УДК: 004.415.53(075)

ОБ ОЦЕНКЕ ЧУВСТВИТЕЛЬНОСТИ К СБОЯМ ТЕСТОВ ПРОГРАММ

Б.М. Басок[1,@], к.т.н., доцент
В.Н. Захаров[2], д.т.н., ученый секретарь
С.Л. Френкель[2], к.т.н., старший научный сотрудник

[1]Московский технологический университет (МИРЭА), Москва, 119454 Россия
[2]Федеральный исследовательский центр "Информатика и управление" РАН, Москва, 119333 Россия
Автор для переписки, e-mail: basok@mirea.ru

В статье предлагается метод оценки восприимчивости тестов программ к сбоям в окружающей их среде. Данный метод основан на статистической проверке влияния на программу ее тестов после кратковременной трансформации содержащихся в них данных. Рассматриваются некоторые примеры, а также возможности сокращения времени при реализации предлагаемого метода.

Ключевые слова: чувствительность теста, случайные сбои, классы эквивалентности отказов, статистический подход.

ABOUT ESTIMATION OF SOFTWARE TESTS SENSITIVITY TO TRANSIENT FAULTS

B.M. Basok[1,@],
V. N. Zakharov[2],
S.L. Frenkel[2]

[1]Moscow Technological University (MIREA), Moscow, 119454 Russia
[2]Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, 119333 Russia
[@]Corresponding author e-mail: basok@mirea.ru

The article suggests a method for estimation of the sensitivity of software tests to transient faults of a computer system. The method is based on statistical analysis of a program work results after performing a functional test with a submitted short term (ransient) defect, This method allows to identify tests helping to localize errors both in a program body and in its environment. It also enables to estimate the level of a program fault tolerance. In order to analyze quantitatively the sensitivity, we suggest a notion of test sensitivity rate regarding the transient faults. Examples of estimation of the sensitivity of software tests are given as well as methods of time cost reducing for the test sensitivity rate estimation.

Keywords: sensitivity of a test, random failures, equivalence classes of failures, statistical approach.

Введение

Организация тестирования различных объектов контроля, как правило, состоит из трех важнейших составляющих:

• синтез и анализ тестов;

• реализация тестовых процедур;
• разработка принципов проектирования тестопригодных объектов.

Обычно при решении этих задач считают, что возможные дефекты располагаются исключительно в самом контролируемом объекте, а среда, в которой он функционирует, находится в исправном состоянии. Однако в последнее время в связи с уменьшением проектных норм обостряется проблема случайных сбоев в современных интегральных схемах, вследствие воздействия на них одиночных частиц: протонов, нейтронов, тяжелых заряженных частиц [1, 2], а также отказов, связанных с взаимовлиянием отдельных элементов СБИС при работе на высоких частотах. В работах [1, 2] рассмотрено влияние сбоев на работу вычислительных устройств и их устранение с помощью статистического подхода [3]. При этом очевидно, что одиночные сбои влияют также на результаты выполнения программ, использующих вычислительные устройства и связанное с ними программное обеспечение. Сбои могут возникать как внутри памяти программы, так и вне её. Они могут исказить данные на внешних входах программы и содержимое программы (значения переменных, констант, кодов операций и т.д.). Кроме того, зачастую программы обнаруживают периодически возникающие нестабильные ошибки, а пользователь по результатам выполнения этих программ не может определить место возникновения данных ошибок.

Представляется целесообразным обнаружение сбоев путем создания тестов программ, разработанных для проверки их функционирования. Кроме того, причиной возникновения ошибок при выполнении программ с использованием произвольных данных могут быть самые различные отказы, в том числе, не обнаруженные при тестировании. Обнаружение ошибок в программах часто требует значительных затрат времени.

Таким образом, задача оценки чувствительности тестов программ к предполагаемым случайным сбоям оборудования является актуальной.

В настоящее время одним из распространенных методов оценки чувствительности программ к сбоям является метод внесения неисправностей (так называемый метод *Fault injection*, FI). Он предполагает искусственное внесение разного рода неисправностей для тестирования отказоустойчивости и, в частности, обработки исключений [4−7]. Однако это весьма дорогостоящая процедура, так как требует дополнительного привлечения значительного объема программных и аппаратных ресурсов [8], не используемых для решения других задач разработки программ, в частности, разработки функциональных тестов.

Цель настоящей работы – предложить и проанализировать оригинальный подход к оценке устойчивости тестов к сбоям в процессе разработки функциональных тестов без использования метода FI.

Современные методы оценки устойчивости программ к сбоям

Как уже упоминалось, одной из основных техник оценки чувствительности вычислительных систем к сбоям является техника *Fault injection* (FI), используемая для оценки надежности тестирования влияния аппаратных или системных сбоев на работу прикладных программ [4−7]. Обычно в технике FI для прикладных программ применяют технологии *instruction-by-instruction*, в которых операнды-источники случайным образом модифицируются до выполнения инструкции, а результат выполнения инструкции также случайным образом искажается, имитируя тем самым действие ошибки. При этом программный счетчик случайно модифицируется в случае, если это инструкция условного перехода (*branch Instruction*). Результирующая последовательность таких модификаций инструкций и счетчика прослеживается для определения уровня защищенности данной программы к указанным сбоям (*soft error rate, SER*).

Анализ возможных сбоев на уровне программного кода (*SW testing and fault inj*) на уровне языков высокого уровня описан в [9]. В ней рассматривали концепцию «*Comfort zone*», т.е. множество входных переменных, в пределах которой тестирование показало правильное функционирование.

Поскольку выше описанные методы требуют наличия у разработчиков достаточно дорогостоящего программного обеспечения, ниже мы рассматриваем подход, не требующий применения указанной техники.

Метод оценки чувствительности

Для оценки восприимчивости теста к кратковременным сбоям введем понятие чувствительности теста (в дальнейшем *чувствительность*). Под *чувствительностью* теста программы S будем понимать способность теста реагировать на случайные сбои в аппаратуре или в программном окружении. Примем, что программа к моменту определения чувствительности отлажена, оттестирована и все найденные в программе дефекты исправлены.

Если тест обладает низкой чувствительностью к сбоям, то следует проанализировать его влияние на выполнение программы.

Возможны два варианта: либо тест обладает низкой полнотой относительно отказов данного типа, либо программа хорошо защищена от подобных отказов. В первом случае в зависимости от результатов можно скорректировать тест, тем самым повысив его полноту; во втором − подтверждается отказоустойчивость контролируемого объекта.

Если тест чувствителен к сбоям, то, с одной стороны, его можно использовать для проверки реакции программы на некорректные данные, например, на входные данные, значения которых выходят за пределы допустимых. Кроме того, тест с высокой чувствительностью к сбоям можно использовать для выявления мест возникновения кратковременных отказов путем многократного его выполнения (зацикливания).

Понятно, что количество вероятных сбоев, воздействующих на реальную программу, может быть очень большим. Поэтому стоит ограничиться анализом влияния конечной статистической выборки [3] вносимых в программу кратковременных сбоев. Аппаратный сбой в окружении программы может породить кратный отказ в программе, однако при рассмотрении чувствительности теста ограничимся одиночными сбоями. Такое ограничение не снижает качества оценки чувствительности, поскольку опирается на известную гипотезу о редкой компенсации кратных отказов [10]. Из гипотезы следует, что тест, обнаруживающий одиночные отказы, обнаружит и любое их сочетание. В существующей практике анализа устойчивости к сбоям методом FI задают случайное изменение (одновременное внесение ошибки) операнда-источника, операнда-результата и программного счетчика, т.е. с точки зрения этой терминологии рассматривают именно кратную неисправность.

В зависимости от влияния на программу будем различать два вида сбоя аппаратных средств: в памяти программы и вне ее. Для сбоев первого типа задача определения чувствительности похожа на задачу определения полноты теста методом последовательного внесения дефектов в программу. Тогда отказ вносится в программу не перед выполнением теста, а в процессе его реализации, в случайный момент времени − для упрощения процедур внесения отказов лучше перед началом произвольно выбранного очередного набора теста. При этом возможно восстановление искаженных данных (например, при записи новых неискаженных данных в переменные). Для ускорения этого процесса целесообразно использовать метод моделирования кратных отказов, описанный в [11,12].

Рассмотрим влияние внешних по отношению к программе сбоев. Поскольку сбои данного типа могут привести к искажению входных данных программы (тестовых, в том числе), то, следовательно, они могут исказить данные теста, поступающие на входы программы при его выполнении. Будем последовательно вносить одиночные дефекты в тест в случайные моменты времени. Если при выполнении теста с внесенным в него дефектом ошибка будет обнаружена, то чувствительность теста повышается.

В качестве вносимых в тест отказов можно ввести так называемые «мягкие» отказы, которые действуют в течение небольшого промежутка времени, например, в течение времени между двумя последовательными наборами теста.

Точки модификации данных в тесте могут быть самые разные: текстовая строка, введенная через графический интерфейс, бинарные данные из файла или, например, значение поля в каком-нибудь сетевом запросе. Могут быть смоделированы также сбои через изменение управляющих элементов интерфейса пользователя, например, как кратковременную в течение одного тестового набора «инверсию» отдельного элемента управления. Если тесты автоматизированы, то ошибка вносится в соответствующий скрипт. Если тест является ручным, то в данном случае имитируется ошибка тестировщика.

Количественно *чувствительность* можно представить как выраженное в процентах отношение количества случайных сбоев, обнаруженных при выполнении теста n, к общему количеству возможных сбоев N:

$$S = \frac{n}{N} \cdot 100\%$$

Проиллюстрируем оценку чувствительности тестов к сбоям входных данных на простых примерах, когда не требуется использовать статистический подход при анализе чувствительности тестов.

Пример 1

Программа подсчитывает количество вхождений в строку заглавных букв. Программа отлажена и оттестирована. (Код символов ASCII). В качестве ошибок в данных теста будем в этом и последующих примерах рассматривать замену символа на любой другой из возможных символов с кодом ASCII.

Тест: ввел заглавную букву (например, «А») – получил результат: «1».

Всего возможных одиночных дефектов, вызванных сбоем данных, N = 255, из них выявляемых, n = 197 (все возможные символы, не являющиеся заглавными буквами). Тогда мера чувствительности теста к сбоям S_1 равна 77.3%.

Пример 2

Теперь тест для программы из Примера 1 состоит из двух символов «Аб». Результат: «1». Всего возможных одиночных дефектов, вызванных сбоем данных, $N_2 = 510$. Тест выявит $n_2 = 256$. Тогда мера чувствительности теста к сбоям S_2 равна 50%.

Пример 3

Тест состоит из двух наборов «Аб» и «вг». Результат: «1».

Всего возможных одиночных дефектов, вызванных сбоем данных, $N_3 = 1020$. Отказы держатся не более одного набора. Тест выявляет n_3 = 256 (первый набор) + 118 (второй набор) = 374. Тогда мера чувствительности теста к сбоям S_3 равна 36.7%.

Время оценки чувствительности тестов к сбоям можно значительно уменьшить, для чего достаточно разбить дефекты на входах теста на классы эквивалентности.

Группа дефектов на входах теста образуют класс эквивалентности, если результаты выполнения модернизированного теста совпадают внутри класса на всех наборах теста, начиная с номера модернизированного набора. Если результаты воздействия дефектов, образующих один класс, не будут отличаться от результатов выполнения исходного теста, то можно не выполнять программу с модифицированными тестами, содержащими отказы из данного класса. Очевидно, эти дефекты не повышают чувствительность теста.

Если результаты воздействия дефектов, образующих один класс, не будут отличаться от результатов выполнения модифицированного теста с уже проанализированным отказом из данного класса, то можно не выполнять программу с модифицированным тестом. В зависимости от того, повышается чувствительность теста при внесении проанализированного дефекта или нет, отказы из данного класса аналогично влияют на чувствительность данного теста.

Если взять Пример 1, то исходя из условий задачи ясно, что подмножество ошибок, связанных с заменой заглавной буквы на заглавную (русскую или латинскую), образуют класс экви-

валентности, результаты воздействия элементов которого совпадают с результатами исходного теста. То есть они не повышают чувствительности теста к сбоям. Все остальные отказы образуют второй класс эквивалентности и повышают чувствительность теста.

При выполнении теста после модификации его входных данных возможны следующие варианты:

- изменения не обнаружены;
- изменения обнаружены;
- программа закончилась аварийно, например, по тайм-ауту.

Первый вариант говорит о нечувствительности теста к сбоям внешнего окружения при имеющихся входных данных, второй – о чувствительности теста, третий – о низком уровне отказоустойчивости программы [13]. Если имеет место третий вариант, то это говорит, что найдена уязвимость программы, которую необходимо ликвидировать.

Если значительно превалирует второй вариант над первым, то тест весьма чувствителен к сбоям и если после его прогона возникает ошибка, то следует обратить внимание на внешнее окружение объекта тестирования. Если значительно превалирует первый вариант над вторым, то, скорее всего, тест не пригоден для выявления внешних сбоев и уязвимостей программы.

Заключение

Рассмотрен метод, позволяющий использовать тесты программ не только для выявления ошибок в данных программах, но и для регистрации кратковременных сбоев в их окружении.

Введено понятие чувствительности теста программы к кратковременным сбоям. Предложен критерий оценки чувствительности тестов к кратковременным сбоям. Предложена модель кратковременного сбоя, используемая для оценки влияния сбоев на выполнение программы на заданном тесте. На простых примерах проиллюстрированы процедуры вычисления оценки чувствительности тестов к кратковременным сбоям. Рассмотрен способ сокращения времени оценки чувствительности тестов к сбоям.

Литература:

1. Осипенко П.Н., Левадский С.А., Антонов А.А. Исследование Архитектурной чувствительности к сбоям с использованием метода статистического внесения сбоев // Программные продукты и системы. 2010. № 4. С. 10–14.

2. Осипенко П.Н. Одиночные сбои – вызов для современных процессоров // Электронные компоненты. 2010. № 1. С. 66–69.

3. Гробман Д.М. Статистический способ определения полноты тестов // Тезисы докладов IV Всесоюзного совещания по технической диагностике. Черкассы. 1979. С. 9–11.

4. Lee C., Potkonjak M., Mangione-Smith W.H. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems // In: Proc. of the 30th annual IEEE/ACM Int'l Symp. on Microarchitecture, Dec. 1997. P. 292–303.

5. Borchert C., Schirmeier, Spinczyk O. Protecting the dynamic dispatch in C++ by dependability aspects // In: 1st GI W'shop on SWBased Methods for Robust Embedded Sys. (SOBRES '12), ser. Lecture Notes in Informatics. German Society of Informatics, Sep. 2012. P. 521–535.

6. Cho H., Mirkhani S., Cher C.-Y., Abraham J.A., Mitra S. Quantitative evaluation of soft error injection techniques for robust system design // I:n 50th Design Automation Conf. (DAC '13). IEEE, May 2013. P. 1–10.

7. Wei J., Thomas A., Li G., Pattabiraman K. Quantifying the accuracy of high-level fault injection techniques for hardware faults // In: 44th IEEE/IFIP Int. Conf. on Dep. Sys. & Netw. (DSN '14). IEEE, Jun. 13. 2014. P. 375–382.

8. Hoijin Yoon, Byoungju Choi. Component customization testing technique using fault injection technique and mutation test criteria // Proc. of the 1st Workshop on Mutation Analysis (MUTATION'00) San Jose, California, 6-7 October 2001. P. 71–78.

9. Reynolds J.C., Just J., Clough L., Maglich R. On-Line Intrusion Detection and Attack Prevention Using Diversity, Generate-and-Test, and Generalization // In: Proc. of the 36th Annual Hawaii International Conference on System Sciences (HICSS), January 2003. P. 1–8.

10. Гробман Д.М. Программный контроль и диагностика неисправностей ЦВМ // Диагностика неисправностей вычислительных машин. М.: Наука, 1965. С. 7–22.

11. Басок Б.М., Гречин А.А. О едином подходе при анализе тестов дискретных устройств и программ // Вопросы радиоэлектроники. Серия ЭВТ. 2010. Вып. 3. С. 140–145.

12. Басок Б.М., Гречин А.А. Об усовершенствовании статистического метода оценки полноты тестов программ и устройств // Инструменты и методы анализа программ. Труды Международной научно-практической конференции, Кострома, 2013. С. 40–45.

13. Фаззинг, фаззить, фаззер: ищем уязвимости в программах, сетевых сервисах, драйверах. Журнал «Хакер». Июль 19, 2010. [Электронный ресурс] URL: https://xakep.ru/2010/07/19/52726/