Моделирование программно-аппаратных систем и анализ их безопасности

[1] *С.А. Зеленова <sophia@ispras.ru>*
[1,2] *С.В. Зеленов <zelenov@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Национальный исследовательский университет Высшая школа экономики, 101000, Москва, ул. Мясницкая, д. 20.*

Аннотация. В данной статье демонстрируется целесообразность применения языка моделирования программно-аппаратных систем AADL и его расширения Error Model Annex для описания требований безопасности проектируемой системы. Наиболее важным аспектом здесь является возможность описания требований безопасности в терминах, используемых в теории безопасности, таких, как марковские цепи или логико-вероятностные функции, т.к. за годы развития теории было накоплено большое количество весьма полезных результатов. Различные подходы к оценке безопасности систем не конкурируют, но дополняют друг друга, так что наличие некоторой универсальности в описании требований безопасности является весьма ценным качеством.

Ключевые слова: моделирование программно-аппаратных систем; безопасность; анализ дерева неисправностей; анализ видов и последствий отказов; марковский анализ.

DOI: 10.15514/ISPRAS-2017-29(5)-13

Для цитирования: Зеленов С.В., Зеленова С.А. Моделирование программно-аппаратных систем и анализ их безопасности. Труды ИСП РАН, том 29, вып. 5, 2017 г., стр. 257-282. DOI: 10.15514/ISPRAS-2017-29(5)-13

1. Введение

Бурное развитие техники в XX веке поставило вопросы защиты человека от всевозможных проблем (аварий, техногенных катастроф) связанных со сложными техническими системами. Увеличение количества техники, усложнение как внутреннего строения, так и внешних связей технических систем затрудняет поиск проблемных «тонких» мест и требует автоматизированных методов оценки надежности и безопасности системы. Кроме того, в силу тех же причин отказы и аварии часто влекут большие экономические и людские потери. Все это привело к возникновению

совершенно новых дисциплин — теории надежности, теории безопасности, теории живучести [3], [8], [15], [21], [22], [26].

Термины надежности, безопасности, живучести хотя и близки, однако имеют большие смысловые отличия:

- *надежность* — способность системы в нормальных условиях работать безотказно;
- *безопасность* — способность системы функционировать не переходя в опасное состояние;
- *живучесть* — способность системы функционировать под воздействием поражающих факторов, возникающих в результате каких-либо экстраординарных событий (взрыв, пожар, землятрясение и т. п.).

Из этих определений видно, что понятия надежности и безопасности структурно очень похожи, различие между ними состоит в том, что в основе понятия надежности лежит понятие отказа, а в основе понятия безопасности — понятие опасного состояния. Опыт показывает, что различия в определениях существенны для реальных систем, так, надежная система может быть и опасной, и неживучей. Корабль, который плавает не один десяток лет, проходит все ремонты и технически исправен, может затонуть при хорошей погоде за несколько минут при получении пробоины — это пример надежной, но неживучей системы (крушение теплохода «Адмирал Нахимов» в 1986 году [6]). Другой пример — склад боеприпасов: трудно представить, какой отказ в нем может произойти в нормальных условиях, т.е. эта система надежна, но между тем, очевидно, что такая система становится опасной в случае взрыва боеприпасов.

В нашей стране теории надежности, безопасности и живучести сложно структурированных систем получили свое развитие в послевоенные годы. Исследованием данных вопросов занимались как инженеры (Дедков В.К., Рябинин И.А., Северцев Н.А., Ильичев А.В., и др), так и математики, (Гнеденко Б.В., Беляев Ю.К., Коваленко И.Н., Каштанов В.А., Соловьев А.Д. и др.) [2], [3], [4], [7]. [8], [9], [10], [26]. В настоящей работе мы упомянем две методологии, использующиеся в данных теориях (см. раздел 2).

Размеры и сложность современных систем таковы, что их ручной анализ был бы чрезвычайно трудоемким и дорогостоящим, а в некоторых случаях и просто невозможным. Таким образом, привлечение автоматизированных средств и, как следствие, разработка формальных моделей системы, становится насущной необходимостью.

Одним из современных средств описания архитектуры программно-аппаратных систем является Architecture Analysis & Design Language (AADL) [18], [33]. Для спецификации модели сбоев системы предназначено специальное расширение Error Model Annex [34] языка AADL.

Разработчики стандарта AADL предоставляют инструмент OSATE [31], который позволяет редактировать и анализировать AADL-модели, в частности выполнять отдельные аспекты анализа рисков [17]. Однако, следует отметить, что OSATE предоставляет лишь некоторую базовую поддержку языка, а в плане анализа моделей в основном полагается на сторонние инструменты, такие как OpenFTA [30] для анализа дерева неисправностей и PRISM [24] для марковского анализа. OSATE производит разбор AADL-модели и подготовку необходимых входных данных для запуска внешнего анализатора. В ряде случаев функциональность анализа рисков в OSATE имеет серьезные ограничения, например, обрабатывается только один уровень вложенности компонентов модели, а также отсутствует поддержка комбинаций различных сбоев в компоненте.

В настоящей статье мы представляем алгоритмы полнофункционального, лишенного указанных недостатков, анализа рисков: анализ дерева неисправностей, анализ видов и последствий отказов, марковский анализ [32]. Алгоритмы реализованы в разрабатываемом в ИСП РАН инструменте MASIW [1], [23], [29] для поддержки автоматизированного проектирования и анализа программно-аппаратных систем.

Основная часть статьи построена следующим образом. В разделе 2 дан обзор методов оценки безопасности. В разделе 3 представлены возможности языка AADL для описания архитектуры системы и для спецификации модели сбоев системы. В разделе 4 приводятся алгоритмы анализа рисков на основе описаний AADL. В разделе 5 рассмотрен пример моделирования и анализа безопасности системы передачи данных. В разделе 6 подводятся итоги статьи.

2. Элементы теории безопасности

2.1 Логико-вероятностный анализ

Основным методом логико-вероятностного анализа (ЛВА) [7], [9], [10], [13], [19], [25] является сопряжение методов математической логики и теории вероятности.

Общая схема может быть описана следующим образом. Исходя из поставленной цели (изучение надежности, безопасности) для исследуемой системы определяются высказывания об элементах системы и вероятность того, что данное высказывание верно (неверно). Например, для имеющегося соединительного провода высказывание может звучать как «провод цел» или «провод оборван». Вероятность истинности такого высказывания берется из соображений технического и опытного характера. На этом этапе первую часть анализа составляет структуризация системы, приводящая к выделению отдельных элементов, высказывания о которых приобретают характер терминальных. Так, можно говорить об отказе или опасном состоянии целого блока не вдаваясь в подробности его устройства, если имеются основания рассматривать данный блок как единое целое. Важно заметить, что об одном

элементе системы можно сделать несколько высказываний («провод оборван», «провод перегорел», «провод перекушен кусачками злоумышленником»), у каждого из которых будет своя вероятность истинности.

Далее, исходя из общей структуры системы, терминальные высказывания об отдельных ее элементах организуются в логическую формулу, которая уже будет говорить об отказе/опасном состоянии системы в целом. То есть в математическом виде прописывается логика возникновения отказа/опасного состояния. Важно понимать взаимосвязи терминальных элементов в смысле зависимости или независимости их вероятностных характеристик. Не всегда можно считать вероятности истинности терминальных элементов независимыми, это создает дополнительные трудности.

На следующем этапе логическая формула отказа/опасного состояния системы должна быть преобразована в вероятностную функцию. Факторы, усложняющие этот процесс: немонотонность логической функции, повторение аргументов (терминальных элементов) и др. Для упрощения обычно используются различные алгоритмы ортогонализации.

Логико-вероятностный анализ широко используется в самолетостроении, часто в виде различных упрощений и модификаций. Примером такой модификации является анализ дерева неисправностей [12], [16], [27], когда логическая функция строится для какого-то одного события верхнего уровня. Отталкиваясь от этого события, исследователь-аналитик постепенно спускается на нижележащие уровни, выискивая всевозможные причины, приводящие к событиям верхних уровней, и, таким образом, строя искомую логическую функцию.

Достоинства ЛВА:

- простота и доступность для понимания;
- наглядность;
- не слишком большой объем;
- возможность использования различных уровней абстракции;
- возможность автоматизации значительной части анализа при наличии формальной модели системы.

Недостатки ЛВА:

- отсутствие понятия времени в описании логической функции и, как следствие этого, необходимость строгого регламентирования всех событий и трудности описания специфических ситуаций, зависящих от времени (повторные отказы, перемежающиеся отказы, наложение одних событий на другие, самовосстанавливающиеся системы);
- трудности построения вероятностной функции в некоторых случаях (повторные терминальные символы, немонотонная логическая функция).

2.2 Вероятностные автоматы и марковские цепи

Другой подход к оценке надежности/безопасности/живучести системы — использование теории марковских процессов [11]. Здесь необходимо ввести одно понятие из теории конечных автоматов.

Вероятностный или *стохастический автомат* — это конечный автомат, в котором нет входных и выходных символов, а переходы между состояниями осуществляются с заданной вероятностью.

Исследуемая на предмет безопасности система может быть описана в виде вероятностного автомата, содержащего особые состояния, соответствующие отказам (или опасным состояниям). Вероятность перехода в такое состояние — это вероятность возникновения отказа/опасного состояния.

Во многих случаях можно считать, что будущее технической системы определяется ее текущим состоянием, так что построенный вероятностный автомат является марковской цепью и к нему применимы все результаты теории марковских цепей.

Марковский анализ [11], [14], [20] целевой системы дает достаточно полную информацию о её надежности и безопасности. Основная проблема в том, что полученная модель может быть очень объемной.

Достоинства марковского анализа:

- наличие понятия временной последовательности событий в самой модели и, вследствие этого, возможность оценки поведения системы в разные моменты времени;
- простота описания циклических событий;
- возможность автоматизации значительной части анализа при наличии формальной модели системы.

Недостатки марковского анализа:

- большой объем модели;
- достаточно сложный математический аппарат.

Как видно, ни марковский анализ, ни логико-вероятностный анализ не имеют неоспоримых преимуществ друг перед другом, и должны использоваться как взаимно дополняющие.

2.3 Специфические методы анализа

Кроме общих методов анализа безопасности, таких как, ЛВА и марковский анализ, имеется большое число специфических методов, направленных на анализ конкретных инженерных решений. Эти методы несомненно полезны, так как хорошо учитывают специфику ситуации, но они не обладают общностью, что сужает область их применения.

3. Язык моделирования архитектуры AADL

Язык AADL (Architecture Analysis and Design Language) [33] в настоящее время де-факто является стандартом для моделирования архитектуры и требований по безопасности в аэрокосмической области. В данном разделе мы продемонстрируем, как на базе выразительных возможностей AADL можно автоматизировать различные виды анализа безопасности, и проиллюстрируем процесс моделирования архитектуры и требований по безопасности на примере фрагмента реальной системы.

3.1 Моделирование архитектуры

Язык AADL предназначен для формального описания архитектуры программно-аппаратных систем. В AADL имеются встроенные средства для описания следующих сущностей и отношений между ними:

- на аппаратном уровне: элементарное устройство, процессор, память, порт, шина;
- на функциональном уровне: процесс, подпрограмма, соединение (канал передачи информации);
- есть возможность описать систему на некотором виртуальном абстрактном уровне: виртуальный процессор, виртуальная шина, поток исполнения.

AADL-описание отражает внутреннюю структуру, подчиненную строгой иерархии: система состоит из подчиненных подсистем и элементов, у этих подсистем могут быть свои подсистемы и элементы и т. д. Кроме того, функциональные компоненты могут быть привязаны к аппаратным напрямую или через несколько виртуальных уровней, что позволяет рассматривать систему на нескольких уровнях абстракции.

3.2 Описание требований по безопасности

Для задания требований безопасности разработчиками языка AADL было создано специальное расширение этого языка — Error Model Annex (EM) [34]. Для каждого компонента системы, описанного на языке AADL, можно указать особые требования, накладываемые на этот компонент, которые могут интерпретироваться как требования безопасности. Error Model Annex включает следующие изобразительные возможности:

- для каждого компонента может быть задан конечный автомат, состояния которого — это штатные и нештатные (опасные/отказные) состояния данного компонента;
- влияние сбоев компонентов системы на другие компоненты описывается посредством указания логических условий распространения ошибок между различными типами компонентов в различных состояниях;

- переходы между состояниями компонента осуществляются при выполнении некоторых логических условий, в зависимости от наличия сбоев — как внутренних, так и «наведенных», т.е. распространившихся в данный компонент извне;
- для внутренних сбоев задаются вероятности их возникновения;
- модель ошибок компонента, составленного из подкомпонентов, описывается как композиция моделей ошибок подкомпонентов, при этом для составления этой композиции используются логические операции.

Таким образом, Error Model Annex использует для описания требований безопасности и термины логико-вероятностного анализа, и термины марковского анализа.

3.3 Математическая модель требований по безопасности

Пусть C — множество всех компонентов данной модели. Рассмотрим отдельный компонент *c*. Связанные с компонентом *c* объекты (множества, функции) будем записывать с префиксом «c.». Напомним, что для множества M запись 2^M означает множество всех подмножеств множества M.

Пусть c.S — множество состояний компонента *c*, c.E — множество внутренних сбоев, c.Q — множество сбоев, приходящих в *c* извне, c.R — множество сбоев, распространяющихся из *c* вовне. Пусть задана функция переходов между состояниями c.F: $c.S \times 2^{c.E} \times 2^{c.Q} \to c.S$, а также функция распространения сбоев в данном состоянии c.G: $c.S \times 2^{c.E} \times 2^{c.Q} \to 2^{c.R}$. Пусть также задана функция st: $C \to \{ c.S \}$, которая для данного компонента возвращает его текущее состояние. Кроме того, пусть для всех внутренних сбоев задана функция p вероятности их возникновения.

В EM-описании функции c.F и c.G имеют следующий вид. Переход из состояния $c.s_i$ в состояние $c.s_j$ осуществляется, если выполняется некоторое заданное логическое условие $c.f_{i,j}(c.e_1, \dots , c.e_L, c.q_1, \dots , c.q_M)$ от внутренних и внешних сбоев. Аналогично, в состоянии $c.s_j$ осуществляется распространение сбоя $c.r_k$, если выполняется некоторое заданное логическое условие $c.g_{j,k}(c.e_1, \dots , c.e_L, c.q_1, \dots , c.q_M)$ от внутренних и внешних сбоев. Терминальные высказывания относительно сбоев в условиях $c.f_{i,j}$ и $c.g_{j,k}$ имеют вид «произошел сбой».

Если компонент *c* является композицией компонентов $d_1, \dots , d_n$, то его состояния задаются функцией c.H: $d_1.S, \dots , d_n.S \to c.S$. В EM-описании функция c.H имеет следующий вид. Компонент *c* находится в состоянии $c.s_j$, если выполняется некоторое заданное логическое условие $c.h_j(d_1.s_1, \dots , d_n.s_n)$. Терминальные высказывания относительно состояний подкомпонентов d_i в условиях $c.h_j$ имеют вид «d_i находится в состоянии $d_i.s_i$», т.е. «$st(d_i) = d_i.s_i$».

4. Анализ рисков на основе описаний AADL

4.1 Анализ дерева неисправностей

4.1.1 Построение дерева неисправностей

Заметим, что каждое логическое условие можно однозначно представить в виде дерева, где листьям соответствуют терминальные операнды, а внутренним узлам — логические операции. Будем называть такое дерево LF-деревом. Верно и обратное: для любого LF-дерева однозначно восстанавливается соответствующее логическое условие.

Дерево неисправностей — это LF-дерево, соответствующее условию перехода некоторого компонента c в состояние $c.s_j$.

Шаг построения дерева неисправностей начинается с рассмотрения целевого состояния $c.s_j$ компонента *c*. На одном шаге построения требуется выявить локальные причины перехода компонента *c* в состояние $c.s_j$. Пусть в строящемся дереве есть узел A, соответствующий условию «$st(c) = c.s_j$».

Если EM-описание для $c.s_j$ имеет вид композиции, то происходит поиск соответствующего условия $c.h_j$ от состояний подкомпонентов. Дерево t логического условия $c.h_j$ вставляется в дерево неисправностей, так что корень t становится узлом A. Далее построение дерева неисправностей продолжается для фигурирующих в $c.h_j$ терминальных высказываний относительно состояний подкомпонентов.

Если EM-описание для $c.s_j$ имеет вид функции перехода из другого состояния, то ищутся все соответствующие исходные состояния $c.s_i$ и условия $c.f_{i,j}$. Далее строится дерево t' для логического условия

$$OR_i \ (st(c) = c.s_i) \text{ AND } c.f_{i,j} ,$$

где «OR_i» означает оператор «ИЛИ» по всем индексам *i*, и вставляется в дерево неисправностей, так что его корень становится узлом A.

Далее построение дерева неисправностей продолжается для терминальных высказываний:

- для условия «$st(c) = c.s_i$» производится следующий шаг построения дерева;
- для условия «в компоненте *c* произошел внутренний сбой c.e» в дереве оставляется соответствующий листовой узел;
- для условия «в компоненте *c* произошел внешний сбой c.q» происходит поиск источников этого внешнего сбоя.

Поиск источников внешнего сбоя c.q для компонента c происходит так. Для всех компонентов b_i, которые распространяют сбой $b_i.r_k = c.q$ в компонент *c*,

ищутся все соответствующие состояния $b_i.s_j$ и условия $b_i.g_{j,k}$. Далее строится дерево t" для логического условия

$$OR_i \ OR_j \ OR_k \ (st(b_i) = b_i.s_j) \text{ AND } b_i.g_{j,k}$$

и вставляется в дерево неисправностей, так что его корень заменяет узел для рассматриваемого высказывания «в компоненте *c* произошел внешний сбой c.q». Построение дерева неисправностей для терминальных высказываний дерева t" продолжается аналогично.

В результате в листьях дерева будут содержаться только условия вида «в компоненте c_i произошел внутренний сбой $c_i.e_{l_i}$»

4.1.2 Минимальные сечения

После построения дерева неисправностей можно вычислить вероятность перехода исходного компонента *c* в состояние $c.s_j$. Для этого необходимо сперва упростить логическую формулу, соответствующую данному дереву, и привести ее к некоторому каноническому виду. Прежде чем приступить к описанию этого канонического вида, напомним следующее определение.

Минимальным сечением называется такая наименьшая комбинация внутренних сбоев, что если все эти сбои произойдут, то это повлечет возникновение в дереве события верхнего уровня (т.е. переход компонента *c* в состояние $c.s_j$).

Вычисление минимальных сечений для данной логической формулы происходит путем приведения формулы к дизъюнктивной форме с использованием булевских законов дистрибутивности и поглощения. В результате формула примет вид дизъюнкции конъюнкций, причем каждая конъюнкция будет соответствовать какому-то минимальному сечению.

Пусть все внутренние сбои являются независимыми событиями. Тогда вероятность любой их конъюнкции равна произведению вероятностей множителей. Что касается вероятности дизъюнкции, то для ее точного вычисления необходимо применять формулу включений-исключений. Однако, поскольку в реальности вероятности внутренних сбоев достаточно малы (по меньшей мере порядка $10^{-4}..10^{-6}$), членами высших порядков в формуле включений-исключений пренебрегают.

Таким образом, если логическая формула для данного дерева неисправностей приведена к дизъюнкции конъюнкций, соответствующих минимальным сечениям, то вероятность события верхнего уровня (приближенно) равна соответствующей сумме произведений вероятностей внутренних сбоев из минимальных сечений.

4.1.3 Ранжирование сбоев

Все внутренние сбои, встречающиеся в данном дереве неисправностей, можно ранжировать по величине их вклада в вероятность возникновения в дереве события верхнего уровня (т.е. перехода компонента *c* в состояние $c.s_j$).

Ранжирование первичных событий в соответствии с их значимостью в смысле наступления события верхнего уровня может производиться разными методами [27]. Пусть S – событие верхнего уровня, E – первичное событие, $\bar{E}$ – отрицание E, p – функция вероятности событий. Напомним, что если A и B – два события, то p(A|B) – это вероятность наступления события A при условии, что наступило событие B.

Мера значимости «*Risk Achievement Worth*» – «стоимость возрастания риска»: p(S|E) / p(S). Эта величина показывает, насколько увеличивается риск при условии, что событие E происходит постоянно (т. е. соответствующий компонент абсолютно не надежен). Это индикатор, насколько важно поддерживать текущий уровень надежности соответствующего компонента.

Мера значимости «*Risk Reduction Worth*» – «стоимость уменьшения риска»: p(S) / p(S|$\bar{E}$). Эта величина показывает, максимальное возможное уменьшение риска, которое можно ожидать при увеличении надежности соответствующего компонента (вплоть до его абсолютной надежности, когда событие E никогда не происходит).

Мера значимости *по Бинбауму* (Birnbaum): $\partial p(S)/\partial p(E) = p(S|E) - p(S|\bar{E})$. Эта величина показывает, в какой степени изменение надежности E влияет на повышение надежности S.

Мера значимости *по Фусселу-Весли* (Fussel-Vesely): $(p(S) - p(S|\bar{E})) / p(S)$. Эта величина показывает степень критичности события E, т.е. относительную величину вклада события E в величину вероятности наступления S.

Мера значимости «*важность диагностики*»: $p(E|S) = p(E)\cdot p(S|E) / p(S)$. Эта величина показывает вероятность сбоя E при условии сбоя S, т.е. насколько приоритетно проверять, был ли сбой E, при возникновении сбоя S.

4.2 Анализ видов и последствий отказов

Пусть модель состоит из компонентов c_1, ... , c_N. Состояние всей модели полностью определяется состояниями всех ее компонентов: $c_1.S \times ... \times c_N.S$.

Рассмотрим отказ следующего вида. Пусть каждый из компонентов c_i находится в некотором состоянии $c_i.s_i$, и пусть в каждом из компонентов c_i произошли некоторые внутренние сбои $\{c_i.e_{i,j}\}$.

В результате этого отказа компоненты модели могут поменять свое состояние и начать распространять какие-то сбои вовне. Далее, в условиях прихода этих внешних сбоев в зависимые компоненты, компоненты модели опять могут поменять свое состояние и начать распространять еще какие-то сбои вовне. И так далее, модель будет эволюционировать до тех пор, пока не достигнет некоторого стабильного состояния.

Требуется найти, в каком состоянии модель стабилизируется. Это состояние модели будет выражать последствия исходного отказа.

Рассмотрим подробно один шаг эволюции модели. В общем случае, в начале шага каждый компонент c_i находится в некотором состоянии $c_i.s_i$ и имеет сбои: внутренние $\{c_i.e_{i,j}\}$ и внешние $\{c_i.q_{i,j}\}$. Применяя для соответствующих компонентов сперва функцию $c_i.F$, а затем функцию $c_i.H$, найдем, в какое новое состояние $c_i.s_i'$ при текущих условиях перейдет каждый компонент. Затем, применяя для каждого компонента функцию $c_i.G$, найдем, какие сбои $\{c_i.r_{i,j}\}$ станет распространять компонент c_i в состоянии $c_i.s_i'$ при условии сбоев $\{c_i.e_{i,j}\}$ и $\{c_i.q_{i,j}\}$. После этого, определим компоненты-приемники найденных сбоев $\{r_{i,j}\}$ и таким образом получим, новый набор внешних сбоев $\{c_i.q_{i,j}'\}$.

Как видим, шаг эволюции модели зависит не только от состояний компонентов, но также и от наборов сбоев $\{c_i.e_{i,j}\}$ и $\{c_i.q_{i,j}\}$. Таким образом, стабильным можно считать такое состояние модели, когда в результате шага эволюции в компонентах не меняются как состояния, так и наборы сбоев.

Поскольку все заданные множества ($c_i.S$, $c_i.E$, $c_i.Q$) являются конечными, процесс эволюции на некотором шаге придет в такое состояние, в котором модель уже находилась ранее. После этого модель начнет эволюционировать по циклу. В том случае, если этот цикл является петлей из одного состояния, модель стабилизируется в этом состоянии. Если же этот цикл содержит более одного состояния модели, можно говорить, что модель задана некорректно. Вопрос изучения критериев корректности модели выходит за рамки настоящей работы.

4.3 Построение марковской цепи

Давно разработан классический математический аппарат [11], [14], [20] для проведения марковского анализа по заданной марковской цепи. Здесь мы опишем алгоритм построения марковской цепи на основе EM-описания.

Для данной модели, состоящей из компонентов c_1, ... , c_N, рассмотрим вероятностный автомат. Состояниями этого автомата являются кортежи состояний всех компонентов $s_1 \times ... \times s_N$. Переход автомата в другое состояние происходит при возникновении некоторого набора внутренних сбоев в компонентах. Для данного состояния s автомата и данного набора внутренних сбоев $\{e_{i,j}\}$ можно запустить процесс анализа последствий этого отказа модели и найти состояние s', в котором модель стабилизируется. В этом случае будем говорить, что в вероятностном автомате имеется переход $s \rightarrow s'$ с вероятностью $\prod_{i,j} p(e_{i,j})$.

Если в модели имеется N компонентов и L различных внутренних сбоев, причем каждый компонент может находиться не менее чем в двух состояниях, то вероятностный автомат будет содержать не менее чем 2^N состояний, и из каждого состояния будет выходить 2^L переходов.

Однако, на практике не все состояния вероятностного автомата являются достижимыми из некоторого начального состояния. Таким образом, если строить вероятностный автомат постепенно, по мере построения переходов из уже достигнутых состояний, то количество полученных состояний будет существенно меньше экспоненты.

Кроме того, в реальных системах отдельные сбои имеют довольно низкую вероятность, порядка 10^{-4} .. 10^{-12}. Соответственно, вероятность одновременного возникновения нескольких сбоев является исчезающе малой величиной. Таким образом, на практике наборы сбоев $\{e_{i,j}\}$ можно ограничить парами или тройками, а значит количество переходов из каждого состояния вероятностного автомата будет ограничено полиномом степени 2 или 3.

Более подробное описание алгоритмов построения марковской цепи и проведения марковского анализа на основе EM-описания можно найти в [5].

5. Пример моделирования и анализа системы

Процесс моделирования системы с использованием поддержки требований безопасности включает следующие шаги:

- анализ общей ситуации
- формулирование различных проблем на уровне системы
- выделение компонентов, ответственных за ситуацию
- классификация и построение описания проблем, связанных с отдельными компонентами виртуального уровня
- анализ связей между виртуальным и физическим уровнем
- классификация и построение описания проблем, связанных с отдельными компонентами физического уровня

Проиллюстрируем процесс моделирования и анализа безопасности на примере сети передачи данных между датчиками, электроприводами, вычислительные модули, дисплеями и т.п. внутри авиационного судна. Абоненты сети, которым требуется передавать друг другу информацию, связаны между собой проводами. Несколько десятилетий назад, когда электрооборудования на самолетах было немного, для повышения безопасности проектировщики руководствовались правилом: каждому каналу передачи информации — свой провод. В настоящее время количество абонентов сети, а с ними и количество каналов выросло настолько, что буквальное следование этому принципу привело бы к непомерному росту суммарного веса необходимых проводов. Решением проблемы стало использование одних и тех же проводов несколькими каналами. Уменьшение количества проводов потребовало более сложной топологии сети, в частности применения коммутаторов (switch) в местах разветвления.

Один из подходов, реализующих указанный принцип, называется Avionics Full-Duplex Switched Ethernet (AFDX) [28] и в настоящее время де-факто является стандартом в этой области.

Канал передачи информации в AFDX-сети не является, как раньше, единоличным владельцем провода, однако на логическом уровне каждый канал моделируется так называемым виртуальным каналом (virtual link). Таким образом, проектировщик мысленно продолжает жить в старом подходе («один канал — один провод»), оперируя виртуальными каналами, а в реальности каждый виртуальный канал использует какой-то свой физический маршрут, состоящий из некоторого набора проводов и коммутаторов сети (см. рис.1).

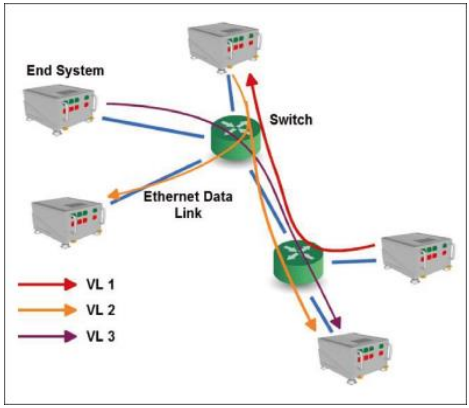

Рис. 1. Виртуальные каналы

Fig. 1. Virtual links

Для обеспечения безопасности в AFDX применяется резервирование: для каждого виртуального канала используются два независимых физических маршрута, по каждому из которых передаются идентичные копии сообщения.

5.1 Моделируемая система

Рассмотрим процесс моделирования и анализа сети передачи данных.

Пусть абонентами сети являются:

- датчик (sensor),
- вычислительный модуль (computer),
- электропривод (actuator),
- дисплей (monitor).

Перечислим логические соединения между абонентами (см. рис. 2):

- датчик отправляет свои текущие показания на вычислительный модуль,

- вычислительный модуль анализирует полученные показания датчика и отправляет управляющие команды на электропривод,
- также вычислительный модуль отправляет текущую информацию о параметрах полета на дисплей,
- кроме того, электропривод в случае возникновения в нем аварийной ситуации отправляет предупреждающий сигнал на дисплей.

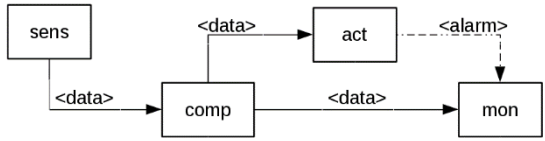

Рис. 2. Функциональная модель

Fig. 2. Functional model

Моделирование сети AFDX состоит в описании виртуальных каналов. Каждому логическому соединению соответствует один виртуальный канал в сети AFDX, а для обеспечения резервирования для каждого вируального канала определены два виртуальных маршрута «a» и «b», каждому из которых соответствует свой набор аппаратных компонентов системы.

Пусть абоненты нашей сети выполняются каждый на своем процессоре, и каждый процессор использует для выхода в AFDX-сеть свою сетевую карту. В местах разветвления информационных потоков в сети используются коммутаторы. На рис.3 представлена аппаратная часть архитектуры системы без резервирования. Собственно сеть (провода и коммутаторы) на рисунке выделена пунктирной рамкой. Для обеспечения резервирования в системе мы организуем два идентичных экземпляра представленной на рисунке сети.

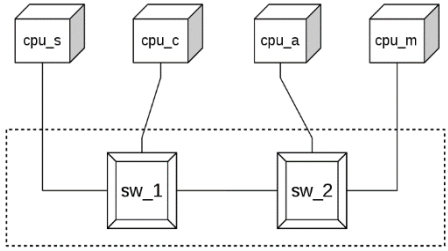

Рис. 3. Аппаратная часть: процессоры и сеть

Fig. 3. Hardware: processors and net

Архитектура системы в целом показана на рис.4 и состоит из трех частей. В верхней части представлена функциональная модель с абонентами сети и логическими соединениями. В средней части представлена модель сети AFDX,

которая содержит четыре элемента — виртуальные каналы для каждого логического соединения. В нижней части представлена аппаратная модель HW, которая содержит процессоры, два экземпляра сети net_a и net_b (чтобы не загромождать рисунок, внутреннее содержимое сетей скрыто), а также энергосистема power. Энергосистема состоит из нескольких батарей, каждая из которых обеспечивает энергией какие-то два устройства (процессоры и комммутаторы).

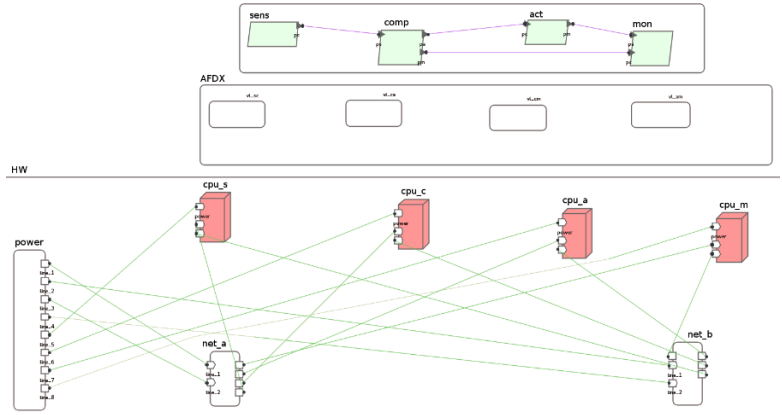

Рис. 4. Модель системы

Fig. 4. System model

На рис.5 подробно показана привязка одного логического соединения к аппаратным компонентам посредством виртуального канала и виртуальных путей.

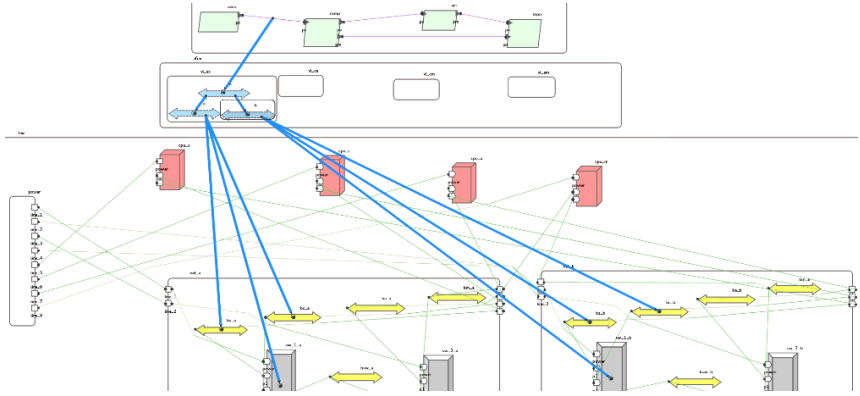

Рис. 5. Привязка логического соединения к аппаратным компонентам

Fig. 5. Binding of a connection to hardware components

5.2 Спецификация опасных состояний компонентов

Изучим возможные повреждения отдельных компонентов и их влияние на работоспособность других компонентов.

Прежде всего сформулируем возможные проблемы, которые могут представлять опасность для системы в целом:

- отказ какого-то процесса-абонента,
- потеря информации для управления самолетом,
- потеря предупреждающей сигнализации.

Далее необходимо выразить перечисленные функциональные опасности через комбинации отказов компонентов на аппаратном уровне.

Отказ процесса-абонента происходит в случае отказа соответствующего процессора.

Потеря информации на логическом соединении происходит в случае если одновременно отказали оба виртуальных пути соответствующего виртуального канала, поскольку виртуальные пути в рамках одного виртуального канала включены параллельно. Отказ виртуального пути происходит в случае отказа хотя бы одного из аппаратных компонентов, к которым привязан этот виртуальный путь, поскольку эти аппаратные компоненты в рамках виртуального пути включены последовательно.

Теперь перейдем собственно к описанию повреждений отдельных компонентов.

Изначально, каждый компонент находится в рабочем (Operational) состоянии. Если в компоненте или в его окружении возникнут какие-то повреждения, то данный компонент может частично или полностью терять свою работоспособность.

Для системы в целом и для процессов-абонентов введем три опасных состояния, соответствующих перечисленным выше функциональным опасностям: Failed_NoService, Failed_AppMsg, Failed_AlarmMsg.

Для остальных компонентов (виртуальные каналы и пути, процессоры, коммутаторы, батареи) введем одно опасное состояние Failed, когда компонент полностью теряет способность выполнять заданное назначение.

EM-описание условия нахождения каждого компонента в конкретном состоянии может иметь одну из двух форм:

- явное описание переходов между состояниями;
- неявное описание переходов в зависимости от состояний подкомпонентов.

Для иллюстрации неявной формы рассмотрим нашу систему в целом. В соответствии с декомпозицией, система переходит, например, в состояние Failed_NoService, если хотя бы один из процессов-абонентов перешел в состояние Failed_NoService.

В EM-описании состояние компонента-композиции задается в виде логической функции от состояний подкомпонентов:

```
composite error behavior
      states
             [   sens.Failed_NoService
              or comp.Failed_NoService
              or act.Failed_NoService
              or mon.Failed_NoService
             ]-> Failed_NoService;
```

Явная форма задания состояний компонента включает следующие описания:

- внутренние сбои (events);
- собственно переходы между состояниями компонента (transitions);
- влияние на соседние компоненты (propagations).

Для примера рассмотрим EM-описание процессора. Внутренним сбоем здесь является поломка всего процессора:

```
events
      Failure : error event;
```

Далее опишем переходы между состояниями процессора. В результате поломки процессора происходит его переход в состояние Failed. В это состояние он также переходит и при отказе энергосистемы:

```
transitions
      Operational -[ Failure ]-> Failed;
      Operational -[ power{NoPower} ]-> Failed;
```

Наконец, опишем влияние процессора в различных его состояниях на соседние компоненты. Отказ процессора повлияет на работоспособность привязанного к нему процесса:

```
propagations
      Failed -[]-> bindings( NoService );
```

5.3 Анализ безопасности

Проведем анализ безопасности описанной выше системы.

Пусть интенсивности внутренних сбоев компонентов равны значениям из следующей таблицы:

Компонент	**Сбой**	**Интенсивность**
Коммутатор	Отказ	$2.5 \cdot 10^{-5}$
Процессор	Отказ	$2.5 \cdot 10^{-5}$
Батарея	Разряжена	$1.35 \cdot 10^{-5}$

Результаты анализа деревьев неисправностей для сформулированных в начале функциональных опасностей системы приведены в следующей таблице:

Функциональная опасность	**Логическое выражение (по дереву неисправностей)**	**Логическое выражение (по минимальным сечениям)**	**Вероятность**
Потеря предупреждающей сигнализации	[⩾2] ((sw_2_a.Failure ⋁ battery_2.Depleted) , (sw_2_b.Failure ⋁ battery_2.Depleted))	(battery_2.Depleted + (sw_2_a.Failure * sw_2_b.Failure))	0.0000135
Потеря информации для управления	([⩾2] ((sw_1_b.Failure ⋁ battery_1.Depleted) , (sw_1_a.Failure ⋁ battery_1.Depl ...	(battery_1.Depleted + battery_2.Depleted + (sw_1_a.Failure * sw_1_b.Failure) + (sw_1_a.Failu ...	0.000027
Отказ	((cpu_s.Failure ⋁ battery_3.Depleted) ⋁ cpu_c.Failure ⋁ (cpu_a.Failure ⋁ battery_4.Depleted) ⋁ cpu_m.Failure)	(battery_3.Depleted + battery_4.Depleted + cpu_a.Failure + cpu_c.Failure + cpu_m.Failure + cpu_s.Failure)	0.000127

Как видно, для опасностей «Потеря предупреждающей сигнализации» и «Потеря информации для управления» вероятности имеют порядок 10^{-5}, хотя из соображений резервирования виртуальных каналов эти вероятности должны иметь порядок $10^{-9}..10^{-10}$ (поскольку вероятности внутренних сбоев имеют порядок 10^{-5}).

Меры значимости внутренних сбоев для опасности «Потеря предупреждающей сигнализации» приведены в следующей таблице:

ID	**Description**	**RAW**	**RRW**	**Birnbaum**	**Fussel-Vesely**	**Diagnostic**
battery_2. Depleted	Батарея разряжена	74071.15	21601.4	1.0	0.9999537	0.9999537
sw_2_a. Failure	Отказ	2.851	1.000046	0.0000245	0.0000463	0.0000713
sw_2_b. Failure	Отказ	2.851	1.000046	0.0000245	0.0000463	0.0000713

Как видно, сбой «Батарея разряжена» имеет чрезвычайно большую значимость для данной функциональной опасности. Анализируя логическое выражение по минимальным сечениям для опасности «Потеря предупреждающей сигнализации», видно, что одно из минимальных сечений состоит из единственного сбоя «Батарея разряжена». Это подтверждает и анализ видов и последствий отказов:

Item(s)	Initial failure mode(s)	End effect	Sev	P
sw_2_a	Отказ	a.[propagation]	9	0.000125
		sw_2_a.Failed	9	0.000125
sw_2_b	Отказ	b.[propagation]	9	0.000125
		sw_2_b.Failed	9	0.000125
battery_2	Батарея разряжена	Model. Failed_AppMsg	10	0.0000675

Из этой таблицы следует, что внутренний сбой в коммутаторе приводит лишь к отказу в коммутаторе и к распространению ошибки на уровне виртуального пути («a» или «b»), а внутренний сбой батареи battery_2 приводит к опасному состоянию Failed_AppMsg («Потеря предупреждающей сигнализации») уровня всей системы с максимальным значением «Важность» (Severity) = 10.

Причина такого положения дел кроется в следующей архитектурной ошибке, которая сводит на нет резервирование сети. А именно, коммутаторы sw_1_a и sw_1_b (резервирующие друг друга) подключены к одной батарее battery_1 (см. рис.6), а коммутаторы sw_2_a и sw_2_b (тоже резервирующие друг друга) подключены к одной батарее battery_2.

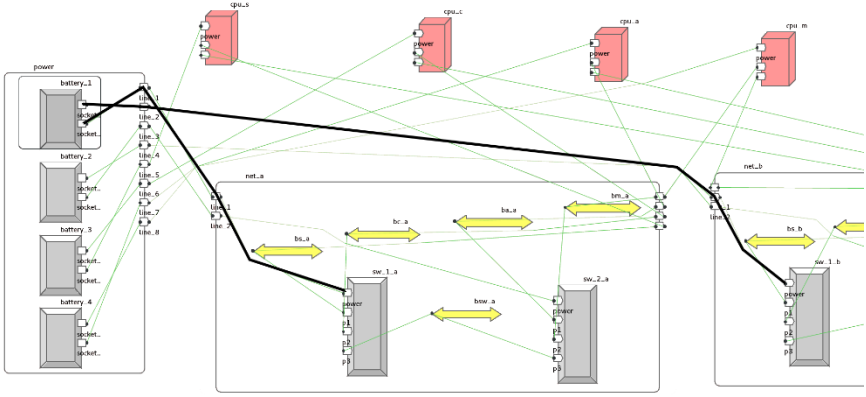

Рис. 6. Два резервирующих друг друга коммутатора подключены к одной батарее

Fig. 6. Two corresponding redundant switches connected to one battery

Для исправления этой ошибки требуется переподключить коммутаторы к батареям, так чтобы резервирующие друг друга коммутаторы были подключены к разным батареям, например, как это изображено на рис.7.

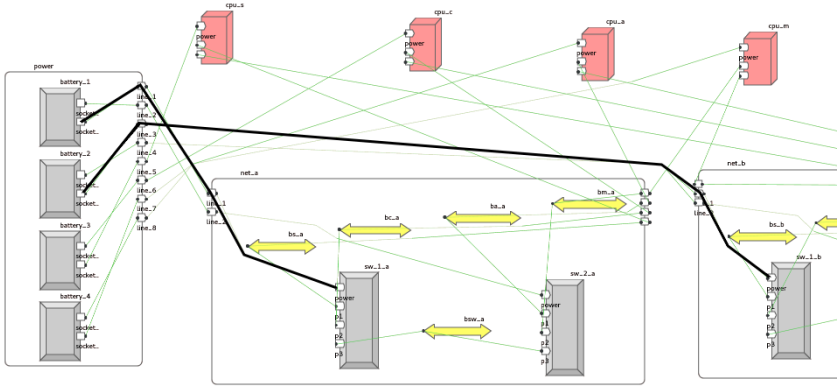

Рис. 7. Правильное подключение коммутаторов к батареям

Fig. 6. Good connecting of switches to batteries

После такой корректировки модели результаты анализа дерева неисправностей будут такими:

Функциональная опасность	**Логическое выражение (по дереву неисправностей)**	**Логическое выражение (по минимальным сечениям)**	**Вероятность**
Потеря предупреждающей сигнализации	[⩾2] ((sw_2_b.Failure ∨ battery_2.Depleted) , (sw_2_a.Failure ∨ battery_1.Depleted))	((battery_1.Depleted * battery_2.Depleted) + (battery_1.Depleted * sw_2_b.Failure) + (battery_2.Depleted * sw_2_a.Failure) + (sw_2_a.Failure * sw_2_b.Failure))	1.4822E-9
Потеря информации для управления	([⩾2] ((sw_1_a.Failure ∨ battery_1.Depleted) , (sw_1_b.Failure	((battery_1.Depleted * battery_2.Depleted) + (battery_1.Depleted * sw_1_b.Failure)	4.03216E-9

	∨ battery_2.Depl ...	+ (...	
Отказ	((cpu_s.Failure ∨ battery_3.Depleted) ∨ cpu_c.Failure ∨ (cpu_a.Failure ∨ battery_4.Depleted) ∨ cpu_m.Failure)	(battery_3.Depleted + battery_4.Depleted + cpu_a.Failure + cpu_c.Failure + cpu_m.Failure + cpu_s.Failure)	0.000127

Теперь для опасностей «Потеря предупреждающей сигнализации» и «Потеря информации для управления» вероятности имеют порядок 10^{-9}, а все минимальные сечения имеют по два элемента. Меры значимости внутренних сбоев для опасности «Потеря предупреждающей сигнализации» таковы:

ID	Description	RAW	RRW	Birnbaum	Fussel-Vesely	Diagnostic
battery_1. Depleted	Батарея разряжена	25974.95	1.54	0.0000385	0.35	0.35
battery_2. Depleted	Батарея разряжена	25974.95	1.54	0.0000385	0.35	0.35
sw_2_b. Failure	Отказ	25974.65	2.85	0.0000385	0.65	0.65
sw_2_a. Failure	Отказ	25974.65	2.85	0.0000385	0.65	0.65

Значимость сбоя «Батарея разряжена» стала сравнима со значимостью отказа коммутатора.

Результаты анализа видов и последствий отказов таковы:

Item(s)	Initial failure mode(s)	End effect	Sev	P
sw_2_a	Отказ	a.[propagation]	9	0.000125
		sw_2_a.Failed	9	0.000125
sw_2_b	Отказ	b.[propagation]	9	0.000125
		sw_2_b.Failed	9	0.000125
battery_1	Батарея разряжена	a.[propagation]	9	0.0000675
		sw_1_a.Failed	9	0.0000675
		sw_2_a.Failed	9	0.0000675
		battery_1.Failed	9	0.0000675
battery_2	Батарея разряжена	b.[propagation]	9	0.0000675
		sw_1_b.Failed	9	0.0000675
		sw_2_b.Failed	9	0.0000675
		battery_2.Failed	9	0.0000675

Теперь единичный сбой в батарее не приводит к опасному состоянию уровня всей системы.

Таким образом, используя анализ рисков, удалось выявить, локализовать и устранить ошибку проектирования, причем сделать это на этапе моделирования системы.

Еще один недостаток проектирования рассмотренной здесь системы выражается в том, что вероятность отказа в каком-либо процессе (функциональная опасность «Отказ») имеет порядок 10^{-4}, что является слишком большим значением. Этот недостаток можно устранить, например, если дублировать процессы, причем выполнять их на разных независимых процессорах. Однако, подробное рассмотрение этого сценария выходит за рамки настоящей статьи.

Следует также отметить, что поскольку для рассмотренной здесь системы не предусмотрено восстановление от ошибок, то использование марковского анализа в данном случае не дает существенной дополнительной информации по сравнению с результатами анализа дерева неисправностей.

6. Заключение

AADL и Error Model Annex позволяют систематически разрабатывать формальные модели состем и формально описывать требования по безопасности. AADL-модели со спецификациями на Error Model Annex позволяют автоматизировать многие разновидности анализа рисков, которые в частности требуются для сертификации самолетов гражданской авиации [32].

В настоящей статье мы представили алгоритмы для следующих видов анализа рисков на основе AADL и Error Model Annex: анализ дерева неисправностей, анализ видов и последствий отказов, марковский анализ.

Представленные в статье алгоритмы реализованы в инструменте MASIW. Их использование позволяет выявлять и локализовывать ошибки проектирования системы на этапе моделирования системы.

Список литературы

[1] Д.В. Буздалов, С.В. Зеленов, Е.В. Корныхин, А.К. Петренко, А.В. Страх, А.А. Угненко, А.В. Хорошилов. Инструментальные средства проектирования систем интегрированной модульной авионики. Труды ИСП РАН, том 26, вып. 1, 2014, стр. 201-230. DOI: 10.15514/ISPRAS-2014-26(1)-6

[2] Б.В. Гнеденко, Ю.К. Беляев, И.Н. Коваленко. Математические вопросы теории надежности. Итоги науки. Сер. Теор. вероятн. Мат. стат. Теор. Кибернет. 1964, 1966. 7-53.

[3] Гнеденко Б.В., Беляев Ю.К., Соловьев А.Д. Математические методы в теории надежности. М.: Наука, 1965.

[4] В.К. Дедков, А.С. Проников, А.Н. Терпиловский. Надежность сложных технических систем. Методы определения и обеспечения надежности промышленной продукции. Под ред. Г. Н. Бобровникова. М.:АНХ 1983.

[5] Карнов А.А., Зеленов С.В. Стохастические методы анализа комплексных программно-аппаратных систем. Труды ИСП РАН, том 29, вып. 4, 2017 г., стр. 191-202. DOI: 10.15514/ISPRAS-2016-29(4)-12
[6] Никольский В.И. Некоторые аварии и катастрофы отечественных и пассажирских судов. СПб.: СПГУВК, 2011.
[7] Рябинин И.А. Концепция логико-вероятностной теории безопасности.// М., «Приборы и системы управления», №10, 1993, с.6-9.
[8] Рябинин И.А. Надежность и безопасность структурно-сложных систем. Спб., Политехника, 2000.
[9] Рябинин И.А. Логико-вероятностный анализ проблем надежности живучести и безопасности. Новочеркасск, Южно-Российский государственный университет. Новочеркасск: Лик, 2009. -600с.
[10] Рябинин И.А., Черкесов Г.Н. Логико-вероятностные методы исследования надежности структурно-сложных систем.// Изд. "Радио и связь", М., 1981.
[11] Альберт Ширяев. Вероятность. 4-е издание, переработанное и дополненное. М.:МЦНМО. 2007.
[12] ГОСТ Р 27.302-2009. Надежность в технике. Анализ дерева неисправностей.
[13] K.K. Aggarwal, J.S. Gupta, and K.B. Misra. A new method for system reliability evaluation. Microelectronics Reliability, 12(5):435–440, Nov 1973.
[14] U.M. Ascher and L.R. Petzold. Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1998.
[15] E.E. Barlow, F. Proschan, and L.C. Hunter. Mathematical Theory of Reliability. Wiley, New York-London-Sydney, 1965.
[16] R.G. Bennetts. On the analysis of fault trees. IEEE Transactions on Reliability, R-24(3):175–185, Aug 1975.
[17] J. Delange, P. Feiler, D. Gluch, J. Hudak. AADL Fault Modeling and Analysis Within an ARP4761 Safety Assessment. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. CMU/SEI-2014-TR-020, 2014.
[18] Peter H. Feiler, David P. Gluch. Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language. Addison-Wesley Professional, 2012.
[19] L. Fratta and U.G. Montanari. A boolean algebra method for computing the terminal reliability in a communication network. IEEE Transactions on Circuit Theory, 20(3):203–211, 1973.
[20] J. Hadamard. Lectures on Cauchy's Problem in Linear Partial Differential Equations. Dover phoenix editions. Dover Publications, 2003.
[21] E.J. Henley and H. Kumamoto. Reliability engineering and risk assessment. Prentice-Hall, 1981.
[22] E.J. Henley and H. Kumamoto. Designing for reliability and safety control. Prentice-Hall International Series in Industrial and Systems Engineering. Prentice-Hall, 1985.
[23] Alexey Khoroshilov, Dmitry Albitskiy, Igor Koverninskiy, Mikhail Olshanskiy, Alexander Petrenko, and Alexander Ugnenko. AADL-based toolset for IMA system design and integration. SAE Int. J. Aerosp., 5:294–299, Oct 2012.
[24] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In Proc. 23rd International Conference on Computer Aided Verification (CAV11), ser. LNCS, volume 6806, pages 585–591. Springer, 2011.
[25] Nils J. Nilsson. Probabilistic logic. Artif. Intell., 28(1):71–88, February 1986.

[26] I.A. Ryabinin. Reliability of Engineering Systems. Principles and Analysis. MIR, Moscow, 1976.
[27] W. Vesely, J. Dugan, J. Fragola, Minarick, and J. Railsback. Fault Tree Handbook with Aerospace Applications. Handbook, National Aeronautics and Space Administration, Washington, DC, 2002.
[28] ARINC 664 part 7, Avionics Full Duplex Switched Ethernet (AFDX) network, 2005.
[29] MASIW: Modular Avionics System Integrator Workplace, 2016. https://forge.ispras.ru/projects/masiw-oss/.
[30] OpenFTA, 2005. http://openfta.com/.
[31] OSATE: Open Source AADL2 Tool Environment, 2016. http://osate.org/.
[32] SAE International standard ARP4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996. http://standards.sae.org/arp4761/.
[33] SAE International standard AS5506C, Architecture Analysis & Design Language (AADL), 2004. Rev. 2017, http://standards.sae.org/as5506c/.
[34] SAE International standard AS5506/1A, Architecture Analysis & Design Language (AADL), Annex E: Error Model Annex, 2011. Rev. 2015, http://standards.sae.org/as5506/1a/.

Modeling and Risk Analysis of Hardware-Software Systems

[1] *S.A. Zelenova <sophia@ispras.ru>*
[1,2] *S.V. Zelenov <zelenov@ispras.ru>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *National Research University Higher School of Economics (HSE) 20 Myasnitskaya Ulitsa, Moscow, 101000, Russia.*

Abstract. Hardware-software systems are widely used now and must be safe and reliable. Manual analysis of risks for structural complex systems is very expensive, so formal automated methods are required. The most important aspect here is the possibility to describe safety requirements in terms used in safety theory, such as Markov chains or logic-probabilistic functions, since for the decades of development of the theory, a large number of very useful results have been accumulated. Different approaches to assessing safety of systems do not compete, but complement each other, so having some universality in describing safety requirements is a very valuable quality.

In this article, we demonstrate the advisability of using the AADL modeling language and its extension Error Model Annex to describe safety requirements of a system under design.

First, we describe a mathematical model of safety requirements expressible in AADL Error Model Annex.

Next, we present algorithms to perform the following automated risk analysis on the base of AADL models: Fault Tree Analysis (including calculation of minimal cut sets and ranking of primary events with respect to different relevant importance measures), Failure Mode and Effects Analysis, and Markovian Analysis.

At last, we consider an example of a real avionic system. We present an architecture of an AADL model of the system under design and describe how to develop Error Model Annex specifications for the model. With the help of risk analysis, we show how one can identify, localize and fix a bug in the architecture of the system on the design stage of the system development.

All presented algorithms are implemented in MASIW framework for design of modern avionics systems.

Keywords: risk analysis; reliability; safety; fault tree analysis; failure mode and effects analysis; markovian analysis.

DOI: 10.15514/ISPRAS-2017-29(5)-13

For citation: Zelenova S.A., Zelenov S.V. Modeling and Risk Analysis of Hardware-Software Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 5, 2017. pp. 257-282 (in Russian). DOI: 10.15514/ISPRAS-2017-29(5)-13

References

[1] D. V. Buzdalov, S. V. Zelenov, E. V. Kornykhin, A. K. Petrenko, A. V. Strakh, A. A. Ugnenko, and A. V. Khoroshilov. Tools for system design of integrated modular avionics. Trudy ISP RAN/Proc. ISP RAS, volume 26, issue 1, pages 201–230, 2014. DOI: 10.15514/ISPRAS-2014-26(1)-6 (Russian)

[2] Gnedenko, B. V.; Beljaev, Ju. K.; Kovalenko, I. N. Mathematical problems in the theory of reliability. (Russian) 1966 Theory of Probability, Math. Statist., Theoret. Cybernet. 1964 (Russian) pp. 7–53 Akad. Nauk SSSR Inst. Naučn. Informacii, Moscow.

[3] B.V. Gnedenko, Y.K. Belyayev, and A.D. Solovyev. Mathematical methods of reliability theory. Nauka, Moscow, 1965. (Russian)

[4] V.K. Dedkov, A.S. Pronikov, A.N. Terpilovskij. Reliability of complex technical systems. Methods for determining and ensuring the reliability of industrial products. Academy of National Economy, Moscow, 1983. (Russian)

[5] Karnov A.A., Zelenov S.V. Stochastic Methods for Analysis of Complex Hardware-Software Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 4, 2017, pp. 191-202. DOI: 10.15514/ISPRAS-2016-29(4)-12

[6] Nikolskij V.I. Some accidents and disasters of domestic passenger ships. St. Petersburg State University of Water Communications, St.Petersburg, 2011. (Russian)

[7] I.A. Ryabinin. The concept of the logic-probabilistic theory of safety. Devices and control system, 10:6–9, 1993. (Russian)

[8] I.A. Ryabinin. Reliability and Safety of Structural Complex Systems. Politechnika, St.Petersburg, 2000. (Russian)

[9] I.A. Ryabinin. Logic-probabilistic Analysis of Problems of Safety, Survivability and Safety. South Russian State University, Lik, Novocherkassk, 2009. (Russian)

[10] I.A. Ryabinin and G.N. Cherkesov. The logic-probabilistic research methods of structure-complex systems reliability. Radio and communication, Moscow, 1981. (Russian)

[11] Albert Nikolaevich Shiryaev. Probability. 2nd edition, 1995.

[12] State Standard 27.302-2009. Dependability in technics. Fault tree analysis. Moscow, Standartinform Publ., 2011. (In Russian)

[13] K.K. Aggarwal, J.S. Gupta, and K.B. Misra. A new method for system reliability evaluation. Microelectronics Reliability, 12(5):435–440, Nov 1973.

[14] U.M. Ascher and L.R. Petzold. Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1998.
[15] E.E. Barlow, F. Proschan, and L.C. Hunter. Mathematical Theory of Reliability. Wiley, New York-London-Sydney, 1965.
[16] R.G. Bennetts. On the analysis of fault trees. IEEE Transactions on Reliability, R-24(3):175–185, Aug 1975.
[17] J. Delange, P. Feiler, D. Gluch, J. Hudak. AADL Fault Modeling and Analysis Within an ARP4761 Safety Assessment. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. CMU/SEI-2014-TR-020, 2014.
[18] Peter H. Feiler, David P. Gluch. Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language. Addison-Wesley Professional, 2012.
[19] L. Fratta and U.G. Montanari. A boolean algebra method for computing the terminal reliability in a communication network. IEEE Transactions on Circuit Theory, 20(3):203–211, 1973.
[20] J. Hadamard. Lectures on Cauchy's Problem in Linear Partial Differential Equations. Dover phoenix editions. Dover Publications, 2003.
[21] E.J. Henley and H. Kumamoto. Reliability engineering and risk assessment. Prentice-Hall, 1981.
[22] E.J. Henley and H. Kumamoto. Designing for reliability and safety control. Prentice-Hall International Series in Industrial and Systems Engineering. Prentice-Hall, 1985.
[23] Alexey Khoroshilov, Dmitry Albitskiy, Igor Koverninskiy, Mikhail Olshanskiy, Alexander Petrenko, and Alexander Ugnenko. AADL-based toolset for IMA system design and integration. SAE Int. J. Aerosp., 5:294–299, Oct 2012.
[24] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In Proc. 23rd International Conference on Computer Aided Verification (CAV11), ser. LNCS, volume 6806, pages 585–591. Springer, 2011.
[25] Nils J. Nilsson. Probabilistic logic. Artif. Intell., 28(1):71–88, February 1986.
[26] I.A. Ryabinin. Reliability of Engineering Systems. Principles and Analysis. MIR, Moscow, 1976.
[27] W. Vesely, J. Dugan, J. Fragola, Minarick, and J. Railsback. Fault Tree Handbook with Aerospace Applications. Handbook, National Aeronautics and Space Administration, Washington, DC, 2002.
[28] ARINC 664 part 7, Avionics Full Duplex Switched Ethernet (AFDX) network, 2005.
[29] MASIW: Modular Avionics System Integrator Workplace, 2016. https://forge.ispras.ru/projects/masiw-oss/.
[30] OpenFTA, 2005. http://openfta.com/.
[31] OSATE: Open Source AADL2 Tool Environment, 2016. http://osate.org/.
[32] SAE International standard ARP4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996. http://standards.sae.org/arp4761/.
[33] SAE International standard AS5506C, Architecture Analysis & Design Language (AADL), 2004. Rev. 2017, http://standards.sae.org/as5506c/.
[34] SAE International standard AS5506/1A, Architecture Analysis & Design Language (AADL), Annex E: Error Model Annex, 2011. Rev. 2015, http://standards.sae.org/as5506/1a/.