Застосувавши аналогічний підхід для отримання кількісних характеристик ризиків програмного проекту, визначених в [3], ми отримуємо можливість удосконалити ймовірністну модель життєвого циклу розробки програмного забезпечення, запропоновану в [2].

Література

1. Project Management Body of Knowledge (PMBOK), PMI Standard Committee
2. Мандрикова Л.В., Манжос Ю.С., Хоменко В.В. Метод идентификации рисков программного проекта на основе вероятностного подхода. // Радіоелектронні і комп'ютерні системи, 2009. – №7(41). – С.207-211.
3. Software Engineering Body of Knowledge (SWEBOK), IEEE, 2004.

УДК 681.3

НАВЧАЛЬНО-МЕТОДИЧНИЙ ПРОГРАМНИЙ КОМПЛЕКС З ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

П.Г.Бивойно
Чернігівський державний технологічний університет

На кафедрі інформаційних та комп'ютерних систем ЧДТУ створено і впроваджено в навчальний процес програмний комплекс, що використовується в процесі вивчення дисципліни «Моделювання». Комплекс доступний студентам у вигляді .jar файлу і може використовуватися під час самостійної роботи по вивченню курсу, у процесі виконання лабораторних робіт і під час модульного контролю.

Комплекс вирішує такі завдання:

- надання студентам доступу до методичних матеріалів, що структуровані по темам та підрозділам тем теоретичної частини курсу;
- розміщення інструкцій до виконання лабораторних робіт;
- доступ до програмних засобів, що використовуються під час виконання лабораторних робіт;
- можливість перевірити набуті теоретичні знання та практичні навички за допомогою тестів.

Комплекс охоплює такі розділи курсу «Моделювання»:

- генератори випадкових чисел. Методи генерації послідовностей випадкових чисел та методи їх дослідження;
- принципи побудови моделей паралельних процесів, що одночасно функціонують у часі, та програмна реалізація механізмів синхронізації псевдопаралельних процесів;
- принципи побудови моделей СМО, агентних та комбінованих моделей на основі фреймворку Simulation Java;

- одно рівневі та багаторівневі однофакторні експерименти з моделями та дисперсійний і регресійний аналіз отриманих результатів, з використання засобів автоматизації експерименту, що надає фреймворк Simulation Java;
- дослідження перехідних процесів у моделях засобами фреймворку Simulation Java;
- організація модельних експериментів з метою пошуку екстремуму.

Особливість викладання дисципліни «Моделювання» на кафедрі полягає у тому, що вона тісно пов'язується з дисципліною «Об'єктно-орієнтоване програмування». Тому у курсі в якості прикладу інструментального засобу для моделювання використовується фреймворк Simulation-Java, що був створений на кафедрі спеціально для курсу моделювання. На основі цього фреймворка побудовано застосування для виконання лабораторних робіт і тестові завдання комплексу. Фреймворк являє собою сукупність пакетів, що містять класи, які вирішують практично усі загальні питання, пов'язані з моделювання систем масового обслуговування та побудовою агентних моделей.

У фреймворці реалізовано поняття «активний об'єкт». Таким об'єктом може бути обслуговуючий пристрій, агент або навіть трансакція, якщо вона має власну поведінку у часі. Для створення таких об'єктів програміст має створити клас, що успадковує абстрактний клас process. Actor фреймворка, і у цьому класі реалізувати правила дії об'єкта у методі rule(). Затримки виконання правил дії на деякий час реалізуються за допомогою методу holdForTime(double). Затримки до виконання умови реалізуються за допомогою методу waitForCondition(IWaitCondition). IWaitCondition – це інтерфейс, що передбачає реалізацію методу boolean testConition(), де описується очікувана умова. Можна використовувати і комбінований метод waitForConditionOr HoldForTime (IWaitCondition,double), який забезпечує затримку до виконання умови, але не більше ніж на заданий час.

Синхронізацію правил дії активних об'єктів у модельному часі забезпечує об'єкт класу process.Dispatcher. У разі необхідності цей об'єкт може сформувати протокол роботи моделі у часі.

Для генерації випадкових величин із різними законами розподілу використовуються класи пакету rnd. Є також візуальний компонент, що дозволяє налаштувати потрібний закон розподілу.

Для накопичення статистичної інформації та її обробки можна використовувати класи пакету stat, які також надають можливість отримати результати обробки у текстовому та графічному вигляді.

Візуалізацію роботи моделі та графічне відображення результатів моделювання забезпечують класи пакету paint.

У разі моделювання систем масового обслуговування для моделювання черг та накопичувачів трансакцій можна використовувати класи пакету

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-