события;

– новые события, возвращенные выполненной активностью, вставляются в список таким образом, чтобы сохранить упорядоченность по планируемому времени. Для этого используется двоичный поиск по списку.

Модель заканчивает работу, когда все дискретные события исчерпаются или при наступлении специально определенных условий.

Имитационная модель может быть описана полностью на уровне дискретных событий с использованием только лишь понятий событие, активность, объект назначения и задание. Как правило, в большинстве случаев это не практично, т. к. требует от разработчика модели на низком уровне реализовывать такие действия, как порождение и управление объектами событий и заданий, кэширование, маршрутизация между активностями и т. д. Кроме того, формализация модели с использованием одних только указанных выше понятий затруднительна, поскольку данные абстракции неспособны отразить статическую структуру и структуру состояния модели. Однако описание некоторых модулей модели на данном уровне абстракции оправдано в нетривиальных случаях, когда по причине накладываемых ограничений описание данных модулей невозможно на более высоком уровне.

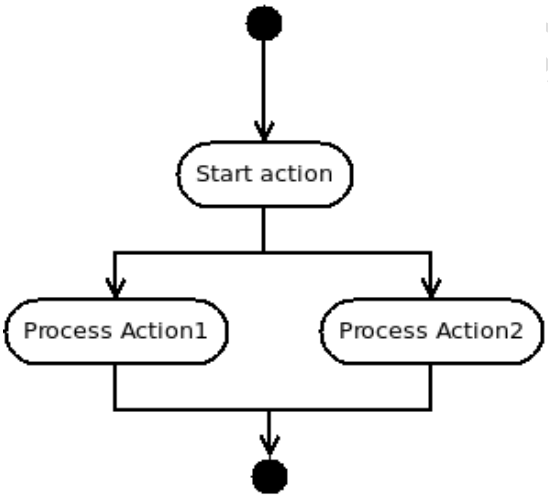

Рисунок 2 – Пример диаграммы активностей

На уровне *связанных активностей* имитационная модель представляется графами переходов между активностями. Это промежуточный несамостоятельный уровень абстракции, который может использоваться для описания модели только в совокупности с более низким или более высоким уровнем. Компоненты инфраструктуры моделирования на данном уровне позволяют решать следующие задачи:

– инкапсуляция системных функций инфраструктуры моделирования, таких как управление порождением и кешированием объектов событий, заданий и т. д., что позволяет описывать активности более компактно;

– отделение кода реализации активностей от кода связей между ними (это решение позволяет повторно использовать код активности в различных частях модели, динамически устанавливая связи между различными активностями);

– формализация и описание отдельных частей модели в виде графа переходов между активностями.

Граф переходов между активностями может быть представлен в виде UML-диаграммы активностей. Пример представлен на рисунке 2.

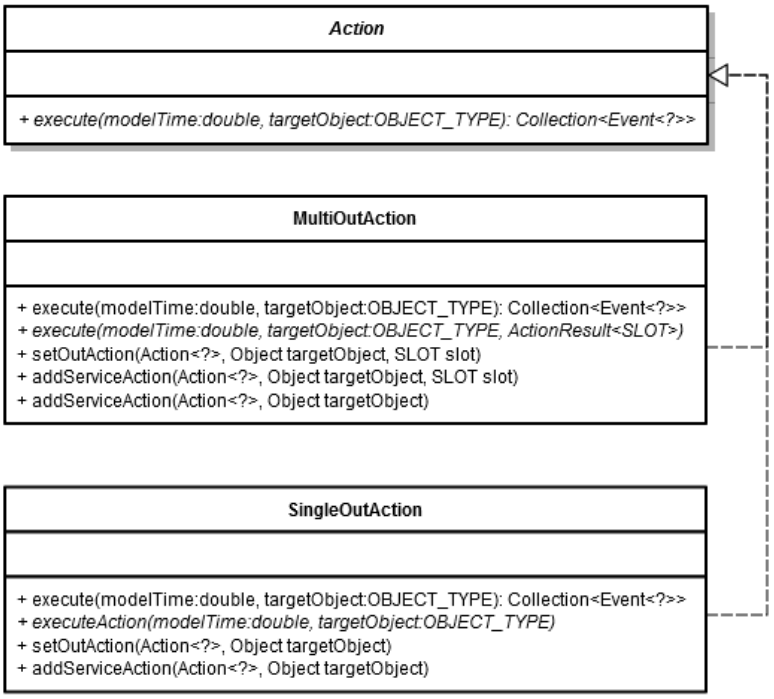

Рисунок 3 – Зависимости между классами на уровне связанных активностей

Основной концепцией на данном уровне абстракции является связываемая активность. API системы имитационного моделирования на данном уровне представлено двумя абстрактными классами SigleOutAction и MultiOutAction. SigleOutAction представляет активность, которая может быть динамически связана в последовательность с другой активностью. После выполнения данной активности управление будет передано активности, которая была связанна с ней. MultiOutAction представляет активность, которая может быть динамически связана с несколькими другими активностями. В зависимости от результата работы кода активности, управление передается одной из предварительно связанных с ней активностей. Конкретная активность, производная от абстрактного класса MultiOutAction, может иметь несколько определенных *слотов* (*выходов*). С каждым слотом может быть динамически связана только одна активность. Основной код активности должен возвращать идентификатор слота, по которому необходимо передать управление в данном случае. Таким образом, при помощи данного API определяются типовые активности, которые затем связываются в графы переходов. Графы переходов между активностями описывают модель работы механизмов обслуживания устройств моделируемой системы. Также классы SigleOutAction и MultiOutAction содержат методы setServiceAction(), позволяющие задавать служебные активности, которые будут вызваны при переходах между основными активностями механизма обслуживания.

Зависимости между классами API данного уровня абстракции системы представлены на рисунке 3. Абстрактные классы SigleOutAction и MultiOutAction содержат служебный код для порождения объектов событий, использующий пулы объектов, код передачи управления связанным активностям и иной код, обеспечивающий функционирование системы моделирования.

На *уровне устройств* имитационная модель представляется графом соединенных устройств системы массового обслуживания (СМО). На данном уровне абстракции используются следующие концепции:

– транзакты (заявки) – сообщения, движущиеся между устройствами;

– генераторы (источники транзактов);

– устройства обслуживания (УО);

– очереди (и другие типы устройств накопления транзактов) для сохранения транзактов перед занятыми устройствами;

– поглотители (уничтожители) транзактов;

Транзакт икапсулирует в себе некоторое сообщение, которое будет обработано иными устройствами модели. Транзакт может содержать состояние, которое может изменяться после обработки транзакта на различных устройствах модели.

Устройства обслуживания икапсулируют в себе определенные механизмы обслуживания транзактов, которые, в свою очередь, описываются графами переходов между активностями на более низком уровне абстракции. Обслуживание начинается с приходом транзакта на устройство и завершается после его ухода. Без транзактов код активностей устройств не получают управление. Транзакт после обслуживания на исходном устройстве может быть перемещен на следующее устройство или быть уничтожен, а также ожидать начала обслуживания или привести к аварийному завершению эксперимента из-за невозможности его обслуживания следующим устройством.

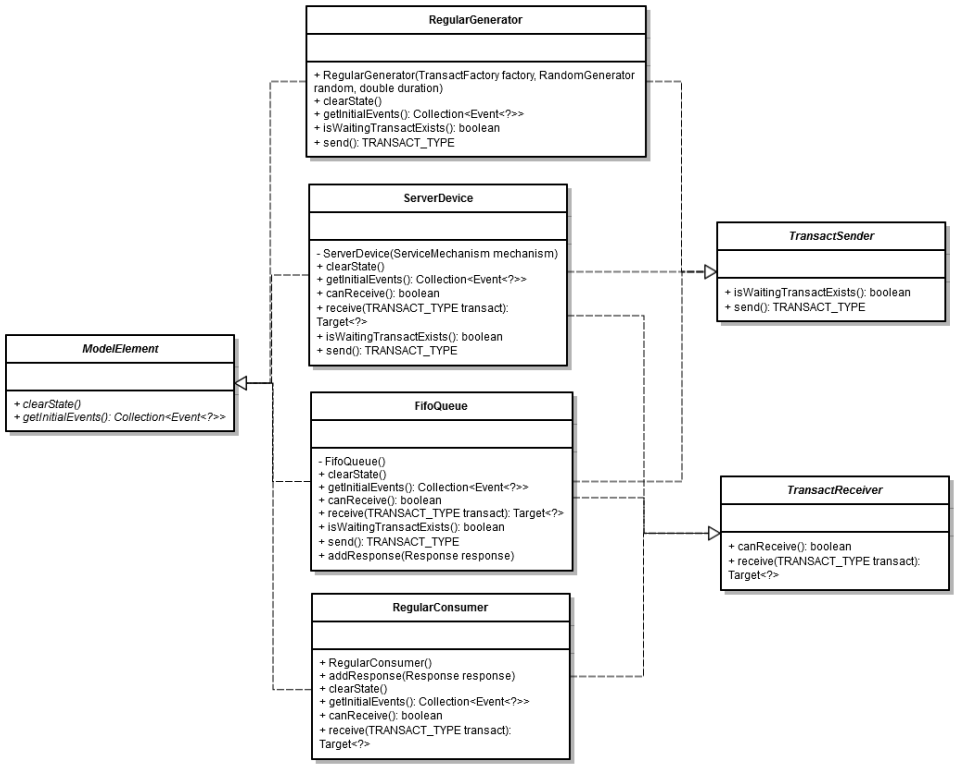

Рисунок 4 — Диаграмма основных классов и интерфейсов API уровня устройств.

Элемент имитационной модели, самостоятельно порождающий транзакты, называется *генератором*. Генераторы позволяют моделировать воздействие внешней среды на моделируемую систему путем создания потока транзактов на обслуживание. Также генераторы могут использоваться для внутренней синхронизации элементов системы или для выполнения ими служебных функций при программировании имитационной модели (отладки или мониторинга состояния имитационной модели).

Элемент имитационной модели, уничтожающий транзакты, называется *поглотителем.* Как правило, поглотители используются для вычисления откликов модели.

Устройство накопления сохраняет транзакты, если следующие за ним устройство занято обработкой иного транзакта. Устройства накопления могут быть организованы как простые очереди, работающие по принципу FIFO (First Input – First Output), так и реализовывать более сложную логику накопления, извлечения и уничтожения транзактов.

Основные классы, интерфейсы и зависимости данного уровня API представлены на рисунке 4.

Заключение. Таким образом, API системы моделирования позволяют описывать различные составные части модели на различных уровнях абстракции. Это позволяет компактно

описывать типовые алгоритмы поведения модели, используя API высокого уровня, и при необходимости программировать нетривиальные аспекты работы модели, используя API более низкого уровня абстракции. Система моделирования адаптирована для имитации большого количества дискретных событий и генерации большого потока транзактов, что является важным аспектом, т.к. имитационное моделирование сетей предполагает моделирование обработки каждого сетевого пакета. Данная архитектура использована для построения проблемно-ориентированного инструментария автоматизации имитационного моделирования сетей нового поколения.

Литература

1. Левчук, В.Д. Программно-технологические комплексы имитации сложных дискретных систем / В.Д. Левчук, И.В. Максимей. – Гомель : ГГУ им. Ф. Скорины, 2006.– 263 с.

2. Левчук, В.Д. Базовая схема формализации системы моделирования MICIC4 / В.Д. Левчук // Проблеми програмування. – 2005. – № 1. – С. 85–96.

3. Хобня, А.И. Разработка универсальной платформы моделирования механизмов обеспечения качества обслуживания в сетях пакетной передачи данных / А.И. Хобня // Тезисы докладов XVIII Респ. научной конф. «Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях». – Гомель : ГГУ им. Ф. Скорины, 2015.

4. Демиденко, О.М. Концептуальная модель генерации VoIP трафика в сети NGN / О.М. Демиденко, А.И. Хобня // Известия Гомельского государственного университета им. Ф. Скорины. – 2014. – № 6 (87). – С. 117–122.

5. Кулинченко, В.Н., Демиденко, О.М., Чечет, П.Л. Об одном подходе к определению попускной способности каналов ЛВС по протоколам TCP/IP, ICMP и UDP / В.Н. Кулинченко, О.М. Демиденко, П.Л. Чечет // Проблемы физики, математики и техники. – 2014.– № 4 (21). – С. 1–3.

Гомельский государственный
университет им. Ф. Скорины

Поступила в редакцию 25.06.2015