УДК 004.312.466

Б. В. СИВКО, магистр технических наук, Белорусский государственный университет транспорта, г. Гомель

МЕТОДИКА ДОКАЗАТЕЛЬСТВА БЕЗОПАСНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ ЖЕЛЕЗНОДОРОЖНОЙ АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

Предложена методика доказательства безопасности программного обеспечения микропроцессорных систем железнодорожной автоматики и телемеханики с помощью формальных методов, в которой рассматриваются валидация и верификация программного обеспечения в неразрывной связи с аппаратными средствами анализируемых систем. Она базируется на опыте доказательства безопасности железнодорожных устройств и на мировой теории и практике работы с критически важными объектами информатизации. Методика позволяет проводить анализ на функциональную, информационную и кибернетическую безопасность произвольных железнодорожных аппаратно-программных комплексов с объемом программного обеспечения до 10 KLOC.

Доказательство корректности является одним из формальных методов верификации программного обеспечения (ПО). Оно успешно применяется на Белорусской железной дороге и используется как во время проектирования и разработки аппаратно-программных комплексов (АПК), так и для систем, находящихся в эксплуатации [1, 2]. Достоинством метода является то, что с помощью него могут быть найдены и устранены ошибки, которые не могут быть обнаружены другими методами [3]. Однако доказательство корректности – это сложный и длительный процесс, и он подвержен влиянию человеческого фактора, что является его недостатком. Поэтому для совершенствования и облегчения доказательства корректности необходимо создавать и применять более совершенные средства, к которым относится рассматриваемая в данной статье методика (далее – методика).

Опыт испытаний железнодорожных устройств на безопасность функционирования в лаборатории БЭМС ТС БелГУТа показывает, что большинство анализируемых СЖАТ отличаются своим разнообразием и не обладают большой сложностью (ПО размером до 10 KLOC [4]). На основании особенностей микроэлектронных устройств, используемых на железной дороге, возможно и актуально создание методов и средств, позволяющих эффективно проводить доказательство безопасности, что и предлагает методика.

Методика базируется как на опыте доказательства безопасности железнодорожных АПК, относящихся к критически важным объектам информатизации (КВОИ), так и на общем теоретическом базисе разработки и верификации систем, критичных к безопасности (*safety-critical systems*) [5]. Первое подразумевает, что она непосредственно создавалась для доказательства безопасности ПО СЖАТ и готова к применению, второе – что она руководствуется и оперирует общими теоретическими принципами и лучшими практиками (*best practices*) как ПО, так и инженерии безопасных систем.

ПО безопасных систем не рекомендуется рассматривать отдельно от аппаратных средств, так как безопасность и корректность поведения является свойством всего АПК, поэтому при доказательстве безопасности рассматривается ПО в интеграции с аппаратным обеспечением системы [6]. В методике рассматривается не только верификация, но и валидация, которая является одним из основных источников ошибок [6, 7]. Методика базируется на основе анализа свойств микропроцессорных СЖАТ, так как в настоящее время показано, что чем более специализирован инструмент поиска ошибок, тем он более эффективен [8, 9]. Поэтому её применение для железнодорожных устройств эффективно, а вне этой области нецелесообразно.

Исходя из требований стандарта IEC 61508, одной из задач является доказательство безопасности систем, выполняемых сторонними организациями, когда проводится полный цикл обратной разработки (*reverse-engineering*) [10]. В связи с этим методика основывается не на определенной модели, а предполагает, что ПО может быть любым. Это ограничивает допустимую сложность ПО, проверяемого на безопасность, до систем в тысячи строк исходного кода (менее 10 KLOC). Для более сложных систем рекомендуется применение специализированных подходов. Например, доказательство корректности ПО объемом порядка 100 KLOC может базироваться на применении строгого контрактного программирования (*design by contract*) с последующей автоматизированной верификацией [11, 12].

Таким образом, методика позволяет выполнять анализ на безопасность ПО СЖАТ размером до 10 KLOC при наличии доступа к исходному коду.

Общие положения. Одной из особенностей доказательства корректности ПО, и, как следствие, доказательства безопасности микропроцессорных СЖАТ является то, что невозможно разработать универсальный метод, позволяющий выполнять формальную верификацию произвольного свойства АПК [13]. Данное ограничение является следствием из общей теории вычислительных систем, в частности, проблемы остановки и проблемы разрешения, а существующие способы автоматического доказательства не способны доказывать свойства программ рассматриваемой сложности (единицы KLOC) [14, 15]. Следовательно, невозможно создать универсальную методику в виде алгоритма действий, позволяющую доказывать истинность произвольной функции в общем случае, поэтому методика базируется на классических методах доказательства корректности, способных к верификации любого ПО [16]. Однако данные методы сложны в применении для программ объемом более тысячи строк кода (1 KLOC), и в связи с этим предлагается способ, заключающийся в анализе общих свойств систем предметной области, позволяющий создать инструменты, которые значительно уменьшат сложность доказательства.

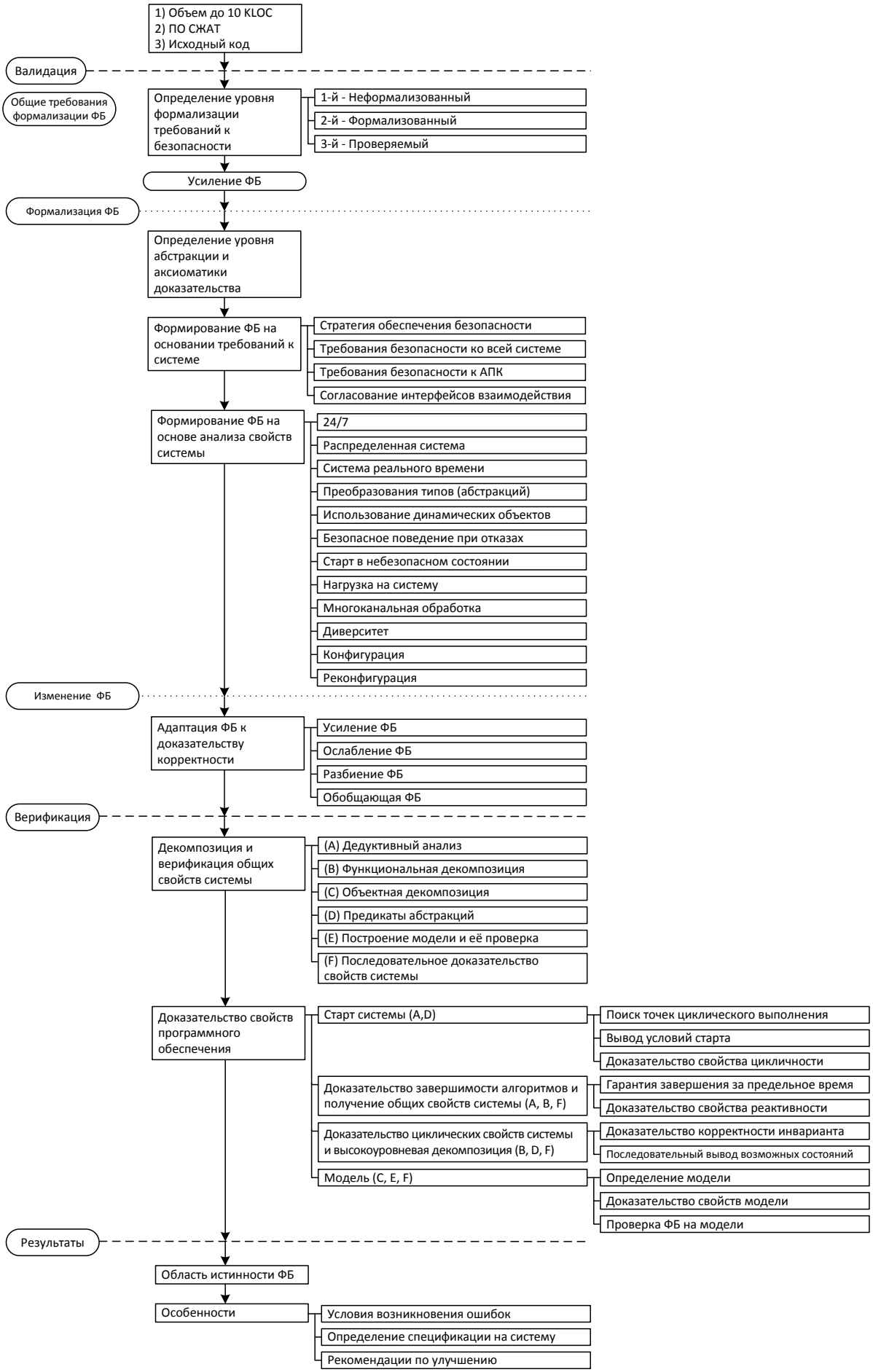

Рисунок 1 – Этапы доказательства безопасности

Доказательство безопасности разделяется на этапы валидации и верификации (рисунок 1). На первом происходит определение функции безопасности (ФБ), а на втором проверяется, что она в рассматриваемом АПК всегда выполняется [17, 18]. В методике данные этапы считаются независимыми, и при этом сначала проводится валидация, а в дальнейшем – верификация, которая выполняется на основании полученной на предыдущем этапе ФБ. Анализ на безопасность рекомендуется выполнять по порядку (см. рисунок 1), так как некоторые шаги могут быть выполнены только после предыдущих.

Валидация. В методике представляет собой процесс независимого определения ФБ, которая в дальнейшем подлежит верификации. Проблемы валидации заключаются в том, что созданные спецификации на рассматриваемое устройство не всегда верны, их ошибки часто проявляются, наносят значительный ущерб и их сложно исправить [6].

Установка общих требований к формализации ФБ. С точки зрения доказательства корректности все рассматриваемые понятия (свойства, функции и пр.) должны быть формализованы, так как в противном случае доказать что-либо с помощью любых формальных методов будет невозможно [19].

Опыт верификации микроэлектронных СЖАТ выявил ряд ситуаций, когда задание ФБ в формализованном виде для некоторых свойств не представляется возможным, например, для описания корректности работы устройств индикации [20]. В этом случае решением является формализованное описание свойств рассматриваемого понятия в качестве задачи доказательства корректности и определения ФБ, а в дальнейшем делается экспертное заключение о том, является ли верифицированное свойство безопасным или нет. Для этого сначала посредством доказательства корректности определяются формализованные свойства системы, а в последующем, на их основании, делается вывод о безопасности системы.

В методике предлагается три уровня формализации (таблица 1). Определить уровень формализации можно так, как показано на рисунке 2.

Таблица 1 – **Уровни формализации**

Название уровня	Пример
Неформализованный	Запрещается перевод стрелки под составом
Формализованный	Если хранимое в памяти значение о состоянии свободности секции стрелки обновлялось и проверялось за последние 400 мс, и при этом получена команда на перевод стрелки, то только в этом случае разрешена её дальнейшая передача на исполнение
Проверяемый	Имеется возможность проверки формализованного свойства посредством тестирования

Первый уровень является вербальной формулировкой. Его недостатком является то, что необходимый впоследствии переход к формальному уровню неоднозначен, что может вести к проблемам безопасности и сложностям при доказательстве [21]. Поэтому переход ко второму уровню делать необходимо, и выполнять его как можно раньше.

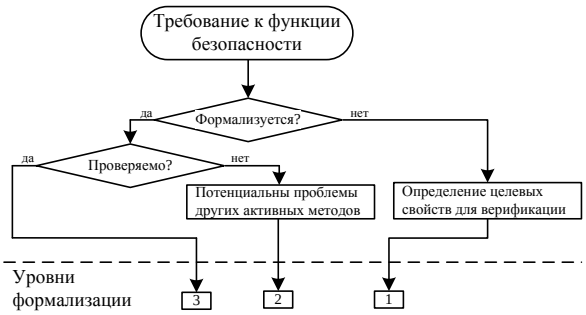

Рисунок 2 – Определение уровня формализации требования к безопасности

Неформализованный уровень появляется изначальным при формулировке, удобен в общении, не требует существенных затрат и является абстрактным [15]. Кроме того, он широко используется в нормативных документах, таких как ГОСТ, ПТЭ, ИСИ и др. Однако в случае рассмотрения конкретного устройства и его доказательства корректности требуется формализация, которая устраняет неоднозначности, улучшает понимание и может быть помещена в некоторую формальную систему для последующего доказательства корректности [22].

Формализованный уровень обладает тем свойством, что он может быть записан в виде знаков некоторой формальной системы. Например, если рассматривается момент времени t, и при этом в системе хранится свободность секции в виде значения $S(t)$, а функции времени от последней проверки объекта x и от времени последнего обновления могут быть выражены в виде $f(x, t)$ и $g(x, t)$ соответственно, то возможность передачи команды на перевод стрелки может быть представлена в виде формулы

$$(f(S(t), t) \le 400) \wedge (g(S(t), t) \le 400). \qquad (1)$$

Однако не всегда формализованный вариант может быть проверяемым (полностью или частично), т.е. соответствовать третьему уровню формализации. Данный уровень предполагает, что свойство должно быть фальсифицируемо в рамках имеющихся ресурсов для доказательства. Отсутствие возможности выполнения проверки может быть обусловлено сложностью системы или формулировки, ограниченностью ресурсов доказательства безопасности, невозможностью проведения эксперимента или другими факторами. Доказательство корректности может работать со вторым уровнем формализации, но отсутствие возможности проверки или её ограниченность сигнализирует о потенциальных проблемах, так как возможно будет затруднительно использовать другие способы верификации, такие как тестирование, имитационные испытания, самопроверка и др. Сама же необходимость перехода на третий уровень согласуется с опытом создания надежных и безопасных систем [7].

Определение уровня абстракции и аксиоматики доказательства корректности. Любое доказательство всегда базируется на некотором множестве утверждений (аксиом), которые заведомо всегда выполняются. В последующем, на основании аксиом и с помощью правил вывода (логики), проводится доказательство (доказывается теорема) и делается вывод о факте выполнения целевых свойств системы, которые могут выполняться, не выполняться, или теорема может оказаться слишком сложной для доказательства. Общая схема описанного процесса показана на рисунке 3.

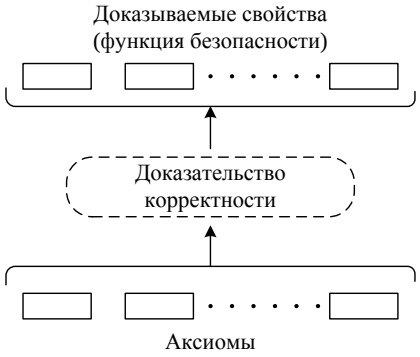

Рисунок 3 – Общая схема доказательства корректности

В качестве аксиом выбираются такие утверждения, которые в наибольшей степени являются неизменными и устойчивыми. Также данные утверждения должны соответствовать рассматриваемому уровню абстракции. Например, при доказательстве корректности ПО на языке ассемблера в качестве аксиом могут быть выбраны утверждения исходя из спецификации команд и состояний процессора. Аксиомы могут быть ослаблены в случае, если рассматривается работа в сложной электромагнитной обстановке, при которой возможны произвольные изменения регистров процессора и ячеек памяти. Также утверждения могут быть усилены в случае рассмотрения ограниченного набора команд процессора.

В общем случае разработка математических идей, к которым относится доказательство корректности, сопровождается аксиоматизацией. Она уточняет понятия, выявляет потенциальные недоразумения и ошибки, детализирует и формулирует черты нового понятия или абстракции. А строгая формулировка аксиом считается одним из мощных средств в борьбе с программными ошибками на всех стадиях жизненного цикла [22].

Таким образом, задачей данного шага является определение уровня абстракции и аксиоматики доказательства корректности.

Анализ требований к системе. ФБ может быть определена исходя из функциональности системы. Методика предлагает определение ФБ на основании стратегии обеспечения безопасности, требований безопасности ко всей системе, требований безопасности к рассматриваемому АПК и интерфейсов взаимодействия [17, 18, 20].

Результатом валидации на данном этапе является ФБ, для которой свойства рассматриваемого ПО выполняют требования безопасности к нему и к его окружению, а также согласованы с используемой стратегией обеспечения безопасности.

Анализ свойств системы. Опыт верификации КВОИ говорит о том, что ФБ формируется, прежде всего, на основании произошедших аварий и катастроф в прошлом [6]. Определенная на предыдущих этапах методика ФБ не способна учитывать опыт эксплуатации систем, а существующие практики и рекомендации по большей части не указывают обстоятельства, из-за которых введена рекомендация [10,23]. В связи с этим для решения проблемы предлагается использование контрольного списка особенностей, представляющего собой набор пар «условие» и «необходимая проверка». Данный контрольный список проверок представлен в работе [9] и дополняется на основании опыта верификации.

Адаптация ФБ к последующему доказательству корректности. Сформированная ФБ требует верификации на последующих этапах. Но её можно изменить и тем самым повлиять на последующее доказательство корректности. Это важно на практике, так как при верификации систем выбор ФБ представляет собой компромисс между имеющимися ресурсами и доказываемыми свойствами [17]. Способы изменения и адаптации ФБ представлены в работах [18, 24].

Верификация. К настоящему моменту ФБ должна быть определена и готова к верификации, во время которой определяется, действительно ли рассматриваемая система выполняет ФБ.

Декомпозиция и верификация общих свойств системы. Способы декомпозиции и верификации общих свойств системы в методике рассматриваются в виде следующего набора инструментов:

А Дедуктивный анализ.

Б Функциональная декомпозиция.

В Объектная декомпозиция.

Г Предикаты абстракций.

Д Верификация модели.

Е Последовательное доказательство свойств системы.

В последующем ссылку на соответствующий способ будем обозначать от (А) до (Е) соответственно.

Доказательство корректности можно полностью провести с помощью классических методов, т.е. посредством *дедуктивного анализа (А)*. Однако для систем большой сложности это является сложной и трудоемкой задачей. В связи с этим предлагается изучать её и упрощать, определяя свойства системы и формируя абстракции более высокого уровня, что показано на рисунке 4 [19, 20].

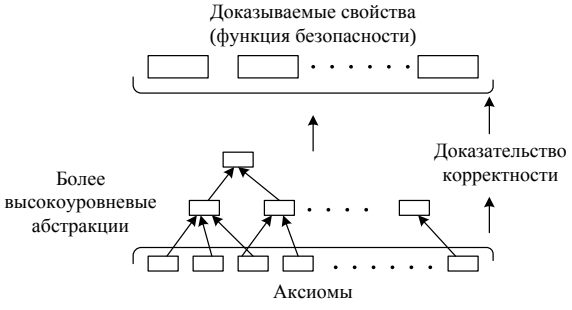

Рисунок 4 – Доказательство корректности на основе абстракций декомпозиции системы

К абстракциям более высокого уровня могут относиться программные модули, функции, объекты, подмножества состояний, модели и др. [22]. Их выбор

определяется исходя из задачи упрощения верификации, т.е. вновь формируемая абстракция должна быть проще той сущности, которую она инкапсулирует. Другими словами, новая абстракция должна скрывать под собой элементы таким образом, чтобы доказательство корректности упрощалось. При этом свойства новых абстракций можно использовать в последующих шагах доказательства, когда по существу идет ссылка на теорему, которая уже доказана [22].

Вторым способом методики является *функциональная декомпозиция (Б).* Программа разбивается на последовательности команд (функции) таким образом, чтобы их можно было рассмотреть в качестве отдельных независимых блоков, каждый из которых представляет собой функцию, изменяющую состояние системы определенным образом при переходе из начальной точки блока в конечную. Функции должны обладать следующими свойствами:

– завершимости – всегда должны заканчивать свое выполнение при любых обстоятельствах в рамках рассматриваемого множества состояний;

– зависимости от фиксированного множества аргументов – функция должна зависеть от определенных аргументов, формировать результат на определенном множестве переменных и не иметь побочных эффектов.

Изначально функции базируются на аксиоматических утверждениях, а в последующем анализ функции может опираться на уже верифицированные функции [20].

Необходимость свойства завершимости следует из специфики разработки и верификации безопасных микроэлектронных СЖАТ вследствие особенностей стратегий обеспечения безопасности. В частности, в случае неопределенно длительной работы функции невозможно гарантировать максимально допустимое время обнаружения одиночного отказа, а также выполнить требования сходимости (*convergence*) для самостабилизирующихся систем [3, 25].

Анализ проводится с учетом ФБ. Например, если установлено, что функция не влияет на ФБ, т.е. идентификаторы являются свободными по отношению к данной функции, то это ведет к упрощению [19].

Третьим предлагаемым способом является *объектная декомпозиция (В)*. Под объектом в методике понимается объединение кода и данных таким образом, что они являются независимыми от другой части системы [20].

Объекты должны обладать следующими свойствами:

– все методы объекта должны обладать свойством завершимости;

– должны быть определены внутренние состояния на основании фиксированного набора переменных;

– доступ к набору переменных должен осуществляться только посредством методов объекта.

Для данного этапа важно провести функциональную декомпозицию (Б), что позволяет выделить потенциальные объекты и предоставить уже готовые свойства функций, являющиеся методами объектов или используются ими. Таким образом, объектная декомпозиция позволяет перейти к более высокоуровневому анализу. Подробнее объектная декомпозиция рассмотрена в работе [20].

Четвертым способом методики является использование *предикатов абстракций* (*predicate abstraction*) *(Г)*. Его задачей является разбиение всего множества состояний системы на подмножества, а в дальнейшем верификация для каждого из них выполняется отдельно. Подробнее см. в работе [20].

Пятым предлагаемым способом является построение модели и её проверка (*model checking*) *(Д)* [26]. Здесь формулируется модель, согласно которой работает рассматриваемая система. В дальнейшем доказывается, что система действительно работает согласно модели, т.е. свойства заявленной модели выполняются. В последующем доказывается, что модель выполняет те требования, которые предъявляются ФБ. Подробнее см. в работе [20].

Следует отметить, что важной задачей с точки зрения методики является выполнение *последовательного доказательства свойств системы (Е),* когда сначала доказываются некоторые общие свойства, а в дальнейшем доказываются на их основе другие свойства, тем самым первые облегчают верификацию последующих. Подробнее см. в работе [20].

В завершении этапа декомпозиции и верификации общих свойств системы важно отметить, что основным резервом и способом обеспечения правильности программ является удачная организация её структуры и обрабатываемых ею данных [15, 22]. Поэтому отсутствие структурной организации ПО или невозможности её выделения ведет к непреодолимым сложностям при верификации и доказательстве её корректности. Это подтверждается верификацией и разработкой как отечественных СЖАТ, так и международным опытом анализа на безопасность КВОИ [2, 6, 22, 23, 27]. Таким образом, если система на этапе проектирования не была подготовлена к тому, чтобы быть верифицируемой, то с большой вероятностью доказать корректность будет затруднительно, а программа будет содержать ошибки.

Последовательность доказательства корректности. Методика предлагает следующие шаги, позволяющие эффективно верифицировать ПО СЖАТ:

1 Доказательство старта системы.

2 Доказательство завершимости алгоритмов и получение общих свойств системы.

3 Доказательство циклических свойств системы и высокоуровневая декомпозиция.

4 Анализ модели.

Доказательство старта системы представляет собой вывод условий, при которых система начинает свою работу в начальном состоянии. Здесь целесообразно применение методов (А) и (Г).

Доказательство завершимости алгоритмов и получение общих свойств системы опирается на факт того, что большинство СЖАТ относится к циклическим системам (*cyclic executive*), для которых характерно разделение выполнения программы на инициализацию и циклическую работу [9]. Данные этапы качественно различны, и поэтому к ним требуется применение разных подходов. Подробнее это рассмотрено в работе [28]. Здесь основными методами верификации являются (А), (Б) и (Е).

Следующим шагом является *доказательство циклических свойств системы и высокоуровневая декомпозиция*, для которого характерны методы (Б), (Г) и (Е). Основными способами доказательства корректности ПО относительно ФБ являются доказательство инварианта на каждом витке цикла и последовательный вывод всех возможных состояний. Подробнее см. в работе [28].

Анализ на модели проводится не всегда, так как доказательство ФБ может быть выполнено без перехода к модели. Здесь характерно применение методов (В), (Д) и (Е). Переход к модели подразумевает, что должны выполняться некоторые допущения [29]. Для СЖАТ к таким допущениям могут относиться: представление внешних и внутренних передаваемых сигналов в терминах модели, представление времени в модели и предположение о нулевой задержке.

Анализ модели проводится последовательностью действий, показанной в таблице 2.

Таблица 2 – **Анализ модели**

Действие	Описание
Определение модели	Установка общих абстракций АПК и модели. Формализация модели. Определение аксиом модели относительно аппаратных средств, её уровня абстракции и условий допущения
Верификация модели	Доказательство того, что модель обладает заявленными в ней свойствами
Проверка на модели	Доказательство свойств модели, верифицирующих ФБ

Таким образом, анализ на модели представляет собой проверку модели, построение которой опирается на уже определенные свойства системы.

Результаты верификации. После выполнения всех вышеописанных действий по доказательству корректности нужно сделать заключение о том, выполняется ФБ или нет, т.е. обнаружены ли ошибки ПО в процессе валидации и верификации. Это является основным результатом. Однако анализ на безопасность с помощью доказательства корректности обладает рядом особенностей, которые позволяют получить дополнительную выгоду от выполненной работы.

Первой особенностью является то, что результат анализа для обнаруженных ошибок позволяет сформулировать условия, при которых они возникают.

Второй особенностью является то, что одним из результатов является определение спецификаций системы как следствие верификации. Например, может быть определено время реакции на конкретные воздействия, формализовано поведение системы при определенных условиях и др. Это позволяет более точно оценить достоинства и недостатки исследуемого АПК, его качественные характеристики, рассмотреть детально особенности функционирования.

Третья особенность связана непосредственно с возможностью повышения качества ПО в будущем. Рекомендации основываются на анализе, и они могут способствовать улучшению не только надёжности и безопасности системы, но и скорости её работы, устойчивости к сбоям, снижению требований к системе, улучшению качественных характеристик всего исследуемого АПК.

Методика явно акцентирует внимание на данных особенностях и дополнительной пользе, которую приносит выполнение доказательства безопасности. Соответственно результаты об условиях возникновения ошибок, об определении спецификаций и о возможности повышения качества оформляются вместе с заключением о корректности ПО.

Применение методики для доказательства информационной и кибернетической безопасности. Методика построена на анализе доказательств функциональной безопасности, а для применения в сфере информационной и кибернетической безопасности необходима адаптация.

Основным инструментом в методике является доказательство корректности, которое предоставляет собой доказательство некоторых свойств системы. Соответственно, если они сформулированы в других терминах, таких как надежность или отказоустойчивость АПК, информационная или кибернетическая безопасность, способность выдерживать нагрузки и т.д., то для них возможно проведение верификации аналогичным методике образом. При этом необходимо провести анализ свойств ПО для новых типов задач, т.е. пересмотреть этапы формирования доказываемой функции во время валидации.

В настоящее время на Белорусской железной дороге задачи информационной и кибернетической безопасности являются актуальными проблемами, требующими решения. Это обусловлено тем, что применяется стороннее ПО, которое может содержать недекларированную функциональность, непосредственным образом оказывающая влияние на информационную и кибернетическую безопасность [30]. Поэтому применение стороннего ПО СЖАТ подлежит обязательной проверке на безопасность, что обусловливает необходимость разработки методов и средств, позволяющих эффективно выявлять программные закладки.

Таким образом, методика может быть адаптирована для решения проблем информационной и кибернетической безопасности ПО СЖАТ.

Заключение. Предложенная и описанная методика доказательства безопасности ПО СЖАТ с помощью формальных методов дает возможность проводить анализ на безопасность произвольных железнодорожных АПК с объемом ПО до 10 KLOC. Данный результат достигнут за счет анализа свойств предметной области, учета опыта работы с КВОИ, последовательного применения методов декомпозиции в сочетании с известными в настоящее время лучшими практиками инженерии безопасных систем. Методика позволяет уменьшить порог вхождения для применения доказательства корректности на практике, ускорить процесс проведения валидации и верификации, а также улучшить качество анализа на безопасность с помощью описанных формальных методов, обосновывать доказательство функциональной, информационной и кибернетической безопасности ПО СЖАТ.

Список литературы

1 **Сивко, Б. В.** Доказательство корректности блока телеуправления 16-1 диспетчерской централизации «Нёман» / Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 1 (24). – С. 18–21.

2 **Харлап, С. Н.** Верификация программного обеспечения микропроцессорной светооптической светодиодной системы / С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 1 (24). – С. 22–25.

3 **Бочков, К. А.** Микропроцессорные системы автоматики на железнодорожном транспорте : учеб. пособие / К. А. Бочков, А. Н. Коврига, С. Н. Харлап. – Гомель : БелГУТ. – 2013. – 254 с.

4 **Kan, S. H.** Metrics and Models in Software Quality Engineering / S. H. Kan. – USA, Boston : Addison-Wesley Professional, 2nd edition. – 2004. – 560 p.

5 **Knight, J. C.** Safety critical systems: challenges and directions / J. C. Knight // ICSE '02 Proceedings of the 24th International Conference on Software Engineering. – 2002. – P. 547–550.

6 **Leveson, N**. Safeware: System Safety and Computers / N. Leveson. – USA, Boston : Addison-Wesley. – 1995. – 680 p.

7 **Шубинский, И. Б.** Функциональная надежность информационных систем : методы анализа / И. Б. Шубинский. – Ульяновск : Изд-во журнала «Надежность». – 2012. – 216 с.

8 **Smith, D. J.** Developments in the Use of Failure Rate Data and Reliability Prediction Methods for Hardware: Aerospace Engineering, Dissertation: Date: 29.05.2000. – Delft University of Technology, Netherlands, Delft, 2000. – 175 p.

9 **Бочков, К. А.** Особенности программного обеспечения микропроцессорных систем железнодорожной автоматики и телемеханики / К. А. Бочков, С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2014. – № 1 (28). – С. 13–18.

10 **Smith, D. J.** Safety Critical Systems Handbook. A Straightforward Guide to Functional Safety, IEC 61508 and Related Standards, Including Process IEC 61511 and Machinery IEC 62061 and ISO 13849 / D. J. Smith, Kenneth G. L. Simpson. – Elsevier Ltd. – 2010. – 270 p.

11 **Paige, R. F.** Symposium on Concurrency, Real-time, and Distribution in Eiffel-like Languages: Proceedings / R. F. Paige, P. J. Brooke // University of York. Department of Computer Science, University of Teesside. – 2006. – 26 p.

12 **Дубова, Н.** Строитель надежных программ / Н. Дубова // Открытые системы. – 2011. – № 10 (176). – С. 52–55.

13 **Сивко, Б. В.** Доказательство корректности программного обеспечения железнодорожной автоматики и телемеханики / Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2013. – № 2 (27). – С. 21–26.

14 **Чень, Ч.** Математическая логика и автоматическое доказательство теорем / Ч. Чень, Р. Ли. – М. : Наука. – 1983. – 360 с.

15 **Dijkstra, E. W.** Programming as a discipline of mathematical nature / Edsger W. Dijkstra // American Mathematical Monthly. – Vol. 81, No. 6. – 1974. – P. 608–612.

16 **Hoare, C. A. R.** An axiomatic basis for computer programming / C. A. R. Hoare // Communications of the ACM. – 1969. – Vol. 12, Is. 10. – P. 576–580.

17 **Сивко, Б. В.** Определение функции безопасности при верификации программного обеспечения микропроцессорных систем железнодорожной автоматики и телемеханики / Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2013. – № 1 (26). – С. 18–20.

18 **Бочков, К. А.** Выбор и определение функции безопасности при верификации микропроцессорных систем железнодорожной автоматики и телемеханики / К. А. Бочков, Б. В. Сивко // Надёжность. – 2014. – № 2 (49). – С. 101–115.

19 **Грис, Д.** Наука программирования / Д. Грис; пер. с англ.; под ред. А. П. Ершова. – М. : Мир. – 1984. – 416 с.

20 **Сивко, Б. В.** Доказательство корректности программного обеспечения многопроцессорных устройств связи с объектами железнодорожной автоматики и телемеханики / Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 2 (25). – С. 27–30.

21 **Parnas, D. L.** Assessment of safety-critical software in nuclear power plants / D. L. Parnas, J. Madey, G. J. K. Asmis // Nuclear Safety. – 1991. – Vol. 32, No. 2. – P. 189–198.

22 **Дал, У.** Структурное программирование / У. Дал, Э. Дейкстра, К. Хоор; пер. с англ.; под ред. Э. З. Любимского и В. В. Мартынюка. – М. : Мир. – 1975. – 245 с.

23 **Gavin, M.** Guidelines for The Use of The C Language in Vehicle Based Software / M. Gavin. – UK, Nuneaton, The Motor Industry Software Reliability Association. – 2004. – 116 p.

24 **Сивко, Б. В.** Выбор и определение функции безопасности при верификации программного обеспечения критически важных объектов информатизации / Б. В. Сивко // Информационные технологии и системы. – Минск : БГУИР. – 2013. – С. 322–323.

25 **Dolev, S.** Self-stabilization / S. Dolev. – Cambridge, USA, Massachusetts : MIT Press. – 2000. – 197 p.

26 **Кларк, Э. М.** Верификация моделей программ: Model checking / Э. М. Кларк ; пер. с англ. под ред. Р. Смелянского. – М. : МЦНМО. – 2002. – 416 с.

27 **Lawford, M.** Formal Verification of Nuclear Systems: Past, Present, and Future / M. Lawford, A. Wassyng // Information & Security: An International Journal. – 2012. – Vol. 28, Is. 2, No. 18. – P. 223–235.

28 **Харлап, С. Н.** Верификация циклических систем железнодорожной автоматики и телемеханики / С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2013. – № 2 (27). – С. 37–41.

29 **Charles, A.** Representation and analysis of reactive behaviors : A synchronous approach / A. Charles // Computational Engineering in Systems Applications IEEE-SMC, University of Nice Sophia Antipolis. – 1996. – P. 19–29.

30 **Бодунов, Н.** Обзор стандартов жизненного цикла программного обеспечения для систем с высокими требованиями к надежности и безопасности / Н. Бодунов, И. Починок // Мир компьютерной автоматизации: встраиваемые компьютерные системы. – № 3 (27/45/104). – 2014. – С. 35–43.

Получено 27.09.2014

B. V. Sivko. The method for proof of safety for software of railway microprocessor systems.

The paper suggests a method for formal proof of safety for railway software systems which includes validation and verification stages and considers software and hardware as one integrated piece. The method is based on as experience of software correctness proof of railway devices as on the worldwide theory and practice for safety-critical systems. The method can do safety analysis of an arbitrary railway hardware-software complex, which has software size up to 10 KLOC. The method can be used to prove functional safety and information security.