УДК 004.4'242

В.Г. Иванов, А.М. Ныцик

Харьковский национальный университет радиоэлектроники, Харьков

РАЗРАБОТКА СРЕДЫ ДОРЕАЛИЗАЦИОННОГО МОДЕЛИРОВАНИЯ И ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Проанализированы классификация ошибок программного обеспечения (ПО), способы контроля качества ПО. В результате их анализа предложена концепция среды дореализационного моделирования и тестирования, которая должна помочь предупредить возникновение определенных типов ошибок на этапах разработки ПО до этапа реализации (анализ, проектирование). Предложена структура среды, принцип ее работы, взаимодействие с пользователем и другими приложениями, которые автоматизируют процесс разработки ПО. Обоснован выбор критериев тестирования созданной модели.

Ключевые слова: *программное обеспечение (ПО), качество программного обеспечения, анализ, проектирование, тестирование, моделирование, реализация, имплементация, UML, E-сети, фишка, переход.*

Введение

Экономический и функциональный эффект от применения программного обеспечения (ПО) напрямую зависит от его качества. Качество ПО регламентируется группой стандартов ISO 9126. В стандартах качество ПО определено в виде всей совокупности характеристик, позволяющих удовлетворить потребности всех заинтересованных лиц. При рассмотрении качества ПО различаются понятия внутреннего качества, связанного с характеристиками самого ПО, без учета его поведения, внешнего качества, характеризующего ПО с точки зрения его поведения в системе, и качество ПО при использовании в различных сценариях работы. Кроме того, на создание качественного ПО существенное влияние оказывает качество технологических процессов его разработки. [1]

Анализ современных методов обеспечения качества ПО. В современных условиях особое внимание уделяется созданию систем, обеспечивающих высокое качество ПО на основе системного объектно-ориентированного подхода, позволяющих проводить оценки на ранних этапах проектирования [2].

В соответствии с классификацией программных ошибок, в зависимости от их места в жизненном цикле ПС можно выделить следующие группы:

– ошибки при постановке задачи и при составлении ТЗ;
– ошибки проектирования;
– ошибки кодирования;
– ошибки отладки и тестирования;
– ошибки и недочеты в документации на ПО;
– ошибки на этапе эксплуатации и модернизации [3].

Из всех перечисленных групп ошибок, в большинстве случаев, самыми серьезными являются те, которые были допущены на ранних стадиях разработки ПО. Это объясняется тем, что поздние этапы итерационного процесса разработки напрямую зависят от более ранних. Таким образом, ошибка, возникшая на этапе постановки задачи или проектирования, скорее всего, нанесет больший убыток, чем ошибка, допущенная на этапе кодирования. Также возможно своего рода наложение: когда в процессе разработки на ошибки одного типа накладываются ошибки другого типа, тем самым, усложняя процесс их установления, локализации и устранения.

Для снижения затрат на исправление ошибок и на разработку ПС в целом зачастую прибегают к:

– выбору наиболее подходящей модели разработки ПС;
– увеличению затрат на проектирование, системный анализ и анализ требований;
– использованию модулей уже готовых решений;
– внедрению в процесс разработки ПО средств автоматизации (проектирования, кодирования и т.п.).

Но даже с применением упомянутых средств и методов избежать всех ошибок невозможно. Усугубляет ситуацию также тот факт, что наиболее широко применяемый метод контроля качества – тестирование, осуществляется на последних стадиях жизненного цикла ПО. Конечно, в случае использования различных моделей жизненного цикла ПО, отличных от классической, можно уменьшить объем тестируемого функционала, разбив его на несколько групп и выполняя их попеременно с другими этапами процесса разработки (спиральная модель). Однако и такие модели имеют свои недостатки, связанные с трудностями контроля и управления временем разработки, повышением требовательности к заказчику и т.п. [4].

Целью настоящей работы является изложение подхода, позволяющего совместить преимущества языка унифицированного объектно-ориентированного описания программ с системой имитационного моделирования на основе E-сетей для оценки корректности и качества создаваемого или уже имеющегося программного обеспечения.

Постановка задачи. Для разграничения программных ошибок разных типов (в соответствии с

классификацией, приведенной в предыдущем пункте) и исправления их на соответствующих этапах разработки ПО предлагается использовать среду, которая позволит построить модель разрабатываемого ПО и проводить ее тестирование. Такой вариант решения задачи тестирования ПО вполне актуален в связи с широким распространением унифицированного языка объектно-ориентированного моделирования UML. UML может применяться начиная с этапа системного анализа и заканчивая этапом кодирования, используя и модифицируя одни и те же модели разрабатываемой системы [5]. Следовательно, имея некую базовую модель ПС на этапе проектирования можно провести ее моделирование и тестирование, чтобы определить допущенные ошибки.

Предлагаемая среда позволит моделировать и тестировать систему на заданном уровне абстракции до этапа кодирования. Тестирование должно давать возможность оценивать качество потока управления и потока данных модели.

Таким образом, получаемая среда должна включать в себя элементы сред моделирования и тестирования, быть интегрированной со средой разработки (проектирования) для того, чтобы использовать модели имплементируемой системы.

Содержание среды дореализационного моделирования и тестирования

Как уже говорилось ранее, среда дореализационного моделирования и тестирования должна работать с моделью системы, представленной в форме совокупности UML диаграмм. В большинстве случаев для полного представления системы на определенном уровне абстракции достаточно использовать три типа диаграмм (моделей):

- классов;
- состояния;
- взаимодействия [5].

Каждый тип диаграмм описывает отдельную сторону системы. Модель классов представляет статические, структурные аспекты системы, связанные с данными. Модель состояний представляет временные, поведенческие, управленческие аспекты системы. Модель взаимодействия представляет кооперацию отдельных объектов, другими словами, все аспекты системы, связанные с взаимодействиями.

Также на вход среды должны поступать параметры моделирования и (или) тестирования имеющейся модели, чтобы на выходе получить соответствующие результаты.

Основываясь на сказанном выше, на рис. 1 представим среду дореализационного моделирования и тестирования в виде «черного ящика».

Получив на вход совокупность UML диаграмм, описывающих систему с единых позиций, среда должна объединить их в одно целое, представив в виде формальной модели, подходящей для проведения моделирования и тестирования.

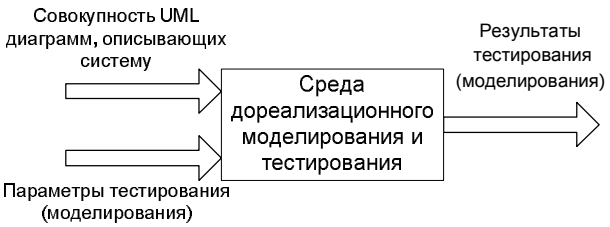

Рис. 1. Представление среды дореализационного моделирования и тестирования в виде «черного ящика»

В качестве внутреннего представления модели в системе предлагается использовать аппарат оценочных сетей.

Оценочные сети (Е-сети) – одно из наиболее мощных расширений аппарата сетей Петри. В отличие от сетей Петри, в Е-сетях [6, 7]:

– имеются несколько типов вершин-позиций: простые позиции, позиции-очереди, разрешающие позиции;

– фишки (метки) могут снабжаться набором признаков (атрибутов);

– с каждым переходом может быть связана ненулевая задержка и функция преобразования атрибутов фишек;

– введены дополнительные виды вершин-переходов;

– в любую позицию может входить не более одной дуги и выходить также не более одной.

В связи с этим, любой переход может быть описан тройкой параметров (1):

$$d_J = \left(S, t(d_J), p(d_J)\right), \qquad (1)$$

где S – тип перехода; $t(d_J)$ – функция задержки; $p(d_J)$ – функция преобразования атрибутов.

Аппарат Е-сетей полностью формализован и обладает следующими свойствами:

– позволяет отслеживать статические и динамические свойства исследуемого объекта;

– описывает дискретные системы;

– описывает системы с глубоким параллелизмом;

– позволяет работать с асинхронными информационными потоками;

– позволяет проводить стохастическое событийное моделирование;

– представляет объект в трех формах: двудольного ориентированного мультиграфа, аналитической и матричной [6, 7].

Из модели системы, представленной в виде оценочной сети, можно достаточно легко получить управляющий граф программы (УГП), так как модель Е-сети содержит все параметры, необходимые для построения УГП, а именно: начальную и конечную вершины, направленность передачи управления, множество управлений и множество операторов.

Особенности тестирования в среде дореализационного моделирования и тестирования

На рис. 2 представлена классификация основных критериев тестирования [8].

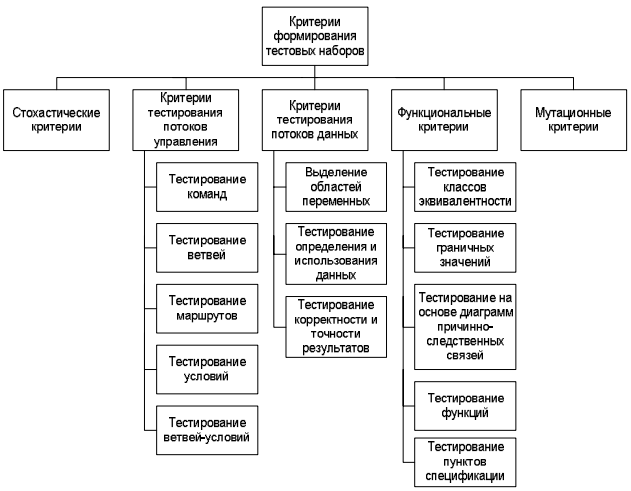

Рис. 2. Классификация критериев тестирования

Рассмотрим каждый из основных критериев тестирования.

Стохастическое тестирование основано на генерации тестовых наборов случайным образом. Стохастическое тестирование выполнимо, если удаётся автоматически и независимым образом определить эталонное множество выходных данных или экспертно указать его распределение.

Критерии тестирования потоков управления (структурные критерии) предполагают тестирование реализации, причём тестовая оснастка реализуется в теле проверяемого элемента программной системы. Структурное тестирование производится по управляющему графу программы. Выделяют ряд частных критериев, используемых при тестировании потоков управления (тестирование команд, ветвей, маршрутов).

Задача тестирования потока данных состоит в анализе корректности обработки данных выполняемых программой (тестирование областей переменных, тестирование корректности определения и использования данных, тестирование корректности обработки каждой переменной и точности результатов вычислений). Критерий тестирования потоков данных во многом пересекается со структурным критерием и осуществляется по информационному графу (ИГ). Под ИГ понимается УГП, который дополнен потоком данных участвующих в вычислениях.

Тестирование в соответствии с функциональными критериями, предполагает осуществление процесса тестирования по принципу “чёрного ящика”. При этом, неизвестна структура программы, недоступны исходные коды, однако известна спецификация программного продукта. Функциональное тестирование основано на тестировании спецификации программы. К частным функциональным критериям относятся: тестирование классов эквивалентности, тестирование граничных значений, тестирование на основе диаграмм причинно-следственных связей, тестирование функций и тестирование пунктов спецификаций.

Мутационный критерий основывается на искусственном внесении ошибок в программу. В программу вносятся мелкие ошибки (мутации). Программы, отличающиеся от исходных программ, искусственно внесёнными ошибками называют мутантами. Если сформированное множество тестовых наборов выявляет все мутации во всех мутантах, то оно соответствует мутационному критерию. Если тестирование исходной программы на заданном множестве тестовых наборов не выявило ошибок, то программа объявляется корректной [8].

Поскольку аппарат Е-сетей позволяет получить УГП и ИГ (т.е. УГП, дополненный потоком данных), то для проведения тестирования целесообразно воспользоваться критериями тестирования потока управления и потока данных. Также эти критерии позволяют протестировать архитектуру разрабатываемого программного средства (структуру и потоки данных).

Недостаток стохастического тестирования заключается в малой вероятности получения оптимального тестового набора, то есть набора, обладающего высокой обнаруживающей способностью. Как следствие в отношении эффект/затраты он проигрывает перед критериями тестирования потоков управления и потоков данных. Но такой критерий имеет смысл использовать для построения тестовых наборов большой мощности. В случае рассматриваемой среды дореализационного моделирования и тестирования, данный критерий лучше всего использовать на заданной пользователем области входных значений, а не на всей области допустимых значений.

Функциональный критерий используется для проверки соответствия спецификации и функционала разработанной программы. В этом случае подразумевается, что тестируется готовый или почти готовый продукт. В среде дореализационного моделирования и тестирования тестированию подвергается программное средство, далекое от завершения. То есть, проверяется не так соответствие спецификации, как архитектура. Следовательно, в этом случае применение данного критерия будет слишком ограничено.

Использование мутационного критерия является достаточно трудоемкой задачей. Процессы анализа Е-сети, внесения туда необходимых мутаций, составления тестовых наборов для нахождения этих мутаций и тестирования исходной программы могут потребовать больше чем просто исполнения последовательности машинных операций, а именно потребовать привлечения разработчика не только к составлению тестового набора, но и к созданию самих тестов. В то же время, среда дореализационного моделирования и тестирования должна быть максимально автоматизи-

рована в отношении тестирования разрабатываемого программного средства. Как следствие, применение мутационного критерия, в нашем случае, может привести к лишним затратам.

Таким образом, из рассмотренных критериев тестирования для среды дореализационного моделирования и тестирования выбраны следующие:

- критерии тестирования потоков управления;
- критерии тестирования потоков данных;
- стохастические критерии.

Указанные критерии хорошо формализуются и позволяют получить качественные и количественные оценки исследуемого программного обеспечения в системе имитационного моделирования E-NetSim.[9]

Выводы

В статье были рассмотрены способы контроля качества ПО, предложено создание среды дореализационного моделирования и тестирования. Использование такой среды позволит предупредить возникновение определенных типов ошибок на ранних этапах разработки ПО до его реализации (анализ, проектирование).

Среда дореализационного моделирования и тестирования должна быть интегрированной со средой разработки (проектирования) для того, чтобы использовать модели имплементируемой системы. Совокупность входных моделей, написанная на UML, должна быть преобразована в единую формализованную внутреннюю форму (Е-сети), после чего производится ее тестирование и (или) моделирование. В качестве критериев тестирования выбраны: стохастические критерии, критерии тестирования потоков управления и тестирования потоков данных.

В перспективе предлагаемая система может стать основой для среды автоматизированного создания ПО, когда на основании разработанной модели будет автоматически генерироваться рабочее приложение.

Список литературы

1. Чумакова Т.Я. Международные стандарты и жизненные циклы программного обеспечения [Електронний ресурс] / Т.Я. Чумакова, С.М. Цыганенко // Математичні машини і системи – 2009. – № 3 – С. 144-150. – Режим доступу до документу: http://www.nbuv.gov.ua/portal/natural/mms/2009_3/03_2009_Chumakova.pdf.

2. Липаев В.В. Методы обеспечения качества крупномасштабных программных средств / В.В. Липаев. – М.: СИНТЕГ, 2003. – 520 с. – (Серия «Управление качеством»).

3. Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / Сэм Канер, Джек Фолк, Енг Кек Нгуен. – М.: ДиаСофт, 2001. – 544 с.

4. Орлов С.А. Технологии разработки программного обеспечения / С.А. Орлов. – СПб.: Питер, 2002. – 464 с.

5. Рамбо Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб.: Питер, 2007. – 544 с.

6. Питерсон Дж. Теория сетей Петри и моделирования систем / Дж. Питерсон. – М.: Мир, 1984. – 264 с.

7. Котов В.Е. Сети Петри / В.Е. Котов – М.: Наука. Главная редакция физико-математической литературы, 1984. – 160 с.

8. Балыков Е. Тестирование программных средств [Електронний ресурс] / Е. Балыков, В. Царев // RSDN Magazine – 2006. – 4 – Режим доступу до документу: http://www.rsdn.ru/article/testing/SoftwareTesting.xml.

9. Представление имитационных моделей дискретно – событийных систем с помощью CASE средства E-NetSim / В.Г.Иванов, И.В. Шевченко // Бионика интеллекта: научно-технич. журнал. – 2009. – № 1 (70). – С. 44-49.

Поступила в редколлегию 9.12.2010

Рецензент: д-р техн. наук, проф. И.В.Гребенник, Харьковский национальный университет радиоэлектроники, Харьков.

РОЗРОБКА СЕРЕДОВИЩА ДОРЕАЛІЗАЦІЙНОГО МОДЕЛЮВАННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В.Г. Іванов, А.М. Ницик

Розглянуті класифікація помилок ПЗ, способи контролю якості ПЗ. В результаті їх аналізу запропоновано створення середовища дореалізаційного моделювання та тестування, яке повинне допомогти попередити появу певного типу помилок на етапах розробки ПЗ до етапу реалізації (аналіз, проектування). Запропонована структура середовища, принцип її роботи, взаємодія з користувачем й іншими програмами, які автоматизують процес розробки ПЗ. Розглянутий та обґрунтований вибір критеріїв тестування створеної моделі.

Ключові слова: *програмне забезпечення, якість програмного забезпечення, аналіз, проектування, тестування, моделювання, реалізація, імплементація, UML, Е-мережі, фішка, перехід.*

DEVELOPMENT OF MODELLING AND TESTING TILL-IMPLEMENTATION ENVIRONMENT FOR SOFTWARE

V.G. Ivanov, A.M. Nytsyk

Software faults classification, methods of quality control are considered. As a result of their analysis creation of modeling and testing till-implementation environment is offered. It should help to warn an occurrence of certain types of errors at development software cycle till a realization stage (the analysis, designing). The structure of environment, a principle of its work, interaction with the user and other applications which automate software development process are offered. The choice of criteria of testing of the created model is considered and proved.

Keywords: *the software, quality of the software, the analysis, designing, testing, modeling, realization, implementation, UML, E-nets, a counter, transition.*