УДК 004.415.532

И. С. СКАРГА-БАНДУРОВА, Я. П. КОВАЛЕНКО

Технологический институт Восточноукраинского национального университета им. В. Даля (г. Северодонецк), Украина

ПОВЫШЕНИЕ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМ КРИТИЧЕСКОГО НАЗНАЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ СТАТИЧЕСКОГО АНАЛИЗА

В статье обсуждается роль тестирования в процессе разработки программного обеспечения критических систем. Представлен обзор методов тестирования программного обеспечения. Исследование сосредоточено на двух взаимосвязанных аспектах: динамическое тестирование и статический анализ кода. Для тестирования программных модулей компьютерных систем с повышенными требованиями к безопасности предлагается на этапе разработки применять статический анализ в качестве дополнительного метода к стандартным динамическим испытаниям. Предложена структурная схема программного интерпретатора, реализующего технологию статического анализа, описаны принципы его работы, внутренняя логическая структура и подходы к реализации.

Ключевые слова: *критическая система, тестирование, метод статического анализа, программный интерпретатор, функциональный блок.*

Введение

Тестирование является одним из наиболее устоявшихся способов обеспечения качества программных компонент и входит в комплекс требований к обеспечению безопасности систем критического назначения. Но, как одна из основных фаз разработки программного обеспечения, тестирование характеризуется большим вкладом в суммарную трудоемкость и составляет 40-45% от общего времени разработки [1]. В процессе сопровождения программного продукта при исправлении ошибок и внесении усовершенствований значительная часть затрат также приходится на тестирование. Программное обеспечение тестируют и исправляют практически на каждом этапе его жизненного цикла.

Цель статьи – рассмотреть методы тестирования программного обеспечения критических систем и предложить способ повышения качества и безопасности программного обеспечения с использованием методов статического анализа.

1. Базовые стратегии тестирования

В процессе разработки программного обеспечения систем, ориентированных на безопасность, традиционно выделяют две базовые стратегии - тестирование «белого ящика» и тестирование «черного ящика» [2]. По своей сути тестирование «черного ящика» - это функциональное тестирование, а тестирование «белого ящика» - структурное тестирование.

Стратегия тестирования «белого ящика» предполагает тестирование кода на предмет логики работы программы и корректности ее работы с точки зрения компилятора, для которого она писалась. Данная стратегия тестирования позволяет проверить внутреннюю структуру программы, т.е. тестировщик получает тестовые данные путем анализа логики работы программы.

Данные базовые стратегии основываются на методе динамического тестирования. Он предполагает, что исходный код исполняется, а разница между стратегиями состоит лишь в той информации, которой владеет тестировщик.

В настоящий момент, в большинстве случаев, тестирование функционального программного обеспечения компьютерных систем высокой надёжности, в частности систем микропроцессорной централизации, выполняется на двух описанных выше базовых стратегиях, организованных на методе динамического тестирования.

Главным системным недостатком представленных методов является необходимость использования специализированного комплекса технических средств и программных инструментальных средств для проведения тестирования, что затрудняет разработку и отладку специализированных прикладных программ и дополнительных функциональных блоков непосредственно на рабочем месте.

Вместе с тем, среди методов направленных на обеспечение качества программных продуктов, при-

© И. С. Скарга-Бандурова, Я. П. Коваленко

сутствует метод статического тестирования. Статический анализ является особенно мощным средством, когда речь заходит о поиске ошибок безопасности. Инструменты статического анализа используются в качестве дополнительного метода к обычным динамическим испытаниям с целью получения гарантий качества и безопасности разработки программного обеспечения критических систем, в том числе, для поиска кода, потенциально содержащего уязвимости [3].

При статическом тестировании программный код не выполняется, анализ программ проводится на основе исходного кода, который вычитывается вручную, либо анализируется специальными инструментами. При этом проверяется корректность построения элементов программы и их взаимодействие друг с другом, выполняется проверка правил структурного построения программы и обработки данных [4].

Тестированию должны быть подвергнуты:

– внутренние структуры данных;
– ветвления;
– обработки ошибочных ситуаций;
– интерфейс модуля;
– граничные условия.

Для оценки результата тестирования используются следующие критерии:

– покрытие операторов (подразумевает выполнение каждого оператора программы, по крайней мере, один раз);

– покрытие решений (подразумевает проверку всех условий в программе при истинных, так и при ложных значениях);

– покрытие условий (подразумевает проверку всех возможных результатов каждого условия в решении).

Применение статического тестирования достаточно эффективно. Для типичных программ, по данным фирмы IBM, можно находить от 30% до 80% ошибок логического проектирования и кодирования [5,6]. Данный метод способствует существенному повышению производительности и надежности программ, позволяет раньше обнаружить ошибки, а значит уменьшить стоимость исправления. Когда программист сам находит свою ошибку и исправляет ее, это не влечет больших затрат необходимых для устранения данной ошибки на более поздних этапах.

2. Программный интерпретатор

В основе предлагаемого подхода к тестированию функционального программного обеспечения систем микропроцессорной централизации лежит стратегия тестирования «белого ящика» основанная на методе статического тестирования отдельных программных модулей.

Для решения отмеченных проблем и упрощения отладки исходных кодов предлагается использовать специализированный программный интерпретатор, основанный на методе статического тестирования, который выполняет покомандный анализ, обработку и выполнение заданного исходного кода, а также отображение значений параметров во время тестирования.

Функционально интерпретатор включает в себя следующие логические единицы [7]:

– диспетчер оперативной базы данных (ДОБД);

– базу описания вызова дополнительных функциональных блоков (ДФБ);

– базу формальных параметров ДФБ;

– базу операторов секции исполнения ДФБ;

– базу групп атрибутов параметров;

– модуль визуализации;

– базу описания макрокоманд;

– модуль тестирования операторов секции исполнения ДФБ;

– диспетчер архивирования.

Структурная схема интерпретатора приведена на рис. 1.

Исходными данными для работы интерпретатора, которые анализируются с помощью ДОБД, являются:

– текст программы ДФБ;

– секции вызова прикладной программы;

– объявление групп атрибутов параметров, необходимых при описании интерфейса объявления дополнительного функционального блока.

Диспетчер оперативной базы данных распознает во входных текстах программы смысловые единицы языка, выполняет поиск операторов секции исполнения, позиций ключевых макрокоманд языка и анализ порядка следования и при их корректности, осуществляет запись данных в соответствующую базу параметров.

В качестве операндов секции исполнения могут использоваться:

– константы;

– формальные имена входных, выходных и настроечных параметров;

– рабочие переменные.

Результатом работы диспетчера являются следующие базы параметров, которые описаны ниже.

База формальных параметров ДФБ состоит из структур со следующими полями: формальное имя параметра; тип параметра; формальное имя операнда; имя атрибута; значение переменной.

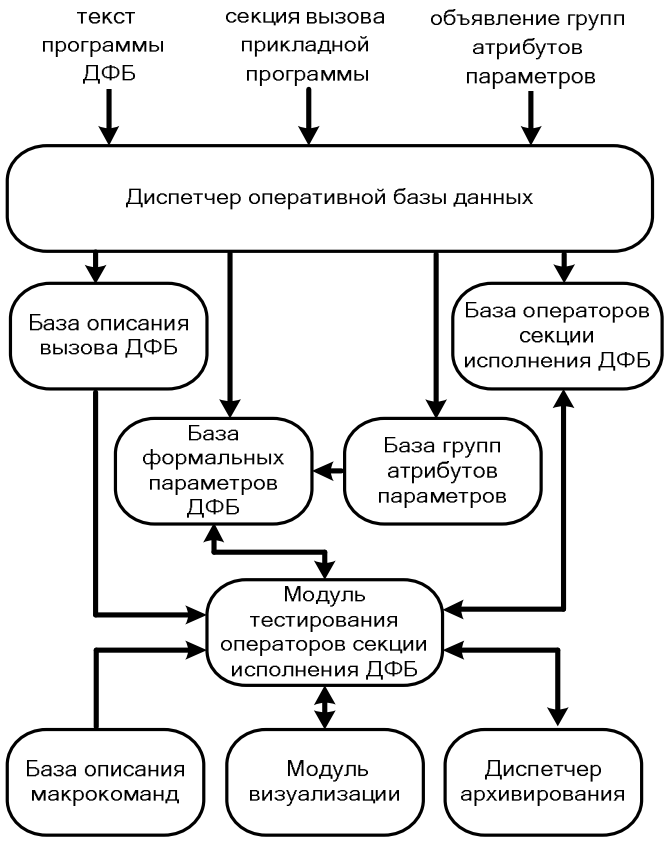

Рис. 1. Структурная схема интерпретатора

База описания вызова ДФБ состоит из структур со следующими полями: ключевое имя, указанное в вызове; формальное имя операнда; имя операнда при вызове.

База групп атрибутов параметров состоит из структур со следующими полями: имя группы; имя атрибута; тип параметра.

Сформированные базы параметров хранятся в динамической памяти программы и необходимы для выполнения дальнейшего анализа и последующей обработки модулем тестирования операторов секции исполнения ДФБ, а также предоставляются пользователю для возможности контролировать и направлять потоки данных во время проведения тестирования.

Также диспетчер оперативной базы данных выполняет функцию поиска операторов технологического языка в секции исполнения дополнительного функционального блока.

В секции исполнения дополнительного функционального блока могут использоваться:

– предопределенные функциональные блоки различных логических функций;

– операторы ветвлений.

Найденные в исходном тексте операторы записываются в отдельную базу для дальнейшего синтаксического анализа, а затем предоставляются пользователю для проведения тестирования.

Заключение

Предполагается, что качественный подход к тестированию исходного кода функционального программного обеспечения критических систем, представленного в виде функциональных блоков, на этапе разработки позволит минимизировать количество ошибок на начальной стадии разработки, сократит время дальнейшей отладки и разработки в целом.

Литература

1. Канер, С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений [Текст] : пер. с англ. / С. Канер, Дж. Фолк, Е. К. Нгуен. – К. : Диа-Софт, 2001. – 544с.

2. Michael, C. C. Risk-Based and Functional Security Testing [Электронный ресурс] / C. C. Michael, K. van Wyk, W. Radosevich. – Режим доступа: https://buildsecurityin.us-cert.gov. – 23.02.2014.

3. Livshits, B. Improving Software Security with Precise Static and Runtime Analysis : PhD dissertation [Text] / B. Livshits ; Stanford University. – USA, 2006. – 229 p.

4. Глухих, М. И. Программная инженерия. Обеспечение качества программных средств методами статического анализа. [Текст] : учеб. пособие / М. И. Глухих, В. М. Ицыксон. – СПб. : Изд-во Политехн. ун-та, 2011. – 150 с.

5. Dusting, E. Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality [Text] / E. Dustin, T. Garrett, B. Gauf. – Addison-Wesley Professional, 2009. – 368 p.

6. Котляров, В. П. Основы тестирования программного обеспечения [Текст] : учеб. пособие / В. П. Котляров, Т. В. Коликова. – М. : Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2006. – 285 с.

7. Коваленко, Я. П. Об одном подходе к функциональному тестированию программного обеспечения систем микропроцессорной централизации [Текст] / Я. П. Коваленко, И. С. Скарга-Бандурова // Перспективы взаимодействия железных дорог и промышленных предприятий: материалы III Междунар. науч.-практ. конф., Днепропетровск, 27-28 февраля 2014 г. – Д. : ДНУЖТ, 2014. – С. 44-45.

Поступила в редакцию 24.02.2014, рассмотрена на редколлегии 25.03.2014

Рецензент: д-р техн. наук, проф. Б. М. Конорев, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Харьков, Украина.

ПІДВИЩЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ КРИТИЧНОГО ПРИЗНАЧЕННЯ З ВИКОРИСТАННЯМ МЕТОДІВ СТАТИЧНОГО АНАЛІЗУ

И. С. Скарга-Бандурова, Я. П. Коваленко

У статті обговорюється роль тестування в процесі розробки програмного забезпечення критичних систем . Представлено огляд методів тестування програмного забезпечення. Дослідження зосереджено на двох взаємопов'язаних аспектах: динамічне тестування і статичний аналіз коду . Для тестування програмних модулів комп'ютерних систем з підвищеними вимогами до безпеки пропонується на етапі розробки застосовувати статичний аналіз в якості додаткового методу до стандартних динамічних випробувань . Запропоновано структурну схему програмного інтерпретатора , що реалізує технологію статичного аналізу , описані принципи його роботи , внутрішня логічна структура і підходи до реалізації.

Ключові слова: критична система, тестування, метод статичного аналізу, програмний інтерпретатор, функціональний блок

THE SOFTWARE QUALITY ENHANCEMENT FOR SAFETY-CRITICAL SYSTEMS WITH STATIC ANALYSIS TECHNIQUES

I. S. Skarga-Bandurova, Y. P. Kovalenko

The article discusses the role of software testing in a software development process for safety-critical computer systems. It focuses on two related topics: dynamic testing and static code analysis. It describes a static analysis as a complementary technique to conventional dynamic testing. The structural scheme of automated testing tool implemented on the principles of static analysis is proposed, the principles of its operation, the internal logical structure and approaches to implementation is described.

Key words: safety-critical computer system, software, testing, code review, static program analyzer, automated tool

Скарга-Бандурова Инна Сергеевна – кан. техн. наук, доцент, доцент кафедры компьютерной инженерии Технологического института Восточноукраинского наионального университета им. В. Даля, Северодонецк, Украина, email: skarga_bandurova@ukr.net.

Коваленко Ян Павлович – аспирант, Технологический институт Восточноукраинского наионального университета им. В. Даля, Северодонецк, Украина, email: czech16@email.ua.