Вестник РУДН. Серия: Инженерные исследования
RUDN Journal of Engineering Research

2025;26(2):155–167

ISSN 2312-8143 (Print); ISSN 2312-8151 (Online)
journals.rudn.ru/engineering-researches

DOI: 10.22363/2312-8143-2025-26-2-155-167
EDN: LZEWFU

Научная статья / Research article

Обеспечение живучести сложной технической системы в специальных условиях

В.В. Алексеев, Д.А. Иванов, И.Г. Рыжов✉

Российский университет дружбы народов, *Москва, Российская Федерация*
✉ ryzhov.ilgen@gmail.com

История статьи

Поступила в редакцию: 3 января 2025 г.
Доработана: 21 марта 2025 г.
Принята к публикации: 5 апреля 2025 г.

Заявление о конфликте интересов

Авторы заявляют об отсутствии конфликта интересов.

Аннотация. Цель исследования — разработка алгоритма обеспечения живучести сложной технической системы в специальных условиях. При проведении исследования были применены принципы и методы системного анализа, формальной верификации и математический аппарат темпоральной логики действий. В результате исследования был разработан алгоритм поиска логических ошибок в проектном решении и программном обеспечении сложной технической системы, базирующийся на темпоральной логике. Отличительными особенностями алгоритма являются возможность формальной верификации проектного решения по системе и наличие механизма обеспечения согласованности проектного решения и реализации. Алгоритм целесообразно применять для обеспечения живучести как вновь разрабатываемых систем на этапах проектирования и ввода в действие, так и уже существующих систем на этапе сопровождения.

Ключевые слова: алгоритм, верификация, информационный поток, темпоральная логика, соответствие проектному решению

Вклад авторов

Алексеев В.В. — общая концепция исследования; *Рыжов И.Г.* — анализ научной литературы, выводы и рекомендации, разработка алгоритма поиска логических ошибок в проектном решении и программном обеспечении, написание текста; *Иванов Д.А.* — методы системного анализа для обеспечения живучести сложной технической системы.

Благодарности

Авторы выражают благодарность Тимакову Алексею Анатольевичу, кандидату технических наук, доценту, за оказанную помощь при проведении данного исследования и Дмитриченко Михаилу Владимировичу за значительный вклад в разработку языка спецификаций TIFL.

Для цитирования

Алексеев В.В., Иванов Д.А., Рыжов И.Г. Обеспечение живучести сложной технической системы в специальных условиях // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. 2025. Т. 26. № 2. С. 155–167. http://doi.org/10.22363/2312-8143-2025-26-2-155-167

Ensuring the Survivability of a Complex Technical System Under Special Conditions

Vladimir V. Alekseev, Dmitry A. Ivanov, Ilya G. Ryzhov✉

RUDN University, *Moscow, Russian Federation*
✉ ryzhov.ilgen@gmail.com

Article history
Received: January 3, 2025
Revised: March 21, 2025
Accepted: April 5, 2025

Conflicts of interest
The authors declare that there is no conflict of interest.

Abstract. The objective of the research presented in this article was to develop an algorithm for ensuring the survivability of a complex technical system under special conditions. The principles and methods of system analysis, formal verification and mathematical apparatus of temporal logic of actions were applied in the research. As a result of the study, an algorithm for searching logical errors in the design solution and software of a complex technical system based on temporal logic was developed. The distinguishing features of the algorithm include the capacity for formal verification of the design solution within the system and the incorporation of a mechanism to ensure the consistency of the design solution and implementation. The application of this algorithm is recommended for the assurance of survivability, encompassing both newly developed systems during the design and commissioning stages, and existing systems during the maintenance stage.

Keywords: algorithm, verification, survivability, information flow, temporal logic, compliance with the design solution

Authors' contribution
Alekseev V.V. — the general concept of research; *Ryzhov I.G.* — analysis of scientific literature, conclusions and recommendations, development of an algorithm for finding logical errors in design solutions and software, writing the text; *Ivanov D.A.* — methods of system analysis to ensure the survivability of a complex technical system.

Acknowledgments
The authors would like to thank Alexey Anatolyevich Timakov, PhD, Associate Professor, for his assistance in conducting this research and Mikhail Vladimirovich Dmitrichenko for his significant contribution to the development of the TIFL specification language.

For citation
Alekseev VV, Ivanov DA, Ryzhov IG. Ensuring the Survivability of a complex technical system under special conditions. *RUDN Journal of Engineering Research.* 2025;26(2):155–167. (In Russ.) http://doi.org/10.22363/2312-8143-2025-26-2-155-167

Введение

В настоящее время трудно представить выполнение сложной технической системой (СТС) функциональных задач без использования информационных технологий. Отказы технических и коммуникационных средств, а также ошибки в проектном решении по системе и программном обеспечении (ПО) могут привести к нарушению качества функционирования системы в специальных условиях. Под специальными условиями понимается высокая вероятность воздействия на СТС деструктивных внешних факторов в процессе ее функционирования, приводящих к нарушению функционирования СТС в целом и, как следствие, снижению живучести.

На данный момент существует множество научных работ, посвященных восстановлению элементов системы и их связей после сбоев и отказов, вызванных внутренними факторами. Результаты, представленные в этих публикациях, не входят в рамки нашего исследования. Большой интерес с точки зрения обеспечения живучести СТС представляют исследования, нацеленные на обнаружение ошибок в проектном решении и ПО, возобновление работы системы в соответствии с предъявляемыми тре-

бованиями после воздействия деструктивных внешних факторов.

Современным направлением исследований является обеспечение живучести СТС, значительную роль в функционировании которых играют информационно-управляющие подсистемы (ИУП) [1]. ИУП управляет объектом с помощью программно реализованных алгоритмов, из чего следует, что функционирование ИУП СТС зависит от качества ПО и корректности алгоритмов. Цена ошибок, допущенных на этапах проектирования СТС (до непосредственной реализации), особенно высока [2].

На основе анализа систематизированных в [3] причин некорректной работы ПО целесообразно выделить в отдельную группу логические ошибки. Под логическими ошибками в данном исследовании будем понимать ошибки в постановке или формализации требований, а также в их реализации, после возникновения которых в результате функционирования системы параметры информационных потоков изменяются не так, как определено конструкторской, программной и эксплуатационной документациями. Следует отметить, что логическая ошибка может быть вызвана деструктивным внешним воздействием на СТС как со стороны злоумышленника, так и действиями легитимного пользователя системы.

При проведении исследования учтено, что информационный процесс представляет собой иерархически упорядоченную структуру информационных процессов и ошибка на низшем уровне иерархии может распространиться на высшие уровни. Ошибки могут привести к функциональному отказу системы самостоятельно или стать причиной получения некорректной или непредусмотренной для пользователя информации, после чего уже пользователь может совершить ошибку и вызвать функциональный отказ системы [1].

Обнаружение подавляющего большинства логических ошибок в ПО на практике выполняется с применением технологий статического и динамического анализа [4; 5]. Это объясняется относительно высокой точностью и незначительными требованиями к ресурсам, необходимым для их использования.

В [6] показано, что применение статических и динамических анализаторов для выявления логических ошибок, приводящих к нарушению параметров информационных потоков, не является эффективным. Автор предлагает механизм формальной верификации для контроля параметров информационных потоков, в основе которого лежит математический аппарат темпоральной логики. Существенным ограничением предложенного в работе механизма видится ориентированность на работу с хранимыми процедурами баз данных и позднее (после разработки ПО) выявление ошибок.

Противоположный, с точки зрения стадии проектирования, подход представлен в [7]. Описывается поиск логических ошибок в проектном решении с помощью построения DFD (Data Flow Diagram) диаграммы с использованием расширенной нотации Демарко [8] и запросов к логической базе знаний. Недостатком данного подхода является сложность интерпретации результата и неприменимость подхода на этапе проверок соответствия реализации техническому заданию[1].

Проведенный анализ доказывает актуальность разработки алгоритма обеспечения живучести СТС в специальных условиях на основе математического аппарата темпоральной логики, к достоинствам которого можно отнести формализацию зависящих от времени свойств, применимость для моделирования поведения системы, наличие инструментов автоматической проверки модели и визуализации последовательности принимаемых состояний модели системы.

1. Материалы и методы

В [7] представлена методика применения метода дедуктивной верификации для поиска ошибок на этапе проектирования системы.

[1] См.: ГОСТ Р 59793-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. Москва : Российский институт стандартизации, 2021. 8 с.; [9].

Особенность этой методики заключается в построении DFD-диаграммы с использованием расширенной нотации Демарко [8], которая затем транслируется в программу на языке логического программирования Prolog [9]. Верификация выполняется посредством запросов к логической базе знаний. В процессе верификации строится сеть распространения особых меток, присвоенных данным, и проводится анализ соответствия меток в сети требованиям к системе. Результатом является список всех обнаруженных нарушений.

В [10] представлен подход для анализа наличия логических ошибок, применяемый на этапе проектирования СТС, в основе которого лежит статический анализ DFD-диаграмм. В статье предложено использовать SecDFD (Security Data Flow Diagram), расширение DFD для построения модели системы. Аналогично [7] данным присваиваются метки, основываясь на требованиях к системе. По диаграмме SecDFD строится граф, в узлах которого хранятся метки данных. Контроль параметров информационных потоков выполняется за счет статической проверки соответствия графовой модели требованиям к системе. Соответствие может быть установлено после распространения меток с помощью обхода в ширину графа согласно семантике узлов.

А.М. Каннер продемонстрировал преимущества применения математического аппарата темпоральной логики и инструментальных средств автоматической верификации моделей (в публикации это модели безопасности), сформулированных на формальном языке, пригодном для верификации, для поиска ошибок в модели [11]. При верификации модели в качестве формального языка в [11] использован язык спецификаций TLA^+ (Temporal Logic of Actions) [12] и метод model checking [13].

Темпоральная логика — это логика, в которой истинное значение логических формул зависит от момента времени, в котором вычисляются значения этих формул. Основная идея темпоральной логики состоит в том, чтобы фиксировать только относительный порядок событий — фактически текущее, будущее и прошедшее время [13].

В [6] представлен алгоритм контроля параметров информационных потоков, проверяющий модель вычислений в системе, выполняемых с помощью Oracle/PLSQL блоков. Верификация модели происходит автоматически с использованием инструмента типа model checker — TLC (Temporal Logic Checker) [12]. Предложенный А.А. Тимаковым алгоритм выявляет запрещенные политиками безопасности информационные потоки. Анализ траекторий функционирования системы, полученных в результате работы TLC, позволяет сформулировать рекомендации по корректировке кода и изменению политики безопасности для устранения нарушений. Данный алгоритм способен обнаружить логические ошибки как на стадии ввода в действие, так и при сопровождении СТС [8].

Разделение анализа наличия логических ошибок на анализ на этапе проектирования и анализ на этапе реализации позволяет достичь композиционности [10]. Однако разделение анализа порождает проблему согласованности реализации и проектного решения. В [14] представлен подход, основанный на двухэтапном анализе наличия логических ошибок, включающий в себя механизм обеспечения согласованности реализации и проектного решения. Моделирование системы и проверка модели происходит аналогично [10]. Затем строится программная модель с помощью фреймворка GRaViTY [16] и выполняется автоматизированное сопоставление элементов программы с элементами SecDFD, с последующим статическим анализом соответствия реализации системы ее модели.

Описанный в [15] фреймворк GRaViTY способен выполнять проверку артефактов разработки системы, а также обеспечивать синхронизацию изменений между ними. Под артефактами разработки в публикации понимаются модели системы различной степени абстракции и программная реализация. Следует отметить простоту интеграции GRaViTY в суще-

ствующие системы, в которых отсутствуют актуальные модели, с помощью автоматизированного получения моделей из программной реализации. Для анализа наличия логических ошибок в артефактах разработки системы в фреймворке используются статические и динамические проверки.

Проведенный анализ современного состояния исследований в области контроля параметров информационных потоков для обеспечения живучести СТС в специальных условиях обосновывает предположение о целесообразности применения алгоритма в составе ИУП, обладающего следующими особенностями:

– в основе алгоритма лежит математический аппарат темпоральной логики;

– реализация алгоритма должна применяться на этапах проектирования, ввода в действие и сопровождения СТС;

– алгоритм способен контролировать параметры информационных потоков СТС;

– алгоритм выявляет несогласованность реализации ПО СТС и проектного решения.

2. Результаты

В качестве подхода для обеспечения живучести СТС в специальных условиях рассмотрим алгоритм поиска логических ошибок в проектном решении и программном обеспечении СТС (рис. 1), базирующийся на темпоральной логике.

Для обнаружения логических ошибок в проектном решении и несоответствия программной реализации проектному решению СТС необходимо выполнить моделирование в виде DFD-диаграммы функций СТС, работающих с наиболее чувствительной информацией непосредственно или через использование других функций.

Под чувствительной информацией будем понимать информацию, доступ к которой ограничен и раскрытие которой может привести систему к снижению эффективности функционирования системы [16]. Поиск функций и объектов, содержащих чувствительную информацию, осуществляется аналитиками системы исходя из понимания особенностей ее функционирования.

По полученной на прошлом шаге DFD-диаграмме необходимо написать TLA^+-спецификацию. Процесс написания TLA^+-спецификаций является трудоемкой и весьма нетривиальной задачей. Это объясняется тем, что TLA^+ ориентирован на описание поведения систем, но не на описание информационных потоков.

Для решения этой проблемы в алгоритм был добавлен шаг составления спецификации на языке TIFL (TIFL-спецификации), описывающей информационные потоки в системе.

Язык TIFL (Trivial Information Flow Language) обладает простым для описания информационных потоков синтаксисом и является доменно-специфичным языком (DSL — Domain Specific Language). Доменно-специфичные языки — это языки, ориентированные на решения проблем в конкретных областях в отличие от языков общего назначения [17]. Обоснованием разработки языка TIFL служит проведенный анализ исследования, опубликованного в [18]. Среди представленных автором доменно-специфичных языков в категории контроля параметров информационных потоков (IFC — Information Flow Control) не было выявлено существующего языка, который можно было бы применить для эффективного описания информационных потоков и последующей трансляции в TLA^+-спецификацию.

TIFL-спецификация является формальным представлением DFD-диаграммы, необходимым для проведения верификации параметров информационных потоков (обнаружения потоков информации, не соответствующих заданным критериям) в автоматическом режиме. В контексте верификации параметров информационных потоков нас интересуют источники и получатели информации, а также условия возникновения информационных потоков.

Проверка TIFL-спецификации выполняется посредством трансляции спецификации на языке TIFL в спецификацию на языке TLA^+

с помощью утилиты tifl2tla[2] и последующего запуска утилиты TLC.

При проигрывании модели с помощью TLC проверяется, что наблюдаемые результаты выполняемых СТС функций не противоречат правилам разграничения доступа к информационным ресурсам, предъявляемым к системе. Далее проверяемое условие будем называть инвариантом. Если в конце проверки модели обнаруживается, что объект системы начинает содержать более чувствительную информацию, то следует повторить проигрывание модели, считая, что данный объект в начальном состоянии содержит информацию, чей уровень чувствительности соответствует уровню текущего конечного состояния.

Повторять проигрывание моделей целесообразно до тех пор, пока при очередном проигрывании моделей не выявится нарушение инварианта или не перестанет меняться уровень чувствительности информации в объектах системы.

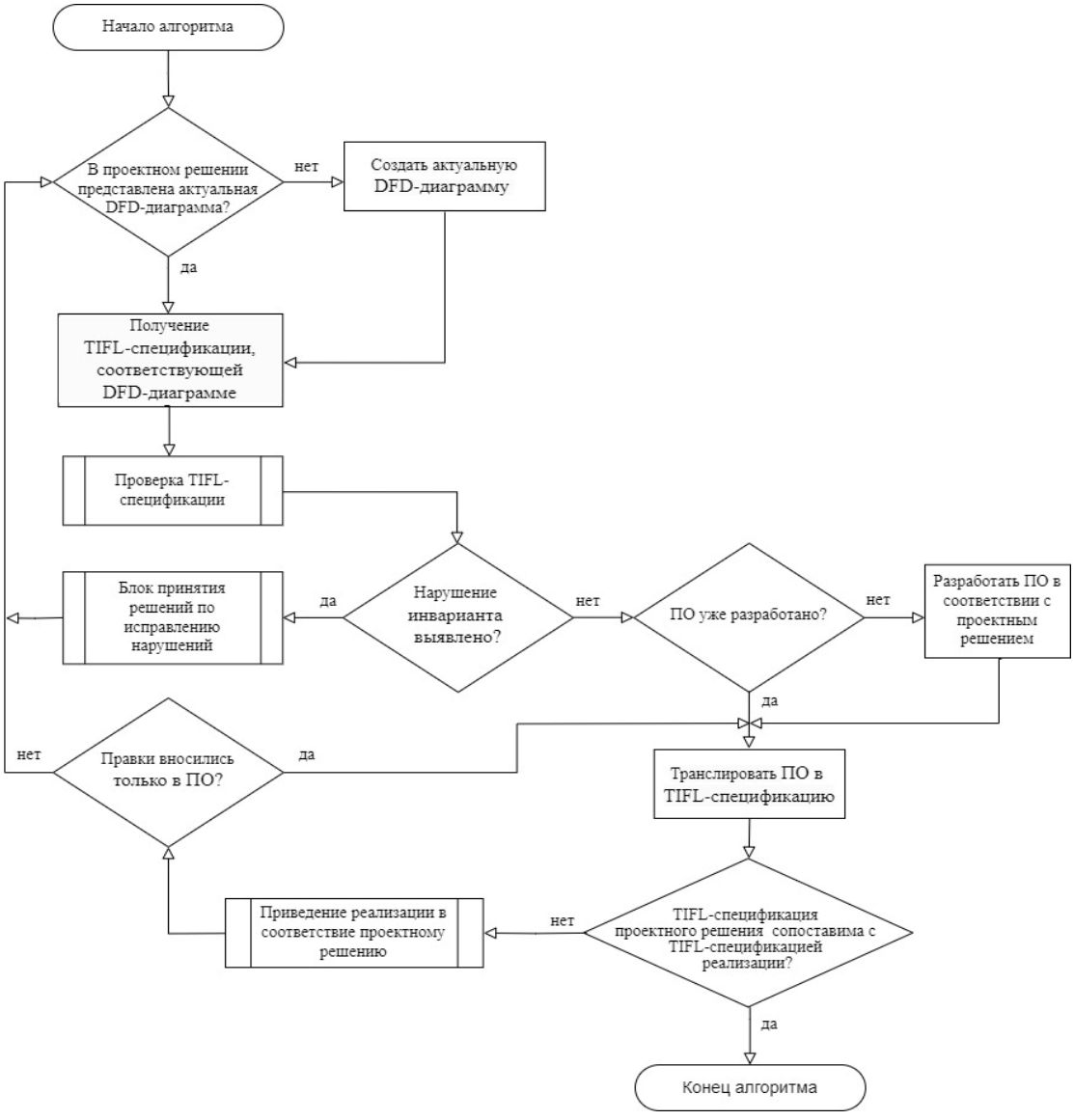

Рис. 1. Схема алгоритма поиска логических ошибок в проектном решении и программном обеспечении СТС

И с т о ч н и к: выполнено В.В. Алексеевым, Д.А. Ивановым, И.Г. Рыжовым

[2] Trivial Information Flow Language (TIFL). URL: https://github.com/IlyaRyzhov/tifl2tla (accessed: 27.11.2024).

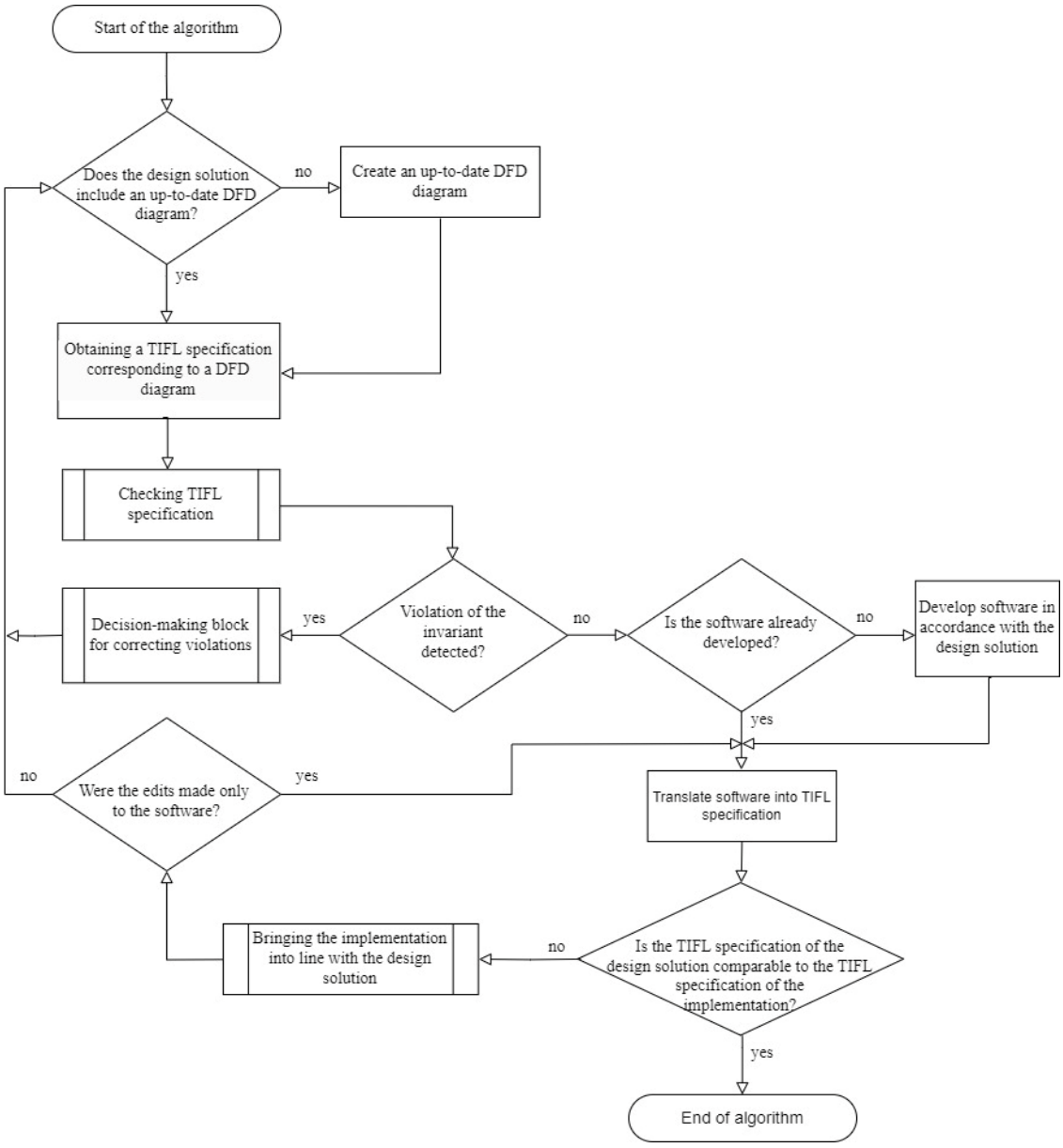

Figure 1. Flowchart of the algorithm for identifying logical errors in the design solution and software of a complex technical system
S o u r c e: by V.V. Alekseev, D.A. Ivanov, I.G. Ryzhov

В случае нарушения инварианта по траектории перемещения информации, полученной в результате работы TLC, можно отследить логические ошибки в проектном решении СТС. В качестве решений по исправлению нарушений, исходя из предположения о корректности работы утилиты трансляции tifl2tla и соответствия DFD-диаграммы TIFL-спецификации, может быть внесение правок в проектное решение или в спецификацию (в случае ложноположительного нарушения инварианта) и соответствующую ей DFD-диаграмму. Если изменениям подверглось проектное решение, то необходимо произвести актуализацию DFD-диаграммы с последующей генерацией спецификаций. После модификации спецификаций вне зависимости от принятого решения необходимо выполнить повторный запуск TLC.

Если же нарушений инварианта не выявлено, то проектное решение системы не содер-

жит логических ошибок, приводящих к нарушению параметров информационных потоков. В таком случае для систем, в которых отсутствует ПО, соответствующее проектному решению, разрешается приступить к его разработке.

Разработанное (для анализа вновь создаваемых систем) или существующее (для анализа эксплуатируемых систем) ПО необходимо транслировать в TIFL-спецификацию.

Особенностью этапа перевода на язык TIFL является то, что он происходит с учетом абстрагирования от реальных вычислений в системе. При описании информационных потоков на языке TIFL рекомендуется использовать как можно меньшее число локальных переменных. Так, например, вместо занесения суммы переменных *a* и *b* в локальную переменную *c* и последующей передачи этой переменной получателю информационного потока *r* можно сразу описать передачу информационному потоку *r* последовательности переменных *a* и *b*. Тем не менее необходимо с осторожностью подходить к объединению информационных потоков и учитывать допустимость отсутствия атомарности процесса, который будет иметь объединенный выходной информационный поток, в реализованной системе. Возможное влияние подобных способов описания спецификаций на точность анализа наличия логических ошибок в программном обеспечении СТС и разрешенные преобразования над спецификациями для приведения их к некоторому нормальному виду (процедура нормализации) требуют отдельных исследований и в данной статье не рассматриваются.

Представленный для демонстрации выразительности языка TIFL фрагмент программы (рис. 2) получен из фрагмента программы на языке Oracle/PLSQL (рис. 3):

– исключением синтаксических конструкций, которые не порождают информационные потоки и не добавляют условия для их возникновения;

– заменой описаний вычислений над строковыми и числовыми типами данных описаниями информационных потоков, представленных с помощью меток, соответствующих переменным и литералам, участвующим в вычислениях.

```
IncomeLevel -> {
    <g::employees.monthly_income><g::employees.name, p::name_in>?-><l::monthly_value>;
    (l::monthly_value, const) ? {
        <const> -> <l::ILevel>;
    } : (l::monthly_value, const) ? {
        <const> -> <l::ILevel>;
    } : (l::monthly_value, const) ? {
        <const> -> <l::ILevel>;
    } : {
        <const> -> <l::ILevel>;
    };
    ->l::ILevel;
}
```

Рис. 2. Фрагмент программы на языке TIFL, соответствующий коду на рис. 3

И с т о ч н и к: выполнено В.В. Алексеевым, Д.А. Ивановым, И.Г. Рыжовым

Figure 2. Program excerpt in the TIFL language corresponding to the code in Figure 3

S o u r c e: by V.V. Alekseev, D.A. Ivanov, I.G. Ryzhov

```
CREATE OR REPLACE Function IncomeLevel ( name_in IN varchar2 ) RETURN varchar2
IS
 monthly_value number(6);
 ILevel varchar2(20);
 cursor c1 is SELECT monthly_income FROM employees WHERE name = name_in;
BEGIN
 open c1;
 fetch c1 into monthly_value;
 close c1;
 IF monthly_value <= 4000 THEN
 ILevel := 'Low Income';
 ELSIF monthly_value > 4000 and monthly_value <= 7000 THEN
 ILevel := 'Avg Income';
 ELSIF monthly_value > 7000 and monthly_value <= 15000 THEN
 ILevel := 'Moderate Income';
 ELSE
 ILevel := 'High Income';
 END IF;
 RETURN ILevel;
END;
```

Рис. 3. Фрагмент программы на языке Oracle/PLSQL
И с т о ч н и к: составлено по Oracle/PLSQL

Figure 3. Program excerpt in Oracle/PLSQL language
S o u r c e: compiled by Oracle/PL SQL

ORACLE PL/SQL. Базы данных. IF-THEN-ELSE ОПЕРАТОР. URL: https://oracleplsql.ru/if-then-else.html (accessed: 27.11.2024).

Теперь, когда получены проверенная TIFL-спецификация проектного решения СТС и непроверенная TIFL-спецификация фактической реализации ПО, необходимо убедиться, что программная реализация соответствует проектному решению СТС. Для этого выполняется сопоставление TIFL-спецификаций, включающее процесс нормализации спецификаций.

Если спецификации на языке TIFL сопоставимы, то считается, что реализация соответствует проектному решению СТС и не содержит логических ошибок, приводящих к нарушению параметров информационных потоков. В противном случае необходимо привести реализацию в соответствие проектному решению.

На этапе приведения реализации в соответствие проектному решению могут как изменяться программная реализация, так и корректироваться проектное решение СТС. Если правки вносились только в ПО, то нужно заново транслировать ПО в TIFL-спецификацию и повторить шаг сопоставления TIFL-спецификаций.

Однако если в ходе приведения реализации в соответствие проектному решению подверглось изменениям проектное решение, то необходимо перейти к шагу актуализации DFD-диаграммы.

Работа алгоритма завершается, когда TIFL-спецификации проектного решения и ПО будут считаться сопоставимыми.

3. Обсуждение

Разработанный алгоритм поиска логических ошибок в проектном решении и программном обеспечении СТС рекомендуется применять для обеспечения живучести совместно с другими существующими подходами, обеспечивающими живучесть, позволяющими обнаруживать логические ошибки в ПО (например, статический и динамический анализы [4; 5]), и подходами, сфокусированными на поиск ошибок в проектных решениях (например, дедуктивной верификацией потоков, представленных в DFD [7]).

Для автоматизации написания TLA$^+$-спецификаций в [6] использована утилита [19], ориентированная на работу с программными блоками баз данных, написанными на языке

Oracle/PLSQL, что дополнительно ограничивает совершенствование подхода развитием вендорской технологии. Использование доменно-специфичного языка TIFL и утилиты трансляции tifl2tla в алгоритме, представленном в данной статье, позволяет не только упростить процесс получения TLA^+-спецификаций, но и абстрагироваться от использованных в реализации СТС технологий. Такое абстрагирование дает возможность независимо развиваться утилитам трансляции, доменно-специфичному языку TIFL и самому подходу к обеспечению живучести СТС.

Алгоритм поиска логических ошибок в проектном решении и программном обеспечении СТС, как и подходы, представленные в [14; 15], обнаруживает ошибки в проектном решении и программном обеспечении СТС, а также обеспечивает согласованность проектного решения и варианта реализации системы, но отличается использованием метода верификации моделей model checking, лежащего в основе алгоритма, который исчерпывающе проверяет пространство состояний модели системы [13].

Определение эффективности предложенного в данной статье алгоритма требует оценки живучести СТС. С точки зрения проведения оценки живучести интерес представляют подходы, описанные в [20–23].

Снижения сложности внедрения алгоритма в процесс создания систем и повышения удобства использования алгоритма можно достичь за счет автоматизации шагов алгоритма. Авторы видят наиболее приоритетными для автоматизации шаги создания актуальной DFD-диаграммы и TIFL-спецификаций, соответствующих DFD-диаграмме и ПО.

Для автоматизации создания DFD-диаграмм, соответствующих словесно сформулированным требованиям, перспективным выглядит использование больших языковых моделей с помощью метода RAG [24].

Автоматизация создания TIFL-спецификации, соответствующей DFD-диаграмме, может быть выполнена посредством специально разрабатываемого редактора, сохраняющего построенные диаграммы в виде TIFL-спецификаций и позволяющего отобразить визуально TIFL-спецификации.

TIFL-спецификацию, соответствующую ПО, следует получать автоматизированно, применяя утилиты трансляции, подобные [19], для исходных кодов, написанных на различных языках программирования.

Заключение

Определен подход и создан алгоритм, позволяющий обеспечивать живучесть СТС в специальных условиях. В ходе исследования установлено, что проблемами с точки зрения обеспечения живучести для СТС являются логические ошибки в требованиях и их реализации. В результате на основе анализа существующих подходов к обнаружению логических ошибок разработан алгоритм, базирующийся на темпоральной логике, который позволяет своевременно обнаруживать такие ошибки и способствует их дальнейшему устранению, тем самым обеспечивая живучесть системы. Данный алгоритм целесообразно применять на этапах проектирования, ввода в действие и сопровождения систем.

Дальнейшие исследования авторы видят в следующей очередности:

1) определение необходимой степени абстракции проверяемых моделей и влияние композиции моделей на точность анализа;

2) выбор подходящей DFD-нотации для построения моделей и разработка алгоритма сопоставления TIFL-спецификаций;

3) разработка алгоритма поддержки принятия решений по исправлению нарушений инварианта и устранению несоответствия реализации проектному решению;

4) определение эффективности предлагаемого в статье алгоритма посредством сравнения оценок живучести до и после его применения;

5) повышение уровня автоматизации шагов алгоритма, простоты внедре-ния алгоритма и удобства его использования.

Список литературы

1. *Шубинский И.Б.* Функциональная надежность информационных систем : методы анализа. Ульяновск : Печатный двор, 2012. 296 с. ISBN: 978-5-7572-0327-0 EDN: QMXPUD

2. *Гутгарц Р.Д.* Особенности проектирования и программирования при создании информационных систем // Программные продукты и системы. 2020. Т. 33. № 3. С. 385–395. https://doi.org/10.15827/0236-235x.131.385-395 EDN: MQLBTZ

3. *Белов А.С., Добрышин М.М., Горшков А.Н., Шугуров Д.Е.* Предложение по определению эксплуатационной надежности программного обеспечения сложных технических систем // Известия Тульского государственного университета. Технические науки. 2022. № 9. С. 143–148. https://doi.org/10.24412/2071-6168-2022-9-143-148 EDN: LUVYDO

4. *Воротникова Т.Ю.* Надежный код : статический анализ программного кода как средство повышения надежности программного обеспечения информационных систем // Информационные технологии в УИС. 2020. № 2. С. 22–27. EDN: YYTHON

5. *Аветисян А.И., Белеванцев А.А., Чукляев И.И.* Технологии статического и динамического анализа уязвимостей программного обеспечения // Вопросы кибербезопасности. 2014. № 3 (4). С. 20–28. EDN: SSYPXV

6. *Тимаков А.А.* Контроль информационных потоков в программных блоках баз данных на основе формальной верификации // Программирование. 2022. № 4. С. 27–49. https://doi.org/10.31857/S0132347422040057 EDN: WEMCXC

7. *Seifermann S., Heinrich R., Werle D., Reussner R.* Detecting violations of access control and information flow policies in data flow diagrams // Journal of Systems and Software. 2022. Vol. 184. P. 111138. https://doi.org/10.1016/j.jss.2021.111138 EDN: QLGGWA

8. *DeMarco T.* Structured analysis and system specification // Pioneers and Their Contributions to Software Engineering / M. Broy, E. Denert (eds.). Springer Berlin Heidelberg, 1979. P. 255–288. https://doi.org/10.1007/978-3-642-48354-7_9

9. *Warren D.S.* Introduction to prolog. Prolog: The Next 50 Years. Cham : Springer Nature Switzerland, 2023. P. 3–19. https://doi.org/10.1007/978-3-031-35254-6_1

10. *Tuma K., Scandariato R., Balliu M.* Flaws in flows: Unveiling design flaws via information flow analysis // 2019 IEEE International Conference on Software Architecture (ICSA). IEEE, 2019. P. 191–200. https://doi.org/10.1109/ICSA.2019.00028

11. *Каннер А.М.* Применение TLA+ нотации для описания модели изолированной программной среды субъектов доступа и ее дальнейшей верификации // Вопросы защиты информации. 2021. № 3. С. 8–11. https://doi.org/10.52190/2073-2600_2021_3_8 EDN: KXLLGD

12. *Lamport L.* Specifying systems: The TLA+ language and tools for hardware and software engineers. Boston : Addison–Wesley Publ.; 2002. 364 с. ISBN: 032114306X, 9780321143068

13. *Карпов Ю.Г.* Model checking. Верификация параллельных и распределенных программных систем. СПб. : БХВ-Петербург, 2010. 560 с. ISBN: 978-9775-0404-1

14. *Tuma K., Peldszus S., Strüber D., Scandariato R., Jürjens Ja.* Checking security compliance between models and code // Software and systems modeling. 2023. Vol. 22. No. 1. P. 273–296. https://doi.org/10.1007/s10270-022-00991-5 EDN: QYOWHY

15. *Peldszus S.* Security Compliance in Model-Driven Software Development // Ernst Denert Award for Software Engineering 2022: Practice Meets Foundations. Cham: Springer Nature Switzerland, 2024. P. 73–104. https://doi.org/10.1007/978-3-031-44412-8_4

16. *Коленченко Ю.В., Петров К.А., Емельянов Д.М., Исмагилов И.Р.* Разработка приложения-агента для предотвращения утечек чувствительной информации // Тинчуринские чтения-2020. Энергетика и цифровая трансформация. 2020. С. 64–67. EDN: GKLNCM

17. *Wąsowski A., Berger T.* Domain-Specific Languages. Springer International Publ.; 2023. 486 p. https://doi.org/10.1007/978-3-031-23669-3

18. *Krausz M., Peldszus S., Regazzoni F., Berger T., Güneysu T.* 120 Domain-Specific Languages for Security. 2024. URL: https://arxiv.org/abs/2408.06219 (accessed: 12.09.2024)

19. *Тимаков А.А., Рыжов И.Г., Лысиков А.В.* Генерация TLA+ спецификаций на основе программных блоков баз данных / Свидетельство о регистрации программы для ЭВМ RU 2023612260, 01.02.2023. Заявка № 2022686671 от 30.12.2022. EDN: JPAFHW

20. *Yakovlev A.V., Alekseev V.V., Volchikhina M.V., Petrenko S.V.* A Combinatorial Model for Determining Information Loss in Organizational and Technical Systems // Mathematics. 2022. Vol. 10. No. 19. P. 3448. https://doi.org/10.3390/math10193448 EDN: ZFYIJS

21. *Махутов Н.А., Петров В.П., Резников Д.О.* Оценка живучести сложных технических систем // Проблемы безопасности и чрезвычайных ситуаций. 2009. № 3. С. 47–66. EDN: MEGOYJ

22. *Черкесов Г.Н., Недосекин А.О., Виноградов В.В.* Анализ функциональной живучести структурно-сложных технических систем // Надежность. 2018. Т. 18. № 2. С. 17–24. https://doi.org/10.21683/1729-2646-2018-18-2-17-24 EDN: USQARX

23. *Алымов Н.* Некоторые вопросы оценки живучести технических систем // Инфокоммуникационные

технологии: актуальные вопросы цифровой экономики : сборник научных трудов I Международной научно-практической конференции. Екатеринбург, 2021. С. 188–192. EDN: VEKBVC

24. *Zhao S., Yang Y., Wang Z., He Zh., Qiu L.K., Qiu L.* Retrieval augmented generation (RAG) and beyond: A comprehensive survey on how to make your llms use external data more wisely. URL: https://arxiv.org/html/2409.14924v1 (accessed: 12.09.2024).

References

1. Shubinsky IB. *Functional reliability of information systems. Methods of analysis.* Ulyanovsk: Pechatny Dvor Publ.; 2012. (In Russ.) ISBN: 978-5-7572-0327-0 EDN: QMXPUD

2. Gutgarts RD. Features of design and programming when creating information systems. *Software products and systems*. 2020;33(3):385–395. (In Russ.) https://doi.org/10.15827/0236-235x.131.385-395 EDN: MQLBTZ

3. Belov AS. et al. Proposal for determining the operational reliability of software of complex technical systems. *Bulletin of Tula State University. Technical sciences.* 2022;(9):143–148. (In Russ.) https://doi.org/10.24412/2071-6168-2022-9-143-148 EDN: LUVYDO

4. Vorotnikova TYu. Reliable code: static analysis of program code as a means of increasing the reliability of software for information systems. *Information techno-logies in the UIS*. 2020;(2):22–27. (In Russ.) EDN: YYTHON

5. Avetisyan AI, Belevantsev AA, Chuklyaev II. The technologies of static and dynamic analyses of detecting software vulnerabilities. *Cybersecurity Issues.* 2014;3(4): 20–28. (In Russ.) EDN: SSYPXV

6. Timakov AA. Control of information flows in software blocks of databases based on formal verification. *Programming and Computer Software.* 2022;48(4):265–285. https://doi.org/10.1134/s0361768822040053 EDN: BATUIZ

7. Seifermann S, Heinrich R, Werle D, Reussner R. Detecting violations of access control and information flow policies in data flow diagrams. *Journal of Systems and Software*. 2022;184:111138. https://doi.org/10.1016/j.jss.2021.111138 EDN: QLGGWA

8. DeMarco T. *Structured analysis and system specification.* In: Broy, M., Denert, E. (eds.) Pioneers and Their Contributions to Software Engineering. Springer Berlin Heidelberg; 1979. P. 255–288. https://doi.org/10.1007/978-3-642-48354-7_9

9. Warren DS. Introduction to prolog. *Prolog: The Next 50 Years.* Cham: Springer Nature Switzerland; 2023. P. 3–19. https://doi.org/10.1007/978-3-031-35254-6_1

10. Tuma K, Scandariato R, Balliu M. Flaws in flows: Unveiling design flaws via information flow analysis. *2019 IEEE International Conference on Software Architecture (ICSA*). 2019. p. 191–200. https://doi.org/10.1109/ICSA.2019.00028

11. Kanner AM. Application of TLA+ notation to describe the model of an isolated software environment of access subjects and its further verification. *Information Security Issues*. 2021;(3):8–11. https://doi.org/10.52190/2073-2600_2021_3_8 EDN: KXLLGD

12. Lamport L. *Specifying systems: the TLA+ language and tools for hardware and software engineers.* Boston: Addison–Wesley Publ.; 2002. ISBN 032114306X, 978-0-32114-306-8

13. Karpov YuG. *Model checking. Verification of parallel and distributed software systems*. SPb.: BHV-Petersburg; 2010. (In Russ.) ISBN 978-9775-0404-1

14. Tuma K, Peldszus S, Strüber D, Scandariato R, Jürjens Ja. Checking security compliance between models and code. *Software and systems modeling*. 2023;22(1): 273–296. https://doi.org/10.1007/s10270-022-00991-5 EDN: QYOWHY

15. Peldszus S. Security Compliance in Model-Driven Software Development. *Ernst Denert Award for Software Engineering 2022: Practice Meets Foundations*. Cham: Springer Nature Switzerland; 2024. p. 73–104. https://doi.org/10.1007/978-3-031-44412-8_4

16. Kolenchenko YuV, Petrov KA, Yemelyanov DM, Ismagilov IR. Development of an agent application to prevent leaks of sensitive information. *Tinchurin readings-2020. Energy and digital transformation.* 2020:64–67. (In Russ.) EDN: GKLNCM

17. Wąsowski A, Berger T. *Domain-Specific Languages*. Springer International Publ.; 2023. https://doi.org/10.1007/978-3-031-23669-3

18. Krausz M, Peldszus S, Regazzoni F, Berger T, Güneysu T. *120 Domain-Specific Languages for Security.* 2024. Available from: https://arxiv.org/abs/2408.06219 (accessed: 12.09.2024)

19. Timakov AA, Ryzhov IG, Lysikov AV. *Certificate of state registration of computer program No. 2023612260 Russian Federation*. Generation of TLA+ specifications based on program blocks of databases: No. 2022686671: declared 30.12.2022: published 01.02.2023.

20. Yakovlev AV, Alekseev VV, Volchikhina MV, Petrenko SV. A Combinatorial Model for Determining Information Loss in Organizational and Technical Systems. *Mathematics.* 2022;10(19):3448. https://doi.org/10.3390/math10193448 EDN: ZFYIJS

21. Makhutov NA, Petrov VP, Reznikov DO. Assessment of survivability of complex technical systems. *Problems of safety and emergency situations.* 2009;(3): 47–66. (In Russ.) EDN: MEGOYJ

22. Cherkesov GN, Nedosekin AO, Vinogradov VV. Analysis of the functional survivability of structurally

complex technical systems. *Reliability*. 2018;18(2):17–24. (In Russ.) https://doi.org/10.21683/1729-2646-2018-18-2-17-24 EDN: USQARX

23. Alymov N. Some issues of assessing the survivability of technical systems. *Infocommunication technologies: current issues of the digital economy. Collection of scientific papers of the I International Scientific and Practical Conference*. Ekaterinburg: Reliability; 2021. p. 188–192. (In Russ.) EDN: VEKBVC

24. Zhao S, Yang Y, Wang Z, He Zh, Qiu LK, Qiu L. *Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your llms use external data more wisely*. 2024. Available from: https://arxiv.org/html/2409.14924v1 (accessed: 12.09.2024)

Сведения об авторах

Алексеев Владимир Витальевич, доктор технических наук, профессор кафедры механики и процессов управления, инженерная академия, Российский университет дружбы народов, Российская Федерация, 117198, г. Москва, ул. Миклухо-Маклая, д. 6; eLIBRARY SPIN-код: 9320-9713, ORCID: 0000-0002-0398-4426; e-mail: vvalex1961@mail.ru

Иванов Дмитрий Александрович, аспирант кафедры механики и процессов управления, инженерная академия, Российский университет дружбы народов, Российская Федерация, 117198, г. Москва, ул. Миклухо-Маклая, д. 6; eLIBRARY SPIN-код: 4761-2024, ORCID: 0009-0004-0182-5095; e-mail: 1142230113@pfur.ru

Рыжов Илья Геннадьевич, аспирант кафедры механики и процессов управления, инженерная академия, Российский университет дружбы народов, Российская Федерация, 117198, г. Москва, ул. Миклухо-Маклая, д. 6; eLIBRARY SPIN-код: 1818-9990, ORCID: 0000-0001-6014-6982; e-mail: ryzhov.ilgen@gmail.com

About the authors

Vladimir V. Alekseev, Doctor of Sciences (Techn.), Professor of the Department of the Department of Mechanics and Control Processes, Academy of Engineering, RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation; eLIBRARY SPIN-code: 9320-9713, ORCID: 0000-0002-0398-4426; e-mail: vvalex1961@mail.ru

Dmitry A. Ivanov, Postgraduate student of the Department of Mechanics and Control Processes, Academy of Engineering, RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation; eLIBRARY SPIN-code: 4761-2024, ORCID: 0009-0004-0182-5095; e-mail: 1142230113@pfur.ru.

Ilya G. Ryzhov, Postgraduate student of the Department of Mechanics and Control Processes, Academy of Engineering, RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation; eLIBRARY SPIN-code: 1818-9990, ORCID: 0000-0001-6014-6982; e-mail: ryzhov.ilgen@gmail.com