УДК 004.052.2

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

ДИВЕРСИТЕТНЫЕ АКСИОМАТИЧЕСКИЕ БАЗИСЫ ДЛЯ РАЗРАБОТКИ БЕЗОПАСНЫХ И ОТКАЗОУСТОЙЧИВЫХ СИСТЕМ

Предложено описание логического основания и общего подхода для проектирования безопасных и отказоустойчивых систем. Введено понятие диверситета аксиоматических базисов. Показано, что разработка аппаратно-программных комплексов на основании диверситетных аксиоматических базисов позволяет формализовать условия диверситета и защиты от отказа по общей причине. Приведены примеры диверситетных аксиоматических базисов и способы их использования для доказательства корректности функционирования аппаратно-программных комплексов. Показана важность потенциальных преимуществ применения диверситета аксиоматических базисов, как с теоретической так и с практической стороны.

Введение. В настоящее время одной из актуальных задач является создание эффективных методов и средств, позволяющих разрабатывать и верифицировать аппаратно-программные комплексы (АПК), относящиеся к критически важным объектам информатизации (КВОИ). К данным системам относится множество устройств, связанных с безопасностью, которые активно используются в различных отраслях промышленности: железнодорожный и морской транспорт, гражданская авиация, телекоммуникации, медицина, космос, опасное химическое производство и др. [1, 2]. При этом повсеместно требуется выполнение высоких показателей безопасности и надежности их функционирования. К тому же в настоящее время практикуется комплексный подход, заключающийся в применении ряда методов и средств на всех этапах жизненного цикла АПК.

Разработка и верификация КВОИ ведется на основании стандарта IEC 61508, пренебрежение которым может быть воспринято как «неразумная практика» в свете международного Акта о здоровье и безопасности труда и как отказ проявить «должное усердие» [3]. Согласно стандарту, связанные с безопасностью системы имеют градацию уровней полноты безопасности от SIL-1 до SIL-4, и большинство КВОИ относится к наиболее строгому уровню SIL-4, для которого стандарт настоятельно рекомендует производить формализацию требований и процесса доказательства безопасности, применять формальные методы верификации (*formal methods*) и учитывать фактор отказа по общей причине (*CCF, common cause failure*), о которых пойдет речь в данной статье.

В настоящее время используется ряд методов, позволяющих получить программный и аппаратный диверситет, но все они являются неформализуемыми. К таким методам относится N-версионное программирование, выбор различных компиляторов и используемого программного обеспечения, экспертная оценка и др. [4, 5] Для оценки степени диверситета может быть использован BETA-метод и модель BETAPLUS, которые рекомендованы стандартом IEC 61508 [3]. Однако данный подход является экспертным и неформализованным.

Основной проблемой существующих методов является то, что само допущение того, что ошибки совершаются независимо, неверно [2, 6]. Инженеры зачастую реализуют одни и те же вещи одинаковым способом, и, как следствие, ошибки возникают в одних и тех же местах [7]. Во время эксплуатации проблемы нарушения диверситета могут проявляться как для программного, так и для аппаратного обеспечения [6]. Кроме того, не всегда возможно сравнить различные версии в случае, когда они работают корректно [8].

Второй проблемой CCF как для разработки, так и для оценки существующих систем является несовершенство существующих моделей, которое заключается в их ограничениях, в неполном отражении зависимости процессов и во влиянии человеческого фактора [9].

Дополнительной необходимостью разработки формализованных диверситетных методов является сложность решения проблемы для программного обеспечения (ПО). Если в случае аппаратного обеспечения для обеспечения диверситета широко используется физическое разделение компонентов и защита от всевозможного влияния друг на друга, то в случае ПО такие подходы применять невозможно [2, 9].

Предполагается, что описываемые ниже методы будут использованы на ранних этапах разработки, что позволит наделять АПК целевыми формализованными свойствами и доказывать их отказоустойчивость и безопасность.

1 Классическое доказательство безопасности. Для каждого из устройств КВОИ требуется доказательство безопасности его функционирования, которое представляет собой цепь дедуктивных умозаключений, базирующихся на аксиоматическом базисе, результатом которых должен быть вывод о целевом качестве рассматриваемой системы.

Понятие аксиоматического базиса для доказательства свойств ПО было введено Хоаром [10]. В общем случае для доказательства свойств систем, связанных с безопасностью, не рекомендуется рассматривать про-

граммное обеспечение в отрыве от аппаратного, поэтому здесь будет рассматриваться аксиоматический базис в отношении ко всему рассматриваемому АПК [2].

Аксиоматическим базисом (далее базис) будем считать некоторое множество утверждений. Если данные утверждения выполняются для системы в рассматриваемом состоянии, то будем считать, что базис истинен (выполнятся) для состояния данной системы. Другими словами, если рассматривается некоторая система в состоянии x, и при этом имеется множество условий $A_1(x)$, $A_2(x)$, ..., $A_n(x)$, которые являются истинными, то базисом будет являться $A(x)$, определенное по формуле (1), а система в данном состоянии будет считаться удовлетворяющей данному базису.

$$A(x) \equiv A_1(x) \wedge A_2(x) \wedge \ldots \wedge A_n(x) \qquad (1)$$

Пример утверждений базиса:

– все инструкции микропроцессора выполняются согласно его спецификации;

– тактовая частота генератора микропроцессора находится в заданных конкретных пределах;

– изменения ячеек памяти из-за агрессивной электромагнитной обстановки происходят не чаще заданного предела по времени и не больше некоторого предела бит.

Утверждения базиса должны быть формализуемыми, т. е., каждое из них можно однозначно записать в виде математического условия $A_i(x)$. Сами же утверждения могут быть любыми, но они задают уровень абстракции таким образом, чтобы в рамках данной аксиоматики верифицировать свойства системы. Так, вышеописанные утверждения базиса позволяют верифицировать АПК, который работает в условиях сложной электромагнитной обстановки, а базис Хоара позволяет верифицировать алгоритм работы программы [10].

Ряд утверждений базиса может подразумеваться неявно, но использоваться при доказательстве безопасности. Кроме того, выбор базиса не всегда может быть полным, но при этом является важным для решения проблем валидации [11].

Таким образом, при доказательстве безопасности выбирается некоторый аксиоматический базис (явно или неявно). Далее посредством дедуктивных умозаключений (логики) делается попытка установить верифицируемые свойства системы. В случае успеха система считается удовлетворяющей проверяемым свойствам.

2 Диверситетные аксиоматические базисы. Рассмотрим два базиса – A и B, каждый из которых выполняется исходя из своего множества условий. Зависимости базисов друг от друга показаны на рисунке 1.

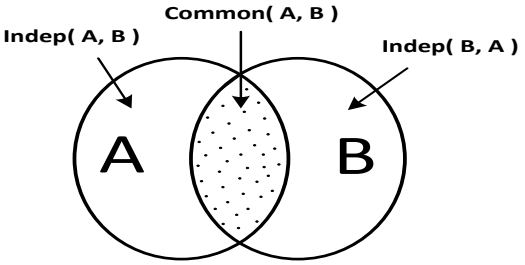

Рисунок 1 – Диверситетные аксиоматические базисы

Если происходит некоторое воздействие на систему (изменение ячеек памяти из-за сбоя, искажение логики работы микропроцессора из-за аппаратного отказа, ошибка разработчика на этапе реализации), то может получиться так, что какие-то условия из этих базисов будут нарушены, и, как следствие, может перестать выполняться как базис A, так и базис B.

Если имеется условие $A_i(x)$, нарушение которого не приводит к нарушению базиса B, то будем называть базис B независимым от условия $A_i(x)$. По аналогии может присутствовать условие $B_j(x)$, нарушение которого не приводит к нарушению базиса A, что означает, что базис A независим от условия $B_j(x)$.

Таким образом, при рассмотрении двух базисов – A и B будем считать, что имеется множество условий базиса A, которые независимы от базиса B, что можно выразить в виде следующей функции:

$$Indep(A, B) \equiv A \wedge (not\ B). \qquad (2)$$

Также определим, что множество условий, нарушение любого из которых приводит к нарушению как базиса A, так и базиса B, является общим базисом A и B, что можно выразить в виде следующей функции:

$$Common(A, B) \equiv A \wedge B. \qquad (3)$$

Базисы A и B определим как независимые, если общий базис пуст (нет ни одного единичного условия, представленного в A и B, которое нарушало бы оба базиса одновременно):

$$Common(A, B) \equiv \varnothing. \qquad (4)$$

Базисы A и B будем считать диверситетными в случае, если они являются независимыми по каким-либо условиям друг от друга, т.е., являются истинными следующие выражения:

$$Indep(A, B) \neq \varnothing; \qquad (5)$$

$$Indep(B, A) \neq \varnothing. \qquad (6)$$

3 Разработка безопасных и отказоустойчивых систем на основании диверситетных аксиоматических базисов. Ключевой идеей данной разработки является создание такой системы, которая одновременно выполняет некоторую функцию F как в базисе A, так и в базисе B, и при этом базисы являются диверситетными. Особенностью такой реализации является то, что в случае нарушения одного из условий, независимых от од-

ного из базисов, АПК остается в состоянии, когда выполняется другой базис, и тем самым система продолжает выполнять функцию *F*.

Первой задачей при применении данного метода является определение таких базисов, которые являются более сильными с точки зрения их диверситета (усиление *Indep(A, B)* и *Indep(B, A)*), и при этом с как можно меньшим общим базисом [как можно меньший *Common(A, B)*].

Вторая задача – удержание внимания на решаемых проблемах, т.е., помещение в *Indep(A, B)* и в *Indep(B, A)* таких условий, которые наиболее важны для функционирования рассматриваемого АПК. Например, если для рассматриваемой системы характерны отказы, характеризующиеся постоянным значением 0 или 1 вне зависимости от попыток записи других значений, то помещение данной особенности в независимые условия от одного из базисов позволит спроектировать систему, которая будет устойчивой к данному типу отказов. Такой подход также считается хорошей стратегией по обеспечению защиты от CCF для отказоустойчивых и безопасных систем в общем случае [9].

Третьей задачей является выбор такой функции *F*, выполнение которой важно с точки зрения системы. Это может быть как функциональность, связанная с безопасностью, так и ключевой элемент в самодиагностике и восстановлении всей системы в случае сбоев.

Одной из особенностей микропроцессорной элементной базы является то, что один и тот же элемент может использоваться множество раз для различных операций, и, как следствие, отказ такого элемента приводит к отказу по общей причине всех функций, которые от него зависят [4]. Данная проблема потенциально решаема с помощью разделения функций часто используемых элементов на различные диверситетные базисы, тем самым может быть снижена или исключена вероятность CCF.

Важно отметить, что диверситетные базисы, имеющие пустой общий базис, в реальных системах являются уязвимыми вне теории, определенной аксиомами базиса. Другими словами, рассматриваемые базисы практически всегда полагаются на истинность других неявных утверждений, которые обеспечивают истинность аксиом базиса. Например, если рассматривается базис в рамках спецификаций языка программирования, то он перестает выполняться в случае, если у микропроцессора произошел отказ тактового генератора и по этой причине исполняемая программа будет остановлена. Проблема такого типа показывает потенциальную вероятность проявления CCF и должна решаться либо с помощью более высокоуровневой абстракции, либо посредством применения других подходов.

Пример аксиоматических базисов логических функций. Рассмотрим два базиса – *A* и *B*, основанных на следующих условиях:

1. Операция OR (логическое ИЛИ) выполняется корректно.

2. Операция AND (логическое И) выполняется корректно.

3. Операция NOT (логическое НЕ) выполняется корректно.

Базис *A* выполняется в случае, если истинны условия 1 и 3, а базис *B* выполняется тогда, когда истинны условия 2 и 3. Здесь базисы являются диверситетными по условиям 1 и 2, что означает, что если одно из них не будет выполняться, то всегда будет базис, который работает корректно. Если же будет нарушено условие 3, которое является общим базисом, то система не сможет выполнить свои функции.

Особенностью данных базисов является то, что каждый из них образует функционально полную систему функций алгебры логики [12]. Другими словами, на основании любого из них можно реализовать любую логическую функцию. Например, если имеется функция *F*, которую требуется вычислить (7):

$$F(x, y, z) = (x \textbf{ and } y) \textbf{ or } z, \qquad (7)$$

то её можно определить в каждом из базисов так, как это показано на рисунке 2.

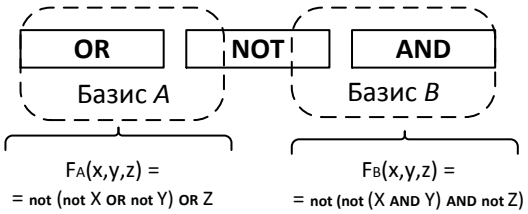

Рисунок 2 – Вычисление функций на основе диверситетных базисов

Как следствие, можно вычислить одну и ту же функцию как в одном базисе, так и во втором. При этом если одно из условий 1 или 2 будет нарушено, то функция будет корректно вычислена в одном из базисов.

Кроме вышеописанных базисов можно сформулировать два диверситетных базиса на условиях корректности операций ИЛИ-НЕ (функция Вебба) и И-НЕ (функция Шеффера) для системы, в которой отказ между данными операциями является независимым [12]. В этом случае общий базис будет пустым, что исключит CCF для вычисления логических функций.

Таким образом, в данном примере показан диверситет базисов с точки зрения выполняемых команд процессора и возможность описанного проектирования для любых логических функций.

Тестирование аксиоматического базиса и диверситет ячеек памяти. Если при рассмотрении диверситетных базисов имеется возможность проверки корректности базиса в реальном времени, то в этом случае можно спроектировать программу или аппаратное обеспечение таким образом, что сначала проверяется выполнение базисов, а в дальнейшем, в зависимости от результата, запускается та или иная процедура или аппаратное обеспечение.

В качестве примера рассмотрим систему, в распоряжении которой имеется два бита регистра *X* и *Y*, каждый из которых может хранить один бит информации. Для данной системы необходимо защититься от отказа постоянного 0 или 1 (когда вне зависимости от команд процессора на присваивание в регистре хранится постоянное число). Представим соответствующе диверситетные базисы с пустым общим базисом:

– базис *A*: попытка чтения с *X* сразу после записи в него бита возвращает точно такое же значение;

– базис *B*: попытка чтения с *Y* сразу после записи в него бита возвращает точно такое же значение.

Проверить на выполнение базис *A* можно следующими командами:

1 $A(x) = true$.

2 $X \leftarrow 1$.

3 Если $X \neq 1$, то базис *A* нарушен ($A(x) = false$).

4 $X \leftarrow 0$.

5 Если $X \neq 0$, то базис *A* нарушен ($A(x) = false$).

Соответственно для тестирования и принятия решения можно применить логику, показанную на рисунке 3.

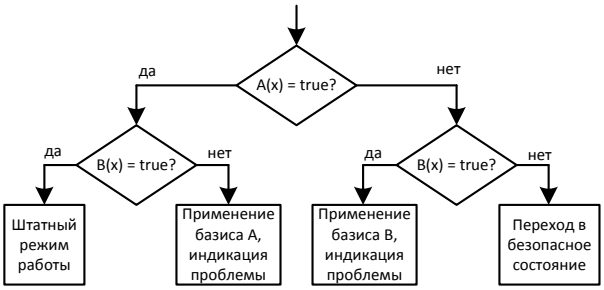

Рисунок 3 – Тестирование базисов

Одной из особенностей данного подхода является то, что сами операции проверки должны быть независимыми от условий выполнения базисов. То есть опираться необходимо либо на общий базис, либо на утверждения, которые находится вне рассматриваемой аксиоматики.

Таким образом, в случае проблем с одним битом регистра система переключится на другой и продолжит выполнять возложенные функции, а время реакции на отказ будет определяться частотой проверок базисов.

Данный алгоритм может быть применен не только к описанным отказам, но и к сбоям ячеек памяти, когда происходит самопроизвольный переход из одного состояния в другое. В этом случае возможна попытка восстановления нарушенного базиса посредством корректного базиса, и в случае успеха система возвращается к исходному состоянию. Данный тип проектирования позволяет создавать АПК, которые являются устойчивыми к ошибкам из-за сбоев по причине электромагнитных помех (*fault-tolerant systems*).

Применение в общем случае для создания отказоустойчивой системы. Рассмотрим последовательность действий, которая позволяет применить диверситетные аксиоматические базисы в общем случае для проектирования отказоустойчивой системы. Этот случай предполагает, что уже присутствует формализация проблемы.

1 Отказ или сбой формируется при условии выполнения некоторых утверждений $X = \{X_1, ..., X_n\}$. Необходимо сформулировать условия, при которых возникает проблема. Например:

– происходит сбой в памяти, при котором изменяется в произвольное состояние не более *N* последовательных ячеек в оперативной памяти с ответственными данными;

– ячейка памяти, выполняющая одну команду ответственной функции программы, изменяется в произвольное состояние;

– отказ воздействует таким образом, что один из бит определенного регистра процессора переходит в фиксированное состояние (постоянное 0 или 1).

2 Найти два таких множества утверждений $Y = \{Y_1, ..., Y_n\}$ и $Z = \{Z_1, ..., Z_n\}$, которые обеспечивают диверситет относительно выполнимости *X*. Соответствующие предыдущим примеры:

– поместить две копии таблицы с данными на адресном расстоянии не менее *N*;

– разработать два экземпляра функции, выполняющей ответственное действие, на основании как *Y*, так и *Z*;

– реализовать выполнение функции на разных подмножествах бит регистра, которое не перекрываются друг с другом.

3 Спроектировать АПК таким образом, чтобы система выполняла свою функцию как на множестве утверждений *Y*, так и на множестве утверждений *Z*.

Данный алгоритм может быть применен не только для проектирования отказоустойчивой системы, но и для её верификации и доказательства безопасности.

Общие выводы. Сложность применения диверситетных базисов зависит от задачи и требуемой глубины решения. Например, простым решением является диверситет по регистрам процессора, когда можно их разделить на два подмножества и реализовывать требуемые функции независимо в каждом из них. В данном случае не требуется специальная подготовка и дополнительная квалификация. Но, с другой стороны, применение может быть достаточно сложным, если необходимо обеспечить диверситет в качестве защиты от нескольких типов отказов с аппаратной интеграцией. В этом случае могут быть востребованы сложные математические модели с применением дополнительных формальных методов для доказательства корректности.

На основании диверситетных базисов могут быть разработаны способы, позволяющие проектировать системы, которые не являются диверситетными, и изменить их функционально так, чтобы обеспечить требуемый диверситет при сохранении необходимых свойств. Примером такого способа является представление любой логической функции на диверситетных базисах ИЛИ-НЕ и И-НЕ, представленных выше. Таким обра-

зом, данный подход может быть интересен с теоретической стороны.

В данной статье не затронуты такие вопросы, как мажорирование, интеграция с аппаратным обеспечением, ввод-вывод, интерфейс между человеком и компьютером, защита от более сложных и непредсказуемых отказов, восстановление после обнаруженного сбоя, формальное доказательство корректности работы диверситетных базисов, самостабилизация при сбоях, вероятностный подход и др. Эти вопросы, как и их сложность и способы решения, зависят от особенностей рассматриваемой задачи, и они могут быть решены на базе описанного общего подхода с помощью диверситетных базисов.

Задачами диверситета в общем случае является множество мероприятий, которые должны сделать две подсистемы как можно более разными. В настоящее время диверситет создается, и его качество оценивается экспертными методами. В данной статье предложен общий подход, позволяющий выполнять данную операцию формализованно, что дает возможность целенаправленно защищаться от типичных отказов, более точно описывать диверситет рассматриваемых систем и заранее задавать уровень диверситета на этапе проектирования или валидации. Описанный общий подход также может быть использован как теоретическая основа построения диверситета и его доказательства с последующей верификацией АПК. Предполагается, что данный подход будет применяться для разработки отказоустойчивых систем, тем самым позволяя улучшить их показатели надежности и безопасности.

Список литературы

1 **Neumann, P. G.** Computer-Related Risks / P. G. Neumann // New York, NY, USA, Addison-Wesley Professional. – 1995. – 384 p.

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

3 **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. – 270 p.

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

5 **Chen, L.** N-Version Programming: A Fault-Tolerance Approach to Reliability of Software Operation / L. Chen, A. Avizienis // Proceedings of the Eighth Annual International Conference on Fault Tolerant Computing. – Toulouse, France. – 1978. – P.3–9.

6 **Knight, J. C.** An Experimental Evaluation of the Assumption of Independence in Multiversion Programming / J. C. Knight and N. G. Leveson // IEEE Transactions on Software Engineering. – 1986. – Vol. SE-12, No. 1.

7 **Brilliant, S.** Analysis of Faults in an N-Version Software Experiment / S. Brilliant, J. Knight, N. Leveson // IEEE Trans. on Software Engineering. – 1990. – Vol. SE-16, No. 2.

8 **Brilliant, S.** The Consistent Comparison Problem in N-Version Programming / J. C. Knight and N. G. Leveson // IEEE Trans. on Software Engineering. – 1989. – Vol. SE-15, No. 11.

9 **Parry, G. W.** Common Cause Failure Analysis: A Critique and Some Suggestions / G. W. Parry // Reliability Engineering and System Safety. – 1991. – Vol. 34, Is. 3. – P. 309–326.

10 **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.

11 **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.

12 **Сапожников, В. В.** Теория дискретных устройств железнодорожной автоматики, телемеханики и связи / В. В. Сапожников, Ю. А. Кравцов, Вл. В. Сапожников; под ред. В. В. Сапожникова // 2-е изд., перераб. и доп. – М. : УМК МПС России. – 2001. – 312 с.

Получено 12.05.2014

B. V. Sivko. Diverse Axiomatic Bases for Development of Safe and Dependable Systems.

In this paper there is an attempt to descript a logic basis and general way to design safe and dependable systems. The notion ‘diverse axiomatic bases’ had been introduced. It is shown that the safe and dependable software and hardware development, which is based on diverse axiomatic bases, allow formalizing terms of diversity and common cause failure. Examples are given of such diverse axiomatic bases and ways how to use for proof of correctness for microprocessor systems. Finally, it is argued that possible important advantages, both theoretical and practical, which may follow from these topics.