Автоматизированное управление тестированием программных систем с применением нейронных сетей

И.С. Полевщиков, Р.А. Файзрахманов

Пермский национальный исследовательский политехнический университет

Аннотация: Статья посвящена решению актуальной проблемы повышения эффективности тестирования программного обеспечения (ПО) на основе разработки автоматизированной системы управления (АСУ) данным процессом. Предложена структура создаваемой АСУ, позволяющей формировать советующие воздействия специалистам для наилучшего осуществления каждого из этапов тестирования ПО. Представлены оптимизационные модели, обеспечивающие корректный подбор значений различных атрибутов планов тестирования, тест-кейсов, отчетов о дефектах и других документов, создаваемых в процессе тестирования. Описаны структура нейронной сети и концепция ее применения для наиболее точного подбора значений атрибутов.
Работа выполнена при поддержке стипендии Президента РФ молодым ученым и аспирантам (№ стипендии СП-100.2018.5), назначенной Советом по грантам Президента Российской Федерации.
Ключевые слова: тестирование программного обеспечения, методы оптимизации, автоматизированная система управления, нейронные сети, тестовое покрытие.

1 Введение. Актуальность исследования

Применение автоматизированных систем (АС) в работе современного предприятия является важной составляющей для успешного создания товаров и услуг. Низкое качество программного обеспечения (ПО) АС может являться одной из причин, вызывающих нарушение деятельности организации и финансовые потери. С целью создания качественного ПО требуется грамотная организация всех этапов разработки [1], и, в частности, этапа тестирования как важнейшего средства для определения степени соответствия ПО заданным требованиям [1-4].

Тестирование – важный и неотъемлемый этап разработки ПО, поскольку ошибки, не найденные на данном этапе, будут обнаружены пользователями, и стоимость их исправления при сопровождении значительно выше, чем в процессе разработки [1]. С другой стороны, тестирование – достаточно трудоемкий процесс, составляющий примерно 40% от суммарной

трудоемкости разработки ПО [1, 2].

Для уменьшения трудоемкости тестирования разрабатываются и применяются средства автоматизации, позволяющие [5, 6]: увеличить скорость выполнения тест-кейсов; снизить влияние человеческого фактора в процессе их выполнения; минимизировать затраты при многократном выполнении тест-кейсов; выполнять тест-кейсы, непосильные человеку в виду сложности, скорости и других причин; собирать, хранить и обрабатывать большие объемы данных.

Однако современные средства автоматизации тестирования ПО обладают значительными недостатками:

– фактически отсутствуют программные продукты, позволяющие в наилучшей степени автоматизировать все фазы процесса тестирования ПО, учитывая взаимосвязь фаз (существующие средства автоматизации в основном предназначены для какого-либо отдельного этапа, например, сравнения реальных и ожидаемых результатов выполнения тестов);

– многие сложные задачи (в частности, выбор методов разработки тест-кейсов для конкретного проекта, разработка тест-кейсов) фактически возложены на специалиста по тестированию (человека) и выполняются в значительной степени на основе его субъективного мнения.

Таким образом, актуальной задачей является разработка моделей, методов и программных продуктов с целью повышения эффективности информационной поддержки специалиста на всех этапах тестирования ПО.

2 Структура системы управления тестированием ПО

Разработана структура автоматизированной системы управления (АСУ) процессом тестирования ПО (рис. 1), обеспечивающей информационную поддержку всех этапов тестирования сложной программной системы (планирование; создание наборов тест-кейсов; выполнение тест-кейсов,

фиксация найденных дефектов; анализ результатов, формирование отчетности), а также вспомогательных функций, таких как сбор, анализ и оценка требований к ПО, обучение специалистов по тестированию ПО.

Рис. 1. – Структура АСУ процессом тестирования ПО

Всем специалистам для осуществления необходимых этапов процесса тестирования ПО предоставлен графический пользовательский интерфейс. АСУ позволяет формировать советующие воздействия специалистам для наилучшего осуществления каждого из этапов. Дополнительные подсистемы позволяют в структурированном виде сформировать перечень требований к программному продукту (подсистема №5 на рис. 1), проверяемых на этапе тестирования, а также повышать квалификацию специалистов по тестированию ПО и обучать начинающих специалистов [7, 8], в том числе на основе опыта реальных программных проектов (подсистема №6). База данных позволяет хранить все настройки и результаты работы, обеспечивает взаимосвязь всех подсистем АСУ.

Разрабатываемая АСУ обладает возможностью интеграции с другими средствами автоматизации тестирования ПО, нашедшими эффективное применение при решении некоторых частных задач (например, юнит-тестирование, тестирование мобильных приложений и т.д.).

3 Разработка оптимизационных моделей для повышения эффективности процесса тестирования ПО

В основе работы АСУ лежит комплекс оптимизационных моделей, позволяющих при минимальных затратах ресурсов (например, времени):

– разрабатывать наборы тест-кейсов, обеспечивающих необходимое тестовое покрытие требований и программного кода и обладающих большой вероятностью систематического обнаружения различных классов еще не раскрытых ошибок в программах;

– правильно подбирать различные атрибуты планов тестирования, тест-кейсов, отчетов о дефектах и других документов, обычно устанавливаемые специалистом на основе субъективного мнения.

Для каждого этапа процесса тестирования (планирование, разработка тест-кейсов и т.д.) создаются оптимизационные модели следующего вида:

$K_{\text{эфф.}} = f_{\text{эфф.}}(A_1, A_2, ..., A_p) \to \max$ – функция оценки эффективности выполнения соответствующего этапа тестирования, зависящая от множества атрибутов A_k ($k = \overline{1, p}$) создающихся на данном этапе документов (тест-планы, тест-кейсы, отчеты о дефектах и т.д.);

$K_{\text{рес.}} = f_{\text{рес.}}(A_1, A_2, ..., A_p) \to \min$ – функция оценки требуемых ресурсов для осуществления данного этапа тестирования;

$\forall g = \overline{1, N_{\text{рес.}}}$, $K_g \le K_g^{\text{огр.}}$ – ограничения на ресурсы.

Рассмотрим основные особенности оптимизационных моделей на примере разработки тест-кейсов как одного из этапов тестирования ПО. Выполнено теоретико-множественное описание атрибутов тест-кейса и связей между атрибутами.

Любой тест-кейс T_i представим как множество атрибутов: $T_i = \left\{ A_{ij}^{\text{тест}} \mid j = \overline{1, N_i^{\text{тест}}} \right\}$. Отдельными атрибутами $A_{ij}^{\text{тест}} \in T_i$ являются: идентификатор; заглавие тест-кейса; требования, модули приложения и

ключевые слова, проассоциированных с тест-кейсом; исходные данные (предварительные условия и необходимые приготовления к выполнению тест-кейса); последовательность шагов тест-кейса; приоритет тест-кейса; оценка времени выполнения и т.д. Рассмотрим особенности некоторых из этих атрибутов, и, в частности, специфику формирования системой рекомендаций (советующих воздействий) по вычислению данных атрибутов.

Атрибут «последовательность шагов тест-кейса» ($A^{\text{тест}}_{\text{шаг.}} \in T_i$) представляется кортежем $A^{\text{тест}}_{\text{шаг.}} = \left\langle a_1^{\text{шаг}}, a_2^{\text{шаг}}, ..., a_v^{\text{шаг}} \right\rangle$, где $a_z^{\text{шаг}} = \left\langle d_z, r_z \right\rangle$ – отдельный z-й шаг тест-кейса ($z = \overline{1,v}$), причем d_z – действие, которое необходимо реализовать в процессе выполнения тест-кейса, r_z – ожидаемый результат (реакция приложения) на выполнение действия. Данные, подающиеся на вход программы при выполнении действий d_z, могут генерироваться автоматически на основе моделей работы программы, таких как потоковый граф [1], диаграммы причин-следствий [9] и т.д.

Приоритет $A^{\text{тест}}_{\text{пр.}} \in T_i$ определяет важность тест-кейса, упрощая работу проектной команды. В частности, приоритеты позволяют принимать решения о том, какие тест-кейсы можно не выполнять, если выполнить все запланированные тест-кейсы не возможно. Приоритет представляет некоторое неотрицательное число в шкале порядка или отношений [10]:

– в шкале порядка числу может соответствовать некоторое словесное описания (например, приоритетам, равным 3, 2, 1 могут соответствовать словесные описания «высокий», «средний», «низкий»);

– в шкале отношений приоритеты задаются числами из определенного диапазона (например, отрезок [0;1], где 0 соответствует наиболее низкому приоритету, а 1 – наиболее высокому).

Рекомендуемое значение приоритета тест-кейса может быть подобрано системой автоматически, согласно зависимости $A^{\text{тест}}_{\text{пр.}} = f^{\text{тест}}_{\text{пр.}}\left(A^{\text{кор.}}_{\text{пр.1}}, A^{\text{кор.}}_{\text{пр.2}}, ..., A^{\text{кор.}}_{\text{пр.}r}\right)$,

где $A_{\text{пр.}h}^{\text{кор.}}$ $(h=\overline{1,r})$ – атрибут, с которым коррелирует приоритет тест-кейса. Такими атрибутами могут являться, в частности, вид $A_{\text{вид}}^{\text{тест}}$ тест-кейса, важность и степень риска, сопоставленные с требованиями $A_{\text{треб.}}^{\text{тест}}$, модулями $A_{\text{мод.}}^{\text{тест}}$, ключевыми словами $A_{\text{ключ.}}^{\text{тест}}$, проассоциированными с тест-кейсом. Аналогично может быть получена оценка времени $A_{\text{врем.}}^{\text{тест}}$, затрачиваемого на выполнение тест-кейса (в частности, на основе анализа времени выполнения подобных тест-кейсов).

Сформулированы функциональные требования к подсистеме разработки тест-кейсов (из рис. 1) в части задания значений атрибутов тест-кейса и представлены в виде диаграммы вариантов использования UML [1] (рис. 2).

Рис. 2. – Функциональные требования к подсистеме разработки тест-кейсов

Задание значений атрибутов может представлять собой ввод произвольного текста, ввод числа или выбор значения из перечня. Как показано на рис. 2, для многих атрибутов системой может быть сформировано рекомендуемое значение. АСУ позволяет также помимо стандартных атрибутов, представленных на рис. 2, добавлять свои атрибуты документов, создающихся в процессе тестирования.

4 Применение нейронных сетей для определения значений атрибутов документов

При решении задачи оптимизации в АСУ применяются нейронные сети, позволяющие корректно определить рекомендуемые значения атрибутов документов, создаваемых на различных этапах тестирования (тест-кейсов, отчетов о дефектах и т.д.). Такими атрибутами являются, например: входные данные, приоритет, оценка времени выполнения тест-кейсов; важность и срочность устранения дефектов.

Корректный подбор значений атрибутов обеспечивается возможностью нейросетей выявлять сложные зависимости между входными и выходными данными, выполнять обобщение [11, 12].

Предлагается использовать многоуровневую нейронную сеть прямого распространения, состоящую из входного слоя (один узел для каждого входного значения), одного или нескольких скрытых слоев и выходного уровня [11]. Методом обучения такой нейронной сети будет являться метод обратного распространения ошибки.

Чтобы использовать нейронную сеть при подборе значений атрибутов, необходимо изначально обучить сеть. Процесс обучения заключается в подготовке обучающей выборки, представляющей собой множество пар входных и соответствующих им заведомо верных выходных векторов [11]. Для обучения могут использоваться данные о тестировании при выполнении предыдущих программных проектов и данные о предыдущих итерациях тестирования для текущего программного проекта. Учитываются взаимосвязи между атрибутами документов, упомянутые выше.

Обобщенно структура нейронной сети и концепция ее применения для рассматриваемой задачи представлены схематично на рис. 3.

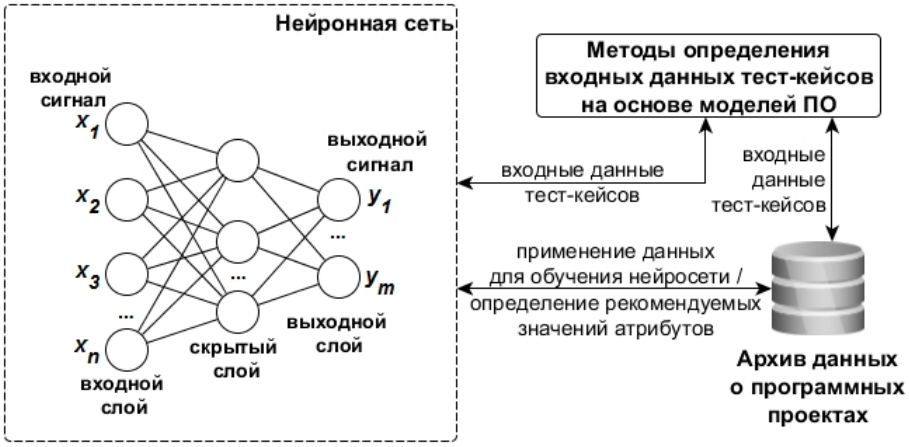

Рис. 3. – Структура нейронной сети и концепция ее применения

Как видно на рис. 3, применение нейросетей дополняет использование существующих алгоритмов формирования входных данных тест-кейсов. При решении каждой конкретной задачи, связанной с определением значений атрибутов того или иного документа, создающегося в ходе тестирования, требуется более точно выбрать топологию, характеристики нейронной сети и параметры обучения.

5 Заключение

Разработанная АСУ обладает рядом преимуществ:

– специалист может наиболее эффективно осуществлять все этапы процесса тестирования ПО благодаря отсутствию рутинных операций и генерации советующих воздействий (посредством алгоритмов, основанных на применении методов математического моделирования и нейросетей);

– обеспечивается комплексный подход к автоматизации тестирования ПО, основанный на систематизированном применении существующих методов и алгоритмов;

– применение современных технологий разработки ПО при создании АСУ, в частности, веб-технологий.

Продолжением исследования будут являться уточнение

функциональных и нефункциональных требований к ПО АСУ, разработка новых алгоритмов, дальнейшее совершенствование созданного прототипа системы. В частности, в системе должно быть организовано управление жизненными циклами тест-кейсов и отчетов о дефектах (на основе разработки имитационных моделей) с целью повышения эффективности нахождения и исправления ошибок в ПО.

Литература

1. Орлов С.А., Цилькер Б.Я. Технологии разработки программного обеспечения: Учебник для вузов. 4-е изд. Стандарт третьего поколения. СПб.: Питер, 2012. 608 с.

2. Основы тестирования программного обеспечения. URL: intuit.ru/studies/courses/48/48/info (дата обращения: 09.11.2018).

3. Бурякова Н.А., Чернов А.В. Классификация частично формализованных и формальных моделей и методов верификации программного обеспечения // Инженерный вестник Дона. 2010. №4. URL: ivdon.ru/ru/magazine/archive/n4y2010/259.

4. Яловой И.О. Анализ требований и управление изменениями программных проектов // Инженерный вестник Дона. 2008. №4. URL: ivdon.ru/ru/magazine/archive/n4y2008/102.

5. Куликов С.С. Тестирование программного обеспечения. Базовый курс. Минск: Четыре четверти, 2017. 312 с.

6. Полевщиков И.С. Автоматизированная система управления процессом тестирования программного обеспечения в ходе промышленной разработки // Решение. 2018. Т. 1. С. 191-193.

7. Fayzrakhmanov R., Polevshchikov I., Khabibulin A. Computer Simulation Complex for Training Operators of Handling Processes // Proc. of the 5th

International Conference on Applied Innovations in IT (ICAIIT), Koethen (Germany), 16 March 2017. Vol. 5. pp. 81-86.

8. Faizrakhmanov R.A., Polevshchikov I.S., Khabibulin A.F., Shklyaev F.I. An energy-saving overload-control technology based on a computerized training-machine system // Russian Electrical Engineering. 2017. Vol. 88, № 11. pp. 725-727.

9. Калин М.В., Полевщиков И.С. Автоматизация процесса тестирования программного обеспечения на основе построения диаграмм причин-следствий // Инновационные технологии: теория, инструменты, практика. 2017. Т. 1. С. 45-51.

10. Новиков Д.А. Статистические методы в педагогических исследованиях (типовые случаи). М.: МЗ-Пресс, 2004. 67 с.

11. Данилов А.Д., Мугатина В.М. Верификация и тестирование сложных программных продуктов на основе нейросетевых моделей // Вестник Воронежского государственного технического университета. 2016. Т. 12. № 6. С. 62-67.

12. Долгова Е.В., Курушин Д.С. Адаптация математической модели нейронной сети для системы распознавания слитного рукописного текста // Вестник Московского государственного областного университета. Серия: Физика-математика. 2011. № 2. С. 90-97.

References

1. Orlov S.A., Tsil'ker B.Ya. Tekhnologii razrabotki programmnogo obespecheniya: Uchebnik dlya vuzov. 4-e izd. Standart tret'ego pokoleniya [Software development technologies: A textbook for universities. 4th ed. The standard of the third generation]. SPb.: Piter, 2012. 608 p.

2. Osnovy testirovaniya programmnogo obespecheniya [Software Testing Basics]. URL: intuit.ru/studies/courses/48/48/info (accessed 09/11/2018).

3. Buryakova N.A., Chernov A.V. Inženernyj vestnik Dona (Rus), 2010, №4. URL: ivdon.ru/ru/magazine/archive/n4y2010/259.

4. Yalovoy I.O. Inženernyj vestnik Dona (Rus), 2008, №4. URL: ivdon.ru/ru/magazine/archive/n4y2008/102.

5. Kulikov S.C. Testirovanie programmnogo obespecheniya. Bazovyy kurs [Software Testing. Basic course]. Minsk: Chetyre chetverti, 2017. 312 p.

6. Polevshchikov I.S. Reshenie. 2018. Vol. 1. pp. 191-193.

7. Fayzrakhmanov R., Polevshchikov I., Khabibulin A. Computer Simulation Complex for Training Operators of Handling Processes. Proc. of the 5th International Conference on Applied Innovations in IT (ICAIIT), Koethen (Germany), 16 March 2017. Vol. 5. pp. 81-86.

8. Faizrakhmanov R.A., Polevshchikov I.S., Khabibulin A.F., Shklyaev F.I. Russian Electrical Engineering. 2017. Vol. 88, № 11. pp. 725-727.

9. Kalin M.V., Polevshchikov I.S. Innovatsionnye tekhnologii: teoriya, instrumenty, praktika. 2017. Vol. 1. pp. 45-51.

10. Novikov D.A. Statisticheskie metody v pedagogicheskih issledovanijah (tipovye sluchai) [Statistical methods in educational research (typically)]. Moscow: MZ-Press, 2004. 67 p.

11. Danilov A.D., Mugatina V.M. Vestnik Voronezhskogo gosudarstvennogo tekhnicheskogo universiteta. 2016. Vol. 12. № 6. pp. 62-67.

12. Dolgova E.V., Kurushin D.S. Vestnik Moskovskogo gosudarstvennogo oblastnogo universiteta. Seriya: Fizika-matematika. 2011. № 2. pp. 90-97.