DOI 10.37882/2223–2966.2020.09.32

МЕТОДЫ СТАТИЧЕСКОГО И ДИНАМИЧЕСКОГО АНАЛИЗА ДЛЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

STATIC AND DYNAMIC ANALYSIS METHODS FOR SOFTWARE TESTING

A. Svyatenko

Summary. Currently, there is no single methodology that allows comprehensive testing of software. The main difference between the methods used for statistical and dynamic analysis of software quality are quantitative and qualitative indicators, as well as the choice of a reference indicator.

The problem is that static analysis methods for software testing lead to estimation inaccuracies, while dynamic analysis methods are time-consuming and can define procedures for selecting optimal values in different ways.

The article suggests using a complex methodology based on static and dynamic analysis methods when testing software.

Keywords: software testing, static analysis methods, dynamic analysis methods, analyzer, tool, program code, quality metrics.

Святенко Александр Сергеевич
Руководитель проектов по тестированию,
ООО «ЕГАР Сервис», г. Москва
info@egartech.com

Аннотация. В настоящее время не существует единой методики, позволяющей всесторонне протестировать программное обеспечение. Основным отличием методик, которые применяются для статистического и динамического анализа качества программного обеспечения, являются количественные и качественные показатели, а также выбор эталонного показателя.

Проблемой является то, что статические методы анализа для тестирования программного обеспечения приводят к неточностям оценки, а динамические методы анализа являются трудоемкими и могут по-разному определять процедуры выбора оптимальных значений.

В статье предложено при тестировании программного обеспечения использовать комплексную методику, основанную на методах статического и динамического анализа.

Ключевые слова: тестирование программного обеспечения, методы статического анализа, методы динамического анализа, анализатор, инструмент, программный код, метрики качества.

Среди методов статистического анализа для тестирования программного обеспечения следует отметить метод кластеризации, основанный на проверке схожести или связей. Задачей методов кластеризации является объединение в одну или множество различных групп дефектов программного обеспечения с целью их последующего предупреждения.

Методы кластеризации бывают прямыми и косвенными. Их различие состоит в том, что при прямой кластеризации создается связь между двумя идентичными предупреждениями. Кластеризация предупреждений о дефектах осуществляется с использованием арифметической связи между индексными переменными при доступе к буферу.

В случае косвенной кластеризации производится группировка предупреждений по синтаксическому или структурированному признаку [1].

Кроме кластеризации в методах статистического анализа распространение получил метод ранжирования, в основе которого находится выстраивание предупреждений при вводе в анализатор так чтобы предупреждения, имеющие большую вероятность, находились в верхней части списка, а меньшую вероятность — в нижней части списка.

Статический метод ранжирования основан на технологиях приоритезации предупреждений. Для реализации этого метода используется статическая модель ранжирования дефектов в программном обеспечении. В этом случае производится наблюдение за программным кодом, его проверка на наличие ошибок. Статическим анализатором устанавливаются факты инициализации ошибочных ситуаций, как при прямой, так и при обратной проверке программного кода.

Кроме статического ранжирования можно использовать ранжирование по истории изменений. В этом случае рекомендуется использовать более высокий приоритет для ошибок в программном коде [5].

Метод отсечения и классификации предупреждений позволяет скрыть предупреждение о реальных ошибках в программном коде. В основе метода находится иден-

тификация предупреждений на основании проведения статистического анализа и последующего сравнения полученных результатов.

В свою очередь, Банчук Г. Г. в качестве статистических методов анализа качества программного кода рекомендуется использовать метод избавления от ложноположительных предупреждений. В основе этого подхода находится инкрементальное расширение контекста для верификации программного кода через вызов функции постепенного расширения [2].

Для улучшения метода Костиным А. В. рекомендуется использовать технику предсказания результата, которая позволит исключить ложноположительные предупреждения и уменьшить количество вызовов для проверки программного кода [9].

Недостатком методов статического анализа для тестирования программного обеспечения является не точность в полученных результатах.

Для изменения характеристик и критериев качества в динамических методах анализа применяются метрики, представленные в виде системы показателей для тестирования программного обеспечения. Метрики могут быть использованы на уровне критериев качества, а также на уровне отдельных характеристик.

Метрики для оценки размера программного обеспечения являются более простыми. В этом случае определяется размер программного обеспечения по количеству строк исходного кода. Оценка размера программы применяется для классификации программ и выделения их объемов [4].

Вторая группа оценок тестирования программного обеспечения включает в состав метрики сложности потока управления программным обеспечением. В состав этих метрик входит плотность управляющих переходов программ в зависимости от переходов. Основной метрикой сложности выступает цикломатическое число Мак-Кейба, позволяющее охарактеризовать трудоемкость тестирования программы. Для графов корректных программ, в которых нет недостижимых участков, сильно связанный граф получается на основании замыкания одной вершины, означающий конец программного продукта с вершиной, означающей точку входа в эту программу.

Следующей группой метрик оценки сложности программного обеспечения являются метрики оценки потока данных. В этом случае применяются модуль обращений к глобальным переменным. В зависимости от наличия в программе обращение к переменной формируются 2 типа пар. Для расчёта вероятности ссылки произвольного модуля на глобальную переменную используется понятие, что чем выше вероятность, тем больше возможность несанкционированного изменения переменной, что оказывает влияние на сложность изменения программы [7].

Для оценки качества программного обеспечения в динамическом анализе также применяют метрики Джилба, позволяющие оценить сложность программного обеспечения на основе количества в программе условных операторов или операторов цикла. Эта метрика позволяет оценить сложность написания программного продукта, а при добавлении показателя максимального уровня вложенности циклических и условных операторов, значимость в метрике существенно увеличивается.

Метод Чепина предусматривает проведение оценки информационной прочности программного обеспечения на основе анализа использования переменных из списка ввода-вывода.

Также применяется метрика Берлингера, в основе которой находится определение частоты появления символов в коде программы и вероятности его проявления [6].

Однако динамические методы анализа программного обеспечения отличаются сложностью применения на практики, особенно при отсутствии инструментов автоматизации.

Для исключения недостатков методов статического и динамического анализа в работах многих авторов рекомендуется их комбинировать. В настоящее время свое распространение получил инструмент Check'N'Crash, в котором объединены инструменты статического анализа ESC/Java и JCrasher генератор тестов, позволяющий извлекать определенные значения свойств специфические для абстрактных условий ошибок, найденных статическим анализатором.

Рассматривая развитие методов обеспечения качества программных продуктов, Кулева Ю. С. рекомендует использовать инструмент DSD-Crasherr, включающий в свой состав методы проверки качества программного кода на трех уровнях [8]:

- исполнения для динамического обнаружения заложенного поведения с целью ограничения множества входных значений;
- статического анализа для выявления наиболее опасных ошибок;
- динамической верификации для обнаружения потенциальных ошибок и проверки их достижимости.

Слайсинг программного обеспечения может применяться для объединения динамического и статического методов анализа качества программного обеспечения. Для осуществления слайсинга применяют инструмент SANTE, в котором интегрированы инструменты для проведения статистического анализа, позволяющего обнаружить ошибки в программном коде и динамического анализа.

На основании этого генерация больших программных кодов сокращается, а также ускоряется процесс оценки качества программного обеспечения.

В работе Банчука Г. Г. предлагается объединить статистический анализ и динамический для обнаружения подозрительных мест в программном коде. Для реализации этого подхода можно использовать статический анализатор SVR, в основе которого находится промышленный компилятор GCC [2].

Статическим анализатором по программному коду формируется модель для инструмента Moped, проверяющего свойства безопасности программного кода. В результате определяется набор путей, приводящих к потенциальному дефекту. На основании этого маршрута формируется блок входных данных и производится проверка достижимости определенных состояний в программном коде.

В Костина А. В. рассматривается подход, объединяющий статистический и динамический анализ в инструменте JNuke. Этот инструмент позволяет абстрагировать алгоритм работы анализатора и предоставить возможность оценки различных свойств программного кода, вычисляемых статически и предоставляемых через единый интерфейс [9].

Идеи комбинирования динамического и статистического анализа реализованы в инструменте SANTE, который позволят анализировать исходный код на языке Си Farma-C и производить динамическое символьное исполнение с целью построения тестовых наборов.

В Лукина В. Н. рекомендуется использовать инструмент ConDroid, позволяющий выполнять анализ программного кода продуктов, работающих под управлением операционной системы Android. Этот инструмент основ на статистическом анализе программного кода и определения критических узлов в программе с использованием динамического анализа [10].

Метод динамического анализа в инструменте ConDroid представлен в виде символьного исполнения определённого набора входных данных. При этом итеративный динамический анализ для каждой операции определяет выходные данные, которые в итоге позволяют привести программу в исполнение.

Инструмент IntelliDroid объединяет методы динамического и статического анализа и представляет результаты в формате байт-кода DEX. Статистическим анализатором применяются данные о месте использования стороннего API вместе с точками вызова и генерацией входных данных через решатель формул, основанный на булевых ограничениях.

Еще одним инструментом, позволяющим генерировать тестовые сценарии, выявлять критические ошибки является программа STAR, которая используется для оценки качества программного обеспечения написанного на JAVA [3].

В инструменте реализуется метод классификации утечки памяти за счет совмещения статистического анализа и динамического символьного анализа.

Следовательно, в настоящее время многие специалисты производят интеграцию методов статистического и динамического анализа программного обеспечения, которая позволяет исключить неточности присущие методам статического анализа и сложность характерную для методов динамического анализа.

ЛИТЕРАТУРА

1. Асрян, С. А. Обнаружение ошибок, возникающих при использовании динамической памяти после ее освобождения /С.А. Асрян // Труды Института системного программирования РАН, 2018.— № 3.— С. 7–20.
2. Банчук, Г. Г. Теоретические аспекты оценки качества программных продуктов / Г. Г. Банчук // Статья в сборнике Информационно-аналитические системы и технологии, 2018.— С. 40–50.
3. Бубарева, О. А. Оценка качества программных систем при связывании объектных спецификаций по семантике онтологического уровня / О. А. Бубарева // Современные наукоемкие технологии, 2018.— № 6.— С. 40–43.
4. Кожомбердиева, Г. И. Получение интегральной оценки качества программного обеспечения на основе формулы Байеса / Г. И. Кожомбердиева // Транспортные интеллектуальные системы, 2017.— С. 209–220.
5. Климов, Г. Ю. Оценка качества и анализ программного обеспечения в информационной системе управления предприятием / Г. Ю. Климов // Статья в сборнике Наука сегодня, 2019.— С. 53–54.

6. Кудеяров, Ю. А. Испытания программного обеспечения средств измерений: учеб. пособие/ Ю. А. Кудеяров.— М.: Академия стандартизации, метрологии и сертификации, 2017.— 250 с.
7. Ленкин, А. В. Анализ требований к интерфейсу современного программного обеспечения / А. В. Ленкин// Постулат, 2017.— № 1.— С. 41–46.
8. Кулева, Ю. С. Развитие методов обеспечения качества программных продуктов / Ю. С. Кулева // Статья в сборнике Инноватика, 2018.— С. 491–494.
9. Костин, А. В. Модель для оценивания функциональности систем машинного перевода / А. В. Костин // Известия Российской академии науки, 2018.— С. 158–172.
10. Лукин, В. Н. Подготовка качественных программистов: проблемы обучения / В. Н. Лукин // Моделирование и анализ данных, 2017.— № 1.— С. 29–41.

Журнал «Современная наука: актуальные проблемы теории и практики»