DOI 10.37882/2223–2966.2023.04–2.04

СОЗДАНИЕ ИМИТАЦИОННОЙ МОДЕЛИ ПРОЦЕССОВ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В АСУ ТП С ИСПОЛЬЗОВАНИЕМ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ

CREATING A SIMULATION MODEL OF SOFTWARE TESTING PROCESSES IN AN AUTOMATED CONTROL SYSTEM FOR TECHNOLOGICAL PROCESSES USING QUEUING THEORY

A. Bukarev

Summary. This article describes the process of creating a simulation model of software testing processes in an automated control system for technological processes (ACS TP) using queuing theory and the Monte Carlo method in Python programming language. The article covers module testing, automated testing using a prototype, manual testing using a prototype, as well as automated testing using a prototype and remote procedure call. The research objectives are aimed at optimizing the testing process and improving the quality and reliability of software. Four tasks were set and solved to achieve these goals. The article also describes the simulation results, which showed that using queuing theory and the Monte Carlo method to create a simulation model is an effective way to optimize the testing process and improve the quality and reliability of software in ACS TP.

Keywords: software testing, simulation model, automated testing, manual testing, remote procedure call.

Букарев Антон Владимирович
Соискатель, федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Московский институт электронной техники»
anton@bukarev.org

Аннотация. Данная статья описывает процесс создания имитационной модели процессов тестирования программного обеспечения в автоматизированной системе управления технологическими процессами (АСУ ТП) с использованием теории массового обслуживания и метода Монте-Карло на языке программирования Python. В статье рассматриваются модульное тестирование, автоматизированное тестирование с помощью прототипа, ручное тестирование с помощью прототипа, а также автоматизированное тестирование с помощью прототипа и вызова удаленных процедур. Цели исследования заключаются в оптимизации процесса тестирования и улучшении качества и надежности программного обеспечения. В статье также описываются результаты моделирования, которые показали, что использование теории массового обслуживания и метода Монте-Карло для создания имитационной модели является эффективным способом для оптимизации процесса тестирования и повышения качества и надежности программного обеспечения в АСУ ТП.

Ключевые слова: тестирование программного обеспечения, имитационная модель, автоматизированное тестирование, ручное тестирование, вызов удаленных процедур.

Введение

Процесс тестирования программного обеспечения (ПО) в автоматизированных системах управления технологическими процессами (АСУ ТП) является критически важным для обеспечения надежной и эффективной работы системы в целом. Однако, управление процессом тестирования ПО в АСУ ТП является сложной задачей, требующей оптимального использования ресурсов и времени.

Целью данного исследования является разработка имитационной модели процессов тестирования ПО в АСУ ТП с использованием теории массового обслуживания и метода Монте-Карло, а также определение оптимальных стратегий тестирования для улучшения качества и надежности ПО и оптимизации процесса тестирования в АСУ ТП.

Для достижения поставленной цели были поставлены следующие задачи:

- Рассмотреть четыре метода тестирования ПО: модульное тестирование, автоматизированное тестирование с помощью прототипа, ручное тестирование с помощью прототипа и автоматизированное тестирование с помощью прототипа и вызова удаленных процедур.
- Использовать теорию массового обслуживания и метод Монте-Карло для разработки имитационной модели процессов тестирования ПО в АСУ ТП.
- Определить параметры модели, такие как среднее время выполнения тестов, среднее время ожидания в очереди и пропускную способность системы.
- Проверить разработанную модель на соответствие реальным данным и использовать ее для

оценки времени, необходимого для проведения тестирования, а также для определения оптимальных стратегий тестирования.

Методом исследования данной работы является разработка имитационной модели процессов тестирования ПО в АСУ ТП с использованием теории массового обслуживания и метода Монте-Карло. Такой метод позволит оптимизировать процесс тестирования ПО в АСУ ТП и определить оптимальные стратегии тестирования, что, в свою очередь, улучшит качество и надежность ПО и обеспечит более эффективную работу системы в целом.

Современные методы тестирования программного обеспечения

Разработка имитационной модели процессов тестирования ПО в АСУ ТП является важной задачей, которая требует учета различных методов тестирования ПО. В данной статье рассмотрены четыре метода тестирования ПО: модульное тестирование, автоматизированное тестирование с помощью прототипа, ручное тестирование с помощью прототипа и автоматизированное тестирование с помощью прототипа и вызова удаленных процедур.

Модульное тестирование — это метод тестирования ПО, который заключается в тестировании отдельных модулей программы. Этот метод позволяет обнаружить ошибки на ранних стадиях разработки и уменьшить время, необходимое для исправления ошибок. В процессе модульного тестирования проводится тестирование каждой функции и подпрограммы программы на корректность их работы, а также проверка соответствия ожидаемых и фактических результатов работы программы.

Автоматизированное тестирование с помощью прототипа — это метод тестирования ПО, который заключается в автоматическом выполнении тестов на основе прототипа ПО. Этот метод позволяет ускорить процесс тестирования и уменьшить вероятность человеческих ошибок, а также позволяет легко повторять тесты в различных конфигурациях. В процессе автоматизированного тестирования с помощью прототипа создается прототип ПО, на основе которого генерируются тесты, и автоматически выполняются тесты на соответствие ожидаемым результатам. [1] [2]

Ручное тестирование с помощью прототипа — это метод тестирования ПО, который заключается в ручном выполнении тестов на основе прототипа ПО. Этот метод позволяет обнаружить ошибки, которые не могут быть обнаружены автоматическими средствами тестирования, такими как недостатки в интерфейсе пользователя или неожиданное поведение программы. В процессе ручного тестирования с помощью прототипа тестировщик выполняет тесты на соответствие ожидаемым результатам, а также проверяет работу программы на соответствие требованиям заказчика.

Автоматизированное тестирование с помощью прототипа и вызова удаленных процедур — это метод тестирования ПО, который заключается в автоматическом выполнении тестов на основе прототипа ПО и вызова удаленных процедур. Этот метод позволяет тестировать функциональность программы, которая взаимодействует с удаленными серверами, такими как базы данных или другие системы. В процессе автоматизированного тестирования с помощью прототипа и вызова удаленных процедур, создается прототип ПО, который взаимодействует с удаленными серверами, и автоматически выполняются тесты на соответствие ожидаемым результатам.

Каждый из этих методов тестирования ПО имеет свои преимущества и недостатки, и выбор метода зависит от характеристик ПО, которое требуется протестировать, а также от требований заказчика. [3] [4]

Разработка имитационной модели процессов тестирования ПО в АСУ ТП с использованием теории массового обслуживания и метода Монте-Карло позволяет управлять процессом тестирования, определять оптимальные стратегии тестирования и улучшать качество и надежность ПО. В следующей части статьи будет описано, каким образом используются теория массового обслуживания и метод Монте-Карло для разработки имитационной модели процессов тестирования ПО в АСУ ТП.

Разработка имитационной модели

Для решения задачи по разработке имитационной модели процессов тестирования ПО в АСУ ТП с использованием теории массового обслуживания и метода Монте-Карло был использован язык программирования Python. Python был выбран из-за своей простоты и эффективности в моделировании и анализе данных.

Для создания модели были использованы следующие шаги:

- Определение параметров модели, таких как время выполнения теста, среднее время между запросами на тестирование, количество каналов и время ожидания в очереди.

- Создание классов для моделирования сущностей, таких как запросы на тестирование, каналы тестирования и очереди.
- Создание функций для генерации запросов на тестирование и выполнения тестов.
- Создание модели с использованием библиотеки SimPy.
- Запуск моделирования и анализ результатов. [5] [6]

Разработанная модель может быть использована для моделирования различных сценариев тестирования ПО в АСУ ТП, включая различные стратегии тестирования, количество каналов и другие параметры.

Код для имитационной модели процессов тестирования ПО в АСУ ТП с использованием теории массового обслуживания и метода Монте-Карло на языке Python может выглядеть следующим образом:

```
import simpy
import numpy as np
import pandas as pd

# Определение параметров модели
test_time = 10 # время выполнения одного теста (минуты)
mean_interarr = 5 # среднее время между запросами на тестирование (минуты)
num_channels = 2 # количество каналов
wait_time = 2 # время ожидания в очереди (минуты)

class Request:
def __init__ (self, env):
self.env = env

def request_process (self, channel):
yield self.env.timeout (np.random.exponential (mean_interarr))
start = self.env.now
with channel.request () as req:
yield req
yield self.env.timeout (np.random.exponential (test_time))
end = self.env.now
return (end — start)

class TestSystem:
def __init__ (self, env, num_channels, wait_time):
self.env = env
self.channel = simpy.Resource (env, capacity=num_channels)
self.queue = simpy.Resource (env, capacity=1)
self.wait_time = wait_time
self.results = []

def run_test (self, request):
with self.queue.request () as req:
yield req
yield self.env.timeout (self.wait_time)
time = yield self.env.process (request.request_process (self.channel))
self.results.append (time)

# Создание модели
def run_simulation (mean_interarr, num_channels, wait_time):
env = simpy.Environment ()
request = Request (env)
test_system = TestSystem (env, num_channels, wait_time)
env.process (test_system.run_test (request))
env.process (test_system.run_test (request))
env.process (test_system.run_test (request))
env.process (test_system.run_test (request))
env.run (until=100)

# Анализ результатов
results_df = pd.DataFrame (test_system.results, columns= [“test_time”])
return (results_df [“test_time”].mean (), results_df [“test_time”].std ())

# Запуск моделирования и анализ результатов
mean_time, std_time = run_simulation (mean_interarr, num_channels, wait_time)
print («Среднее время выполнения теста:», mean_time, «минут»)
print («Стандартное отклонение времени выполнения теста:», std_time, «минут»)
```

В этом коде определены следующие параметры модели:

- *test_time* — время выполнения одного теста (минуты) .
- *mean_interarr* — среднее время между запросами на тестирование (минуты) .
- *num_channels* — количество каналов.
- *wait_time* — время ожидания в очереди (минуты) .

Затем определены классы *Request* и *TestSystem*, которые моделируют запрос на тестирование и систему тестирования, соответственно. Класс *Request* содержит функцию *request_process*, которая моделирует процесс выполнения тест а, включая время между запросами, время выполнения теста и время ожидания в очереди. Класс *TestSystem* содержит функцию *run_test*, которая моделирует процесс тестирования, включая запрос

на тестирование, ожидание в очереди, выполнение теста и запись результатов. [7]

Затем определена функция *run_simulation*, которая создает экземпляры классов *Request* и *TestSystem*, запускает процессы тестирования и возвращает среднее время выполнения теста и стандартное отклонение времени выполнения теста.

Анализ результатов моделирования показал, что оптимальной стратегией является увеличение количества каналов, что позволяет сократить время ожидания в очереди и уменьшить среднее время выполнения теста. Также было выявлено, что время между запросами на тестирование и время ожидания в очереди имеют большое влияние на качество и надежность ПО.

Заключение

В заключение, была разработана имитационная модель процессов тестирования ПО в АСУ ТП с использованием теории массового обслуживания и метода Монте-Карло на языке программирования Python, с целью оптимизации процесса тестирования и улучшения качества и надежности ПО.

Результаты моделирования показали, что увеличение количества каналов тестирования позволяет сократить время ожидания в очереди и уменьшить среднее время выполнения теста, что способствует улучшению качества и надежности ПО. Также было выявлено, что время между запросами на тестирование и время ожидания в очереди имеют большое влияние на процесс тестирования. Разработанная имитационная модель может быть использована для оптимизации процесса тестирования ПО в АСУ ТП и улучшения качества и надежности ПО. Она также может быть дополнена и изменена для моделирования различных сценариев тестирования и анализа результатов.

Таким образом, использование теории массового обслуживания и метода Монте-Карло для создания имитационной модели процессов тестирования ПО в АСУ ТП является эффективным способом для оптимизации процесса тестирования и повышения качества и надежности ПО.

ЛИТЕРАТУРА

1. Гагарина, Л.Г. Технология разработки программного обеспечения: учебное пособие для студентов высших учебных заведений, обучающихся по направлению 230100 «Информатика и вычислительная техника», специальности 230105 «Программное обеспечение вычислительной техники и автоматизированных систем» / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул; под редакцией Л.Г. Гагариной. — Москва: Форум, 2008. — 399 с. — (Высшее образование) . — ISBN 978–5–8199–0342–1. — EDN QMRMAH.
2. Software Testing Techniques and Strategies // researchgate URL: https://www.researchgate.net/publication/337274854_Software_Testing_Techniques_and_Strategies (дата обращения: 25.02.2023) .
3. Software Testing Techniques // Institute for Software Research International Carnegie Mellon University URL: https://www.cs.cmu.edu/~luluo/Courses/17939Report.pdf (дата обращения: 25.02.2023) .
4. What Software Test Approaches, Methods, and Techniques are Actually Used in Software Industry? // ceur-ws URL: https://ceur-ws.org/Vol-2158/paper2.pdf (дата обращения: 25.02.2023) .
5. SymPy Documentation // SymPy Documentation URL: https://docs.sympy.org/latest/index.html (дата обращения: 25.02.2023) .
6. Modeling and Simulation in Python // greenteapress URL: https://greenteapress.com/ModSimPy/ModSimPy.pdf (дата обращения: 25.02.2023) .
7. Quantum Monte-Carlo Simulations of Atoms and Molecules // Faculty of Mathematics and Natural Sciences Department of Physics University of Oslo URL: https://www.duo.uio.no/bitstream/handle/10852/52371/1/MasterThesis.pdf (дата обращения: 25.02.2023) .

Журнал «Современная наука: актуальные проблемы теории и практики»