СТАТЬИ

УДК 004.04
DOI 10.17513/snt.39609

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ ФОРМАЛИЗОВАННОГО ПОДХОДА ДЛЯ ОПТИМИЗАЦИИ ПРОЦЕССА ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ (АСУ ТП)

Букарев А.В.

ФГАОУ ВО «Национальный исследовательский университет «Московский институт электронной техники», Москва, e-mail: anton@bukarev.org

Автоматизация тестирования программного обеспечения (ПО) является важным элементом процесса его разработки. Оптимизация тестирования, подразумевающая снижение сроков проведения, а также затрат используемых ресурсов (как финансовых, так и человеческих), важная для достижения высокого качества продукта, может быть произведена с использованием автоматизированной системы управления технологическими процессами (АСУ ТП). В данной статье описывается формализованное представление процесса тестирования ПО в АСУ ТП с помощью множественного представления, графов, теории массового обслуживания и имитационной модели методом Монте-Карло. Основная цель данного исследования – разработать практически применимый подход для оптимизации и сокращения времени процесса тестирования программного обеспечения в АСУ ТП. В частности, статья охватывает зависимости между аппаратным обеспечением, драйверами и программным обеспечением, использующим драйвера. Последовательность и взаимосвязи между шагами процесса тестирования и определение времени выполнения каждого из них представляется нагруженным графом; совокупная производительность тестирования оценивается с использованием методов теории массового обслуживания; ввиду высокой сложности процесса моделирование возможных ошибок производится с использованием имитационной модели методом Монте-Карло. Результаты исследования показали, что такое формализованное представление процесса тестирования программного обеспечения позволяет значительно ускорить и оптимизировать процесс тестирования в АСУ ТП, а также повысить качество продукта. Разработана программная реализация модели на языке Python, которая обладает широким спектром возможностей для математических вычислений и моделирования сложных систем.

Ключевые слова: автоматизированное тестирование программного обеспечения, формализованное представление, множественные представления, графы, теория массового обслуживания, имитационная модель Монте-Карло, зависимости аппаратного обеспечения, зависимости драйверов, этапы тестирования

PRACTICAL APPLICATION OF A FORMALIZED APPROACH FOR OPTIMIZING THE SOFTWARE TESTING PROCESS IN AUTOMATED SYSTEMS FOR MANAGING TECHNOLOGICAL PROCESSES

Bukarev A.V.

National Research University of Electronic Technology, Moscow, e-mail: anton@bukarev.org

Automation of software testing (SW) is an important element of the software development process. Test optimization, which implies a reduction in the timing, as well as the cost of the resources used (both financial and human), which is important for achieving high product quality, can be done using an automated process control system (APCS). This article describes a formalized representation of the software testing process in automated process control systems using multiple representation, graphs, queuing theory and a Monte Carlo simulation model. The main goal of this study is to develop a practical approach to optimize and reduce the time of the software testing process in ICS. Specifically, the article covers dependencies between hardware, drivers, and software that uses the drivers. The sequence and relationships between the steps of the testing process and the determination of the execution time of each of them is represented by a loaded graph; cumulative testing performance is estimated using queuing theory methods; due to the high complexity of the process, the modeling of possible errors is carried out using a simulation model using the Monte Carlo method. The results of the study showed that such a formalized representation of the software testing process can significantly speed up and optimize the testing process in the process control system, as well as improve the quality of the product. A software implementation of the model in the Python language has been developed, which has a wide range of capabilities for mathematical calculations and modeling of complex systems.

Keywords: automated software testing, formalized representation, multiple representations, graphs, queuing theory, Monte Carlo simulation model, hardware dependencies, driver dependencies, testing steps

Производство технически сложных устройств, в том числе программно-аппаратных устройств, таких как банкоматы, телефоны и прочие различные бытовые устройства, представляет собой два сильно связанных направления, ведущихся параллельно: разработка и производство аппаратной части и обеспечивающей ее функционирование программной части. Последняя по своей архитектуре разделяется на ряд уровней (аппаратный микрокод, драйвера и интерфейс прикладного программного обеспечения), взаимодействующих между собой. При модернизации как аппаратно-

го сегмента комплекса, так и какой-либо из программных его составляющих требуется комплексное тестирование. В данной работе развиваются методы повышения эффективности тестирования программной части технически сложных устройств.

Разработка качественного программного обеспечения является сложным процессом, требующим соблюдения определенных стандартов и процедур. Тестирование является одной из важнейших стадий этого процесса, которая необходима для проверки работоспособности и соответствия программного обеспечения требованиям. Однако с развитием автоматизации процессов в различных областях возникает необходимость в автоматизированном тестировании программного обеспечения, включая системы управления технологическими процессами (АСУ ТП) [1].

Цель данного исследования – разработать формализованное представление процесса тестирования программного обеспечения в АСУ ТП с использованием теории графов и теории массового обслуживания, а также создание имитационной модели Монте-Карло. Данный подход должен помочь в оптимизации и сокращении времени процесса тестирования и улучшить качество программного обеспечения. В результате производительность и функциональность АСУ ТП должны быть улучшены.

Множественное представление аппаратно-программных платформ

Одной из проблем, с которой сталкиваются разработчики, является необходимость создавать программное обеспечение, которое может работать с разными версиями аппаратного обеспечения, драйверов и операционных систем. Распространенными случаями являются производство мобильных телефонов, устройств «умного дома» и других бытовых устройств. Например, каждый телефон может иметь разные модели камер, разные версии драйверов и разные версии операционной системы.

Чтобы понять, насколько сложным может быть этот процесс, рассмотрим возможные комбинации аппаратного обеспечения, драйверов и операционных систем. Предположим, что у нас есть три модели камер, две версии драйверов и три версии операционной системы. Количество возможных комбинаций аппаратного обеспечения, драйверов и операционных систем будет равно произведению количества моделей камер, версий драйверов и версий операционной системы: $3 \times 2 \times 3 = 18$. Это означает, что у устройства может быть до 18 разных комбинаций аппаратного обеспечения, драйверов и операционных систем, с которыми должно работать наше программное обеспечение.

Множественное представление аппаратно-программных платформ позволяет создавать несколько версий программного обеспечения, каждая из которых оптимизирована для конкретной комбинации аппаратного обеспечения, драйверов и операционной системы. Например, можно создать одну версию программного обеспечения для телефонов с камерами от компании А, использующих драйверы версии 1 и операционную систему версии 1, и другую версию для телефонов с камерами от компании В, использующих драйверы версии 2 и операционную систему версии 2.

Для отслеживания множественных представлений аппаратно-программных платформ можно использовать систему управления версиями, позволяющими контролировать все версии ПО. Также возможно использовать систему управления версиями для отслеживания изменений, внесенных в каждую версию приложения, и для разрешения конфликтов, которые могут возникнуть при интеграции нескольких версий.

Еще одним важным аспектом множественного представления аппаратно-программных платформ является тестирование. Для каждой версии программного обеспечения мы должны провести отдельное тестирование, чтобы убедиться, что она работает правильно с соответствующей комбинацией аппаратного обеспечения, драйверов и операционной системы. Это может занять много времени и ресурсов, поэтому мы должны убедиться, что наши тесты максимально автоматизированы [2].

Множественное представление аппаратно-программных платформ позволяет создавать несколько версий программного обеспечения, каждая из которых оптимизирована для конкретной комбинации аппаратной реализации, драйверов и операционной системы. Это позволяет максимизировать совместимость и производительность и улучшить процесс тестирования и разработки. Использование множественного представления является эффективным способом управления сложностью аппаратного и программного обеспечения.

Формализованное представление автоматизированного процесса тестирования с помощью графов

В процессе разработки ПО тестирование играет важную роль в обеспечении качества и надежности продукта. Для улучшения про-

цесса тестирования, его оптимизации и автоматизации в процессах АСУ ТП, которая включает в себя несколько шагов, начиная от создания тестовых сценариев и заканчивая анализом результатов тестирования. Одним из эффективных способов формализации и оптимизации процесса тестирования является представление шагов тестирования с помощью теории графов (рисунок) [3, 4].

Представление процесса тестирования ПО

Первый шаг тестирования – создание тестовых сценариев. Этот шаг включает в себя описание сценариев тестирования, включающих в себя список функций и аспектов, которые будут протестированы. Каждый сценарий может содержать несколько тест-кейсов, которые покрывают все возможные состояния программного обеспечения. Для создания тестовых сценариев можно использовать как формальные, так и неформальные методы.

Второй шаг – разработка тестов в соответствии с тестовыми сценариями. Этот шаг включает в себя написание тестовых сценариев на одном из языков программирования, выбранном для тестирования. Разработка тестов включает в себя создание тестовых данных, выполнение тестов на программном обеспечении, а также запись результатов тестирования.

Третий шаг – подготовка программного обеспечения к запуску тестов в облаке. Этот шаг включает в себя загрузку программного обеспечения на облачную платформу, создание виртуальных машин для запуска тестов, а также настройку облачной платформы для обеспечения тестирования на разных устройствах.

Четвертый шаг – запуск тестов на устройствах в облаке. Этот шаг включает в себя запуск тестов на различных устройствах, которые находятся в облаке. Этот шаг может занять длительное время, в зависимости от количества тестов и количества устройств, используемых для тестирования.

Пятый шаг – анализ результатов тестирования. Этот шаг включает в себя анализ результатов тестирования, включая отчеты обо всех найденных ошибках и проблемах. Результаты тестирования также могут быть представлены в графическом виде, чтобы облегчить понимание и визуализацию данных [5, 6].

Каждый узел представляет определенный шаг тестирования, а каждое ребро указывает на зависимости между шагами. Например, на графе выше зависимость между созданием тестовых сценариев и разработкой тестов показана стрелкой от узла «Создание тестовых сценариев» к узлу «Разработка тестов».

Представление каждого шага тестирования в виде графа позволяет оптимизировать процесс тестирования путем идентификации и устранения неоптимальных мест. Например, на графе можно выделить узлы, которые занимают больше всего времени, и улучшить их производительность, чтобы ускорить весь процесс тестирования.

Представление шагов тестирования в АСУ ТП с помощью теории графов является эффективным способом оптимизации процесса тестирования. Графы позволяют представить каждый шаг тестирования в виде узла, а зависимости между шагами – в виде ребер графа, что упрощает понимание и визуализацию данных. Это позволяет оптимизировать процесс тестирования и улучшить процесс разработки программного обеспечения.

Представление процесса тестирования с помощью теории массового обслуживания

Представление шагов тестирования программного обеспечения в автоматизи-

рованной системе управления технологическими процессами с помощью теории графов – это важный шаг для оптимизации процесса тестирования. Теория графов позволяет наглядно представить зависимости между шагами тестирования, что может помочь ускорить процесс и облегчить коммуникацию между участниками проекта.

Однако помимо теории графов существует другой подход к представлению шагов тестирования – это теория массового обслуживания. В этом подходе шаги тестирования представляются в виде очереди, где каждый шаг является отдельным обслуживанием.

Опишем, как можно представить шаги тестирования с помощью теории массового обслуживания. Для начала нужно определить основные этапы тестирования. В контексте данного исследования рассматриваются следующие этапы тестирования: создание тестовых сценариев, разработка тестов в соответствии с тестовыми сценариями, подготовка программного обеспечения к запуску тестов в облаке, запуск тестов на устройствах в облаке и анализ результатов тестирования.

Каждый шаг тестирования может быть представлен в виде отдельного обслуживания в очереди. Каждый обслуживаемый процесс имеет свои характеристики, такие как время обслуживания и время ожидания в очереди [7].

Характеристики процесса модели тестирования ПО

Этап тестирования	Время обслуживания (часы)	Время ожидания в очереди (часы)
Создание тестовых сценариев	2	0
Разработка тестов	4	2
Подготовка программного обеспечения	1	1
Запуск тестов	8	4
Анализ результатов тестирования	2	2

Как видно из таблицы, каждый этап тестирования имеет свою продолжительность обслуживания и время ожидания в очереди. Например, создание тестовых сценариев занимает два часа и не имеет времени ожидания в очереди, так как является первым этапом тестирования. Разработка тестов занимает четыре часа и имеет два часа времени ожидания в очереди.

Моделирование процесса тестирования с использованием теории массового обслуживания может помочь определить время, которое займет тестирование, и понять, какие шаги тестирования могут быть оптимизированы.

Кроме того, для моделирования процесса тестирования с использованием теории массового обслуживания необходимо учесть следующие параметры:

– Число пользователей, которые одновременно выполняют тестирование. Число пользователей влияет на длину очереди и время ожидания в очереди.

– Распределение времени обслуживания. Распределение времени обслуживания может быть равномерным или неравномерным.

– Требования к производительности системы, такие как время отклика и время загрузки страниц.

Когда модель тестирования создана, она может быть использована для оптимизации процесса тестирования. Например, если анализ результатов моделирования показывает, что время ожидания в очереди для этапа «Разработка тестов» является слишком высоким, то можно принять меры для оптимизации этого шага тестирования [8].

Представление шагов тестирования программного обеспечения с помощью теории массового обслуживания является эффективным инструментом для оптимизации процесса тестирования. Моделирование процесса тестирования с помощью этого подхода может помочь определить время, которое займет тестирование, и выявить узкие места, которые можно оптимизировать.

Имитационная модель тестирования методом Монте-Карло

Имитационное моделирование Монте-Карло – это метод анализа процессов, основанный на генерации случайных значений и оценке средних значений на основе повторных экспериментов. Данный метод используется для моделирования процессов, которые не могут быть точно описаны математическими уравнениями.

Преимущества использования метода Монте-Карло заключаются в возможности создания точной и подробной модели процесса тестирования, учете всех факторов, влияющих на процесс, и оценке процесса тестирования на основе множества случайных входных данных.

Рассмотрим пример создания имитационной модели Монте-Карло для анализа процесса тестирования программного обе-

спечения. Предположим, что проводится тестирование программного обеспечения, которое должно быть запущено на различных устройствах в облаке. Нам нужно определить оптимальное количество устройств и время, необходимое для выполнения тестов.

Начнем с определения параметров модели:

```
num_devices = 10
test_time_mean = 5
test_time_std = 1
device_time_mean = 1
device_time_std = 0.1
sim_time = 1000
```

Далее необходимо определить класс для устройств. Каждое устройство имеет метод *run()*, который используется для моделирования времени, необходимого для выполнения тестов.

```
class Device(object):
    def __init__(self, env):
        self.env = env
        self.action = env.process(self.run())
    def run(self):
        while True:
            yield self.env.timeout(random.gauss(device_time_mean, device_time_std))
```

Далее определяется функция *run_tests()*, которая запускает тесты на устройствах. Эта функция вызывается периодически для запуска новых тестов.

```
def run_tests(env, devices):
    while True:
        yield env.timeout(random.gauss(test_time_mean, test_time_std))
        device = random.choice(devices)
        yield env.process(device.run())
```

Следующим шагом необходимо создать окружение моделирования с помощью библиотеки SimPy и создать необходимое количество устройств. Также запускается процесс тестирования, который использует функцию run_tests() для запуска тестов на устройствах.

```
env = simpy.Environment()
devices = [Device(env) for i in range(num_devices)]
env.process(run_tests(env, devices))
```

В заключительном шаге запускается моделирование, используя метод *run()* окружения моделирования. Метод *until=sim_time* указывает, что моделирование должно быть остановлено после определенного количества времени.

```
env.run(until=sim_time)
```

Результатом использования данного метода являются результаты, показывающие, сколько времени заняло выполнение каждого теста, сколько времени устройства были заняты, а также другие характеристики процесса тестирования. Данный метод можно расширять для более сложных процессов разработки ПО [9, 10].

Заключение

В данной статье рассмотрена разработка формализованного представления процесса тестирования программной составляющей программно-аппаратных платформ. Цель исследования заключалась в разработке подхода, который позволит оптимизировать процесс тестирования, сократить время, затрачиваемое на него, и улучшить качество программного обеспечения. Для достижения этой цели были использованы множественное представление, графы, теория массового обслуживания и имитационная модель методом Монте-Карло.

Результаты исследования показали, что создание формализованного представления процесса тестирования программного обеспечения в АСУ ТП с помощью графов и использование имитационной модели Монте-Карло может значительно ускорить и оптимизировать процесс тестирования, а также повысить качество продукта. Использование теории массового обслуживания позволило определить оптимальный размер команды и количество устройств в облаке для максимальной производительности.

В дальнейшем можно расширить модель автоматической проверкой кода и внедрением новых технологий. Также можно провести дополнительные исследования в этой области, чтобы улучшить эффективность тестирования и оптимизировать процесс разработки программного обеспечения. Разработка формализованного представления процесса тестирования программного обеспечения в АСУ ТП имеет большой потенциал для улучшения процесса тестирования и ускорения процесса разработки.

Список литературы

1. Обзор заказной разработки программного обеспечения // TAdviser [Электронный ресурс]. URL: https://www.tadviser.ru/index.php/Статья:Российский_рынок_заказной_разработки_ПО._Обзор_TAdviser (дата обращения: 05.04.2023).

2. Таран В.Н., Щербина И.О. Технологии автоматизации тестирования и их внедрения в процесс создания игровых приложений // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2020. № 4 (271). С. 75–86.

3. Паршина И.С., Фролов Е.Б. Разработка цифрового двойника производственной системы на базе современных цифровых технологий // Экономика промышленности. 2020. Т. 13, № 1. С. 29–34. DOI: 10.17073/2072-1633-2020-1-29-34.

4. Фролов Е.Б., Паршина И.С., Зайцев А.С., Климов А.С. Индустрия 4.0: «Цифровой двойник» как средство повыше-

ния эффективности производственной системы // Наукоемкие технологии в машиностроении. 2019. № 2 (92). С. 42–48.

5. Developing Services for the Wireless Internet: Software Development Processes // springer [Электронный ресурс]. URL: http://link.springer.com/chapter/10.1007%2F978-1-84628-589-9_2 (дата обращения: 20.02.2023).

6. Graph Theory with Applications // zib.de [Электронный ресурс]. URL: https://www.zib.de/groetschel/teaching/WS1314/BondyMurtyGTWA.pdf (дата обращения: 20.02.2023).

7. Basic Queueing Theory // irh.inf.unideb.hu [Электронный ресурс]. URL: https://irh.inf.unideb.hu/~jsztrik/education/16/SOR_Main_Angol.pdf (дата обращения: 20.02.2023).

8. Monte Carlo methods for risk analysis // risk-engineering.org [Электронный ресурс]. URL: https://risk-engineering.org/static/PDF/slides-monte-carlo.pdf (дата обращения: 20.02.2023).

9. Monte Carlo Simulation // Researchgate [Электронный ресурс]. URL: https://www.researchgate.net/publication/362478619_Monte_Carlo_Simulation (дата обращения: 20.02.2023).

10. Multi Model Monte Carlo with Python (MXMCPy) // NASA [Электронный ресурс]. URL: https://ntrs.nasa.gov/api/citations/20200003111/downloads/20200003111.pdf (дата обращения: 20.02.2023).