УДК 622.621

Система автоматического тестирования технологического программного обеспечения систем микропроцессорной централизации

Наседкин Олег Андреевич — канд. техн. наук, доцент, руководитель испытательного центра. Научные интересы: методы доказательства безопасности железнодорожной автоматики, нормативное, организационное и методическое обеспечение процесса разработки систем на современной элементной базе. E-mail: nasedkin@crtc.spb.ru

Васильев Денис Анатольевич — старший научный сотрудник. Научные интересы: разработка средств испытаний программного обеспечения систем железнодорожной автоматики. E-mail: denvas@crtc.spb.ru

Бутузов Максим Алексеевич — старший научный сотрудник, руководитель экспертного отдела. Научные интересы: экспертная оценка безопасности систем, методическое обеспечение процесса испытаний на безопасность. E-mail: max@crtc.spb.ru

Испытательный центр железнодорожной автоматики и телемеханики, Петербургский государственный университет путей сообщения Императора Александра I, Россия, 190031, Санкт-Петербург, Московский пр., 9

Для цитирования: Наседкин О. А., Васильев Д. А., Бутузов М. А. Система автоматического тестирования технологического программного обеспечения систем микропроцессорной централизации // Интеллектуальные технологии на транспорте. 2025. № 2 (42). С. 93–102. DOI: 10.20295/2413-2527-2025-242-93-102

Аннотация. *Актуальность автоматизированного тестирования технологического программного обеспечения систем микропроцессорной централизации обусловлена критической ролью железнодорожной автоматики в обеспечении безопасности движения.* ***Введение:*** *с ростом сложности программных компонентов МПЦ методы ручного тестирования становятся недостаточно эффективными.* ***Цель:*** *разработка системы автоматического тестирования ТПО МПЦ на основе скриптового подхода, обеспечивающей полноту проверки функциональных требований и корректности реализации алгоритмов.* ***Методы:*** *гибридный подход, сочетающий скриптовый язык Lua для описания тестовых сценариев, виртуальную среду имитации работы напольного оборудования, автоматическую генерацию тестов и интеграцию с экспертной системой анализа протоколов.* ***Результаты:*** *модульная система тестирования, включающая: библиотеку тестовых скриптов, интерпретатор со специализированным API для взаимодействия с ТПО, механизмы автоматической валидации.* ***Практическая значимость:*** *доказана эффективность подхода на реальных конфигурациях МПЦ. Определены направления развития: интеграция с CI/CD, расширение покрытия тестами отказоустойчивости.* ***Обсуждение:*** *выявлены преимущества скриптового подхода: независимость тестов от конкретной станции, возможность повторного использования сценариев.*

Ключевые слова: *микропроцессорная централизация, технологическое ПО, автоматическое тестирование, скриптовые тесты, железнодорожная автоматика, язык программирования Lua, виртуальная среда*

2.3.5 *— математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей (технические науки);* ***2.3.6*** *— методы и системы защиты информации, информационная безопасность (технические науки)*

Введение

Обеспечение технологической безопасности [1] является одной из задач, которые реализуют системы железнодорожной автоматики (СЖАТ) в общем контексте обеспечения безопасности функционирования. Характерной особенностью современных СЖАТ является реализация средствами программного обеспечения большей части функций, в том числе и технологических. Поэтому в микропроцессорных СЖАТ (МП СЖАТ) программное обеспечение становится одним из ключевых компонентов системы, определяющих их уровень безопасности. В процессе проведения работ по обеспечению и доказательству безопасности МП СЖАТ значительная часть усилий должна быть направлена на оценку безопасности программного обеспечения.

Методы и средства оценки программного обеспечения

Широко используемый сегодня подход нормирования уровня безопасности разрабатываемых систем в виде вероятностных показателей позволяет учитывать стохастическую природу ошибок проектирования и программирования при реализации расчетных методов оценки безопасности. Однако в основе этой оценки лежит практическое подтверждение полноты и корректности реализации прикладных задач средствами программного обеспечения. В этих условиях процесс доказательства безопасности ПО в составе систем управления заключается в последовательном, поэтапном подтверждении правильности решаемой задачи от момента определения исходных требований к системе до их интеграции в единую систему.

Такой подход предполагает использование различных средств экспертизы и испытаний на всех стадиях разработки микропроцессорных систем ЖАТ. Для каждой из них должны быть установлены требования, на соответствие которым нужно проверять систему согласно принятой последовательности ее разработки.

В системах микропроцессорных централизаций технологические функции системы реализуются в технологическом программном обеспечении (ТПО), которое играет роль технологического «ядра» системы, принимая решение о допустимости выполнения тех или иных команд оператора в процессе управления станционными объектами и реализации тех или иных технологических алгоритмов. В этом случае под безопасностью ТПО будем понимать корректную постановку технологических функций системы и их корректную программную реализацию. Как правило, полноту и правильность технологических функций оценивают эксперты, анализируя функциональные требования, базовый состав которых отражен в соответствующих нормативных документах [2]. Принимаются во внимание также архитектурные аспекты реализации ТПО с целью возможности его корректного тиражирования и функционального расширения.

ТПО обладает высокой степенью сложности, поскольку реализует взаимозависимости и при этом реализуется в значительной мере независимо от базового и системного программного обеспечения. Такие свойства ТПО обуславливают возможность и необходимость проведения его тестирования не только в составе системы МПЦ, но и отдельно от нее, с использованием специально разработанных программных имитаторов. Функциональная структура имитатора для проведения испытаний технологического программного обеспечения приведена на рис. 1.

Основная задача проверки технологического ПО с использованием данного имитатора состоит в проверке правильности обработки ТПО поступающей информации (входные директивы дежурного по станции (ДСП), состояние напольных объектов) и полноты реализуемых ТПО функций.

Рассмотрим функциональные возможности представленной структуры, назначение ее основных модулей и их взаимодействие.

В процессе испытаний оператор, используя меню команд и отображение плана станции на экране, формирует директивы двух типов:

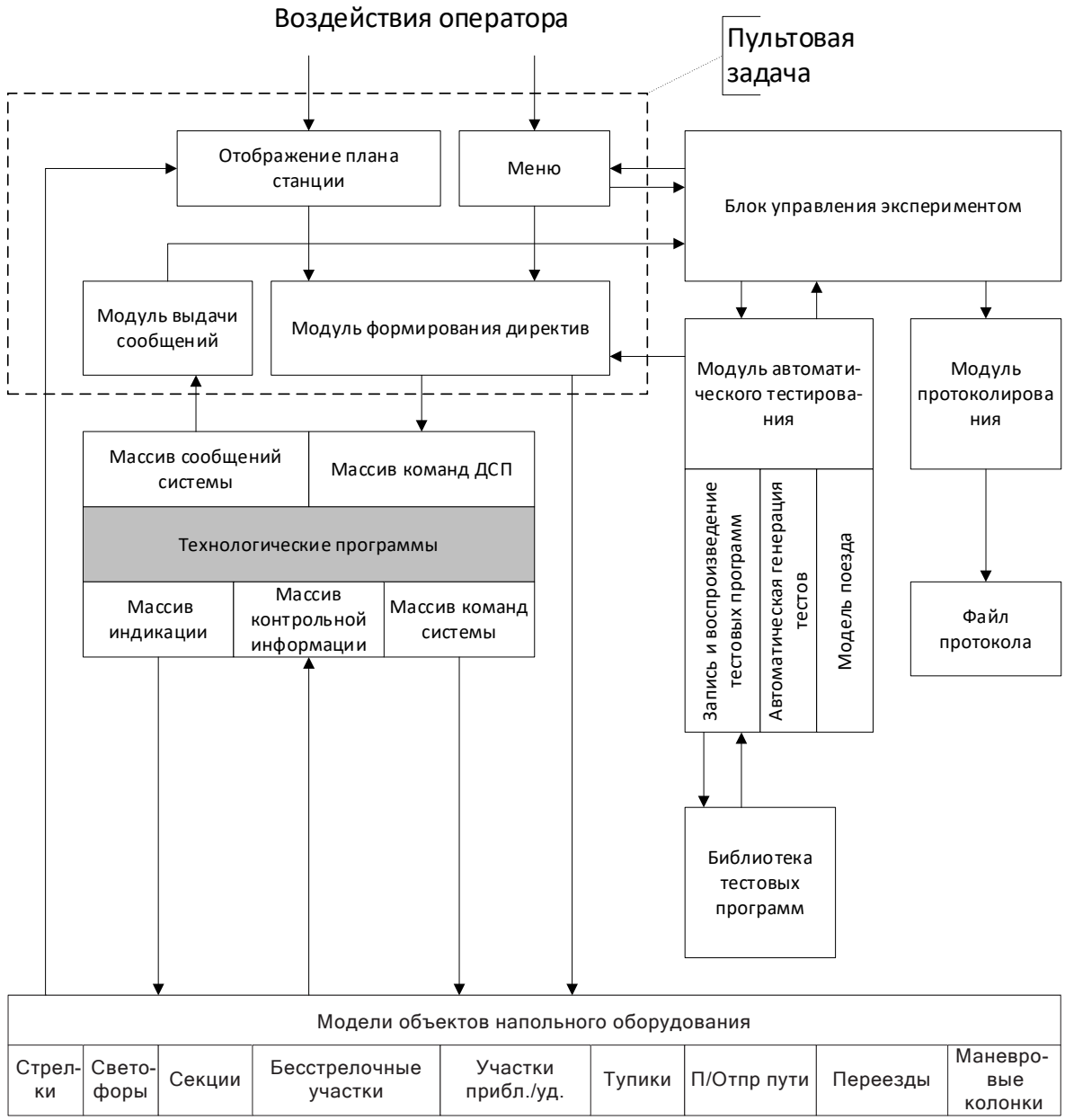

Рис. 1. Функциональная структура имитатора для проведения испытаний ТПО

– управляющих директив дежурного по станции;

– директив изменения состояния напольного оборудования.

Модуль выдачи сообщений обрабатывает информацию, получаемую из массива сообщений системы. Информация о сообщениях системы выводится на экран и в файл протокола.

Модули отображения плана станции, меню команд, формирования директив и выдачи сообщений в совокупности реализуют экранный интерфейс имитатора.

Модели объектов напольного оборудования имитируют работу реальных устройств, а также выполняют отображение их состояния на плане станции. Из системы МПЦ на модели объектов напольного оборудования поступает следующая информация:

– команды системы (через массив команд системы);

– информация для отображения состояния объектов (через массив индикации).

Модуль автоматического тестирования обеспечивает проведение испытаний в автоматическом

режиме, а также процесс формирования тестовых программ в ручном режиме работы. Формируемый файл тестовой программы включается в состав библиотеки тестовых программ. Тестовая программа, включенная в библиотеку, может быть воспроизведена неограниченное количество раз в автоматическом режиме работы. Кроме воспроизведения заранее записанной тестовой программы в имитаторе есть возможность автоматической генерации тестов по заданным алгоритмам (модуль автоматической генерации тестов) или автоматической имитации движения поезда по путевому развитию станции (модель поезда).

Модуль протоколирования обеспечивает ведение протокола испытаний. Возможны два режима протоколирования: основной и дополнительный. В основном режиме в протокол заносятся все формируемые в имитаторе директивы, изменения в состоянии напольного оборудования и сообщения, генерируемые технологическими программами. В дополнительном режиме, кроме того, фиксируются все изменения в массивах обмена информацией между имитатором и испытуемой системой, что позволяет проверить отсутствие несанкционированных управляющих воздействий со стороны технологических программ системы МПЦ.

Блок управления экспериментом управляет взаимодействием программных модулей имитатора в ручном и автоматическом режимах, а также обеспечивает режимы приостановки процесса имитации, пошаговой работы имитатора и изменение масштаба времени имитации.

Использование такого подхода, когда испытание ТПО проводится вне реального аппаратного и программного обеспечения — посредством средств виртуализации — имеет ряд преимуществ:

– возможность управления генерацией входных воздействий. Программные имитаторы задают произвольную последовательность входных данных в сочетании с наиболее неблагоприятными состояниями систем, что в условиях эксплуатации практически невозможно;

– возможность изменения интенсивности воздействия на испытываемый объект, что в значительной степени сокращает сроки испытаний;

– возможность программной имитации произвольной среды функционирования с учетом степени готовности системы на данном этапе разработки.

Примеры имитаторов для испытаний ТПО и специфика их построения описаны в [3].

Существуют различные подходы к тестированию программного обеспечения. Наиболее распространенным является деление подходов на тестирование по методу «черного ящика» и «белого ящика». В случае «черного ящика» для проектирования тестов используются внешние требования к ТПО, и целью является проверка соответствия реализации ТПО этим требованиям. При использовании подхода «белого ящика» тесты проектируются исходя из знания о структуре конкретной реализации ТПО.

Оба подхода применяются при тестировании ТПО, но использование метода «белого ящика» более типично для тестирования организацией-разработчиком в процессе реализации. При проведении экспертизы или декларировании ТПО при реализации процедуры оценки соответствия основным подходом является метод «черного ящика», реализующий функциональное тестирование, «процесс, нацеленный на выявление расхождений между поведением программы и внешней спецификацией» [4]. В случае с ТПО внешней спецификацией являются требования к технологическим функциям, реализуемым в ТПО. Например, при проверке реализации в ТПО технологической функции перевода стрелки (индивидуального или маршрутного) необходимо проверить, что для выдачи команды на перевод стрелки ТПО проверяет свободность секции, в которую входит данная стрелка, и неучастие ее в другом маршруте.

При тестировании ТПО возможны два основных режима: ручной, при котором тесты формируются тестировщиком вручную, и автоматический, при котором воздействия на ТПО и проверки результатов генерируются в соответствии с заданным алгоритмом в соответствии с выбранной методикой.

Автоматическое тестирование имеет ряд безусловных преимуществ перед ручным тестированием. По словам Б. Бейзера, «требование надежности обуславливает использование автоматического те-

стирования» [5]. В [6] отмечается, что «сравнение ручного и автоматизированного подхода показывает тенденцию современного тестирования, ориентирующуюся на максимальную автоматизацию процесса тестирования и генерацию тестового кода». Хотя построение автоматических тестов не всегда является тривиальной задачей, что усложняет внедрение автоматического тестирования, тем не менее несомненна необходимость максимально увеличить долю автоматических тестов в общем объеме испытаний.

При проведении автоматического тестирования необходимо обеспечить проверку корректности поведения тестируемого ТПО. Это может быть реализовано как в самом тесте, когда формируются тестовые воздействия с заранее известным результатом, так и с помощью внешней экспертной системы, получающей на вход протокол испытаний и выдающей предупреждения о некорректном поведении ТПО. Подробно тема применения экспертной программы в имитаторе для испытаний ТПО МПЦ рассмотрена в [7].

Можно выделить два основных подхода к автоматизации проектирования тестов: захват/воспроизведение данных и разработка специализированной программы — генератора тестов [5]. Оба этих метода применимы для тестирования ТПО МПЦ.

При использовании метода захвата/воспроизведения данных воздействия оператора в процессе ручного тестирования на имитаторе версии ТПО для конкретной станции, а также реакция ТПО записываются в виде тестового сценария, который может быть воспроизведен в автоматическом режиме. Преимуществом этого подхода является относительная простота получения теста. Основным недостатком этого подхода является привязка к конкретному объекту тестирования.

Специализированное программное обеспечение, порождающее тестовые воздействия по определенному алгоритму, существенно сложнее в разработке, но позволяет использовать написанные один раз тесты для любого типа ТПО на любой станции.

В составе имитатора для испытаний ТПО этот функционал реализуется системой автоматического тестирования (САТ).

Система автоматического тестирования

Система автоматического тестирования предназначена для автоматической генерации тестовых воздействий на ТПО на основе заданных алгоритмов тестирования.

На рис. 2 показана схема взаимодействия элементов САТ с имитатором для испытаний ТПО.

Рис. 2. Взаимодействие составных частей САТ с имитатором для испытаний ТПО

Основными составляющими САТ являются:

- набор тестовых скриптов, описывающих алгоритмы тестирования ТПО;
- интерпретатор тестовых скриптов;
- интерфейс тестирования имитатора.

Набор тестовых скриптов САТ представлен в виде файлов на скриптовом языке, распределенных по каталогам в соответствии с категориями тестов. В качестве языка для записи тестов используется язык программирования Lua.

Интерпретатор тестовых скриптов загружает скрипты, выбранные для проведения испытаний, и на их основе формирует команды и проверки для ТПО. Интерпретатор основан на стандартном интерпретаторе языка Lua, дополненном прикладным интерфейсом имитатора для тестирования.

Интерфейс тестирования имитатора предоставляет набор функций, доступных изнутри тестовых скриптов, используемых для воздействия на ТПО и проверки его реакции.

Тестовые скрипты

В САТ тесты представлены в виде тестовых скриптов. Тестовые скрипты представляют из себя текстовые файлы на языке программирования Lua, причем каждый тест описывается одним файлом. В каждом файле могут быть указаны характеристики применимости данного теста, в частности тип системы МПЦ, для которой данный тест является применимым, и станции, для которой описан данный тест. По умолчанию считается, что тест применим для любой станции и любой системы МПЦ.

Файлы тестовых скриптов могут быть распределены по директориям файловой системы, в этом случае имя директории задает общую категорию для каждого из включенных в нее тестовых файлов. Категориями могут быть, например, «тестирование перевода стрелок» или «проверка установки маршрутов». При запуске тестирования есть возможность выбрать категорию тестов, подлежащих исполнению.

Язык Lua выбран в качестве языка для тестовых скриптов в силу ряда преимуществ перед другими решениями. Язык Lua специально предназначен для встраивания в программы на языках C и C++. Он обладает простым, но выразительным синтаксисом, включающим кроме обычных конструкций структурного программирования такие возможности, как, например, замыкания, которые широко используются для реализации тестовых скриптов. Интерпретатор языка Lua компактен, доступен в виде исходных кодов на языке C, обладает высокой степенью совместимости с различными компиляторами и платформонезависимостью [8].

Предполагается, что на момент запуска теста имитатор и ТПО находятся в исходном состоянии, эквивалентном состоянию при начальном запуске имитатора, даже если выполняется несколько тестов подряд. Это может достигаться либо перезагрузкой ТПО и моделей в имитаторе после завершения каждого теста, либо самим построением тестового скрипта. Во втором случае, если по каким-либо причинам при завершении теста имитатор не пришел в исходное состояние, это проявится при выполнении следующих тестов, что может использоваться как дополнительная проверка корректности функционирования ТПО.

Тест содержит набор манипуляций и проверок над некоторым множеством объектов тестируемой станции. Выборка объектов для проверок производится по типам объектов («стрелка», «светофор», «маршрут» и т. п.) и атрибутам объектов (например, все стрелки, входящие в заданную секцию, или все стрелки, входящие в заданный маршрут). Тесты описываются в технологических терминах именованных состояний и команд для объектов контроля и управления и логических объектов, общих для всех систем МПЦ, а не специфичных для каждого типа ТПО внешних входов или выходов.

Команды и состояния объектов задаются строковыми литералами кириллицей с использованием принятой в отрасли терминологии, что делает тестовые скрипты понятными для специалистов в области железнодорожной автоматики, но не являющихся программистами. Это, в свою очередь, облегчает разработку тестов и проверку реализации в них функциональных требований.

Интерфейс тестирования

Интерфейс тестирования предоставляет набор функций имитатора для использования из тестовых скриптов. Эти функции реализуются специальным программным модулем на языке C++, входящим в состав имитатора, и регистрируются в интерпретаторе Lua. Параметрами функций интерфейса тестирования являются переменные, задающие объекты станции, и текстовые строки, задающие команды, состояния и связи объектов. При вызове функций с некорректными параметрами в протокол тестирования будет записано сообщение об ошибке.

Рассмотрим основные функции и конструкции, поддерживаемые интерфейсом тестирования.

- cmd() — выполнить команду определенного типа для определенного объекта. Имитатор поддерживает следующие типы команд:
  - «ДСП» — директива дежурного по станции (например, индивидуально перевести стрелку или установить маршрут).

Пример директивы дежурного на индивидуальный перевод в минус стрелки, заданной переменной *s*:

```
cmd(s, «ДСП», «перевести в минус»)
```

  - «КИ» — команда имитатора, позволяющая установить состояние модели объекта контроля и управления в имитаторе (например, ввести занятость секции).

Пример команды имитатора, устанавливающей занятость секции, заданной переменной *sp*:

```
cmd(sp, «КИ», «занять»)
```

- flag() — получить бинарное состояние объекта. Вид состояния объекта, который можно получить:
  - «СО» — физическое состояние объекта, состояние, формируемое моделями объектов контроля и управления, которое подается на входы ТПО.

Пример получения признака занятости секции, заданной переменной *sp*:

```
flag(sp, «СО», «занятость»)
```

  - «СОЛ» — логическое состояние объекта, т. е. состояние, формируемое ТПО на основании контрольной информации, сформированной моделями объектов контроля и управления в имитаторе (например, логический признак занятости секции).

Пример получения признака занятости логического объекта секции, заданной переменной *sp*:

```
flag(sp, «СОЛ», «занятость»)
```

- link() — получить для объекта ссылку на другой объект по имени связи.

Например, следующий код присваивает переменной *sp* ссылку на секцию для стрелки, заданной переменной *s*:

```
sp = link(s, "секция")
```

- run() — запустить процесс имитации на заданное время.

Пример команды run(), запускающей моделирование на 3000 миллисекунд:

```
run(3000)
```

- error() — прервать выполнение тестов и выдать сообщение об ошибке. Функция error() используется при проверке условий.

Например, в следующем фрагменте кода проверяется наличие минусового контроля стрелки после завершения перевода:

```
if not flag(sw, "СО", "минус") then
error("ошибка перевода в минус стрелки " .. name(sw))
end
```

- int_attr() — получить целочисленный атрибут объекта.

Например, следующий код запрашивает время перевода стрелки, заданной переменной *s*:

```
int_attr(s, "время перевода мс")
```

Важнейшей конструкцией интерфейса, используемой при построении скриптовых тестов, является итератор объектов тестирования. Эта конструкция основана на понятии итератора в том виде, в каком оно используется в языке Lua: «Итератор — это любая конструкция, которая позволяет вам перебирать элементы набора» [7]. Итератор в Lua реализован через механизм замыканий.

Итератор объектов реализован в виде функции objects_of_type(), получающей в качестве параметра строковый тип объекта. В тестовом скрипте итератор используется, как правило, совместно с оператором цикла для обхода всех объектов кон-

кретного типа. Например, для обхода всех стрелок используется следующая конструкция:

```
for s in objects_of_type("стрелка") do
end
```

При выполнении цикла for, в теле цикла переменная *s* будет последовательно ссылаться на каждую стрелку тестируемой станции.

Для обхода маршрутов в тестовых скриптах используется специальный итератор routes(), не получающий параметров.

Интерпретатор тестовых скриптов

Интерпретатор тестовых скриптов — это подсистема имитатора, выполняющая следующие функции:

1. Формирование среды выполнения тестов.
2. Загрузка и выполнение файлов тестовых скриптов в зависимости от заданных оператором параметров.
3. Протоколирование процесса тестирования и выдача сообщения о результатах тестирования, в том числе информации об ошибке в случае неудачного выполнения теста.

Формирование среды выполнения тестов предполагает создание и инициализацию интерпретатора языка Lua и регистрацию в нем функций интерфейса тестирования имитатора для последующего вызова их из тестовых скриптов, а также функций обработки ошибок.

CAT предоставляет разнообразные возможности для задания набора тестов, выполняемых при конкретном запуске автоматического тестирования. Кроме возможности выполнить конкретный тест или все тесты, есть возможность запустить все тесты одной категории или сформировать специальный файл со списком выполняемых тестов и передать его CAT. Набор таких файлов может быть использован для тестирования различных аспектов поведения ТПО. Выполнение набора тестов прекращается после первого появления сообщения об ошибке в одном из тестов. Выдача сообщения об ошибке происходит в тестовом скрипте при неудачном результате проверки условия. Сообщение об ошибке включает имя теста, в котором произошла ошибка, номер строки в файле теста и текстовое сообщение, заданное в тестовом скрипте в точке проверки условия.

Для протоколирования процесса тестирования интерпретатор использует подсистему протоколирования имитатора, в который встраивается CAT. Сообщения, вносимые в протокол, могут быть также отправлены на анализ в экспертную программу с целью выявления некорректного поведения ТПО.

Заключение

Специфика программного обеспечения как объекта разработки и экспертизы требует особого подхода в части инструментального обрамления процесса его создания и подтверждения реализации заявленных требований. Применение различных подходов к применению средств испытаний, в том числе и испытаний с использованием виртуальной внешней среды, позволяет получить интегральную оценку достигнутого уровня безопасности.

СПИСОК ИСТОЧНИКОВ

1. Понятийный аппарат экспертизы и испытаний на безопасность железнодорожной автоматики / Д. С. Марков, О. А. Наседкин, Д. А. Васильев, М. А. Бутузов // Автоматика на транспорте. 2018. Т. 4, № 1. С. 30–45.

2. ГОСТ 33894—2016. Системы железнодорожной автоматики и телемеханики на железнодорожных станциях. Требования безопасности и методы контроля = Railway automatics and telemechanics systems on railway stations. Safety requirements and methods of checking: межгосударственный стандарт: утвержден и введен в действие в качестве национального стандарта приказом Федерального агентства по техническому регулированию и метрологии от 31 марта 2017 г. № 233-ст: дата введения 2017-11-01. М.: Стандартинформ, 2017. 29 с.

3. Наседкин О. А., Васильев Д. А., Белоус А. М. Методическое и техническое обеспечение испытаний микропроцессорных систем // Автоматика, связь, информатика. 2013. № 12. С. 23–27.

4. Майерс Г., Баджетт Т., Сандлер К. Искусство тестирования программ. Третье издание = The Art of Software Testing. Third Edition / пер. с англ. и ред. А. Г. Гузикевича. М.: Вильямс, 2012. 272 с.

5. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем = Black-Box Testing. Techniques for Functional Testing of Software and Systems / пер. с англ. А. Раздобарина. СПб.: Питер, 2004. 318 с.

6. Котляров В. П., Коликова Т. В. Основы тестирования программного обеспечения: учебное пособие. М.: НОУ ИНТУИТ; БИНОМ. Лаборатория знаний. 2006. 285 с.

7. Экспертная программа для проведения испытаний технологического программного обеспечения систем микропроцессорной централизации / Д. А. Васильев, С. В. Гизлер, О. А. Наседкин, М. П. Шайфер // Развитие элементной базы и совершенствование методов построения устройств железнодорожной автоматики и телемеханики: сборник научных трудов / отв. ред. Вл. В. Сапожников. СПб.: ПГУПС, 2014. С. 39–42.

8. Иерузалимски Р. Программирование на языке Lua. Третье издание = Programming in Lua. Third Edition / пер. с англ. А. В. Борескова. М.: ДМК Пресс, 2016. 382 с.

Дата поступления: 13.05.2025
Решение о публикации: 24.05.2025

Automated Testing of Software Reliability for Computer-Based Interlocking systems

Oleg A. Nasedkin — PhD in Engineering, Associate Professor, Head of the Testing Center. Research interests: methods of proving the safety of railway automation; regulatory, organizational and methodological support for the development of systems based on modern elements. E-mail: nasedkin@crtc.spb.ru

Denis A. Vasiliev — Senior Researcher. Research interests: development of software testing tools for railway automation systems. E-mail: denvas@crtc.spb.ru

Maksim A. Butuzov — Senior Researcher, Head of the Expert Department. Research interests: expert assessment of system safety, methodological support of the safety testing process. E-mail: max@crtc.spb.ru

The Testing Center of Railway Automation and Telemechanics, Emperor Alexander I St. Petersburg State Transport University, 9, Moskovsky ave., Saint Petersburg, 190031, Russia

For citation: Nasedkin O. A., Vasiliev D. A., Butuzov M. A. Automated Testing of Software Reliability Computer-Based Interlocking systems. *Intellectual Technologies on Transport*, 2025, No. 2 (42), Pp. 93–102. DOI: 10.20295/2413-2527-2025-242-93-102. (In Russian)

Abstract. *Automated testing of technological software for computer-based interlocking systems is of critical importance in ensuring the safety of railway traffic.* ***Introduction:*** *as the CBI software components become more complex, so manual testing methods are no longer adequate.* ***Purpose:*** *to develop an automated testing system for the CBI software based on a scripting approach that ensures the overall verification of functional requirements and the correctness of the algorithms.* ***Methods:*** *a hybrid approach combining the Lua scripting language for describing test scenarios, a virtual environment for simulating the operation of outdoor equipment, automatic test generation, and the integration with an expert protocol analysis system.* ***Results:*** *a modular testing system that includes a library of test scripts, an interpreter with a specialized API for interacting with the computer software, and automatic validation mechanisms have been designed.* ***Practical significance:*** *the approach demonstrated its effectiveness in real CBI configurations. The development directions have been outlined as follows: integration with CI/CD and expansion of coverage with fault tolerance tests.* ***Discussion:*** *the research has revealed the advantages of the script approach. These include the independence of tests from a specific station and the possibility of reusing scripts.*

Keywords: *computer based interlocking, microproccessor interlocking, technological software, automated testing, script tests, railway automation, Lua programming language, virtual environment*

REFERENCES

1. Markov D. S., Nasedkin O. A., Vasil'ev D. A., Butuzov M. A. Ponyatiynyy apparat ekspertizy i ispytaniy na bezopasnost zheleznodorozhnoy avtomatiki [Definitions and Terminology of Expertise and Testing of Railway Automation for Safety], *Avtomatika na transporte [Automation on Transport]*, 2018, Vol. 4, No. 1, Pp. 30–45. (In Russian)

2. GOST 33894—2016. Sistemy zheleznodorozhnoy avtomatiki i telemekhaniki na zheleznodorozhnykh stantsiyakh. Trebovaniya bezopasnosti i metody kontrolya [GOST 33894—2016. Railway automatics and telemechanics systems on railway stations. Safety requirements and methods of checking]. Effective from November 01, 2017. Moscow, StandartInform Publishing House, 2017, 29 p. (In Russian)

3. Nasedkin O. A., Vasilyev D. A., Belous A. M. Metodicheskoe i tekhnicheskoe obespechenie ispytaniy mikroprotsessornykh system [Methodological and technical support for testing microprocessor systems], *Avtomatika, svyaz, informatika [Automation, Communications, Informatics]*, 2013, No. 12, Pp. 23–27. (In Russian)

4. Myers G. J., Badgett T., Sandler C. Iskusstvo testirovaniya programm. Tretye izdanie [The Art of Software Testing. Third Edition]. Moscow, Williams Publishing House, 2012, 272 p. (In Russian)

5. Beizer B. Testirovanie chernogo yashchika. Tekhnologii funktsionalnogo testirovaniya programmnogo obespecheniya i sistem [Black-Box Testing. Techniques for Functional Testing of Software and Systems]. Saint Petersburg, Piter Publishing House, 2004, 318 p. (In Russian)

6. Kotlyarov V. P., Kolikova T. V. Osnovy testirovaniya programmnogo obespecheniya: uchebnoe posobie [Fundamentals of Software Testing: a tutorial]. Moscow, INTUIT, BINOM. Laboratoriya znaniy, 2006, 285 p. (In Russian)

7. Vasiliev D. A., Ghizler S. V., Nasedkin O. A., Shaifer M. P. Ekspertnaya programma dlya provedeniya ispytaniy tekhnologicheskogo programmnogo obespecheniya sistem mikroprotsessornoy tsentralizatsii [The Expert Program for Computer Based Interlocking Application Software Test], *Razvitie elementnoy bazy i sovershenstvovanie metodov postroeniya ustroystv zheleznodorozhnoy avtomatiki i telemekhaniki: sbornik nauchnykh trudov [Component Base Development and Railway Automation and Remote Control Devices Design Methods Improvement: Scientific Proceeding]*. Saint Petersburg, St. Petersburg State Transport University, 2014, Pp. 39–42. (In Russian)

8. Ierusalimschy R. Programmirovanie na yazyke Lua. Tretye izdanie [Programming in Lua. Third Edition]. Moscow, DMK Press, 2016, 382 p. (In Russian)

Received: 13.05.2025
Accepted: 24.05.2025