queues, об'єкти яких здатні накопичувати статистичну інформацію та забезпечувати динамічну індикацію свого стану у вигляді часових діаграм.

До складу фреймворка входять також компоненти, що дозволяють автоматизувати проведення багаторівневих одно факторних експериментів з моделями та виконувати дисперсійний та регресійний аналіз отриманих результатів. Компоненти фреймворка надають також можливість досліджувати перехідні процеси у моделях.

Запропоновано шаблон проектування моделі, який узагальнює багаторічний досвід роботи.

Використання фреймворка у курсі моделювання надає можливість студентам не тільки отримати знання з цієї дисципліни, але й познайомитися з механізмами реалізації проблем побудови імітаційних моделей та закріпити знання та навички з курсу ООП.

Литература

УДК 004.031.6

ВЕРИФИКАЦИЯ МОДЕЛЕЙ ОБЬЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

И.В.Богдан

Черниговский государственный технологический университет, Украина

Прежде, чем переходить к непосредственной разработке программного обеспечения необходимо выполнить верификацию модели данного объектно-ориентированного программного обеспечения. Модель состоит из множества UML-диаграмм, среди которых чаще всего создаваемыми являются такие виды, как диаграмма вариантов использования, диаграмма классов, диаграмма состояний и диаграмма последовательностей Верификация представляет собой процесс достижения гарантии того, что верифицируемая модель соответствует требованиям и удовлетворяет проектным спецификациям и стандартам [1].

Верификация модели начинается с верификации диаграмм вариантов использования. Так как варианты использования, которые описаны на диаграммах вариантов использования, не являются представлениями программного обеспечения, а представляют собой требования, которым должно соответствовать программное обеспечение, то чаще всего варианты использования формулируются на естественном языке. В результате верификация диаграмм вариантов использования фактически сводится к проверке синтаксиса.

На следующем этапе происходит верификация диаграмм классов. Верификация классов охватывает все виды деятельности, ассоциированные с проверкой реализации класса на точное соответствие спецификации этого класса [2]. Если реализация корректна, то каждый экземпляр этого класса ведет себя подобающим образом. Верификация классов производится пу-

тем построения экземпляров этих классов и анализа поведения этих экземпляров. В результате, одним из наиболее популярных подходов к верификации классов является верификация путем разработки драйвера, который создает экземпляры классов и окружает эти экземпляры соответствующей средой, то есть экземплярами тех классов, с которыми взаимодействует данный класс и которые по умолчанию считаются правильными.

Далее выполняется верификация диаграмм состояний. Каждая диаграмма состояний представляет некоторый автомат, описывающий поведение отдельного объекта в форме последовательности состояний, которые охватывают все этапы его жизненного цикла, начиная от создания объекта и заканчивая его уничтожением. Таким образом, верификация диаграмм состояний происходит путем проверки тех условий, которые вытекают с теории автоматов и специальной семантики языка UML.

На следующем этапе происходит верификация диаграмм последовательностей, которая выполняется, прежде всего, для того, чтоб убедиться, что происходит правильный обмен сообщениями между объектами, классы которых уже прошли верификацию ранее. Одним из наиболее распространенных и простых подходов к верификации диаграмм последовательностей является верификация путем анализа протоколов. По мере того, как некоторый объект вступает во взаимодействие с другими объектами, увеличивается количество сообщений, которые он получает. Все эти сообщения упорядочиваются в определенную последовательность. Верификация по протоколу проверяет реализацию на соответствие этой последовательности. Различные протоколы, в которых принимает участие тот или иной объект, могут быть логически выведены из пред- и постусловий, регламентирующих выполнение отдельных операций, объявленных в классе этого объекта. Идентифицирующая последовательность вызовов методов, в которую объединяются методы, чьи постусловия удовлетворяют предусловиям следующего метода, образуют протокол.

Верификация модели объектно-ориентированного программного обеспечения является важнейшим этапом его создания. Только прошедшее верификацию всеми описанными методами программное обеспечение можно считать готовым к разработке.

Литература

1. Синицын С.В., Налютин Н.Ю. Верификация программного обеспечения: Курс лекций. - М.: МИФИ (ГУ), 2006. - 158 с.
2. Zhou X. Auto-generation of Class Diagram from Free-text Functional Specificftions and Domain On-tology / X. Zhou, N. Zhou. – 20 p.