***Бочков К.А., Сивко Б.В.***

# ВЫБОР И ОПРЕДЕЛЕНИЕ ФУНКЦИИ БЕЗОПАСНОСТИ ПРИ ВЕРИФИКАЦИИ МИКРОПРОЦЕССОРНЫХ СИСТЕМ ЖЕЛЕЗНОДОРОЖНОЙ АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

*Рассмотрены вопросы определения, формализации и выбора функции безопасности, применяемой при разработке и доказательстве корректности программного обеспечения микропроцессорных систем железнодорожной автоматики и телемеханики. Приведены способы выбора функции безопасности на основании технического задания, ограниченности ресурсов, используемой стратегии обеспечения безопасности и общих требований к характеристикам системы.*

***Ключевые слова:*** *верификация, валидация, функциональная безопасность, функция безопасности, доказательство корректности, критически важные объекты информатизации.*

В настоящее время в новых разработках широко используется микроэлектронная база в системах железнодорожной автоматики и телемеханики (СЖАТ), что расширяет её возможности и позволяет реализовывать и предоставлять более широкую функциональность для эксплуатируемых систем. Но в то же время разработка, верификация и последующая эксплуатация данных систем должны соответствовать и удовлетворять принятому на железнодорожном транспорте уровню безопасности. Традиционно в СЖАТ использовалась релейная база, когда построение происходило на принципе аппаратной реализации функций безопасности, а микропроцессорные системы представляют собой аппаратно-программные комплексы (АПК), большинство функций которых реализуется программно. В то же время при построении современных микропроцессорных СЖАТ преобладающим способом является использование COTS-технологий, но при этом разработка программного обеспечения (ПО) является наиболее сложным элементом этих систем. Кроме того, для микроэлектронных СЖАТ отсутствуют единые, универсальные и общепризнанные способы доказательства безопасности и в связи с этим необходимо применять комплекс методов и средств по повышению уровня безопасности на всех этапах жизненного цикла системы, а разработка новых способов доказательства безопасности является актуальной задачей.

Одним из возможных способов поиска ошибок и повышения качества ПО в рамках применяемого комплекса подходов является доказательство корректности, которое относится к формальным методам (*Formal Methods*) [1] и успешно используется для верификации микропроцессорных устройств на Белорусской железной дороге [2, 3, 4]. Для систем, связанных с безопасностью, стандарт IEC 61508 имеет градацию уровней полноты безопасности от SIL-1 до SIL-4, а железнодорожные систе-

мы должны соответствовать наиболее строгому уровню SIL-4, который настоятельно рекомендует применение формальных методов для критически важных систем информатизации [5].

Формальные методы в качестве доказательства корректности могут быть применены как для готового ПО, так и на ранних этапах разработки всего АПК, но в любом случае одним из первых шагов верификации является определение функции безопасности, подлежащей проверке на корректность [4, 6].

Функция безопасности представляет собой формализованное условие по отношению к верифицируемой системе, выполнение которого позволяет сделать заключение о безопасности функционирования СЖАТ. Для одного и того же АПК функция безопасности может быть определена по-разному, а выбор доказываемого условия может происходить на разных этапах жизненного цикла системы.

Разработка и эксплуатация ПО, а также ряд исследований говорят о том, что чем позже обнаруживается ошибка, тем сложнее как выявить её, так и исправить, и тем больше проблем она может принести [7, 8]. При этом исправление ошибок, допущенных при формулировании требований к системе, обходится в десятки раз дороже ошибок, допущенных во время реализации [9, 10]. Определение функции безопасности, которое относится к формализации решаемой задачи, является спецификацией по отношению к доказательству корректности и обладает теми же свойствами, что и постановка требований при разработке ПО. Потенциальные ошибки, допускаемые при определении данной функции, негативно влияют на качество верификации и могут приводить к искажению результатов доказательства корректности и, как следствие, его полному пересмотру.

На рис. 1 показана последовательность этапов анализа на безопасность с определением функции безопасности.

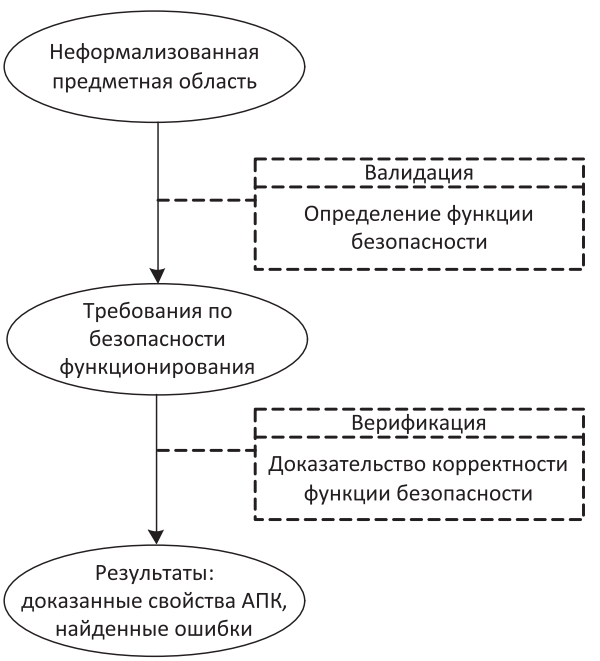


Рис. 1. Последовательность этапов анализа на безопасность

Условия для определения функции безопасности устанавливаются на этапе валидации на основании характеристик применяемых компонентов, множества используемых методов, стратегий обеспечения безопасности и практического опыта в рассматриваемой предметной области [11]. Данный процесс независим от последующей верификации – он определяет подлежащие проверке

свойства и формирует исходные данные, на основании которых задается функция безопасности, используемая в доказательстве корректности. Если допускаются ошибки на этапе валидации, либо не принимаются во внимание особенности поведения, влияющие на безопасность, то это непосредственным образом влияет на качество последующей верификации. Кроме того, какие бы эффективные и диверситетные методы и средства не использовались во время доказательства корректности, они не в состоянии выявить и решить проблемы, созданные во время проектирования, так как они работают с одинаковой спецификацией и только конечный пользователь может указать на ошибку, допущенную при составлении требований.

Мировой опыт эксплуатации критически важных систем информатизации говорит о том, что аварии и катастрофы происходят из-за множества факторов, при этом значительная часть происшествий происходит (в том числе) из-за ошибок, допущенных при формулировке требований к системе [11, 12, 13]. Например, имеется большое количество причин, из-за которых морской корабль может быть подвержен риску: столкновение с айсбергом, коррозия, взрыв груза и др. Инженерам не обязательно знать все источники рисков, но при этом во время проектирования могут быть приняты формализованные решения для минимизации опасности: корабль должен оставаться на плаву при заданном пределе количества пробоин, необходимо наличие спасательных средств и требуется проведение предварительных мероприятий. Пароход “Титаник” был спроектирован так, чтобы оставаться на плаву в случае затопления 4 или менее первых отсеков, что можно назвать функцией безопасности. К сожалению, столкновение с айсбергом привело к затоплению 5 первых отсеков [11]. Функция безопасности была задана исходя из практического опыта и знаний того времени, и после её определения проектирование системы проводилось уже на её основании. Но, как показывает история, такой подход не означает, что все возможные опасности устранены.

При проектировании систем могут приниматься неявные допущения, которые напрямую не связаны с безопасностью функционирования, но могут повлиять на работу всего АПК. Например, допущение того, что траектории самолётов всегда будут выше уровня моря, может привести к ошибкам ПО во время полета над территориями ниже уровня моря и отказу всей системы [14].

Условия безопасности работы системы могут отличаться при изменении окружения или условий функционирования, что актуально для СЖАТ, и в частности это в значительной степени проявляется при переходе с релейной базы на микроэлектронную. Например, при проведении испытаний на безопасность схем блочной маршрутно-релейной централизации проверка зависимостей по стрелочно-путевым секциям проводится один раз независимо от положения стрелок, входящих в секцию, а также независимо от рода и направления маршрута, задаваемого через секцию. Независимость от перечисленных факторов обуславливается свойствами реле первого класса надежности и схемными решениями по проверке взаимозависимостей. Однако в случае применения микропроцессорной базы с симметричными отказами АПК не обладает теми же свойствами, и верификация ПО должна проводиться с учётом всех возможных вариантов, которые могут быть в рассматриваемых условиях функционирования с рассмотрением всех возможных отказов микроэлектронной элементной базы в соответствии с ТНПА.

Таким образом, одной из проблем валидации является определение условий, подлежащих проверке, а особенности данного процесса таковы, что после формализации нет однозначного критерия и уверенности в том, что утвержденная доказываемая функция является необходимой и достаточной [15]. Во время последующей разработки или анализа на безопасность может быть выяснено, что заданные рамки слишком строги и доказательство корректности провести невозможно, или напротив, слишком слабы, из-за чего уменьшается вероятность нахождения ошибок в ПО.

Микроэлектронные СЖАТ обладают сложностью, которая затрудняет формализацию допускаемого и безопасного поведения. Из-за этого с большой вероятностью могут быть допущены ошибки. Для решения данной проблемы может быть определена такая функция безопасности, в которой данные недостатки отсутствуют. Кроме того, при разработке и при доказательстве корректности нет необходимости строгого выбора функции безопасности – это может быть любая функция, удовлетворяющая условиям на рис. 2.


Рис. 2. Выбор доказываемой функции безопасности

Описание для указанных областей:

А – по каким-либо причинам система не выполнила условие доказываемой функции безопасности, но это не привело к опасному отказу;

Б – поведение системы удовлетворяет условию функции безопасности;

Таким образом, доказываемая функция безопасности всегда должна быть такой же или более строгой, чем допускаемое безопасное поведение. Поведение разработанной системы должно удовлетворять условию доказываемой функции безопасности.

Накопленный авторами опыт верификации критически важных объектов информатизации микропроцессорных СЖАТ говорит о том, что определение доказываемой функции безопасности разрабатываемых и существующих АПК необходимо проводить на основании:

– используемой стратегии обеспечения безопасности – например, во время проектирования может быть использована стратегия применения логических элементов с несимметричными отказами ($h_1$-надежные элементы) и система должна придерживаться её во время всего жизненного цикла [16];

– требований безопасности ко всей системе – например, верифицируемый компонент должен выдерживать заданные временные диапазоны сигналов взаимодействия с другими компонентами системы;

– требований безопасности к рассматриваемому АПК – например, система обязана сохранять инвариант и переходить в безопасное состояние в случае внутренних сбоев.

Таким образом, результатом верификации является доказательство того, что свойства рассматриваемого ПО выполняют требования безопасности к нему и к его окружению, а также согласованы с используемой стратегией обеспечения безопасности.

Определение доказываемой функции безопасности может проводиться на основании только технического задания (ТЗ) разрабатываемого АПК, но в этом случае формализация функции безопасности может оказаться сложным и трудоемким процессом. В связи с этим возможен переход к доказательству более строгой функции безопасности, нежели той, которая определена исходя из в требований безопасности ТЗ и учёта полноты критериев опасного отказа.

Рассмотрение требований к системе в отношении стратегии обеспечения безопасности, безопасного внутреннего поведения и согласования взаимодействия с внешними компонентами позволяет быстрее и эффективнее разбивать доказываемую цельную функцию на эквивалентные, но более простые функции, что уменьшает сложность верификации и улучшает её качество.

Формулировка функции безопасности не является конечным решением и при необходимости функция безопасности может быть изменена на любую другую, удовлетворяющую условиям, показанным на рис. 2. Опыт верификации говорит о том, что переход к другой функции безопасности следует проводить тогда, когда в новом рассматриваемом ракурсе поведение системы становится:

– более детерминированным – можно более точно сказать, как ведет себя система в тех или иных ситуациях;

– менее сложным – снижаются затраты анализа на безопасность и время, требуемое на понимание процессов, имеющих место в системе.

Основными применяемыми авторами способами изменения функции безопасности является её расширение (ослабление, более слабое определение) и сужение (усиление, более строгое определение). Кроме этого, функция безопасности может быть изменена не как строгое ослабление или усиление, но в любом случае, она должна находиться в рамках допускаемого безопасного поведения (см. рис. 2).

Рассмотрим три функции безопасности $f_1, f_2$ и $f_3$, каждая из которых зависит от вектора аргументов $\bar{\alpha}$ и имеет область значений истина/ложь (*true/false*). Тогда усилением функции $f_2$ является переход к такой функции $f_3$, при котором выполняются условия (1) и (2):

$$\forall\ (f_3(\alpha) = true\ )\quad f_2(\alpha) = true \tag{1}$$

$$\exists\ (f_2(\alpha) = true\ )\quad f_3(\alpha) = false. \tag{2}$$

Ослаблением функции $f_2$ является переход к такой функции $f_1$, при котором выполняются условия:

$$\forall\ (f_2(\alpha) = true)\quad f_1(\alpha) = true \tag{3}$$

$$\exists\ (f_1(\alpha) = true)\quad f_2(\alpha) = false. \tag{4}$$

Таким образом, ослаблением функции является переход от одной функции $f_2$ к другой функции $f_1$ таким образом, что всегда, когда истинна $f_2$, то истинна и $f_1$, но при этом существуют такие истинные значения $f_1$, при которых $f_2$ ложна. Усилением является аналогичный обратный переход. Графически отношения между функциями показаны на рис. 3.

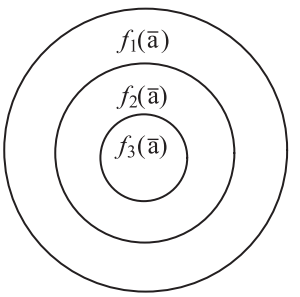


Рис. 3. Усиление и ослабление функций безопасности

Рассмотрим пример функций безопасности, выбор которых может повлиять на доказательство корректности. Предположим, что имеется ПО АПК, вся функциональность которого выполняется в замкнутом цикле, у которого каждое последующее выполнение должно быть отличимым от предыдущего и для этого в памяти хранится идентификатор *id*. Будем считать, что число циклов конечно и каждый из них пронумерован последовательно во времени от *1* до *n*, и, соответственно, существует множество идентификаторов $\alpha = \{id_1, id_2, \ldots, id_n\}$.

Для рассматриваемого случая приведем несколько вариантов функции безопасности. Первый пример:

$$f_1(\alpha) = true, \forall\ (i \in N, i < n)\quad id_i \neq id_{i+1}.$$

Данная функция гарантирует отличие идентификатора от предыдущего и может быть использована для безопасного обновления входящей информации.

Вторая функция безопасности гарантирует уникальность идентификатора за все время работы АПК с момента его запуска и может быть использована для обновления информации, которое происходит не на каждом витке полного цикла:

$$f_2(\alpha) = true, \forall\ (i \neq j \in N;\ i, j \leq n)\quad id_i \neq id_j.$$

Следующая функция безопасности гарантирует, что каждый последующий идентификатор ровно на 1 больше предыдущего и может быть использована для расчёта количества полных циклов между событиями:

$$f_3(\alpha) = true, \forall\ (i \in N, i < n)\quad id_{i+1} = 1 + id_i.$$

Функция $f_3$ более строгая, чем функция $f_2$, которая в свою очередь, более строгая, чем $f_1$.

Верифицировать более строгую функцию сложнее, чем более слабую – на это тратится большее количество ресурсов и не всегда это удается. Но, если есть возможность, то рекомендуется доказывать корректность более строгой функции, так как это имеет следующие положительные эффекты:

– получаем более точное представление о том, как работает система – свойства и поведение определяются более строго;

– снижается анализируемая сложность функционирования и тем самым повышается вероятность нахождения ошибок;

– доказанные функции могут быть использованы в дальнейшем для более эффективного проведения других доказательств корректности рассматриваемого АПК.

Однако в случае анализа на безопасность, когда невозможно доказать корректность в предложенном виде или отсутствуют ресурсы для проведения такого объема работ, возможно ослабление проверяемой функции при условии, что это позволит сделать заключение о безопасности ПО.

Определение функции безопасности для проведения верификации является важным этапом анализа на безопасность и её выбор представляет собой компромисс между имеющимися ресурсами и доказываемыми свойствами. Практика показывает, что избежать компромисса удается только в случае, если система подготовлена к тому, чтобы быть верифицируемой, когда задачи определения функции безопасности решаются до этапов разработки и проектирования, что возможно только для разрабатываемых и проектируемых АПК [4, 6].

Описываемое определение выбора функции безопасности было опробовано на верификации ПО устройств связи с напольными объектами СЖАТ, такими как блоки телеуправления 16-1 [2], светооптические светодиодные системы [3] и многопроцессорные блоки ТУ-8Б и ТС-16Б микропроцессорной централизации «Ипуть» [6].

Таким образом, разработанные общие принципы построения функций безопасности позволяют улучшить формализацию спецификаций доказательства безопасности ПО сложных АПК критически важных систем информатизации, а задача доказательства безопасности упрощается, если в процессе разработки используются методы построения контролепригодного ПО.

## Литература

1. **Butler R.W.** «What is Formal Methods?» NASA LaRC Formal Methods Program, 2001.

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

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

4. **Сивко Б.В.** Проектирование безопасного программного обеспечения микропроцессорных устройств автоматики и телемеханики // Проблемы безопасности на трансп.: тезисы докл. VI Междунар. науч. – практ. конф., Гомель, 29–30 ноября 2012 г. / М-во образования Респ. Беларусь, М-во трансп. и коммуникаций Респ. Беларусь, Бел. ж. д., Белорус. гос. ун-т трансп.: редкол.: В.И.Сенько (отв. ред.) [и др.]. – Гомель, 2012. – С.205.

5. **David Smith 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» / David J. Smith and Kenneth G. L. Simpson // Elsevier Ltd., 2010.

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

7. **Fagan M.E.** Design and code inspections to reduce errors in program development, IBM Systems Journal, Volume 15 Issue 3, September 1976, p. 182–211.

8. **Boehm B.W.** Software engineering. IEEE Transactions on Computers 25:1226–1241, 1976.

9. **Telles M., Hsieh Y., Telles M.A.** The Science of Debugging // The Coriolis Group, 2001.

10. **Boehm B.W., Papaccio P.N.** Understanding and controlling software costs // IEEE Trans Softw Eng 14(10):1462–1477, October 1988.

11. **Nancy G. Leveson,** Software safety in embedded computer systems. Communications of the ACM, 34(2):34–46, February 1991.

12. **Charles Perrow.** Normal Accidents: Living with High Risk Technologies. Basic Books, New York, NY, 1984.

13. **Ivars Peterson**, Fatal Defect: Chasing Killer Computer Bugs, Times Books, New York, 1995.

14. **Nancy G. Leveson.** Safeware: System Safety and Computers. Addison-Wesley, 1995.

15. **Gerhart S.L., Yelowitz L.,** Observations of Fallibility in Applications of Modern Programming Methodologies // IEEE Trans. Software Eng., vol. 2, no. 3, 1976, pp. 195–207.

16. **Сапожников В.В., Кравцов Ю.А., Сапожников Вл.В.** Дискретные устройства железнодорожной автоматики и телемеханики // М. Транспорт, 1988.