ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р МЭК 61508-7— 2007

ФУНКЦИОНАЛЬНАЯ БЕЗОПАСНОСТЬ СИСТЕМ ЭЛЕКТРИЧЕСКИХ, ЭЛЕКТРОННЫХ, ПРОГРАММИРУЕМЫХ ЭЛЕКТРОННЫХ, СВЯЗАННЫХ С БЕЗОПАСНОСТЬЮ

Часть 7

Методы и средства

IEC 61508-7:2000
Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures (IDT)

Издание официальное

БЗ 4—2007/65

Москва
Стандартинформ
2008

Предисловие

Цели и принципы стандартизации в Российской Федерации установлены Федеральным законом от 27 декабря 2002 г. № 184-ФЗ «О техническом регулировании», а правила применения национальных стандартов Российской Федерации — ГОСТ Р 1.0—2004 «Стандартизация в Российской Федерации. Основные положения»

Сведения о стандарте

1 ПОДГОТОВЛЕН обществом с ограниченной ответственностью «Корпоративные электронные системы» и Техническим комитетом по стандартизации ТК 10 «Перспективные производственные технологии, менеджмент и оценка рисков» на основе собственного аутентичного перевода стандарта, указанного в пункте 4

2 ВНЕСЕН Управлением развития, информационного обеспечения и аккредитации Федерального агентства по техническому регулированию и метрологии

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 27 декабря 2007 г. № 581-ст

4 Настоящий стандарт идентичен международному стандарту МЭК 61508-7:2000 «Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 7. Анализ методов и средств» (IEC 61508-7:2000 «Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures»).

Наименование настоящего стандарта изменено относительно наименования указанного международного стандарта для приведения в соответствие с ГОСТ Р 1.5—2004 (подраздел 3.5).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты, сведения о которых приведены в дополнительном приложении С

5 ВВЕДЕН ВПЕРВЫЕ

Информация об изменениях к настоящему стандарту публикуется в ежегодно издаваемом информационном указателе «Национальные стандарты», а текст изменений и поправок – в ежемесячно издаваемых информационных указателях «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ежемесячно издаваемом информационном указателе «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет

Содержание

Введение

Системы, состоящие из электрических и/или электронных компонентов, в течение многих лет используются для выполнения функций безопасности в большинстве областей применения. Компьютерные системы [обычно называемые программируемыми электронными системами (PES)], используемые во всех областях применения для выполнения задач, не связанных с безопасностью, во все возрастающих масштабах используются для решения задач обеспечения безопасности. Для эффективной и безопасной эксплуатации технологий, основанных на использовании компьютерных систем, важно, чтобы лица, ответственные за принятие решений, имели в своем распоряжении практические руководства по вопросам безопасности.

Настоящий стандарт устанавливает общий подход к вопросам обеспечения безопасности всего жизненного цикла систем, состоящих из электрических и/или электронных и/или программируемых электронных компонентов [электрических/ электронных/ программируемых электронных систем (E/E/PES)], используемых для выполнения функций безопасности. Этот унифицированный подход был принят для разработки рациональной и последовательной технической концепции для всех электрических систем, связанных с безопасностью. Основной целью настоящего стандарта является содействие разработке стандартов для их применения в различных предметных областях.

Обычно безопасность систем достигается использованием в них нескольких систем защиты, в которых используются различные (например механические, гидравлические, пневматические, электрические, электронные, программируемые электронные) технологии. Следовательно, любая стратегия безопасности должна учитывать не только элементы, входящие в состав отдельных систем (например датчики, управляющие устройства и исполнительные механизмы), но также и подсистемы, связанные с безопасностью, входящие в состав комбинированной системы, связанной с безопасностью. Таким образом, хотя настоящий стандарт в основном распространяется на электрические / электронные / программируемые электронные (E/E/PE) системы, связанные с безопасностью, он может также дать представление об общей структуре, в рамках которой рассматриваются системы, связанные с безопасностью, основанные на других технологиях.

Признанным фактом является существование огромного разнообразия применений E/E/PES в различных предметных областях, отличающихся разной степенью сложности, опасностями и возможными рисками. В каждом конкретном применении использование необходимых мер безопасности будет зависеть от многочисленных факторов, специфичных для этого конкретного применения. Настоящий стандарт, являясь базовым, позволяет формулировать такие меры для вновь разрабатываемых международных стандартов для различных предметных областей.

Настоящий стандарт:

- рассматривает все соответствующие этапы жизненного цикла систем безопасности в целом, а также подсистем E/E/PES и программного обеспечения (начиная с исходной концепции, включая проектирование, разработку, эксплуатацию, техническое обслуживание и вывод из эксплуатации), в ходе которых E/E/PES используются для выполнения функций безопасности;

- разработан с учетом быстрого развития технологий; его структура является достаточно устойчивой и полной для удовлетворения потребностей разработок, которые могут появиться в будущем;

- делает возможной разработку стандартов областей применения, в которых используются системы E/E/PES; разработка стандартов для областей применения в рамках общей структуры, вводимой настоящим стандартом, должна приводить к более высокому уровню согласованности (например основные принципы, терминология и т.п.) как для отдельных областей применения, так и для их совокупности; это дает преимущества как для безопасности, так и в сфере экономики;

- предоставляет метод разработки спецификаций для требований безопасности, необходимых для достижения требуемой функциональной безопасности E/E/PE систем, связанных с безопасностью;

- использует уровни полноты безопасности для задания планируемого уровня полноты безопасности функций, которые должны быть реализованы E/E/PE системами, связанными с безопасностью;

- использует для определения уровней полноты безопасности подход, основанный на оценке рисков;

- устанавливает количественные значения отказов E/E/PE систем, связанных с безопасностью, которые связаны с уровнями полноты безопасности;

- устанавливает нижний предел планируемых значений отказов в режиме опасных отказов, который может быть задан для отдельной E/E/PE системы, связанной с безопасностью; для E/E/PE систем, связанных с безопасностью работающих:

- в режиме с низкой интенсивностью запросов нижний предел для выполнения планируемой функции по запросу устанавливают на средней вероятности отказов 10^{-5};

- в режиме с высокой интенсивностью запросов нижний предел устанавливают на вероятности опасных отказов 10^{-9} в час.

Примечание — Конкретная E/E/PE система, связанная с безопасностью, не обязательно предполагает одноканальную архитектуру;

- применяет широкий набор принципов, методов и мер для достижения функциональной безопасности E/E/PE систем, связанных с безопасностью, но не использует концепцию безаварийности, которая может иметь важное значение в случае, если виды отказов хорошо определены, а уровень сложности является относительно невысоким. Концепция безаварийности признана неподходящей из-за широкого диапазона сложности E/E/PE систем, связанных с безопасностью и подпадающих под область применения настоящего стандарта.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФУНКЦИОНАЛЬНАЯ БЕЗОПАСНОСТЬ СИСТЕМ ЭЛЕКТРИЧЕСКИХ, ЭЛЕКТРОННЫХ, ПРОГРАММИРУЕМЫХ ЭЛЕКТРОННЫХ, СВЯЗАННЫХ С БЕЗОПАСНОСТЬЮ

Часть 7

Методы и средства

Functional safety of electrical, electronic, programmable electronic safety-related systems.
Part 7. Techniques and measures

Дата введения — 2008—09—01

1 Область применения

1.1 Настоящий стандарт содержит общее описание различных методов и средств, обеспечивающих выполнение требований МЭК 61508-2 и МЭК 61508-3.

Примечание — Ссылки должны рассматриваться как базовые ссылки на методы и инструменты либо как примеры, и они могут не отражать современное состояние области.

1.2 Стандарты МЭК 61508-1 — МЭК 61508-4 представляют собой основополагающие стандарты по безопасности, хотя этот статус не применяется в контексте E/E/PE систем, связанных безопасностью, имеющих небольшую сложность (см. МЭК 61508-4, пункт 3.4.4). Как основополагающие стандарты по безопасности они предназначены для использования техническими комитетами при подготовке стандартов в соответствии с МЭК Руководство 104 и ИСО/МЭК 51. Международный стандарт МЭК 61508 предназначен, кроме того, для использования в качестве самостоятельного стандарта.

В круг обязанностей технического комитета входит использование, где это возможно, основополагающих стандартов по безопасности при подготовке собственных стандартов. В этом случае требования, методы проверки или условия проверки настоящего основополагающего стандарта по безопасности не будут применяться, если это не указано специально, или они будут включаться в стандарты, подготовленные этими техническими комитетами.

Примечания

1 Функциональная безопасность систем E/E/PE, связанных с безопасностью, может достигаться только в том случае, когда все соответствующие требования удовлетворены. Поэтому важно, чтобы все эти требования были тщательно проанализированы и на них были даны адекватные ссылки.

2 В США и Канаде до тех пор, пока стандарты для конкретного сектора применения стандартов МЭК 61508 (например МЭК 61511 [18]) не будут опубликованы в качестве международных стандартов США и Канады, существующие там национальные стандарты по безопасности в обрабатывающих секторах, основанные на МЭК 61508, могут быть применены вместо МЭК 61508.

Издание официальное

1.3 Общая структура МЭК 61508-1 – МЭК 61508-7 с указанием роли МЭК 61508-7 в достижении функциональной безопасности E/E/PE систем, связанных с безопасностью, показана на рисунке 1.

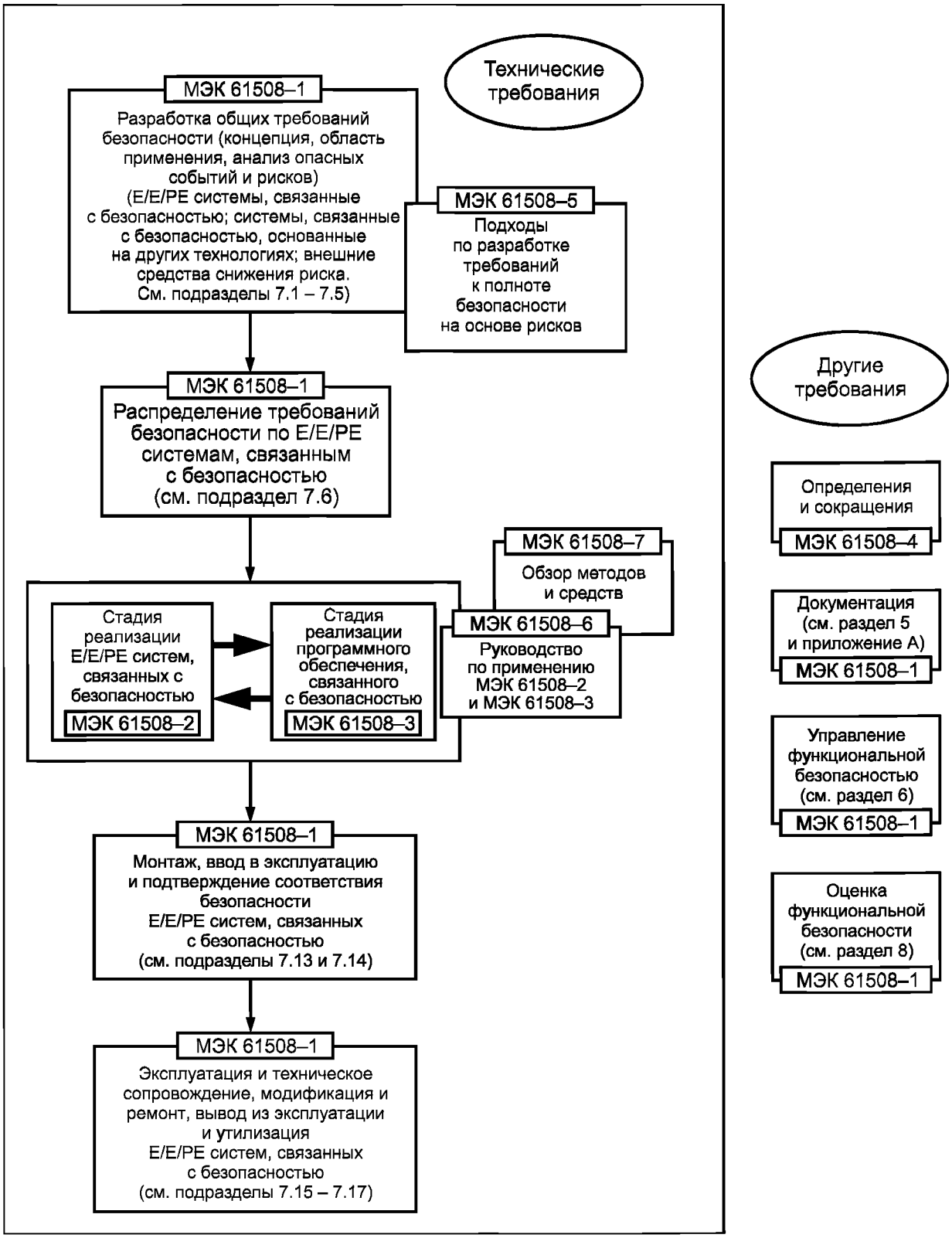

Рисунок 1 — Структура настоящего стандарта

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ИСО/МЭК Руководство 51:1999 Руководство по включению в стандарты аспектов, связанных с безопасностью

МЭК Руководство 104:1997 Руководство по подготовке стандартов, связанных с безопасностью, и по роли комитетов с функциями определения направлений и разработки стандартов в области безопасности

МЭК 61508-1:1998 Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 1. Общие требования

МЭК 61508-2:2000 Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 2. Требования к системам

МЭК 61508-3:1998 Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 3. Требования к программному обеспечению

МЭК 61508-4:1998 Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 4. Термины и определения

МЭК 61508-5:1998 Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 5. Рекомендации по применению методов определения уровней полноты безопасности

МЭК 61508-6:2000 Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 6. Руководство по применению МЭК 61508-2:2000 и МЭК 61508-3:1998

IEEE 352:1987 Руководство IEEE по основным принципам анализа надежности систем безопасности атомных энергетических станций

3 Термины и определения

В настоящем стандарте применяют термины и определения по МЭК 61508-4.

Приложение А
(справочное)

Анализ методов и средств для E/E/PES: контроль случайных отказов оборудования (см. МЭК 61508-2)

А.1 Электрические

Главная цель: Управление отказами в электромеханических компонентах.

А.1.1 Отказы, обнаруживаемые мониторингом в режиме «онлайн»

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.2, А.3, А.7 и А.14 — А.19).

Цель: обнаружение отказов путем контроля поведения E/E/PE систем, связанных с безопасностью, в процессе нормального (в режиме онлайн) функционирования управляемого оборудования (далее — EUC).

Описание: при определенных условиях отказы могут быть обнаружены с помощью информации, например о поведении во времени EUC. Например, если коммутатор, который является частью E/E/PE систем, связанных с безопасностью, нормально активизируется со стороны EUC и если при этом коммутатор не изменяет состояния в предполагаемое время, то отказ может быть обнаружен. Обычными способами невозможно локализовать такой отказ.

А.1.2 Мониторинг релейных контактов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.2 и А.15).

Цель: обнаружение отказов (например пайки) релейных контактов.

Описание: активизируемые контактные реле (или положительно управляемые контакты в реле) спроектированы так, что их контакты жестко связаны между собой. Рассмотрим два переключаемых контакта *a* и *b*. Если нормально разомкнутый контакт *a* оказался спаянным (залипшим), то нормально замкнутый контакт *b* не может замкнуться, если обмотка реле обесточивается. Следовательно, контроль замыкания нормально замкнутого контакта *b* при обесточенной обмотке реле может быть использован для подтверждения того, что нормально разомкнутый контакт *a* разомкнут. Отсутствие замыкания нормально замкнутого контакта *b* указывает на отказ контакта *a*, поэтому схема контроля должна обеспечить надежное отключение или обеспечить продолжение отключения при любом управлении оборудования контактом *a*.

Литература:

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Ver-riegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations- und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld.

Anlagensicherung mit Mitteln der MSR-Technik. G. Strohrman, Oldenburg, 1983.

А.1.3 Компаратор

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.2, А.3, А.4).

Цель: оперативное обнаружение (не одновременное) отказов в независимом модуле обработки или в компараторе.

Описание: сигналы независимых модулей обработки сравнивают циклически или непрерывно компаратором аппаратных средств. Сам компаратор может быть внешне тестируемым или же может использовать самоконтролируемую технологию. Обнаруживаемые компаратором различия в поведении процессоров независимых модулей формируют информацию для сообщений об отказах.

А.1.4 Схема голосования по мажоритарному принципу

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.2, А.3 и А.4).

Цель: обнаружение и парирование отказов, по меньшей мере, в одном из трех аппаратных каналов.

Описание: модуль голосования, использующий мажоритарный принцип (2 из 3, 3 из 4 или *m* из *n*), используется для обнаружения и парирования отказов. Сама схема голосования может внешне тестироваться или же она может использовать самоконтролирующие технологии.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Anlagensicherung mit Mitteln der MSR-Technik. Praxis der Sicherheitstechnik, Vol 1, Dechema, 1988.

Sicherung von Anlagen der Verfahrenstechnik mit Mitteln der Mess-, Steuerungs- und Regelungstechnik. VDI/VDE Blatt 1 to 5, 1984 to 1988.

А.1.5 Отсутствие питания (отключение энергии)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.2, А.9, А.14 и А.15).

Цель: выполнение функции безопасности при выключении или отсутствии питания.

Описание: функция безопасности выполняется, если контакты реле разомкнуты и ток не проходит. Например, при использовании тормозов для остановки опасного вращения двигателя тормоза отпускаются замыкающими контактами в системах, связанных с безопасностью, и включаются размыкающими контактами в системах, связанных с безопасностью.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

А.2 Электроника

Главная цель: управление отказами в транзисторных компонентах.

А.2.1 Тестирование избыточным оборудованием

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.3, А.16, А.17 и А.19).

Цель: обнаружение отказов с использованием избыточных аппаратных средств, то есть с использованием дополнительных аппаратных средств, не требующихся для реализации функций обработки.

Описание: избыточные аппаратные средства могут быть использованы для тестирования на соответствующей частоте заданных функций безопасности. Такой подход обычно требуется для реализации положений пунктов А.1.1 или А.2.2.

Литература:

DIN V VDE 0801: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems), Beuth-Verlag, Berlin, 1990.

А.2.2 Принципы динамического управления

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.3).

Цель: обнаружение статических отказов путем динамической обработки сигналов.

Описание: принудительное изменение других статических сигналов (генерируемых извне или внутри) помогает обнаруживать статические отказы в компонентах. Этот метод часто ассоциируется с электромеханическими компонентами.

Литература:

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations- und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

А.2.3 Стандартный тестовый порт доступа и архитектура граничного сканирования

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.3, А.16 и А.19).

Цель: управление и наблюдение за происходящим на каждом контакте интегральной схемы (IC).

Описание: тестирование граничного сканирования представляет собой метод построения IC, который повышает тестируемость IC, разрешая проблему доступа к внутренним точкам тестируемой схемы. В типичной сканируемой по границам интегральной схеме, содержащей внутренние логические схемы, а также входные и выходные буферы, между внутренними логическими схемами и входными/выходными буферами, соединенными с внешними контактами IC, размещается ступень сдвигового регистра. Содержимое каждого сдвигового регистра находится в ячейке граничного сканирования. Ячейка граничного сканирования может управлять и наблюдать за происходящим на каждом входном и выходном контакте IC через стандартный тестовый порт доступа. Тестирование внутренних логических схем IC проводится путем отключения размещенных на чипе внутренних логических схем от входных сигналов, получаемых от окружающих компонентов, и последующего выполнения внутреннего тестирования. Эти тесты могут быть использованы для обнаружения отказов в IC.

Литература:

IEEE 1149.1:1990, Standard Test Access Port and Boundary-Scan Architecture. IEEE 1149.1:1990.

А.2.4 Отказоустойчивое оборудование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.3).

Цель: перевести систему в безопасное состояние в случае появления отказов.

Описание: в аппаратно реализованных системах считается, что устройство работает отказоустойчиво, если:

- конкретный набор отказов может приводить к безопасному состоянию и
- отказы обнаруживаются.

Пример — К конкретному набору отказов могут относиться постоянные отказы, постоянные отказы типа «разомкнутый», короткие замыкания внутри и между компонентами и направленные короткие замыкания.

Литература:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations-und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

А.2.5 Избыточный контроль

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.3).

Цель: обнаружение отказов путем создания нескольких функциональных модулей, контроля поведения каждого из них для обнаружения отказов и последующего инициирования перехода в безопасное состояние при обнаружении какого-либо несоответствия в поведении.

Описание: функция безопасности выполняется, по меньшей мере, двумя аппаратными каналами. Выходы этих каналов контролируются и безопасное состояние инициируется при обнаружении отказа (в случае, если выходные сигналы из всех каналов не идентичны).

Литература:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations-und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

А.2.6 Электрические/электронные компоненты с автоматической проверкой

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.3).

Цель: обнаружение отказов путем периодической проверки способности выполнения функции безопасности.

Описание: аппаратные средства тестируются до запуска процесса и затем тестируются повторно через соответствующие интервалы. EUC продолжает работу только при условии успешного прохождения каждого теста.

Литература:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations-und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993.

А.2.7 Текущий контроль аналоговых сигналов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.3 и А.14).

Цель: повышение достоверности результатов измерений аналоговых сигналов.

Описание: везде, где возможно, используются аналоговые приборы, а не цифровые. В аналоговых приборах отключение или безопасные состояния представляются уровнями аналоговых сигналов обычно с контролем устойчивости уровня этого сигнала. Это обеспечивает непрерывный контроль и высокую степень достоверности передачи сигнала, снижает частоту необходимого гарантийного тестирования функции чувствительности передатчика. Внешние интерфейсы, например импульсные линии, также нуждаются в тестировании.

Литература:

UKOOA Guidelines for Instrument-Based Systems, UK Offshore Operators Association Limited, December 1995.

А.2.8 Снижение номинального значения

Цель: повышение надежности компонентов аппаратных средств.

Описание: компоненты аппаратных средств нормально выполняют свои функции на уровнях напряжений, которые заложены при проектировании системы и являются более низкими, чем максимально специфицированные номинальные значения. Снижение номинального значения — это практика, обеспечивающая, что при всех нормальных условиях эксплуатации компоненты будут нормально функционировать при уровнях напряжений ниже их максимальных значений.

А.3 Модули обработки

Главная цель: распознавать отказы, которые приводят к неправильным результатам в модулях обработки.

А.3.1 Программное самотестирование: предельное количество комбинаций (одноканальное)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.4).

Цель: оперативное обнаружение отказов в модулях обработки.

Описание: аппаратные средства создаются с использованием стандартных методов, не учитывающих специальных требований к безопасности. Обнаружение отказов реализуется целиком дополнительными программными функциями, которые выполняют самотестирование с использованием, по меньшей мере, двух дополнительных комбинаций данных (например 55hex и AAhex).

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.3.2 Программное самотестирование: блуждающий бит (одноканальное)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.4).

Цель: оперативное обнаружение отказов в физической памяти (например, в регистрах) и дешифраторе команд процессора.

Описание: обнаружение отказов полностью реализуется дополнительными программными функциями, которые выполняют самотестирование с использованием комбинации данных (например комбинации блуждающих битов), которая тестирует физическую память (регистры данных и адресные регистры) и дешифратор команд. Однако диагностический охват составляет только 90 %.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.3.3 Самотестирование, обеспечиваемое оборудованием (одноканальное)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.4).

Цель: оперативное обнаружение отказов в процессоре с использованием специальных аппаратных средств, которые увеличивают скорость и расширяют область обнаружения отказов.

Описание: дополнительные специальные аппаратные средства обеспечивают функции самотестирования для обнаружения отказов. Например, таким средством может быть аппаратный модуль, который циклически контролирует выход на наличие конкретной битовой комбинации, используя механизм сторожевой схемы.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.3.4 Закодированная обработка (одноканальная)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.4).

Цель: оперативное обнаружение отказов в процессоре.

Описание: процессоры могут быть спроектированы с встроенными специальными функциями распознавания или исправления отказов. До сих пор эти функции применялись только в относительно простых схемах и не получили широкого распространения. Однако такие функции не должны исключаться в будущих разработках.

Литература:

The Coded Microprocessor Certification. P. Ozello, Proc. SAFECOMP '92, 185-190, 1992.

Vital Coded Microprocessor Principles и Application for Various Transit Systems. P. Forin, IFAC Control Computers Communications in Transportation, 79-84, 1989.

Le Processeur Codé: un nouveau concept appliqué a la sécurité des systèmes de transports. Gabriel, Martin, Wartski, Revue Générale des chemins de fer, No. 6, June 1990.

А.3.5 Программное обнаружение несовпадений

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.4).

Цель: оперативное обнаружение отказов в процессоре путем динамического программного сравнения.

Описание: два модуля взаимно обмениваются данными (включая результаты, промежуточные результаты и тестируемые данные). Если при сравнении данных, выдаваемых с использованием программных средств в каждом модуле, обнаруживаются различия, то это приводит к выдаче сообщений об отказе.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.4 Постоянная память

Главная цель: выявление модификаций информации в постоянной памяти.

А.4.1 Сохранение слов с многобитовой избыточностью (например контроль ROM с модифицированным кодом Хэмминга)

Примечание — См. также А.5.6 и С.3.2. Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.5).

Цель: обнаружение всех однобитовых ошибок, всех двухбитовых ошибок и некоторых ошибок во всех битах в 16-битовом слове.

Описание: каждое слово в памяти расширяется несколькими избыточными битами для формирования модифицированного кода Хэмминга с расстоянием, равным 4 (по меньшей мере). При каждом считывании слова проверка избыточных битов может указывать, произошло или нет искажение. При обнаружении различия вырабатывается сообщение об ошибке. Эта процедура может также использоваться для обнаружения ошибок адресации путем вычисления избыточных битов для объединения слова данных с его адресом.

Литература:

Error detecting and error correcting codes. R. W. Hemming, The Bell System Technical Journal 29 (2), 147-160, 1950.

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967.

А.4.2 Модифицируемая контрольная сумма

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.5).

Цель: обнаружение всех ошибок нечетных битов, то есть приблизительно 50 % всех возможных битовых ошибок.

Описание: контрольная сумма блока памяти образуется соответствующим алгоритмом, который обрабатывает все слова в блоке памяти. Эта контрольная сумма может храниться как дополнительные слово в ROM, либо может быть добавлена как дополнительное слово в блок памяти для того, чтобы алгоритм контрольной суммы выработал заранее заданное значение. В последнем тестировании памяти контрольная сумма создается снова с использованием того же алгоритма, и результат сравнивается с запомненным или заданным значением. При обнаружении различий вырабатывается сообщение об ошибке.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.4.3 Сигнатура одного слова (8 бит)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.5).

Цель: обнаружение всех однобитовых ошибок и всех многобитовых ошибок в слове при достижении приблизительно 99, 6 % всех возможных битовых ошибок.

Описание: содержимое блока памяти сжимается (с использованием аппаратных или программных средств) в одно слово памяти с использованием алгоритма контроля с помощью избыточного циклического кода (CRC). Типичный алгоритм CRC рассматривает все содержимое блока памяти как побайтовый или побитовый последовательный поток данных, в котором выполняется непрерывное полиномиальное деление с использованием полиномиального генератора. Остаток от деления сохраняется и представляет собой сжатое содержимое памяти — «сигнатуру» памяти. Сигнатура вычисляется еще раз в последующем тестировании и сравнивается с уже запомненным значением. При обнаружении различий выдается сообщение об ошибке.

Литература:

Calculating an error checking character in software. S. Vasa, Computer Design, 5, 1976.

Berechnung von Fehlererkennungswahrscheinlichkeiten bei Signaturregistern. D. Leisengang, Elektronische Rechenanlagen 24, H. 2, S. 55-61, 1982.

А.4.4 Сигнатура двойного слова (16 бит)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.5).

Цель: обнаружение всех однобитовых ошибок и всех многобитовых ошибок в слове составляет примерно 99, 998 % всех возможных битовых ошибок.

Описание: данная процедура вычисляет сигнатуру с использованием алгоритма контроля с помощью CRC, однако длина результирующего значения составляет, по меньшей мере, два слова. Расширенная сигнатура заносится в память, повторно вычисляется и сравнивается как одно слово. При обнаружении различий между сохраненной и повторно вычисленной сигнатурами выдается сообщение об ошибке.

Литература:

Signaturanalyse in der Datenverarbeitung. D. Leisengang, M. Wagner, Elektronik 32, H. 21, S. 67-72, 1983.

Signaturregister für selbsttestende ICs. B. Könemann, J. Mucha, G. Zwiehoff, Groβtintegration/ NTG-Fachtagung Baden-Baden, S. 109-112, April 1977.

А.4.5 Повторение блока (например дублирование ROM в аппаратном и программном исполнении)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.5).

Цель: обнаружение всех битовых ошибок.

Описание: адресное пространство дублируется в двух областях памяти. Первая область памяти работает в нормальном режиме. Вторая — содержит ту же информацию и доступна параллельно с первой. Их выходы сравниваются, и при обнаружении различий выдается сообщение об ошибке. Для обнаружения некоторых видов битовых ошибок данные должны запоминаться инверсно в одной из двух областей памяти и инвертироваться обратно при чтении.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Computers can now perform vital safety functions safely. Otto Berg von Linde, Railway Gazette International, Vol. 135, No. 11, 1979.

А.5 Память с произвольным доступом

Главная цель: обнаружение отказов во время адресации, записи, запоминания и считывания.

А.5.1 Тесты «шахматная доска» и «марш» для памяти с произвольным доступом

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.6).

Цель: обнаружение преимущественно статических битовых ошибок.

Описание: расположенная в шахматном порядке битовая комбинация нулей и единиц записывается в ячейки памяти с битовой организацией. Затем эти ячейки анализируются попарно с тем, чтобы убедиться в их одинаковости и правильности. Адрес первой ячейки такой пары является переменным, а адрес второй ячейки этой пары образуется путем битового инвертирования первого адреса. При первом прохождении диапазон адресов памяти проходится в направлении более высоких адресов переменных адресов, а при втором прохождении — в направлении более низких адресов. После этого оба прохождения повторяются с заранее заданным инвертированием. При обнаружении какого-либо различия выдается сообщение об отказе.

При «маршевом» тестировании памяти с произвольным доступом ячейки памяти с битовой организацией инициализируются унифицированным потоком битов. При первом прохождении ячейки анализируются в нисходящей последовательности; проверяется правильность содержимого каждой ячейки и ее содержимое инвертируется. Основа, созданная в первом прохождении, рассматривается при втором прохождении в убывающем порядке и так же обрабатывается. Первые прохождения повторяются с инвертируемыми предварительными значениями в третьем и четвертом прохождениях. При обнаружении различий выдается сообщение об отказе.

Литература:

Memory testing.. W. G. Fee, LSI Testing (Tutorial at COMPCON 77 in San Francisco), IEEE Computer Society, W. G. Fee (ed.), 81-88, 1978.

Memory testing. P. Rosenfield, Electronics and Power, H. 1, P. 26-31, 1979.

Halbleiterspeicher-Testfolgen. Th. John, E. Schaefer, Elektronikpraxis, H. 6, 18-26 and H. 7, 10-14, 1980.

А.5.2 Тест «прогулочная дорожка» для памяти с произвольным доступом

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.6).

Цель: обнаружение статических и динамических ошибочных битов и перекрестных помех между ячейками памяти.

Описание: тестируемая область памяти инициализируется унифицированным потоком битов. Затем первая ячейка инвертируется и остальная часть памяти анализируется на правильность. После этого первая ячейка повторно инвертируется для возврата в исходное значение, и вся процедура повторяется для следующей ячейки. Второе прохождение «модели блуждающего бита» осуществляется при инверсии всех первоначально назначенных значений памяти. При обнаружении различий выдается сообщение об ошибке.

Литература:

Memory testing W. G. Fee, LSI Testing (Tutorial at COMPCON 77 in San Francisco), IEEE Computer Society, W. G. Fee (ed.), 81-88, 1978.

Techniques for testing the microprocessor family. W. Barraclough, A. Chiang, W. Sohl, Proceedings of IEEE 64 (6), 943-950, 1976.

А.5.3 Тест «бегущий код» для памяти с произвольным доступом

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.6).

Цель: обнаружение статических битовых ошибок и больших пропорций динамических связей.

Описание: при тестировании памяти с произвольным доступом «попарной записью-считыванием» выбранная область памяти сначала инициализируется унифицированно (то есть все 0 или все 1). После этого первая ячейка памяти тестируется и затем инвертируется, и все остальные ячейки анализируются на правильность содержимого. После каждого доступа по чтению к одной из оставшихся ячеек инвертированная ячейка также про-

веряется. Эта процедура повторяется для каждой ячейки в выбранной области памяти. Второе прохождение выполняется противоположно первому. Любые различия приводят к выдаче сообщения об ошибке.

Тестирование «прозрачной попарной записью-считыванием» представляет собой вариацию описанной выше процедуры: вместо инициализации всех ячеек в выбранной области памяти существующее содержимое остается неизменным, а для сравнения содержимого набора ячеек используются контрольные суммы (сигнатуры). Выбирается первая тестируемая ячейка области памяти и вычисляется и сохраняется сигнатура *S*1 всех оставшихся ячеек области. Затем тестируемые ячейки инвертируются, и повторно вычисляется сигнатура *S*2. (После каждого доступа по чтению к одной из оставшихся ячеек инвертируемая ячейка также проверяется). Сигнатура *S*2 сравнивается с сигнатурой *S*1 и при любом различии выдается сообщение об ошибке. Тестируемая ячейка повторно инвертируется для повторного установления исходного содержимого и сигнатура *S*3 всех оставшихся ячеек повторно вычисляется и сравнивается с сигнатурой *S*1. Любые различия приводят к выдаче сообщения об ошибке. Все ячейки памяти в выбранной области тестируются тем же способом.

Литература:

Entwurf von Selbsttestprogrammen für Mikrocomputer. E. Maehle, Microcomputing. Berichte der Tagung III/79 des German Chapter of ACM, W. Remmele, H. Schecher, (ed.), Stuttgart, Teubner, 204-216, 1979.

Periodischer Selbsttest einer mikroprocessorgesteuerten Sicherheitsschaltung. U. Stinnesbek, Diplomarbeit am Institut für theoretische Elektrotechnik der RWTH Aachen 1980.

А.5.4 Тест «Абрахам» для памяти с произвольным доступом

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.6).

Цель: обнаружение всех постоянных отказов и отказов в соединениях между ячейками памяти.

Описание: диагностический охват выше, чем при тесте «попарная запись-считывание». Число операций, необходимых для выполнения всего тестирования памяти, составляет примерно 30 *n*, где *n* - число ячеек памяти. Тестирование может быть «прозрачным» при выполнении запоминания и тестирования в различных временных сегментах в периоде рабочего цикла.

Литература:

Efficient Algorithms for Testing Semiconductor Random-Access Memories. R. Nair, S. M. Thatte, J. A. Abraham, IEEE Trans. Comput. C-27 (6), 572-576, 1978.

А.5.5 Однобитовая избыточность (например контроль памяти с произвольным доступом с помощью бита четности)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.6).

Цель: обнаружение 50 % всех возможных битовых отказов в тестируемой области памяти.

Описание: каждое слово в памяти расширяется на один бит (бит четности), который дополняет каждое слово до четного или нечетного числа логических единиц. Четность слова данных проверяется при каждом чтении. При обнаружении ложного числа единиц выдается сообщение об ошибке. Выбор четности или нечетности должен осуществляться так, чтобы всякий раз в случае отказа не выдавалось ничего кроме нулевого (0) и единичного (1) слова, вырабатывалось уведомление о том, что это слово неправильно закодировано. Контроль четности также может быть использован для обнаружения ошибок адресации, если четность определяется для объединения слова данных с его адресом.

Литература:

Integrierte Digitalbausteine. K. Reiβ, H. Liedl, W. Spichall, Berlin, 1970.

А.5.6 Контроль памяти с произвольным доступом с помощью модифицированного кода Хэмминга или обнаружение ошибок данных с кодами обнаружения и исправления ошибок (EDC)

Примечание — См. также А.4.1 и С.3.2. Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.6).

Цель: обнаружение всех нечетных битовых отказов, всех двухбитовых отказов, некоторых трехбитовых отказов и некоторых многобитовых отказов.

Описание: каждое слово в памяти расширяется несколькими избыточными битами для выработки модифицированного кода Хэмминга с расстоянием Хэмминга равным, по меньшей мере, 4. При каждом считывании слова проверка избыточных битов может указывать, произошло или нет искажение. При обнаружении различий выдается сообщение об отказе. Эта процедура может быть также использована для обнаружения ошибок адресации при вычислении избыточных битов для объединения слова данных с его адресом.

Литература:

Error detecting и error correcting codes. R. W. Hamming, The Bell System Technical Journal 29 (2), 147-160, 1950.

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967.

А.5.7 Дублирование со сравнением памяти с произвольным доступом с аппаратными или программными средствами и тестирование чтением/записью

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.6).

Цель: обнаружение всех битовых отказов.

Описание: адресное пространство содержит две части. Первая часть памяти функционирует в нормальном режиме. Вторая часть памяти содержит ту же информацию и доступна параллельно с первой. Выходы этих частей памяти сравниваются. При обнаружении различий выдается сообщение об ошибке. Для обнаружения некоторых видов битовых ошибок данные должны сохраняться инверсно в одной из двух частей памяти и обратно инвертироваться при чтении.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Computers can now perform vital function safely. Otto Berg von Linde, Railway Gazette International, Vol. 135, No. 11, 1979.

А.6 Устройства ввода-вывода и интерфейсы (внешний обмен)

Главная цель: обнаружение отказов на устройствах ввода и вывода (цифровые, аналоговые, последовательные или параллельные) и предотвращение дальнейшей передачи недопустимых выходных данных.

А.6.1 Тестирующая комбинация

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.7, А.14 и А.15).

Цель: обнаружение статических отказов (постоянные отказы) и перекрестных помех.

Описание: этот метод реализует независимое от потока данных циклическое тестирование входных и выходных элементов. В нем используются определенные тестирующие комбинации для сравнения с соответствующими этим тестирующим комбинациям предполагаемыми значениями. Информация тестирующей комбинации, восприятие и оценка тестирующей комбинации должны быть независимы друг от друга. Тестирующие комбинации не должны неблагоприятно влиять на операции EUC.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.6.2 Кодовая защита

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.7, А.16, А.17 и А.19).

Цель: обнаружение случайных отказов аппаратных средств и систематических ошибок в потоке ввода/вывода данных.

Описание: процедура, реализующая кодовую защиту, защищает вводимую и выводимую информацию от систематических и случайных отказов аппаратных средств. Кодовая защита обеспечивает зависимое от потока данных обнаружение отказов входных и выходных модулей, основываясь на избыточности информации и/или временной избыточности. Обычно избыточная информация налагается на входные и/или выходные данные; тем самым обеспечиваются средства для мониторинга правильности операций входных и выходных схем. Возможно применение многих методов — например, сигнал несущей частоты может налагаться на выходной сигнал датчика. После этого логический модуль может проверить наличие частоты несущей, либо на выходе канала могут быть добавлены избыточные кодовые биты для контроля действительности прохождения сигнала между логическим модулем и оконечным исполнительным механизмом.

Литература:

Standard input/output tests and monitoring procedures — Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.6.3 Многоканальное параллельное выходное устройство

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.7).

Цель: обнаружение случайных отказов аппаратных средств (константные отказы), отказов, обусловленных внешними воздействиями, временных сбоев, отказов адресации, постепенных отказов и самоустраняющихся отказов.

Описание: это зависимое от потока данных многоканальное параллельное выходное устройство с независимыми выходами для обнаружения случайных аппаратных отказов. Обнаружение отказов осуществляется с помощью внешних компараторов. При появлении отказа EUC непосредственно отключается. Это устройство действует только в том случае, если поток данных изменяется в интервале диагностического тестирования.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.6.4 Средство контроля выходов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.7).

Цель: обнаружение случайных отказов, отказов, обусловленных внешними воздействиями, временных сбоев, отказов адресации, постепенных отказов (для аналоговых сигналов) и самоустраняющихся отказов.

Описание: это устройство, зависимое от потока данных, сравнивает выходные данные с независимыми входными данными для обеспечения совместимости с областью допустимых значений (время, диапазон). Обнаруженный отказ не всегда относится к неправильному выходному сигналу. Это устройство действует только в том случае, если поток данных изменяется в интервале диагностического тестирования.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

MSR-Schutzeinrichtungen. Anforderungen und Massnahmen zur gesicherten Funktion. DIN V 19251, February 1995.

А.6.5 Сравнение/голосование входных данных

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.7 и А.14).

Цель: обнаружение случайных отказов, отказов, обусловленных внешними воздействиями, временных сбоев, отказов адресации, постепенных отказов (для аналоговых сигналов) и самоустраняющихся отказов.

Описание: это устройство, зависимое от потока данных, сравнивает выходные данные с независимыми входными данными для обеспечения совместимости с областью допустимых значений (время, диапазон). Реализуемая избыточность может быть 1 из 2, 2 из 3 или более высокая. Это устройство действует только в том случае, если поток данных изменяется в интервале диагностического тестирования.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

MSR-Schutzeinrichtungen. Anforderungen und Massnahmen zur gesicherten Funktion. DIN V 19251, February 1995.

А.7. Маршруты данных (внутренний обмен)

Главная цель: обнаружение отказов, обусловленных искажениями при передаче информации.

А.7.1 Однобитовая аппаратная избыточность

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.8).

Цель: обнаружение всех нечетно-битовых ошибок, то есть 50 % всех возможных битовых ошибок в потоке данных.

Описание: шина расширяется на одну линию (бит) и эта дополнительная линия (бит) используется для обнаружения отказов путем проверки на четность.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.7.2 Многобитовая аппаратная избыточность

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.8).

Цель: обнаружение отказов в процессе передачи по шине и в последовательных каналах связи.

Описание: шина расширяется на две или более линий (битов) и эти дополнительные линии (биты) используются для обнаружения отказов методом кода Хэмминга.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.7.3 Полная аппаратная избыточность

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.8).

Цель: обнаружение отказов в процессе передачи данных путем сравнения сигналов двух шин.

Описание: шина дублируется и дополнительные линии (биты) используются для обнаружения отказов.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.7.4 Анализ с использованием тестирующих комбинаций

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.8).

Цель: обнаружение статических отказов (постоянных отказов) и перекрестных помех.

Описание: осуществляется независимое от потока данных циклическое тестирование маршрутов данных. Используется определенная тестирующая комбинация для сравнения наблюдаемых значений с соответствующими предполагаемыми значениями.

Восприятие информации о тестирующей комбинации и ее оценка должны быть независимы друг от друга. Тестирующие комбинации не должны неблагоприятно влиять на операции EUC.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.7.5 Избыточность при передаче

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.8).

Цель: обнаружение самоустраняющихся отказов в обмене по шине.

Описание: информация передается последовательно несколько раз. Повторение осуществляется только для обнаружения самоустраняющихся отказов.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.7.6 Информационная избыточность

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.8).

Цель: обнаружение отказов в обмене по шине.

Описание: данные передаются блоками наряду с вычислениями контрольной суммы для каждого блока. После этого приемник повторно вычисляет контрольную сумму полученных данных. Результат сравнивается с полученной контрольной суммой.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.8 Питание

Главная цель: обнаружение или устойчивость к отказам, обусловленным источником питания.

А.8.1 Защита от броска напряжения с помощью безопасного выключения

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.9).

Цель: защита систем, связанных с безопасностью, от броска напряжения.

Описание: бросок напряжения обнаруживается достаточно рано с тем, чтобы все выходы могли быть переключены в безопасное состояние процедурой отключения питания или переключились на второй блок питания.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

А.8.2 Управление напряжением (вторичным)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.9).

Цель: контроль вторичных напряжений и инициализация безопасного состояния, если значение напряжения не находится в заданном диапазоне.

Описание: вторичное напряжение контролируется и питание отключается либо происходит переключение на второй блок питания, если не напряжение находится в заданном диапазоне.

А.8.3 Выключение питания с соблюдением безопасности

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.9).

Цель: выключение питания с безопасным сохранением имеющейся информации.

Описание: бросок напряжения или недонапряжение обнаруживается достаточно рано с тем, чтобы сохранить внутреннее состояние в неэнергозависимой памяти (при необходимости) и тем самым установить все выходы в безопасное состояние процедурой отключения питания, или же все выходы переключить в безопасное состояние процедурой отключения питания, либо происходит переключение на второй блок питания.

А.9 Временной и логический контроль последовательности выполнения программ

Примечание — На эту группу методов или средств даны ссылки в МЭК 61508-2 (см. приложение А, таблицы А.16, А.17 и А.19).

Главная цель: обнаружение искаженных программных последовательностей. Искаженная программная последовательность появляется, если отдельные элементы программы (например программные модули, подпрограммы или команды) обрабатываются в неправильной последовательности или в несоответствующий период времени, или если сбилась тактовая частота процессора.

А.9.1 Контрольный датчик времени с отдельной временной базой без временного окна

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.10 и А.12).

Цель: контроль поведения и последовательности выполнения программ.

Описание: внешние средства определения времени с отдельной базой времени (например контрольный датчик времени) периодически переключаются для контроля поведения компьютера и последовательности выполнения программ. Важно, чтобы моменты переключения были правильно расположены в программе. Контрольный датчик времени не переключается с некоторым фиксированным периодом, однако задается максимальный интервал.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.9.2 Контрольный датчик времени с отдельной временной базой и временным окном

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.10 и А.12).

Цель: контроль поведения и последовательности выполнения программ.

Описание: внешние средства определения времени с отдельной базой времени (например контрольный датчик времени) периодически переключаются для контроля поведения компьютера и последовательности выполнения программ. Важно, чтобы моменты переключения были правильно расположены в программе. Контрольный датчик времени не переключается с некоторым фиксированным периодом, однако задается максимальный интервал. Если последовательности программ выполняются больше или меньше ожидаемого времени, то выполняется действие чрезвычайного случая.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.9.3 Логический контроль последовательности выполнения программ

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.10 и А.12).

Цель: контроль правильной последовательности выполнения отдельных частей программы.

Описание: правильная последовательность выполнения отдельных частей программы контролируется с помощью программных средств (процедур учета, ключевых процедур) или с использованием внешних средств контроля. Важно, чтобы точки проверки располагались в программе правильно.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.9.4 Комбинация временного и логического контроля программной последовательности

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.10 и А.12).

Цель: контроль поведения и правильной последовательности выполнения отдельных частей программы.

Описание: средство определения времени (например контрольный датчик времени), контролирующее программную последовательность, вновь запускается только в случае, если последовательность модулей программы выполнятся правильно.

Литература:

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

А.9.5 Первоначальный тест при включении

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.10 и А.12).

Цель: обнаружение отказов при первоначальном тесте.

Описание: при запуске проводится первоначальный тест. Запуск возможен только в случае, если первоначальный тест прошел успешно. Например, датчик температуры может быть проверен нагретым резистором при запуске.

А.10 Вентиляция и температура

Примечание — На эту группу методов и средств дана ссылка в МЭК 61508-2 (см. приложение А, таблицы А.17 и А.19).

Главная цель: управление отказами в системах вентиляции и температурных приборах и/или их контроль, если они связаны с безопасностью.

А.10.1 Датчик температуры

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.11).

Цель: обнаружение температурного перегрева или недогрева до того, как система начнет действовать вне заданных требований.

Описание: датчик температуры контролирует температуру в наиболее критических точках E/E/PES. Прежде чем температура выйдет из заданного диапазона, происходит аварийное действие.

А.10.2 Управление вентиляцией

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.11).

Цель: обнаружение отказов в работе вентилятора.

Описание: работа вентиляторов контролируется. Если вентилятор находится в нерабочем состоянии, то предпринимаются действия по восстановлению его рабочего состояния (или его аварийному отключению).

А.10.3 Безопасное выключение с использованием плавкого предохранителя

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.11).

Цель: выключение системы, связанной с безопасностью, до того как параметры системы выйдут из заданных температурных режимов.

Описание: плавкий предохранитель используется для выключения систем, связанных с безопасностью. В системе PES выключение осуществляется процедурой отключения питания, которая хранит информацию, необходимую при аварийных действиях.

А.10.4 Пороговые сообщения от термодатчиков и условная тревога

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.11).

Цель: показать, что системы, связанные с безопасностью, работают также за пределами допусков по температуре.

Описание: измеряется температура, а при ее выходе из заданного диапазона выдается аварийный сигнал.

А.10.5 Соединение устройства принудительного охлаждения воздуха и индикатора состояний

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.11).

Цель: не допустить перегрева путем искусственного воздушного охлаждения.

Описание: измеряется температура. Если температура превышает заданный предел, то включается искусственное воздушное охлаждение. Пользователь информируется об измеренном значении температуры.

А.11 Обмен и запоминающее устройство большой емкости

Главная цель: контроль отказов в процессе обмена между внешними источниками и запоминающим устройством большой емкости.

А.11.1 Отделение линий электрического питания от информационных линий

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.13).

Цель: минимизировать перекрестные помехи информационных линий, индуцируемые сильным током системы питания.

Описание: линии, обеспечивающие электрическое питание, отделяются от линий, переносящих информацию. Электрическое поле, которое может индуцировать на информационных линиях всплески напряжения, уменьшается с увеличением расстояния.

А.11.2 Пространственное разделение групповых линий

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.13 и А.17).

Цель: минимизировать перекрестные помехи, индуцируемые током системы питания в групповых линиях.

Описание: линии с дублирующими сигналами отделяются друг от друга. Электрическое поле, которое могут индуцировать броски напряжений в групповых линиях, уменьшается с увеличением расстояния. Это отделение линий снижает также отказы по общей причине.

А.11.3 Повышение устойчивости к электромагнитным воздействиям

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.13, А.17 и А.19).

Цель: минимизировать электромагнитные влияния на систему, связанную с безопасностью.

Описание: создание таких методов, как экранирование и фильтрация, для уменьшения чувствительности систем, связанных с безопасностью, к электромагнитным полям, которые могут наводиться на линии питания или сигнальные линии, либо возникать в результате электростатических разрядов [7].

Литература:

Noise Reduction Techniques in Electronic Systems. H. W. Ott, John Wiley Interscience, 2nd Edition, 1988.

EMC for Product Designers. Tim Williams, Newnes, 1992, ISBN 0-7506-1264-9.

Grounding and Shielding Techniques in Instrumentation. John Wiley & Sons, New York, 1986.

Principles and Techniques of Electromagnetic Compatibility. C Christopoulos, CRC Press, 1995.

Gestaltung von Maschinensteuerungen unter Berücksichtigung der elektromagnetischen Verträglichkeit. F. Börner, Sicherheitstechnisches Informations-und Arbeitsblatt 330260, BIA-Handbuch. 20. Lfg. V/93, Erich Schmidt Verlag, Bielefeld.

А.11.4 Передача неэквивалентных сигналов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицы А.13 и А.17).

Цель: обнаружение одинаковых индуцированных напряжений в групповых линиях передачи сигналов.

Описание: вся дублируемая информация передается с антивалентными сигналами (например логические 1 и 0). Ошибки по общей причине (например, вызванные электромагнитными излучениями) могут быть обнаружены антивалентным компаратором.

Литература:

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations-und Arbeitsblatt 330220, BIA-Handbuch. 20. Lfg. V/93, Erich Schmidt Verlag, Bielefeld.

А. 12 Датчики

Главная цель: обнаружение отказов в датчиках систем, связанных с безопасностью.

А.12.1 Эталонный датчик

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.14).

Цель: обнаружение отказа датчика.

Описание: для контроля работоспособности датчика используется независимый эталонный датчик. Все входные сигналы в подходящие временные интервалы проверяются эталонным датчиком для обнаружения отказов в работе проверяемого датчика.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

А.12.2 Положительно-управляемый переключатель

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.14).

Цель: разомкнуть контакт с помощью непосредственного механического соединения между кулачком переключателя и контактом.

Описание: положительно управляемый переключатель размыкает свои обычно замкнутые контакты непосредственным механическим соединением между кулачком переключателя и контактом. Разомкнутость контактов переключателя обеспечивается всякий раз, когда кулачок переключателя находится в рабочем положении.

Литература:

Verriegelung beweglicher Schutzeinrichtungen. F. Kreutzkampf, K. Becker, Sicherheitstechnisches Informations-und Arbeitsblatt 330210, BIA-Handbuch. 1. Lfg. IX/85, Erich Schmidt Verlag, Bielefeld.

А.13 Оконечные элементы (приводы)

Главная цель: обнаружение отказов в оконечных элементах систем, связанных с безопасностью.

А.13.1 Мониторинг

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.15).

Цель: обнаружение отказа привода.

Описание: операции привода контролируются (например положительно управляемыми контактами реле; см. контроль релейных контактов в А.1.2). Избыточность, вносимая этим контролем, может быть использована для переключения на аварийный режим.

Литература:

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelungseinrichtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations-und Arbeitsblatt 330212, BIA-Handbuch. 17. Lfg. X/91, Erich Schmidt Verlag, Bielefeld.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

А.13.2 Перекрестный контроль групповых приводов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.15).

Цель: обнаружение отказов в приводах путем сравнения результатов контроля.

Описание: каждый групповой привод контролируется своим аппаратным каналом. При обнаружении различий вырабатывается аварийное действие.

А.14 Средства против физического воздействия окружающей среды

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (см. приложение А, таблицу А.17).

Цель: предотвратить влияние физической окружающей среды (влажности, пыли, коррозийных субстанций), вызывающих отказы.

Описание: покрытия оборудования должны противостоять чрезмерным внешним воздействиям [2].

Приложение В
(справочное)

Анализ методов и средств для E/E/PES: исключение систематических отказов (см. МЭК 61508-2 и МЭК 61508-3)

Примечание — Многие методы, представленные в данном приложении относятся и к программным средствам, но в приложении С они не описаны.

В.1 Общие методы и средства

В.1.1 Управление проектами

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.1 — В.6).

Цель: устранение отказов с использованием организационной модели, правил и средств по разработке и тестированию систем, связанных с безопасностью.

Описание: наиболее значимыми и лучшими средствами являются:

- создание организационной модели в основном для обеспечения качества (см. стандарты серий ИСО 9000-1— ИСО 9004-1 или аналогичные им), описанное во многих работах по обеспечению качества, и
- установление правил и определение средств для создания и подтверждения соответствия систем, связанных с безопасностью, в руководствах по взаимосвязанным и отдельным проектам.

Для управления проектами установлены следующие важные базовые принципы:

- при выборе проектной организации определяются:
  - задачи и ответственности подразделений конкретной организации,
  - полномочия департаментов по обеспечению качества,
  - независимость гарантии качества (при выполнении внутренней проверки) от разработки;
- план последовательных действий (модель действий) формируется как:
  - определение действий по выполнению проекта, включая внутренние проверки и график их проведения,

обновление проекта;
- стандартная последовательность для внутренней проверки определяется как:
планирование, проведение и контроль проверки (теория проверки),
использование различных механизмов проверок для составных частей,
сохранение результатов повторных проверок;
- управление конфигурацией реализуется как:
администрирование и проверка версий,
выявление результатов модификаций,
проверки согласованности после модификаций;
- вводятся количественные оценки для средств обеспечения качества в виде:
установления требований,
статистики отказов;
- вводятся автоматизированные универсальные методы, инструменты и средства обучения персонала.

Литература:

IEEE: Software Engineering Standards. IEEE/Wiley-Interscience, New York, 1987.
Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.
Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.1.2 Документация

Примечания

1 Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.1 — B.6).

2 См. также МЭК 61508-1 (раздел 5 и приложение А).

Цель: устранение отказов и упрощение процедуры оценки безопасности с помощью систем документирования каждого шага процесса разработки.

Описание: во время процедуры оценки, наряду со всеми составляющими, включенными в разработку, необходимо также уделять внимание эксплуатационным характеристикам и безопасности. В процессе разработки и обеспечения в любой момент времени проверки доказательств безопасности особое внимание уделяется документации на систему [17].

Основными общими подходами к документированию являются введение руководящих принципов создания документов и использование автоматизации, в том числе:
- руководящие принципы:
определяют структуру документа,
используют таблицы контрольных проверок для формирования содержания документа и
определяют формат документа;
- автоматизация управляет документированием и создается структурированная библиотека проекта.

К конкретным методам создания документов относятся:
- разделение в документации описаний:
требований,
системы (документация пользователя) и
разработки (включая внутреннюю проверку);
- группирование разработанной документации в соответствии с жизненным циклом безопасности;
- определение стандартных модулей документации, из которых могут быть скомпилированы документы;
- ясная идентификация составных частей документа;
- формализованное обновление версий;
- выбор ясных и понятных средств описания:
формализованная нотация для определений,
естественный язык для введений, обоснований и представления намерений,
графическое представление для описания примеров,
семантическое определение для графических элементов и
терминологические справочники.

Литература:

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers — Software Development and Systems Documentation. Verlag TÜV Rheinland, Köln, 1985.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Entwicklungstechnik sicherheitsverantwortlicher Software in der Eisenbahn-Signaltechnik. U. Feucht, Informatik-Fachberichte 86, Springer Verlag, Berlin, 184—195, 1984.

Richtlinie zur Erstellung und Prüfung sicherheitsrelevanter Software. K. Grimm, G. Heiner, Informatik Fachberichte 86, Springer Verlag, Berlin, 277—288, 1984.

B.1.3 Разделение систем, связанных с безопасностью, и систем, не связанных с безопасностью

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.1 и B.6).

Цель: предотвращение влияния систем, связанных с безопасностью, на системы, не связанные с безопасностью, в непредвиденных ситуациях.

Описание: в спецификации должно быть определено, возможно ли разделение систем, связанных и не связанных с безопасностью. Должны быть установлены четкие спецификации взаимодействия между системами, связанными и не связанными с безопасностью. Четкое их разделение снижает затраты на тестирование систем, связанных с безопасностью.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.1.4 Разнообразие аппаратных средств

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы А.16, А.17 и А.19).

Цель: обнаружение систематических отказов при выполнении операций EUC с использованием разнообразных компонентов с различными частотами и типами отказов.

Описание: для разнообразных каналов систем, связанных с безопасностью, используются различные типы компонентов. Это снижает вероятность отказов по общей причине (например увеличение напряжения по сравнению с номиналом, электромагнитные влияния) и повышает вероятность обнаружения таких отказов.

Существование различных средств выполнения требуемой функции, например применение других физических принципов, предполагает возможность использования других способов решения проблемы обнаружения систематических отказов. Возникает разнообразие типов используемых способов. Это функциональное разнообразие дает возможность использовать различные подходы для достижения одного и того же результата.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York,1993.

B.2 Спецификация требований к безопасности E/E/PES

Главная цель: создание спецификации, которая, по возможности, была бы полна, свободна от ошибок, противоречий и проста для проверки.

B.2.1 Структурирование спецификации

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.1 и В.6).

Цель: уменьшение сложности путем создания иерархической структуры частичных требований. Исключение ошибок интерфейса между требованиями.

Описание: данный метод разделяет функциональную спецификацию на частичные требования так, чтобы между ними существовали, по возможности, простейшие отношения. Этот метод применяется последовательно до тех пор, пока не будут получены небольшие четкие частичные требования. В результате получается иерархическая структура частичных требований, которая создает основу для спецификации полных требований. Данный метод выделяет интерфейсы между частичными требованиями и особенно эффективен при его использовании для исключения ошибок интерфейса.

Литература:

Structured Analysis and System Specification. T. De Marco, Yourdon Press, Englewood Cliffs, 1979.

ESA PSS 05-02, Guide to the user requirements definition phase, European Space Agency, 1989.

B.2.2 Формальные методы

Примечания

1 Подробные сведения о конкретных формальных методах приведены в С.2.4.

2 Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.1 и В.6).

Цель: создание недвусмысленной и согласованной спецификации с целью обнаружения ошибок и пропусков.

Описание: формальные методы обеспечивают средства разработки и описания системы на конкретном этапе ее спецификации или проектирования. Результирующее описание имеет математическую форму и может быть подвержено математическому анализу для обнаружения различных классов несогласованностей или некорректностей. Более того, такое описание может быть в некоторых случаях проанализировано на ЭВМ подобно тому, как синтаксис исходной программы проверяется компилятором, или поддержано средствами анимации для изображения различных аспектов поведения описываемой системы. Анимация может дать дополнительное подтверждение соответствия системы как реальным, так и формально заданным требованиям, поскольку анимация улучшает восприятие человеком поведения системы.

Формальный метод может в общем случае предлагать нотацию (в основном некоторые методы дискретной математики), средства для вывода описания в данной нотации и различные виды анализа для проверки корректности различных свойств описаний.

Проектирование, начиная с математически формальной спецификации, с помощью последовательности пошаговых уточнений может привести к созданию логической схемы.

Литература:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report Number 68, 1985.

В.2.3 Полуформальные методы

Цель: создание недвусмысленных и согласованных частей спецификации с целью обнаружения ошибок и пропусков.

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.1, В.2 и В.6) и в МЭК 61508-3 (таблицы А.1, А.2 и А.4).

В.2.3.1 Общие положения

Цель: убедиться в том, что проект соотвествует своей спецификации.

Описание: полуформальные методы обеспечивают средства создания описания системы на стадиях ее разработки (например спецификации, проектирования или кодирования). Описание может быть в некоторых случаях проанализировано на ЭВМ или для отображения различных аспектов поведения системы использована анимация. Анимация может придать дополнительную уверенность в том, что система соответствует как реальным, так и специфицированным требованиям.

В следующих пунктах настоящего приложения описаны два полуформальных метода.

В.2.3.2 Метод конечных автоматов /диаграммы переходов состояний

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы В.5 и В.7).

Цель: проведение моделирования, специфицирование или реализация структуры управления системы.

Описание: многие системы могут быть описаны в терминах их состояний, входов и действий. Таким образом, находясь в состоянии S1 и при получении входа I, система может выполнить действие А и перейти в состояние S2. Путем описания действий системы для каждого входа в каждом состоянии можно полностью описать систему. Такая модель системы называется «автоматом с конечными состояниями». Ее часто изображают в виде так называемой «диаграммы переходов состояний», которая показывает, как система переходит из одного состояния в другое, или в виде матрицы, в которой для каждого состояния и входа задаются действия по переходу в новое состояние.

В случае, если система усложняется или имеет естественную структуру, то это может быть отражено в уровневой структуре «автомата с конечными состояниями».

Спецификация (проект), выраженная(ый) в виде «автомата с конечными состояниями», могут быть проверены:

- на полноту (система должна иметь действие и новое состояние для каждого входа в каждом состоянии);
- согласованность (только одно изменение состояния описывается для каждой пары состояние/вход) и
- достижимость (можно или нельзя перейти из одного состояния в другое при любой последовательности входов).

Это важные свойства для критических систем. Инструменты для обеспечения этих проверок легко разработать. Существуют также алгоритмы, позволяющие автоматически генерировать тестовые примеры для верификации реализаций «автомата с конечными состояниями» или анимации модели «автомата с конечными состояниями».

Литература:

Introduction to theory of Finite State Machines. A. Gill, McGraw-Hill, 1962.

В.2.3.3 Метод сетей Петри

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы В.5 и В.7).

Цель: моделирование соответствующих аспектов поведения системы, оценка и, возможно, повышение безопасности и эксплуатационных требований путем анализа и повторного проектирования.

Описание: сети Петри относятся к классу моделей, описываемых теорией графов, и используются для представления информации и управления потоками в системах, в которых процессы конкурентны и асинхронны.

Сеть Петри — это сеть позиций и переходов. Позиции могут быть «маркированными» или «немаркированными». Переход считают «активизированным», если все его входы маркированы. В активизированном состоянии позиции разрешается (но не требуется) быть «возбужденной». Если позиция «возбуждена», то вход, поступающий на переход, становится немаркированным, а вместо него каждый выход из перехода оказывается маркированным.

Потенциальные опасности могут быть представлены в виде конкретных состояний (маркировок) в модели сети Петри. Модель может быть расширена с тем, чтобы обеспечить возможности моделирования систем во времени. И хотя «классические» сети Петри концентрируются на моделировании потоков управления, существуют некоторые расширения модели сети Петри, в которых моделируются потоки данных.

Литература:

Petri nets: Properties, analysis and applications. T. Murata, Proc. IEEE 77 (4), 541—580, April 1989.

Safety analysis using Petri nets. N. G. Leveson, J. L. Stolzy, Proc. 15th Ann. Int. Symp on Fault-Tolerant Computing, 358—363, IEEE, 1985.

Using Petri nets for safety analysis of unmanned Metro system. M. El Koursi, P. Ozello, Proc. SAFECOMP '92, 135—139, Pergamon Press, 1992.

Net theory and applications. W. Brauer (ed.), Lecture Notes in Computer Science, Vol. 84, Springer Verlag, 1980.

Petri net theory and modelling of systems. J. L. Peterson, Prentice Hall, 1981.

A tool requirements specification and analysis real time software based on timed Petri nets. S. Bologna, F. Pisacane, C Ghezzi, D. Mandrioli, Proc. SAFECOMP 88, 9—11 November 1988. Fulda, Fed. Rep. of Germany, 1988.

В.2.4 Автоматизированные средства разработки спецификации

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.1 и В.6) и в МЭК 61508-3 (таблицы А.1 и А.2).

В.2.4.1 Общие положения

Цель: использование формальных методов спецификации для упрощения автоматического обнаружения неоднозначностей и полноты.

Описание: данный метод создает спецификацию в виде базы данных, которая может автоматически анализироваться для оценки согласованности и полноты. Инструмент спецификации может в интересах пользователя использовать анимацию различных аспектов специфицированной системы. В общем случае данный метод поддерживает создание не только спецификаций, но и этап проектирования, а также другие этапы жизненного цикла. Инструменты спецификаций могут быть классифицированы в соответствии со следующими пунктами настоящего приложения.

В.2.4.2 Инструменты, не ориентированные на конкретный метод

Цель: помощь пользователю в составлении правильной спецификации, применяя подсказки и формируя связи между соответствующими частями.

Описание: инструмент спецификаций освобождает пользователя от некоторой рутинной процедуры, поддерживает управление проектом и не представляет из себя какую-либо конкретную методологию разработки спецификаций. Относительная независимость пользователей от метода позволяет быть более свободными при выборе конкретного метода и дает немного специальной поддержки, необходимой при создании спецификаций, что усложняет освоение системы.

Литература:

Integrierte Rechnerunterstützung für Entwicklung, Projektmanagement und Produktverwaltung mit EPOS. R. Lauber, P. Lempp, Elektron. Rechenanlagen 27, Heft 2, 68—74, 1985.

В.2.4.3 Процедура, ориентированная на модель с иерархическим анализом

Цель: помощь пользователю в создании правильной спецификации, обеспечении согласованности между описаниями процессов и данных на различных уровнях абстрагирования.

Описание: данный метод дает функциональное представление о необходимой системе (структурный анализ) на различных уровнях абстрагирования (степень точности). Структурный анализ проводится на различных уровнях как с процессами, так и с данными. Оценка неоднозначности и полноты возможна между иерархическими уровнями, а также между двумя функциональными единицами (модулями) на одном и том же уровне.

Литература:

Structured Analysis for Requirement Definition. D. T. Ross, K. E. Schomann jr, IEEE Trans. on SE, Januay 1977.

В.2.4.4 Модели сущности

Цель: помощь пользователю в создании правильной спецификации на основе использования при описании системы сущностей и отношений между ними.

Описание: описание проектируемой системы в виде совокупности объектов и отношений между ними позволяет определять, какие отношения могут интерпретироваться системой. В общем случае эти отношения позволяют описывать иерархическую структуру объектов, поток данных, отношения между данными и данные, зависимые от конкретных производственных процессов. Этот классический подход расширен применением управления процессами. Возможности обследования и поддержка пользователя зависят от разнообразия проиллюстрированных отношений. Но множество возможностей представления усложняет применение этого метода.

Литература:

PSL/PSA Computer-aided Technique for Structured Documentation and Analysis of Information Processing. D. Teichroew, E. A. Hershey, IEEE Trans on SE, Jan 1977.

Computer Aided Software Development. D. Teichroew, E. A. Hershey, Y. Lamamoto, Beitrag in: Verfahren und Hilfsmittel fьr Spezifikation und Entwurf von Prozeβautomatisierungs-systemen. Hommel (ed.), Bericht KfK-PDV 154, Kernforschungszentrum Karlsruhe, 1978.

PCSL und ESPRESO — zwei Ansätze zur Formalisierung der Prozessrechner Software-spezifikation. J. Ludewig, GI-Fachtagung Prozessrechner 1981, Informatik-Fachberichte Bd. 39, Springer Verlag, Berlin, 1981.

В.2.4.5 Стимул и ответ

Цель: помощь пользователю в создании правильной спецификации путем идентификации взаимоотношений «стимул — ответ».

Описание: взаимоотношения между объектами системы определены в нотации «стимулы» и «ответы». Используется простой и легко расширяемый язык, который содержит элементы языка, представляющие объекты, взаимоотношения, характеристики и структуры.

Литература:

A Requirements Engineering Methodology for Real-time Processing Requirements. M. W. Alford, IEEE Trans on SE, January 1977.

The Specification System X-SPEX — Introduction and Experiences. G. Dahll, J. Lahti, Proc. SAFECOMP '83, 111—118.

B.2.5 Таблица контрольных проверок

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.1, B.2 и B.6) и в МЭК 61508-3 (таблицы A.10 и B.8).

Цель: рассмотрение и управление критическими оценками всех важных аспектов системы на этапе жизненного цикла безопасности, обеспечивая исчерпывающий охват без установления точных требований.

Описание: специалист, заполняющий таблицу контрольных проверок, должен дать ответ на ряд вопросов. Многие вопросы носят общий характер, и он должен интерпретировать их как наиболее подходящие к конкретной оцениваемой системе. Таблицы контрольных проверок допускается использовать на всех этапах полного жизненного цикла безопасности, жизненного цикла безопасности E/E/PES и жизненного цикла безопасности программного обеспечения. Такие таблицы, в частности, полезны в качестве инструмента для оценки функциональной безопасности [4].

Для сокращения широкого разнообразия проходящих подтверждение соответствия систем большинство таблиц контрольных проверок содержат вопросы, которые применимы ко многим типам систем. Поэтому в используемой таблице контрольных проверок может оказаться множество вопросов, которые не уместны для конкретной системы и должны игнорироваться. Кроме того, может также возникнуть необходимость дополнить стандартную таблицу контрольных проверок вопросами, специально ориентированными на конкретную систему.

Использование таблицы контрольных проверок в большой степени зависит от экспертной оценки и суждения специалиста, который выбирает и применяет таблицу контрольных проверок. Принятые им решения относительно выбранных(ой) таблиц(ы) контрольных проверок и любые дополнительные или игнорируемые вопросы должны быть полностью задокументированы и обоснованы. Необходимо стремиться к тому, что если применение таблиц контрольных проверок пересматривается, то гарантируется получение одних и тех же результатов, если только не используются различные критерии.

Описание системы в заполненной таблице контрольных проверок должно быть как можно более кратким. При необходимости исчерпывающего обоснования оно должно быть дано в виде ссылок на дополнительные документы. Для документирования результатов каждого вопроса должен использоваться ответ «успешно», «безуспешно» или «не завершено», либо аналогичный набор ответов. Эта лаконичность значительно упрощает процедуру оформления общего заключения результатов оценки в виде таблицы контрольных проверок [16].

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Programmable Electronic Systems (PES) in Safety Related Application. Health and Safety Executive, UK, 1987.

Dependability of Critical Computer Systems 2, F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

B.2.6 Экспертиза спецификации

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.1 и B.6).

Цель: исключить некомплектность и противоречивость спецификации.

Описание: общий метод анализа спецификации для ее оценки с различных сторон независимой командой экспертов. Такая команда задает вопросы разработчику, который должен дать ей удовлетворительные ответы. Анализ должен (по возможности) проводиться командой экспертов, которая не принимала участия в создании спецификации. Требуемая степень независимости оценки определяется уровнями полноты безопасности, задаваемыми для системы. Команда независимых экспертов должна быть способна реконструировать эксплуатационную функцию системы бесспорным способом без ссылок на любые последующие спецификации. Специалисты команды независимых экспертов должны также убедиться в том, что охвачены все уместные аспекты безопасности и технические аспекты в эксплуатационных и организационных средствах. Общий метод экспертизы спецификации доказал свою высокую эффективность на практике [12].

Литература:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

B.3 Проектирование и разработка E/E/PES

Главная цель: создание устойчивого проекта системы, связанной с безопасностью, в соответствии со спецификацией.

B.3.1 Соблюдение руководящих материалов и стандартов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица B.2).

Цель: рассмотрение стандартов прикладного сектора (не рассматриваемых в настоящем стандарте).

Описание: во время проектирования системы, связанной с безопасностью, должны составляться руководящие материалы, которые должны, во-первых, приводить к созданию систем, связанных с безопасностью, и быть практически свободны от ошибок и, во-вторых, упрощать последующее подтверждение соответствия безопасности. Такие руководящие материалы могут быть универсальными, специальными для проекта или специальными только для отдельного этапа проекта.

Литература:

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers — Software Development and Systems Documentation. Verlag TÜV Rheinland, Kôln, 1985.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

Deutsche Bundesbahn: Richtlinien-Entwürfe 42500 to 42550 für das Handbuch «Grundsatze zur technischen Zulassung in der Signal- und Nachrichtentechnik». Bundesbahn-Zentralamt München, August 1987.

Richtlinie zur Erstellung und Prüfung sicherheitsrelevanter Software. K. Grimm, G. Heiner, Informatik Fachberichte 86, Springer Verlag, Berlin, 277-288, 1984.

B.3.2 Структурное проектирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.2 и B.6).

Цель: снижение сложности проектирования путем создания иерархической структуры частичных требований. Исключение ошибок взаимосвязей между требованиями. Упрощение верификации.

Описание: при проектировании аппаратных средств должны использоваться конкретные критерии или методы. Например, может потребоваться:

- проектирование иерархически структурированных схем;
- использование изготовленных и проверенных частей схем.

При проектировании программных средств использование структурных схем также позволяет создать однозначную структуру программных модулей. Данная структура показывает взаимосвязь модулей друг с другом, конкретные данные, которые передаются между модулями, и конкретное управление, существующее между модулями [16].

Литература:

Structured Development for Real-Time Systems (3 Volumes). P. T. Yourdon, P. T. Yourdon Press, 1985.

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

Essential Systems Analysis. St. M. McMenamin, F. Palmer, Yourdon Inc, 1984.

The Use of Structured Methods in the Development of Large Software-Based Avionic Systems. D. J. Hatley, Proceedings DASC, Baltimore, 1984.

An Overview of JSD, J. R. Cameron, IEEE Trans SE-12 No. 2, February 1986.

System Development. M. Jackson, Prentice-Hall, 1983.

MASCOT 3 User Guide. MASCOT Users Forum, RSRE, Malvern, England, 1987.

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward, S. J. Mellor, Yourdon Press, 1985.

Structured Analysis for Requirements Definition, D. T. Ross, K. E. Schoman Jr, IEEE Trans. Software Eng, Vol. SE-3, 6—15, 1977.

Structured Analysis (SA): A language for communicating ideas. D. T. Ross, IEEE Trans. Software Eng, Vol. SE-3 (1), 16—34.

Applications and Extensions of SADT. D. T. Ross, Computer, 25-34, April 1985.

Structured Analysis and Design Technique — Application on Safety Systems. W. Heins, Risk Assessment and Control Courseware, Module B1, chapter 11, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands, 1989.

B.3.3 Использование достоверно испытанных компонентов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.2 и B.6).

Цель: снижение риска многих оригинальных и необнаруживаемых отказов путем использования компонентов с конкретными характеристиками.

Описание: выбор достоверно испытанных компонентов для целей безопасности выполняется производителем в соответствии с надежностью компонентов (например, использование эксплуатационно-тестируемых физических модулей для удовлетворения высоких требований безопасности или хранение относящихся к безопасности программ только в безопасной памяти). Обеспечение безопасности памяти может касаться устранения несанкционированного доступа, влияний несанкционированной среды (электромагнитная совместимость, радиация, и т.д.), а также отклика компонентов в случае обнаружения отказов [13].

Литература:

Reliability Testing for Industrial use. W. T. Greenwood, Computer 10 (7), 26—30, 1977.

Independent Test Labs: Caveat Emptor. E. Rubinstein, IEEE Spectrum, 14 (6), 44—50, 1977.

Microcomputers in safety technique — an aid to orientation for developer and manufacturer. H. Hölscher, J. Rader, Verlag TÜV Rheinland, Köln, 1986, ISBN 3-88585-315-9.

Zuverlässigkeit elektronischer Komponenten. T. Bajenescu, VDE-Verlag, Berlin, 1985.

В.3.4 Модульное проектирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.2 и В.6).

Цель: снижение сложности и исключение ошибок, связанных с интерфейсами между подсистемами.

Описание: каждая подсистема на всех уровнях проектирования четко определена и ограничена по размеру (только небольшим набором функций). Интерфейсы между подсистемами поддерживаются как можно более простыми, и пересечения (разделяемые данные, обмен информацией) минимизированы. Сложность отдельных подсистем также ограничивается.

Литература:

EWICS European Workshop on Industrial Computer Systems, TC 7: Safety Related Computers — Software Development and Systems, Documentation. TÜV Rheinland, Köln, 1985.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Software Reliability — Principles и Practices. G. J. Myers, Wiley-Interscience, New York, 1976.

Software Design for Real-time Systems. J. E. Cooling, Chapman и Hall, 1991.

В.3.5 Средства автоматизированного проектирования

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.2 и В.6) и в МЭК 61508-3 (таблица А.4).

Цель: более систематическое выполнение процедуры проектирования. Включение в проект подходящих автоматически сконструированных элементов, уже созданных и проверенных.

Описание: инструменты автоматизированного проектирования (CAD) должны использоваться в процессе проектирования как аппаратных, так и программных средств, если они доступны и их использование обосновано сложностью системы. Корректность использования таких инструментов должна быть продемонстрирована конкретным тестированием, обширной предысторией удовлетворительного использования, либо независимой верификацией их результата для конкретной проектируемой системы, связанной с безопасностью.

Литература:

Verification — The Practical Problems. J. T. Webb и D. J. Mannering, SARSS 87, November 1987, Altrincham, England, Elsevier Applied Science, 1987, ISBN 1-85166-167-0.

An Experience in Design and Validation of Software for a Reactor Protection System. S. Bologna, E. de Agostino et al, IFAC Workshop, SAFECOMP 1979, Stuttgart, 16—18 May 1979, Pergamon Press, 1979.

В.3.6 Моделирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.2, В.5 и В.6).

Цель: проведение систематических и полных проверок как функционирования электрических/электронных схем, так и для корректного задания значений параметров их компонентов.

Описание: функция схемы, реализующая систему, связанную с безопасностью, имитируют на компьютере с помощью запрограммированной модели ее поведения. Поведение каждого отдельного компонента схемы моделируют отдельно, и отклик схемы, в которую он входит, анализируют при задании предельных значений параметров для каждого компонента.

Литература:

Proc. Working Conference on Prototyping. Namur, October 1983, Budde et al, Springer Verlag, 1984.

Using an executable specification for an information system. S. Urban et al, IEEE Trans Software Engineering, Vol. SE-11 No. 7, July 1985.

Verification and validation of Real-time Software. W. J. Quirk (ed.), Springer Verlag, Berlin, 1985.

VDI-Gemeinschaftsausschuβ Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

В.3.7 Проверка (обзор и анализ)

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.2 и В.6).

Цель: выявление рассогласования между спецификацией и реализацией.

Описание: проверяются заданные функции системы, связанной с безопасностью. Оценивается соответствие системы, связанной с безопасностью, требованиям, приведенным в спецификации. Все вызывающие сомнение ситуации при реализации и использовании изделий документируют с целью их последующего разрешения. В отличие от сквозного контроля во время процедуры проверки разработчик системы пассивен, а эксперт — активен.

Литература:

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuβ Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993. ANSI/IEE Std. 1028:1997, IEEE Standard reviews и audits.

В.3.8 Сквозной контроль

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица В.6).

Цель: выявление рассогласования между спецификацией и реализацией.

Описание: проверяются заданные функции системы, связанной с безопасностью. Оценивается соответствие системы, связанной с безопасностью, требованиям, приведенным в спецификации. Все вызывающие сомнение ситуации при реализации и использовании изделий документируются с целью их последующего разрешения. В отличие от процедуры проверки во время сквозного контроля автор должен быть активен, а эксперт — пассивен.

Литература:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

Methodisches Testen von Programmen. G. J. Myers, Oldenbourg Verlag, München, Wien, 1987.

VDI-Gemeinschaftsausschuβ Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

ANSI/IEE Std. 1028:1997, IEEE Standard for software reviews and audits.

B.4 Процедуры эксплуатации и обслуживания E/E/PES

Главная цель: разработка процедур, которые исключают ошибки во время эксплуатации и обслуживания систем, связанных с безопасностью.

B.4.1 Инструкции по эксплуатации и обслуживанию

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица B.4).

Цель: исключение ошибок во время эксплуатации и обслуживания систем, связанных с безопасностью.

Описание: инструкции пользователя содержат важную информацию о способах использования и обслуживания систем. В особых случаях эти инструкции могут содержать также примеры общих способов установки систем, относящихся к безопасности. Все инструкции должны легко восприниматься. Для описания сложных процедур и зависимостей должны использоваться рисунки и схемы.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

B.4.2 Удобство общения с пользователем

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица B.4).

Цель: снижение сложности во время эксплуатации систем, связанных с безопасностью.

Описание: правильность эксплуатации систем, связанных с безопасностью, в определенной степени зависит от оператора. Рассматривая конкретные проекты системы и рабочего места, разработчик систем, связанных с безопасностью, должен предусмотреть:

- необходимость минимального вмешательства человека;
- необходимое вмешательство наиболее простым;
- возможность минимального ущерба от ошибок оператора;
- эргономические требования при проектировании средств вмешательства и индикации;
- простые, имеющие четкую маркировку и удобные для использования средства оператора;
- неперенапряженность оператора даже в экстремальной ситуации;
- адаптированность обучения процедурам и средствам процесса вмешательства к уровню знаний и мотивации обучаемого пользователя.

B.4.3 Удобство общения с обслуживающим персоналом

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица B.4).

Цель: упрощение процедуры обслуживания систем, связанных с безопасностью, и проектирование необходимых средств для эффективной диагностики и ремонта.

Описание: профилактическое обслуживание и ремонт часто проводятся в сложных условиях давления предельных сроков. Поэтому разработчик систем, связанных с безопасностью, должен предусмотреть чтобы:

- средства, относящиеся к обслуживанию безопасности, требовались как можно реже или вообще не требовались;
- использовались достаточно чувствительные и легко управляемые диагностирующие инструменты для неизбежных ремонтов; эти инструменты должны включать в себя все необходимые интерфейсы;
- было достаточно времени (если отдельные инструменты диагностики необходимо разработать или приобрести).

B.4.4 Сокращение работ на стадии эксплуатации

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.4 и B.6).

Цель: снизить эксплуатационные возможности для обычного пользователя.

Описание: этот подход снижает эксплуатационные возможности путем:

- ограничения операций в рабочих режимах, например коммутаторами ключей;
- ограничения числа используемых в работе элементов;
- ограничения числа возможных в общем случае рабочих режимов.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

В.4.5 Эксплуатация только квалифицированным оператором

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.4 и В.6).

Цель: исключение отказов, обусловленных ошибками оператора.

Описание: оператор системы, связанной с безопасностью, обучен до степени, соответствующей сложности и степени безопасности систем, связанных с безопасностью. В обучение входит изучение основ процесса производства и взаимосвязей между системами, связанными с безопасностью и EUC.

Литература:

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

В.4.6 Защита от ошибок оператора

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица В.6).

Цель: защита системы от всех видов ошибок оператора.

Описание: ложные входные сообщения (значение, время, и т. д.) обнаруживаются проверками достоверности или контролем EUC. Для того, чтобы объединить эти средства в проекте, необходимо на самом раннем этапе определить, какие из входных сообщений возможны и какие допустимы.

В.4.7 (Не используется)

В.4.8 Защита от модификаций

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица А.18).

Цель: защита системы, связанной с безопасностью, от модификаций аппаратных средств техническими способами.

Описание: модификации или манипуляции обнаруживаются автоматически, например проверками достоверности сигналов датчиков, обнаружением техническим процессом и автоматическим запуском тестирования. При обнаружении модификации выдается аварийный сигнал.

В.4.9 Подтверждение ввода

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы А.18 и А.19).

Цель: обнаружение ошибки во время работы самим оператором до активизации EUC.

Описание: информация, вводимая в EUC через систему, связанную с безопасностью, представляется оператору до передачи в EUC с тем, чтобы оператор имел возможность обнаружить и исправить ошибки. Проектируемая система должна реагировать на неправильные, самопроизвольные действия оператора и учитывать нижние/верхние пределы скорости и направление реакции оператора. Это позволит исключить, например, более быстрое, чем предполагается, нажатие клавиш оператором, и настроить систему воспринимать двойное нажатие клавиши как одинарное или двойное за счет того, что система (изображение на экране) слишком медленно реагирует на разовое нажатие клавиши. Последовательное нажатие одной и той же клавиши при вводе критических данных должно восприниматься системой как одноразовое; нажатие клавиш «ввод» (enter) или «да» (yes) неограниченное число раз не должно приводить к нарушению безопасности системы.

Должны быть предусмотрены процедуры формирования временных пауз с возможностью выбора разных ответов (да/нет и т. п.) с тем, чтобы обеспечить возможность для размышления оператору, а системе - режим ожидания.

Любая перезагрузка PES, связанной с безопасностью, делают систему уязвимой, если программные/аппаратные средства не спроектированы с учетом этой ситуации.

Литература:

DIN V VDE 0801: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems) Beuth-Verlag, Berlin, 1990.

В.5 Интеграция E/E/PES

Главная цель: исключение отказов системы на стадии интеграции и обнаружение любых отказов во время этой и предыдущей стадий.

В.5.1 Функциональное тестирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.3 и В.5) и в МЭК 61508-3 (таблицы А.5 — А.7).

Цель: обнаружение отказов на стадиях создания спецификации и проектирования. Исключение отказов во время реализации и интеграции программных и аппаратных средств.

Описание: в процессе функционального тестирования определяется, достигнуты ли заданные характеристики системы. В систему поступают входные данные, которые адекватно характеризуют обычное выполнение операций. Наблюдаемые выходные результаты сравниваются с заданными в спецификации. Отклонения от спецификации и указания на неполноту спецификации документируются.

Функциональное тестирование электронных компонентов, предназначенных для многоканальной архитектуры, обычно включает в себя промышленные компоненты, каждый из которых поставщик уже протестировал и предварительно подтвердил соответствие. Помимо этого рекомендуется, чтобы покупные промышленные компоненты были протестированы в сочетании с другими компонентами поставщика из той же партии, чтобы выявить неисправности группового типа, которые в противном случае остались бы невыявленными.

Также о достаточных рабочих возможностях системы см. руководящие материалы (приложение С, подраздел С.5.20).

Литература:

Functional Program Testing and Analysis. W. E. Howden, McGraw-Hill, 1987. The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

В.5.2 Тестирование методом «черного ящика»

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.3, В.5 и В.6) и в МЭК 61508-3 (таблицы А.5 — А.7).

Цель: проверка динамического поведения системы в реальных условиях функционирования. Выявление несоответствия функциональной спецификации и оценка ее полезности и устойчивости.

Описание: функции системы или программы выполняются в заданном окружении с заданными данными тестирования, которые систематически формируются из спецификации в соответствии с установленными критериями. Это позволяет поведение системы сравнить с ее спецификацией. При проведении тестирования никакие сведения о внутренней структуре системы не используются. Основная цель состоит в том, чтобы определить, правильно ли выполняет функциональный модуль функции, требуемые спецификацией. Метод формирования эквивалентных классов служит примером критерия тестирования данных методом «черного ящика». Массив входных данных подразделяется на конкретные диапазоны входных значений (эквивалентные классы) на основе спецификации. После этого формируются тестовые примеры из:

- данных из допустимых диапазонов;
- данных из недопустимых диапазонов;
- данных предельных значений диапазонов;
- экстремальных значений и
- комбинаций из перечисленных выше классов.

Могут оказаться эффективными также другие критерии выбора тестовых примеров в различных режимах тестирования (модуля, интеграции и системы). Например критерий «экстремальные эксплуатационные условия» используется при тестировании системы в процессе подтверждения соответствия.

Литература:

Functional Testing and Analysis. W. E. Howden, McGraw-Hill Book Comppany, New York, 1987.

Software Testing and Validation Techniques. E. Miller, W. E. Howden, IEEE Computer Society, New York, 1978.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Methodik systematischen Testens von Software. K. Grimm, 30 (4), 1988.

VDI-Gemeinschaftsausschuβ Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

В.5.3 Статистическое тестирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.3, В.5 и В.6).

Цель: проверка динамического поведения системы, связанной с безопасностью, и оценка ее полезности и устойчивости.

Описание: при этом подходе тестируется система или программа с входными данными, выбранными в соответствии с предполагаемым статистическим распределением реальных эксплуатационных входных данных — эксплуатационный профиль.

Литература:

Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

Aspects of Development and Verification of Reliable Process Computer Software. W. Ehrenberger, IFAC-IFIP Conference Proceedings, 35—48, 1980.

Verification and validation of Real-time Software. W. J. Quirk (ed.), Springer Verlag, Berlin, 1985.

VDI-Gemeinschaftsausschuβ Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

В.5.4 Полевые испытания

Примечания

1 См. также приложение С, подраздел С.2.10 — аналогичные средства, а в приложении D — статистический подход — то и другое в контексте программного обеспечения.

2 Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.3 и В.5).

Цель: использование полевых испытаний из различных областей применения в качестве одного из средств исключения сбоев во время интеграции E/E/PES и/или в процессе подтверждения соответствия безопасности E/E/PES.

Описание: использование компонентов или подсистем, которые при их использовании показали путем испытаний отсутствие или наличие только несущественных ошибок и существенно не изменялись в течение достаточно длительного периода времени во многих различных применениях. В частности, для сложных компонентов с множеством функций (например операционной системы интегральных схем) разработчик должен обратить внимание на функции, которые были фактически протестированы методом полевых испытаний. Например, должны быть рассмотрены подпрограммы самотестирования для обнаружения сбоев: при отсутствии сбоев аппаратных средств в период эксплуатации о подпрограммах нельзя сказать, что они протестированы, поскольку они никогда не выполняли функций обнаружения своих сбоев.

При использовании полевых испытаний должны быть соблюдены следующие требования:

- неизменность спецификации;
- наличие 10 систем в различных применениях;
- длительность работы 10^5 ч и, по меньшей мере, один год сервисной поддержки.

Полевые испытания документируются поставщиком и/или эксплуатирующей организацией; документация должна, по меньшей мере, содержать:

- точное обозначение системы и ее компонентов, включая управление версиями аппаратных средств;
- сведения о пользователях и времени применения;
- отработанное время в часах;
- процедуры выбора системы и прикладные программы, использованные при испытаниях;
- процедуры обнаружения и регистрации сбоев, а также процедуры устранения их последствий и причин возникновения.

Литература:

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993.

В.6 Подтверждение безопасности E/E/PES

Главная цель: Подтвердить, что система E/E/PE, связанная с безопасностью, соответствует спецификации требований безопасности E/E/PES.

В.6.1 Функциональные испытания в условиях окружающей среды

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица В.5).

Цель: оценка защищенности системы, связанная с безопасностью, от типовых воздействий окружающей среды.

Описание: систему помещают в различные условия окружающей среды (например, в соответствии со стандартами серии МЭК 60068 или серии МЭК 61000) и оценивают способности системы выполнять функции безопасности (на соответствие требованиям стандартов, указанных выше) [1], [5].

Литература:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

В.6.2 Испытания на устойчивость к пиковым выбросам внешних воздействий

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.5 и В.6).

Цель: проверка способности систем, связанных с безопасностью, справляться с пиковыми выбросами внешних воздействий.

Описание: в систему загружается типичная прикладная программа и все периферийные линии (цифровые, аналоговые и последовательные интерфейсы, шины, источники питания и т. д.) подвергаются воздействию стандартных шумовых сигналов. Для того, чтобы получить их количественную оценку, целесообразно внимательно подходить к предельным значениям выбросов внешних влияний. Класс помех считается выбранным неверно, если функция системы не выполняется [6].

В.6.3 (Не используется).

В.6.4 Статический анализ

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.5 и В.6) и в МЭК 61508-3 (таблица А.9).

Цель: исключение систематических дефектов, которые могут приводить к отказам в тестируемой системе вначале, либо после продолжительной эксплуатации.

Описание: этот систематический и, возможно, автоматизированный метод позволяет исследовать конкретные статические характеристики опытных образцов системы для обеспечения полноты, согласованности, отсут-

ствия неоднозначностей относительно сформулированных требований (например в руководящих материалах по конструированию, системных спецификациях и инструкциях о применении). Статический анализ должен быть воспроизводим и применим к опытному образцу, который доведен до четко определенной завершающей стадии. Ниже приведены некоторые примеры статического анализа аппаратных и программных средств:

- анализ согласованности потока данных (например, при тестировании, если данные об объекте интерпретируются как имеющие одно значение);
- анализ управления потоком (например определение маршрутов, кода недоступности);
- анализ интерфейсов (например исследование передачи переменных между различными программными модулями);
- анализ потока данных для обнаружения вызывающих сомнения последовательностей для переменных: создание — использование для обращения — удаление;
- тестирование строгого соблюдения конкретных руководящих материалов (например по вопросам: длина пути утечки тока и зазоры, расстояние между совокупностями модулей, расположение физических модулей, механически чувствительные физические модули, индивидуальное использование физических модулей при их внедрении).

Литература:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuβ Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.6.5 Динамический анализ

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.5 и B.6) и в МЭК 61508-3 (таблицы A.5 и A.9).

Цель: обнаружение ошибок в спецификации путем исследования динамического поведения опытных образцов на завершающих стадиях.

Описание: динамический анализ систем, связанных с безопасностью, проводится при подаче на вход опытного образца системы, связанной с безопасностью, входных данных, которые типичны для заданного эксплуатационного окружения. Анализ будет удовлетворительным, если наблюдаемое поведение системы, связанной с безопасностью, соответствует требуемому поведению. Любой отказ системы, связанной с безопасностью, должен быть устранен, после чего новые варианты эксплуатации системы должны быть проанализированы.

Литература:

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

VDI-Gemeinschaftsausschuβ Industrielle Systemtechnik: Software-Zuverlässigkeit. VDI-Verlag, 1993.

B.6.6 Анализ отказов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.5 и B.6).

B.6.6.1 Виды отказов и анализ их последствий

Цель: проведение анализа проекта системы с исследованием всех возможных причин отказов компонентов системы и определением влияния этих отказов на поведение и безопасность системы.

Описание: анализ обычно проводится экспертным методом. Каждый компонент системы анализируется по очереди с тем, чтобы выявить набор режимов отказов для компонента, их причины и результаты, процедуры обнаружения и рекомендации. При выдаче рекомендаций они документируются в виде корректирующих действий [3].

Литература:

System Reliability Engineering Methodology: A Discussion of State of Art. J. B. Fussel, J. S. Arend, Nuclear Safety 20 (5), 1979.

Reliability Technology. A. E. Green, A. J. Bourne, Wiley-Interscience, 1972.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0942, Division of System Safety Office of Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, DC 20555, 1981.

B.6.6.2 Диаграммы последовательностей событий

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы A.10, B.3 и B.4).

Цель: моделирование с помощью диаграмм последовательности событий, которые могут представить проект системы в виде последовательности комбинаций базовых событий.

Описание: данное средство может рассматриваться как комбинация анализов на основе дерева отказов и дерева событий. Начиная с критических событий, граф последовательностей причин просматривается в прямом и обратном направлениях. Прохождение в обратном направлении эквивалентно дереву отказов, где критическое событие представлено в виде события, описанного на верхнем уровне. Прохождение в прямом направлении позволяет определять возможные последствия, возникающие из события. В узле графа могут быть символы, описывающие условия распространения причин по различным ветвям от этого узла. Временные задержки также могут учитываться. Эти условия распространения причин также могут быть описаны с помощью деревьев отказов.

Для того, чтобы диаграмма выглядела более компактной, пути распространения причин могут быть объединены с логическими символами. Должен быть определен набор стандартных символов для использования в диаграммах последовательностей причин. Такие диаграммы могут быть использованы для вычисления вероятности появления определенных критических последовательностей.

Литература:

The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis. B. S. Nielsen, Riso-M-1374, 1971.

B.6.6.3 Анализ дерева событий

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица B.4).

Цель: моделирование с помощью диаграмм последовательности событий, которая может возникнуть в системе после появления инициализирующего события и указать на возможные опасные последствия.

Описание: в верхней части диаграммы записывается последовательность условий, относящихся к формированию последовательности событий, следующих за инициализирующим событием. Начиная с инициализирующего события, являющегося целью анализа, проводится прямая линия к первому условию последовательности. Наличие ветвей «да» и «нет» диаграммы указывает на зависимость будущего события от условий. Каждая из двух ветвей продолжается до следующего условия. Однако не все условия выполняются на этих ветвях. Какая-то из них продолжится до окончания последовательности условий, но каждая ветвь дерева, построенная таким способом, представляет возможную последовательность. Дерево событий может быть использовано для вычисления вероятностей различных последовательностей, основываясь на значениях вероятностей условий и их числе в последовательности.

Литература:

Event Trees and their Treatment on PC Computers. N. Limnious и J. P. Jeannette, Reliability Engineering, Vol. 18, No. 3, 1987.

B.6.6.4 Виды отказов и анализ влияний событий на критичность компонентов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы A.10 и B.4).

Цель: ранжирование критичности компонентов, которые могут вызвать нарушения, повреждения или ухудшение работы системы при одиночных ошибках с целью определить, каким компонентам может потребоваться особое внимание и какие средства управления необходимы в процессе проектирования или эксплуатации.

Описание: критичность может быть ранжирована многими методами. Наиболее сложный метод описан Обществом автомобильных инженеров (Society for Automotive Engineers — SAE) в ARP 926. В этом методе значение критичности для любого компонента определяется числом отказов конкретного вида, предполагаемым в процессе выполнения каждого миллиона операций, реализуемых в критическом режиме. Критичность является функцией девяти параметров, большинство из которых должны быть измерены. Очень простой метод определения критичности состоит в умножении вероятности отказа компонента на величину ущерба, который может быть при этом нанесен; этот метод аналогичен простой оценке показателя риска [3].

Литература:

Design Analysis Procedure for Failures Modes, Effects и Criticality Analysis (FMECA). Aerospace Recommended Practice (ARP) 926, Society of Automotive Engineers (SAE), США, 15 September 1967.

B.6.6.5 Анализ дерева отказов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица B.4).

Цель: помощь в анализе событий или комбинации событий, которые вызывают угрозы или опасные последствия.

Описание: начиная с события, которое может непосредственно вызвать угрозу или опасные последствия («событие вершины дерева»), анализ проводится по ветвям дерева. Комбинации причины описываются логическими операторами («И», «ИЛИ» и т. д.). Затем анализируются промежуточные причины тем же способом и т. д., возвращаясь к базовым событиям, где анализ прекращается.

Данный метод является графическим, и для изображения дерева отказов используется набор стандартизованных символов. Рассматриваемый метод предназначен в основном для анализа аппаратных средств, но допускается также применять его к анализу ошибок программного обеспечения [8].

Литература:

System Reliability Engineering Methodology: A Discussion of State of Art. J. B. Fussel и J. S. Arend, Nuclear Safety 20 (5), 1979.

Fault Tree Handbook. W. E. Vesely et all, NUREG-0492, Division of System Safety Office of Nuclear Reactor Regulation, US Nuclear Regulatory Commission Washington, DC 20555, 1981.

Reliability Technology. A. E. Greene и A. J. Bourne, Wiley-Interscience, 1972.

B.6.7 Анализ наихудшего случая

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы B.5 и B.6).

Цель: исключение систематических ошибок, возникающих в результате неблагоприятных сочетаний условий окружающей среды и допусков на параметры компонентов системы.

Описание: эксплуатационные возможности системы и размеры компонентов исследуются или вычисляются теоретически. При этом для условий окружающей среды задаются их допустимые предельные значения. Анализируются и сопоставляются со спецификацией наиболее существенные характеристики системы.

В.6.8 Расширенное функциональное тестирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.5 и В.6).

Цель: обнаружение отказов на стадиях спецификации, проектирования и разработки системы. Проверка поведения системы, связанной с безопасностью, в случае редких или неспецифицированных операций ввода информации.

Описание: расширенное функциональное тестирование проверяет функциональное поведение системы, связанной с безопасностью как реакцию на входные условия, которые ожидаются только в редких случаях (например глобального отказа) или не охватываются спецификацией системы, связанной с безопасностью, (например некорректные операции). Для редко встречающихся условий наблюдаемое поведение системы, связанной с безопасностью, сравнивается со спецификацией. В тех случаях, когда реакция системы, связанной с безопасностью, не специфицирована, следует убедиться в том, что заданная безопасность сохранена в наблюдаемой реакции системы.

Литература:

Functional Program Testing and Analysis. W. E. Howden, McGraw-Hill, 1987.

The Art of Software Testing. G. J. Myers, Wiley & Sons, New York, 1979.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

В.6.9 Испытания в наихудших случаях

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.5 и В.6).

Цель: тестирование ситуаций, специфицированных во время анализа наихудших случаев.

Описание: эксплуатационные возможности системы и размеры компонентов тестируются для наихудших случаев. При этом для условий окружающей среды задают их допустимые предельные значения. Анализируется и сопоставляется со спецификацией наиболее существенные характеристики системы.

В.6.10 Испытания с введением неисправностей

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблицы В.5 и В.6).

Цель: внесение или имитация неисправностей в аппаратные средства системы и документирование реакции системы.

Описание: представленный метод оценки зависимостей является качественным. Для описания местоположения и типа неисправностей, а также способа их внесения предпочтительно используются детализированные функциональные блоки, схемы и схемные диаграммы: например, питание может не поступать на различные модули; линии питания, линии общей шины или адресные линии могут быть разомкнуты/коротко замкнуты; компоненты или их порты могут быть разомкнуты или закорочены; реле могут быть замкнуты или разомкнуты, либо их действия могут выполняться в неправильные моменты времени и т. д. Возникающие в результате отказы системы классифицируются, например, в МЭК 60812 (таблицы I и II). Обычно вводятся одиночные неисправности в устойчивом состоянии системы. Однако, в случае, если неисправность не обнаруживается тестом встроенной диагностики или оказывается не очевидной, она может сохраниться в системе и вызвать следующую неисправность. При этом количество неисправностей может быстро возрасти многократно [3], [9].

Такие испытания проводятся многопрофильным коллективом специалистов. Поставщик системы должен при этом присутствовать и получать рекомендации. Для отказов, приводящих к опасным последствиям, вычисляют и оценивают среднее время наработки на отказ. Если это время мало, необходима модификация системы.

Литература:

Integrity Testing of Process Control Systems. R. J. Lasher, Control Engineering 36 (11), 152—164, October 1989.

Приложение С
(справочное)

Анализ методов и средств достижения полноты безопасности программного обеспечения (см. МЭК 61508-3)

С.1 Общие положения

Анализ методов, содержащийся в настоящем приложении, не следует рассматривать как полный или исчерпывающий.

Литература:

System — Safety Society of America System Safety Analysis Handbook. System Safety Society, New Mexico Chapter. PO Box 95424, Albuquerque NM, USA.

Dependability of Critical Computer Systems 3. P. G. Bishop et al, Elsevier Applied Science, 1990, ISBN 1-85166-544-7.

Encyclopaedia of Software Engineering. Ed. J. Marciniak. John Wiley & Sons, 1994, ISBN 0-471-54004-8.

Software Engineer's Reference Book. Ed. J. McDermid. Butterworth-Heinemann, 1991, ISBN 0-7506-1040-9.

С.2 Требования и детальное проектирование

Примечание — Соответствующие методы и средства приведены в В.2.

С.2.1 Структурные методы

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.2 и А.4).

С.2.1.1 Общие положения

Цель: основная цель методов анализа структуры (структурных методов) состоит в обеспечении качества разработки программного обеспечения. Данные методы в основном используются на ранних стадиях жизненного цикла создаваемой системы. Структурные методы используют как точные, так и интуитивные процедуры и нотации (поддерживаемые компьютерами), а также определяют и позволяют документировать требования и возможности реализации в логической последовательности и структурированным способом.

Описание: существует достаточно много структурных методов. Некоторые из них созданы для выполнения традиционных функций обработки данных и транзакций, другие (MASCOT, JSD, Yourdon в режиме реального времени) — в большей степени ориентированы на процессы управления и задачи реального времени (для систем, реализующих такие задачи, характеристика безопасности является более критичной, чем для других систем).

Структурные методы можно считать «интеллектуальными инструментами», предназначенными для обобщенного восприятия и структуризации конкретной проблемы или системы. К их основным свойствам относятся:

- использование логики в рассуждениях и выводах, декомпозиция сложной проблемы на управляемые стадии;
- анализ и документирование общей системы, включая окружающую среду, а также разрабатываемую систему;
- декомпозиция данных и функций в разрабатываемой системе;
- использование контрольных таблиц, то есть списков типов объектов, нуждающихся в анализе;
- малая интеллектуальная перегрузка - простота, интуитивность и практичность при представлении проблемы или системы.

Нотации, используемые для анализа и документирования проблем и объектов системы (например, на основе процессов и потоков данных), ориентированы на строгость, однако нотации для выражения функций обработки выполняемых этими объектами являются более неформальными. В то же время некоторые методы частично используют формальные нотации (например, JSD использует регулярные выражения; Yourdon, SOM и SDL используют теорию конечных автоматов). Увеличение точности нотации не только повышает уровень понимания, но и обеспечивает возможность автоматизированной обработки.

Другим преимуществом структурных нотаций является их наглядность, которая позволяет пользователю интуитивно проверять возможности спецификации или проекта при неполной информации.

Настоящее приложение содержит подробное описание пяти структурных методов: представление требований, разработка системы по Джексону, MASCOT, Yourdon для систем реального времени и методология структурного анализа и проектирования (SADT).

Литература:

Software Design for Real-time Systems. J. E. Cooling, Chapman and Hall, 1991.

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward and S. J. Mellor, Yourdon Press, 1985.

Essential Systems Analysis. St. M. McMenamin, F. Palmer, Yourdon Inc, New York, 1984.

The Use of Structured Methods in Development of Large Software-Based Avionic Systems. D. J. Hatley. Proc. DASC, Baltimore, 1984.

С.2.1.2 CORE — представление требований

Цель: обеспечение требований, определений и формулировок.

Описание: данный метод должен устранить пробел между потребителем/конечным пользователем и аналитиком. Он не основан на математически строгой теории, а является средством коммуникации. Метод CORE создан для представления требований, а не для спецификаций. Данный метод является структурированным, все его представления проходят через различные уровни уточнений. Метод CORE используется для широкого круга проблем, учитывает сведения об окружающей среде, в которой система функционирует, а также различные точки зрения разных типов пользователей. Метод CORE содержит руководящие материалы и тактические подходы для того, чтобы упростить сложный проект. Такое упрощение может быть скорректировано, либо явным образом идентифицировано и задокументировано. Таким образом, спецификации могут быть неполными, однако выявленные нерешенные проблемы и области высокого риска должны быть рассмотрены при последующем проектировании.

Литература:

Software Design for Real-time Systems. J. E. Cooling, Chapman и Hall, 1991.

С.2.1.3 JSD — метод разработки системы по Джексону

Цель: разработка метода, охватывающего создание программных систем от стадии формирования требований до стадии кодирования, специально для систем реального времени.

Описание: метод JSD представляет собой поэтапную процедуру разработки, в которой разработчик моделирует поведение реального мира, на котором основываются функции системы, определяет эти функции, вводит их в модель и преобразует образовавшуюся в результате спецификацию в одну из надежно действующих в заданной окружающей среде. Поэтому данный метод охватывает традиционные этапы, такие как создание спецификаций, проектирование и разработка, но несколько отличается от традиционных методов и не является методом нисходящего проектирования.

Данный метод уделяет большое внимание выявлению на ранней стадии сущностей реального мира, относящихся к создаваемой системе, а также моделированию этих сущностей и того, что может с ними произойти. Как только анализ «реального мира» будет выполнен и создана его модель, анализируются функции системы с тем, чтобы определить, как они вписываются в модель «реального мира». Модель результирующей системы дополняется структурным описанием всех процессов модели и затем преобразуется в программы, которые могут работать в заданной программно-аппаратной среде.

Литература:

An Overview JSD. J. R. Cameron. IEEE Transactions on Software Engineering, SE-12, No. 2, February 1986.

System Development. M. Jackson, Prentice-Hall, 1983.

С.2.1.4 MASCOT — модульный метод построения, эксплуатации и тестирования программных средств

Цель: проектирование и реализация систем реального времени.

Описание: MASCOT представляет собой модульный метод проектирования, который поддерживается программно. Данный метод описывает структуры систем реального времени способом, не зависящим от типа аппаратных средств или языка реализации. MASCOT высокоорганизованно реализует проектирование, что порождает строго модульную структуру, обеспечивая близкое соответствие между функциональными элементами проекта и элементами конструкции, появляющимися при интеграции системы. Система представляется в терминах сети конкурирующих процессов, которые взаимодействуют через каналы. Каналами могут быть либо совокупности файлов, либо очереди (конвейеры) данных. Управление доступом к каналу описывается независимо от процессов в терминах механизмов доступа, которые также активизируют правила планирования процессов. Последняя версия MASCOT была спроектирована с учетом реализации языка программирования ADA.

MASCOT обеспечивает стратегию приемлемости, основанную на тестировании и верификации как отдельных программных модулей, так и более крупных совокупностей функционально взаимосвязанных программных модулей. Реализация MASCOT ориентирована на ядро MASCOT – набор примитивов планирования, которые лежат в основе реализации и обеспечивают механизмы доступа.

Литература:

MASCOT 3 User Guide MASCOT Users Forum. RSRE, Malvern, England, 1987.

С.2.1.5 Метод Йордона (Yourdon) для систем реального времени

Цель: спецификация и проектирование систем реального времени.

Описание: данный метод реализует процесс разработки системы, состоящий из трех этапов. На первом этапе происходит создание «сущностной модели», которая описывает поведение системы в целом. На втором этапе строится модель реализации, описывающая структуру и механизмы, которые, будучи реализованными, отражают требуемое поведение системы. На третьем этапе происходит фактическое построение аппаратных и программных средств системы. Три этапа строго соответствуют традиционным спецификации, проектированию и разработке, но главное, что разработчик на каждом этапе должен активно заниматься моделированием.

Сущностная модель состоит из двух частей:

- модели окружающей среды, содержащей описание границ между системой и ее окружением, а также внешних событий, на которые должна реагировать система;

- модели поведения, которая содержит схемы, описывающие преобразования, выполняемые системой в ответ на события, и описание данных, которые система должна содержать для выдачи ответов.

Модель реализации подразделяется на две подмодели, описывающие распределение отдельных процессов в процессорах и декомпозицию процессов на программные модули.

Для создания сущностной модели и модели реализации данный метод использует множество хорошо известных подходов: построение диаграмм потоков данных, преобразование графов, структурированный английский язык, диаграммы переходов состояний и сети Петри. Кроме того, данный метод обеспечивает средства моделирования предложенного проекта системы в письменном или механическом виде из отображенных моделей.

Литература:

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward and S. J. Mellor. Yourdon Press, 1985.

Strategies for Real-time System Specification. D. Hatley, E. Pirbhai, Dorset Publishing House,1988.

С.2.1.6 SADT — методология структурного анализа и проектирования

Цель: моделирование и анализ на уровне информационных потоков процессов принятия решений и задач управления в сложных системах, представленных в виде диаграмм.

Описание: в методологии SADT концепция активностных диаграмм играет главную роль. Активностная диаграмма состоит из активностей, сгруппированных в так называемые «блоки действий». Каждому блоку действий присваивается простое имя, и он связывается отношениями (изображаются стрелками) с другими блоками действий, которым также присваиваются уникальные имена. Каждый блок действий может быть иерархически декомпозирован на блоки действий более низкого уровня и отношения между ними. Каждая из четырех сторон блока действий имеет определенное в методологии SADT особое назначение: входы, управления, механизмы и выходы:

- вход — указывается стрелкой, входящей в блок действий слева. Входы могут быть представлены материальными или нематериальными объектами и предназначены для обработки одним или несколькими блоками действий;
- управление – обычно это инструкция, процедура, критерий выбора и т. п. Управление реализует выполнение действий и изображается стрелками сверху блока действий;
- механизм — ресурс в виде персонала, организационного подразделения или описания, необходимый к действию для выполнения своих задач;
- выход — все, что вырабатывается в результате действия; изображается стрелкой с правой стороны блока действий.

Если действия связаны между собой большим числом отношений, то лучше объединить эти действия в единую группу, поместить в один блок действий, не детализируя в дальнейшем его содержание. Основополагающий принцип группирования действий в блоки действий состоит в том, что образуемые в результате блоки действий должны соединяться между собой только небольшим числом отношений.

Декомпозиция моделей активностных диаграмм реализуется до тех пор, пока не потеряет смысл дальнейшая детализация блоков действий. Этот этап достигается, если действия внутри блоков становятся неразделимыми или если последующая детализация действий внутри блоков выходит за область анализа системы.

Литература:

Structured Analysis for Requirements Definition. D. T. Ross, K. E. Schoman Jr. IEEE Transactions on Software Engineering, Vol. SE-3, 1, 6—15, 1977.

Structured Analysis (SA): A Language for Communicating Ideas. D. T. Ross. IEEE Transactions on Software Engineering, Vol. SE-3, 1, 16—34, 1977.

Applications and Extensions of SADT. D. T. Ross. Computer, 25—34, April 1985.

Structured Analysis and Design Technique — Application on Safety Systems. W. Heins. Risk Assessment and Control Courseware, Module B1, Chapter 11, Delft University of Technology, Safety Science Group, PO Box 5050, 2600 GB Delft, Netherlands, 1989.

С.2.2 Диаграммы потоков данных

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы В.5 и В.7).

Цель: программная поддержка описания потока данных в виде диаграмм.

Описание: диаграммы потоков данных описывают преобразование входных данных в выходные для каждого компонента схемы, представляющего различные преобразования.

Диаграммы потоков данных состоят из трех компонентов:

- аннотированные стрелки — обозначают поток данных, входящих и исходящих из блоков преобразования с кратким описанием этих данных;
- аннотированные кружки — обозначают блоки преобразования с кратким описанием преобразований;
- операторы (and, xor) — эти операторы используются для связи аннотированных стрелок.

Каждый аннотированный кружок на диаграмме потока данных может рассматриваться как самостоятельный блок, который при появлении на его входах данных преобразует их в выходные. Одним из основных преимуществ является то, что они показывают преобразования, не предполагая, как они реализуются. Чистая диаграмма потоков данных не включает в себя управляющую информацию или информацию о последовательности процесса, так как управление реализуется в расширениях для реального времени, как в методе Йордона для систем реального времени (см. С.2.1.5).

Создание диаграмм потока данных является наилучшим подходом при анализе систем в направлении от входов к выходам. Каждый кружок на диаграмме должен обозначать разное преобразование — его выходы должны отличаться от его входов. Не существует правил определения общей структуры диаграммы, и создание диаграммы потока данных является одним из творческих аспектов создания проекта системы в целом. Подобно всем проектам, процедура, уточняющая начальную диаграмму для создания конечной, является итеративной [20], [22].

Литература:

Software Engineering. I. Sommerville, Addison-Wesley, 3rd Edition, 1989.

С.2.3 Структурные диаграммы

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.5).

Цель: представление структуры программы в виде схемы.

Описание: структурные диаграммы дополняют диаграммы потоков данных. Они описывают программируемую систему и иерархию ее компонентов, а также отображают их графически в виде дерева. Структурные диаграммы описывают способ реализации элементов диаграммы потоков данных в виде иерархии программных модулей.

Структурная диаграмма показывает взаимоотношения между программными модулями, не указывая при этом порядок активизации программных модулей. Структурные диаграммы изображаются с использованием следующих четырех символов:

- прямоугольника с именем модуля;
- линии, соединяющей эти прямоугольники, формирующие структуру;
- стрелки, отмеченной незаштрихованным кругом, с именем данных, передаваемых в направлении элементов структурной диаграммы и обратно (обычно такая стрелка изображается параллельно линиям, соединяющим прямоугольники схемы);
- стрелки, отмеченной заштрихованным кругом, с именем сигнала управления, проходящего в структурной диаграмме от одного модуля к другому, и эта стрелка также изображается параллельно линии, соединяющей два модуля.

Из любой нетривиальной диаграммы потока данных можно создать множество различных структурных диаграмм.

Диаграммы потоков данных отображают взаимоотношение между информацией и функциями системы. Структурные диаграммы отображают способ реализации элементов системы. Оба метода представляют собой действующие, хотя и различные точки зрения на конкретную систему.

Литература:

Software Engineering. I. Sommerville, Addison-Wesley, 3rd Edition, 1989.

Structured Design. L. L. Constantine and E. Yourdon, Englewood Cliffs, New Jersey, Prentice Hall, 1979.

Reliable Software Through Composite Design. G. J. Myers, New York, Van Nostrand, 1975.

С.2.4 Формальные методы

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.1, А.2, А.4 и В.5).

С.2.4.1 Общие положения

Цель: разработка программных средств, основанных на математических принципах. К этим средствам относятся методы формального проектирования и формального кодирования.

Описание: на основе формальных методов разработаны средства описания системы для решения отдельных задач на этапах спецификации, проектирования или реализации. Создаваемое в результате описание представляет собой строгую нотацию, математически анализируемую для обнаружения различных видов несогласованностей или некорректностей. Более того, такое описание может быть в некоторых случаях проанализировано автоматически по аналогии с проверкой компилятором синтаксиса исходной программы или, с целью показать различные аспекты поведения описываемой системы, использована анимация. Анимация может дать дополнительную уверенность в том, что система соответствует как реальным, так и формально специфицированным требованиям, поскольку это улучшает восприятие человеком специфицированного поведения системы.

Формальный метод обычно предлагает нотацию (как правило, используется один из методов дискретной математики), метод вывода описания в данной нотации и различные методы анализа описания для проверки корректности различных свойств системы.

Примечание — Приведенное выше описание содержится также в В.2.2.

Ряд формальных методов CCS, CSP, HOL, LOTOS, OBJ, временная логика, VDM и Z описан в подпунктах настоящего пункта. Следует заметить, что другие методы, например метод конечных автоматов (см. В.2.3.2) и сети Петри (см. В.2.3.3), в зависимости от корректности использования методами соответствующего строгого математического аппарата, могут рассматриваться как формальные.

Литература:

The Practice of Formal Methods in Safety-Critical Systems. S. Liu, V. Stavridou, B. Dutertre, J. Systems. Software 28, 77—87, Elsevier, 1995.

Formal Methods: Use and Relevance for Development of Safety-Critical Systems. L. M. Barroca, J. A. McDermid, The Computer Journal 35 (6), 579-599, 1992.

How to Produce Correct Software — An Introduction to Formal Specification and Program Development by Transformations. E. A. Boiten et al, The Computer Journal 35 (6), 547—554, 1992.

С.2.4.2 CCS — расчет взаимодействующих систем

Цель: описание и анализ поведения систем, реализующих параллельные коммуникационные процессы.

Описание: CCS — это математический аппарат, описывающий поведение систем. Проект системы моделируется в виде сети независимых процессов, реализующихся последовательно или параллельно. Процессы могут взаимодействовать через порты (аналогичные каналам CSP), и взаимодействие осуществляется только при готовности обоих процессов. Отсутствие детерминизма может быть смоделировано. Начиная с описания всей системы на высоком уровне абстрагирования (трассирование), можно выполнять пошаговое уточнение системы (стратегия сверху вниз) в рамках композиции взаимодействующих процессов, общее поведение которых формирует также поведение всей системы. В равной степени можно выполнять и стратегию снизу вверх, комбинируя процессы и получая в результате необходимые свойства формируемой системы, используя правила вывода композиционного типа.

Литература:

Communication and Concurrency. R. Milner, Prentice-Hall, 1989.

The Specification of Complex Systems. B. Cohen, W. T. Harwood and M. I. Jackson, Addison Wesley, 1986.

С.2.4.3 CSP — взаимодействующие последовательные процессы

Цель: спецификация конкурирующих программных систем, то есть систем, процессы которых реализуются одновременно.

Описание: метод CSP обеспечивает язык для системных спецификаций процессов и подтверждения соответствия реализации процессов их спецификациям (описанным как «трасса — допустимая последовательность событий»).

Система моделируется в виде сети независимых процессов, составленных последовательно или параллельно. Каждый независимый процесс описывается в терминах всех его возможных поведений. Независимые процессы могут взаимодействовать (синхронно или обмениваться данными) через каналы, и взаимодействие происходит только при готовности обоих процессов. Может быть промоделирована относительная синхронизация событий.

Теоретические положения метода CSP были непосредственно включены в архитектуру транспьютера INMOS, а язык OCCAM позволил непосредственно реализовывать на сетях транспьютеров системы, специфицированные в языке CSP.

Литература:

Communicating Sequential Processes. C A. R. Hoare, Prentice-Hall, 1985.

С.2.4.4 HOL — логика высшего порядка

Цель: спецификация и верификация аппаратных средств.

Описание: HOL представляет собой разработанную в компьютерной лаборатории Кембриджского университета конкретную логическую нотацию и систему, которая ее автоматически поддерживает. Логическая нотация взята в основном из простой теории типов Черча, а машинная реализация основана на теории LCF (логике вычислимых функции).

Литература:

HOL: A Machine Orientated Formulation of Higher Order Logic. M. Gordon, University of Cambridge Technical Report, No. 68, 1985.

Specification and Verification Using Higher-Order Logic: A Case Study, F. K. Hanna and N. Daeche, in: Formal Aspects of VLSI Design: Proceedings of 1985 Edinburgh Workshop on VLSI, pp. 179 — 213, G. Milne and P. A. Subrahmanyam (Eds.), North Holland, 1986.

Application of formal methods to VIPER microprocessor. W. J. Cullyer, C H. Pygott, Proc. IEEE 134, 133 — 141, 1987.

С.2.4.5 LOTOS

Цель: описание и анализ поведения систем, реализующих параллельные коммуникационные процессы.

Описание: LOTOS (язык для спецификации процессов, упорядоченных во времени) основан на CCS с дополнительными возможностями из близких алгебраических теорий CSP и CIRCAL (теория цепей). LOTOS преодолевает недостатки CCS в управлении структурами данных и представлении значений выражений, объединяя его со вторым компонентом, основанным на языке абстрактных типов данных ACT ONE. Процесс описания компонентов в LOTOS может быть также использован для других формальных методов при описании абстрактных типов данных [24].

С.2.4.6 OBJ

Цель: обеспечение точной спецификации системы в процессе диалога с пользователем и подтверждение соответствия системы до ее реализации.

Описание: OBJ представляет собой алгебраический язык спецификаций. Пользователи определяют требования в терминах алгебраических выражений. Системные аспекты (поведение или конструктивы) специфицируются в терминах операций, действующих над абстрактными типами данных (ADT). ADT подобен языку ADA, где поведение оператора наблюдаемо, однако подробности реализации скрыты.

Спецификация OBJ и последующая пошаговая реализация подвергаются тем же формальным методам проверки, что и другие формальные методы. Более того, поскольку конструктивные аспекты спецификации OBJ автоматически исполнимы, существует непосредственная возможность подтверждения соответствия системы на основе самой спецификации. Исполнение — это по существу оценка функций системы путем подстановки выражений (перезаписыванием), которая продолжается до тех пор, пока не будут получены конкретные выходные значения. Эта исполнимость позволяет конечным пользователям рассматриваемой системы получать «облик» планируемой системы на этапе ее спецификации без необходимости знакомства с методами, лежащими в основе формальных спецификаций.

Как и все другие методы ADT, метод OBJ применим только к последовательным системам или к последовательным аспектам параллельных систем. Метод OBJ применяют для спецификации как малых, так и крупных промышленных приложений.

Литература:

An Introduction to OBJ: A language for Writing and Testing Specifications. J. A. Goguen and J. Tardo, Specification of Reliable Software, IEEE Press 1979, reprinted in Software Specification Techniques, N. Gehani, A. McGrettrick (eds), Addison-Wesley, 1985.

Algebraic Specification for Practical Software Production. C. Rattray, Cogan Press, 1987.

An Algebraic Approach to the Standardisation and Certification of Graphics Software. R. Gnatz, Computer Graphics Forum 2 (2/3), 1983.

C.2.4.7 Временная логика

Цель: непосредственное выражение требований к безопасности и эксплуатации, а также формальное представление сохранения этих качеств на последующих этапах разработки.

Описание: стандартная предикатная логика первого порядка не содержит концепций времени. Временная логика расширяет логику первого порядка добавлением модальных операторов (например, «с этого момента» и «случайно»). Эти операторы могут использоваться для уточнения суждений о системе. Например, свойства безопасности могут потребовать использовать модальный оператор «с этого момента», но может потребоваться, чтобы и другие необходимые состояния системы были достигнуты «случайно» из некоторого другого начального состояния. Временные формулы интерпретируются последовательностями состояний (поведениями). Представление состояния зависит от выбранного уровня описания. Оно может относиться ко всей системе, системным компонентам или компьютерной программе.

Квантифицированные временные интервалы и ограничения во временной логике не обрабатываются явно. Абсолютное время обрабатывается путем образования дополнительных временных состояний, что является частью описания состояния.

Литература:

Temporal Logic of Programs. F. Kroger. EATCS Monographs on Computer Science, Vol. 8, Springer Verlag, 1987.

Design for Safety using Temporal Logic. J. Gorski. SAFECOMP 86, Sarlat, France, Pergamon Press, October 1986.

The Temporal Logic of Programs. A. Pnueli, 18th Annual Symposium on Foundations of Computer Science, IEEE, 1977.

Verifying Concurrent Processes Using Temporal Logic, Hailpern, T. Brent, Springer Verlag, 1981.

C.2.4.8 VDM, VDM++ — метод разработки Vienna

Цель: систематическая спецификация и реализация последовательных (VDM) и параллельных (VDM++) программ реального времени.

Описание: VDM — это математический метод спецификации и уточнения реализаций, который позволяет доказать их корректность относительно спецификации.

В этом основанном на модели методе спецификации состояние системы моделируется в терминах теоретико-множественных структур, в которых описаны инварианты (предикаты), а операции над этим состоянием моделируются путем определения их пред- и пост-условий в терминах системных состояний [29]. Операции могут проверяться на сохранение системных инвариантов.

Выполнение спецификаций осуществляется путем реализации состояния системы в терминах структур данных в заданном языке и уточнения операций в терминах программы на заданном языке. Этапы реализации и уточнения позволяют логически вывести свойства, устанавливающие корректность этих этапов. Выполняются или нет эти свойства, определяет разработчик.

В принципе VDM используется на этапе создания спецификации, но может также использоваться на этапах проектирования и реализации исходного кода. VDM может быть также применен к последовательно структурированным программам или к последовательным процессам в параллельных системах.

Объектно-ориентированное и параллельное для реального времени расширения VDM, VDM++ представляют собой язык формализованных спецификаций, основанный на языке VDM-SL, созданном в ИСО, и на объектно-ориентированном языке Smalltalk.

VDM++ имеет широкий диапазон конструкций, что позволяет пользователю формально специфицировать параллельные системы реального времени в объектно-ориентированной среде. В VDM++ полная формальная спецификация содержит совокупность спецификаций классов и отдельных характеристик рабочего пространства.

К средствам описания реального времени на языке VDM++ относятся:

- временные выражения, предусмотренные для представления как текущего момента, так и момента вызова метода внутри тела метода;

- выражение, описывающее синхронизирующий сигнал, которое может быть добавлено к методу для спецификации верхних (или нижних) пределов времени исполнения для корректности реализаций;

- переменные непрерывного времени, которые должны быть введены. С условными операторами и операторами действия допускается специфицировать отношения (например дифференциальные уравнения) между этими временными функциями, что оказалось очень полезно при спецификации требований к системам, действующим в среде с непрерывным временем. Уточняющие шаги приводят к дискретным программным решениям для программ реального времени.

Литература:

Conformity Clause for VDM-SL, G. I. Parkin and B. A. Wichmann, Lecture Notes in Computer Science 670, FME'93 Industrial-Strength Formal Methods, First International Symposium of Formal Methods in Europe. Editors: J. C P. Woodcock and P. G. Larsen. Springer Verlag, 501—520.

Major VDM+ - Language characteristics: http://www.ifad.dk/products/vdmlangchar.html

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall. 2nd Edition, 1990.

Software Development - A Rigorous Approach. C. B. Jones. Prentice-Hall, 1980.

Formal Specification and Software Development. D. Bjorner and C. B. Jones, Prentice-Hall, 1982.

The Specification of Complex Systems. B. Cohen, W. T. Harwood and M. I. Jackson. Addison Wesley, 1986.

C.2.4.9 Z

Цель: Z — это нотация языка спецификаций для последовательных систем и метод проектирования, позволяющий разработчику выполнять работу, начиная со спецификации на языке Z до исполнительных алгоритмов, обеспечивая при этом доказательство их корректности по отношению к спецификации.

Язык Z в принципе используется на этапе спецификации, однако данный язык был разработан для использования от этапа составления спецификации до проектирования и реализации систем. Более всего он подходит для разработки последовательных систем, ориентированных на данные.

Описание: как и в VDM, в методе спецификации, реализованном в языке Z, состояние системы моделируется в терминах теоретико-множественных структур, в которых описаны инварианты (предикаты), а операции над этим состоянием моделируются путем определения их пред- и пост-условий в терминах системных состояний. Операции допускается проверять на сохранение системных инвариантов для демонстрации их согласованности. Формальная часть спецификации подразделяется на схемы, которые обеспечивают возможность структурирования спецификаций путем их усовершенствования.

Обычно спецификация Z представляет собой сочетание формального текста на языке Z и неформального пояснительного текста на естественном языке. Формальный текст сам по себе может оказаться слишком сжатым для простого восприятия и часто его смысл необходимо пояснять, тогда как неформальный, естественный язык может оказаться неоднозначным и неточным.

В отличие от VDM язык Z представляет собой скорее нотацию, чем завершенный метод. Однако был разработан близкий метод (метод B), который может быть использован в сочетании с языком Z. Метод B основан на принципе пошагового уточнения.

Литература:

The Z Notation — A Reference Manual. J. M. Spivey. Prentice-Hall, 1992. Specification Case Studies. Edited by I. Hayes, Prentice-Hall, 1987.

The B Method. J. R. Abrial et al, VDM '91 Formal Software Development Methods, (S. Prehen и W. J. Toetenel, eds), Springer Verlag, 398 — 405, 1991.

Specification of the UNIX Filestore. C Morgan and B. Sufrin. IEEE Transactions on Software Engineering, SE-10, 2, March 1984.

C.2.5 Программирование с защитой

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.4).

Цель: создание программ, выявляющих во время их исполнения аномальные потоки управления, данных или значения данных и реагирующих на них заранее определенным и приемлемым способом.

Описание: в процессе разработки программ допускается использовать разные методы для проверки аномалий в потоках управления или данных. Эти методы могут применяться систематически в процессе программирования системы для снижения вероятности ошибочной обработки данных.

Существуют два пересекающихся множества методов защиты. Внутренние методы защиты от ошибок проектируются в программных средствах для преодоления недостатков в процессе создания этих программных средств. Эти недостатки могут быть обусловлены ошибками при проектировании или кодировании, либо ошибочными требованиями. Ниже перечислены некоторые из рекомендаций по защите:

- проверка диапазона значений переменных;

- проверка значений переменных на их достоверность (если возможно);

- проверка типа, размерности и диапазона значений параметров процедур на входе процедур.

Представленные три рекомендации помогают гарантировать допустимость значений, обрабатываемых в программах, как с точки зрения терминов программных функций, так и физических значений переменных.

Параметры «только для чтения» и параметры «для чтения-записи» должны быть разделены, и доступ к ним должен проверяться. Программные функции должны рассматривать все параметры в качестве параметров «только для чтения». Символьные константы не должны быть доступны для записи. Это помогает обнаруживать случайные перезаписи или ошибочное использование переменных.

Устойчивые к ошибкам программные средства проектируются в «предположении», что ошибки существуют в самой программной среде, либо используются выходящие за номиналы значения условий или используются предполагаемые условия, но программные средства ведут себя заранее определенным способом. В этом случае применяют следующие проверки:

- проверку на достоверность физических значений входных и промежуточных переменных;
- проверку влияния выходных переменных, предпочтительно путем прямого наблюдения соответствующих изменений состояния системы;
- проверку самими программными средствами своей конфигурации, включая наличие и доступность предполагаемых аппаратных средств, а также завершенность самих программ, что особенно важно для поддержки полноты безопасности в процессе их эксплуатации.

Некоторые из методов защиты программ, например проверки последовательности потока управления, также справляются и с внешними ошибками.

Литература:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

Software Engineering Aspects of Real-time Programming Concepts. E. Schoitsch, Computer Physics Communications 41, North Holland, Amsterdam, 1986.

С.2.6 Стандарты по проектированию и кодированию

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.4).

С.2.6.1 Общие положения

Цель: упрощение верификации, с тем чтобы поддержать групповой объективный подход и установить стандартный метод проектирования.

Описание: в самом начале между участниками проекта должны быть согласованы необходимые правила, охватывающие рассмотренные ниже методы проектирования и разработки (например JSP, MASCOT, сети Петри, и т. д.), а также соответствующие стандарты кодирования (см. С.2.6.2).

Данные правила создаются для облегчения разработки, верификации, оценки и эксплуатации. При этом должны учитываться доступные инструментальные средства, в частности, для аналитиков и развитие средств проектирования [4].

Литература:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988. ISBN 1-85166-203-0.

Verein Deutscher Ingenieure. Software-Zuverlassigkeit — Grundlagen, Konstruktive Mass-nahmen, Nachweisverfahren. VDI-Verlag, 1993, ISBN 3-18-401185-2.

С.2.6.2 Стандарты кодирования

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.1).

Цель: упростить верификацию разработанного кода.

Описание: до выполнения кодирования должны быть согласованы подробные правила, которых следует придерживаться. К таким правилам обычно относят:

- иметь подробные сведения о модульности, например о виде интерфейса, размерах программных модулей;
- использовать инкапсуляцию, наследование (ограниченное по глубине) и полиморфизм в случае объектно-ориентированных языков;
- исключать или использовать ограниченно некоторые языковые конструкции, например «goto», «equivalence», динамические объекты, динамические данные, структуры динамических данных, рекурсию, указатели и т. п.;
- ограничивать прерывания, допустимые при выполнении критичного для безопасности кода;
- распечатывать программный код (формировать листинг);
- исключать безусловные переходы (например «goto») в программах на языках высокого уровня.

Данные правила созданы для облегчения процессов тестирования программных модулей, верификации, оценки и обслуживания. При этом должны учитываться доступные инструментальные средства, в частности, для аналитиков.

Примечание — Более подробная информация по этим правилам приведена в С.2.6.3 — С.2.6.7

С.2.6.3 Отказ от динамических переменных или динамических объектов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.1).

Цель: исключить:

- нежелательные или необнаруживаемые наложения в памяти;
- узкие места ресурсов в процессе (связанном с безопасностью) выполнения программы.

Описание: в случае применения этого метода динамические переменные и динамические объекты получают определяемые во время выполнения программы определенные и абсолютные адреса в памяти. Объем (размер) распределяемой памяти и ее адреса зависят от состояния системы в момент распределения памяти и не могут быть проверены компилятором или другим автономным инструментом.

Так как число динамических переменных и объектов и существующее свободное пространство памяти для размещения новых динамических переменных или объектов зависит от состояния системы в момент их размещения, то при размещении или при использовании переменных или объектов возможны сбои. Например, если объем свободной памяти, распределяемый системой, недостаточен, то содержимое памяти другой переменной может быть неумышленно стерто. Если динамические переменные или объекты не используются, то появление этих ошибок исключено.

С.2.6.4 Проверка создания динамических переменных или динамических объектов при выполнении программы

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.1).

Цель: убедиться в том, что память, в которой должны быть размещены динамические переменные и объекты, свободна до ее загрузки, а размещение в ней динамических переменных и объектов во время выполнения программы не повлияет на уже существующие в ней переменные, данные или коды.

Описание: в случае применения этих средств к динамическим переменным относят те переменные, которые имеют свои конкретные и абсолютные адреса в памяти, устанавливаемые во время выполнения программы (в этом смысле динамические переменные являются также атрибутами экземпляров объектов).

Аппаратными либо программными средствами память проверяется на то, что она свободна до размещения в ней динамических переменных или объектов (например, для того, чтобы исключить переполнение стека). Если размещение не разрешается (например, если памяти по конкретному адресу недостаточно), должны быть предприняты соответствующие действия. После использования динамических переменных или объектов (например, после выхода из подпрограммы) вся используемая ими память должна быть освобождена.

Примечание — Альтернативным методом является демонстрация статического распределения памяти.

С.2.6.5 Ограниченное использование прерываний

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.1).

Цель: сохранение верифицируемости и тестируемости программного обеспечения.

Описание: использование прерываний должно быть ограничено. Прерывания могут использоваться, если они упрощают систему. Использование программных средств для обработки прерываний должно быть запрещено в критических ситуациях для выполняемых функций (например критичность по времени, критичность изменений данных). Если прерывания используются, то непрерываемые фрагменты должны иметь специфицированное максимальное время вычисления с тем, чтобы можно было вычислить максимальное время, в течение которого прерывание запрещено. Использование прерываний и их маскирование должны подробно документироваться.

С.2.6.6 Ограниченное использование указателей

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.1).

Цель: исключение проблем, связанных с доступом к данным без предварительной проверки типа и диапазона указателя. Обеспечение модульного тестирования и верификации программных средств. Ограничение последствия отказов.

Описание: в прикладных программных средствах указатель арифметических операций может быть использован на уровне исходного кода только в случае, если тип и диапазон значений указателя данных (для гарантии того, что ссылка указателя находится внутри корректного адресного пространства) будут проверены перед доступом. Связи между задачами в прикладных программах не должны осуществляться с помощью непосредственных ссылок между задачами. Обмен данными должен осуществляться через операционную систему.

С.2.6.7 Ограниченное использование рекурсий

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.1).

Цель: исключение неверифицируемого и нетестируемого использования вызовов подпрограмм.

Описание: если используется рекурсия, то должен быть определен четкий критерий, который делает глубину рекурсии предсказуемой.

С.2.7 Структурное программирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.4).

Цель: проектирование и реализация программы с использованием практического анализа программы без ее выполнения. Программа может содержать только абсолютный минимум статистически нетестируемого поведения.

Описание: для минимизации структурной сложности программы следует применять следующие принципы:

- разделять программу на подходящие небольшие минимально связанные программные модули, все взаимодействия между которыми точно специфицированы;
- составлять поток управления программными модулями с использованием таких структурированных конструкций, как последовательности, итерации и выбор;
- обеспечивать небольшое число возможных путей через программные модули и возможно более простые отношения между входными и выходными параметрами;
- исключать сложные ветвления и, в частности, безусловные переходы (goto) при использовании языков высокого уровня;
- по возможности, связывать ограничения цикла и ветвление с входными параметрами;
- исключать использование сложных вычислений в ветвлении и цикле.

Свойства языков программирования, которые способствуют указанному выше методу, следует использовать, предпочитая их другим свойствам, которые (как утверждают) более эффективны, за исключением случаев, когда эффективность приобретает абсолютный приоритет (например некоторые критичные к безопасности системы).

Литература:

Notes on stuctured programming. E. W. Dijkstra, Stuctured Programming, Academic Press, London, 1972, ISBN 0-12-200550-3.

A Discipline of Programming. E. W. Dijkstra. Englewood Cliffs NJ, Prentice-Hall, 1976.

A Software Tool for Top-down Programming. D. C Ince. Software — Practice and Experience, Vol. 13, No. 8, August 1983.

Verification — The Practical Problems. J. T. Webb and D. J. Mannering, SARSS 87, Nov. 1987, Altrincham, England, Elsevier Applied Science, 1987, ISBN 1-85166-167-0.

An Experience in Design and Validation of Software for a Reactor Protection System. S. Bologna, E. de Agostino et al, IFAC Workshop, SAFECOMP, 1979, Stuttgart, 16—18 May 1979, Pergamon Press, 1979.

С.2.8 Ограничение доступа/инкапсуляция информации

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.9).

Цель: предотвращение непреднамеренного доступа к данным или процедурам и обеспечение тем самым качественной структуры программных средств.

Описание: общедоступные для всех программных компонентов данные могут быть случайно или некорректно модифицированы любым из этих компонентов. Любые изменения этих структур данных могут потребовать подробной проверки программного кода и серьезных исправлений.

Ограничение доступа представляет собой общий метод к минимизации указанных выше проблем. Ключевые структуры данных «скрыты», и с ними можно работать только через конкретный набор процедур доступа, это позволяет модифицировать внутренние структуры данных или добавлять новые процедуры и при этом не оказывать влияния на функциональное поведение остальных программных средств. Например, имя директории может иметь процедуры доступа «вставить», «удалить» и «найти». Процедуры доступа и структуры внутренних данных могут быть изменены (например, при использовании различных методов просмотра или запоминании имен на жестком диске), не оказывая влияния на логическое поведение остальных программных средств, использующих эти процедуры.

В данном случае следует использовать концепцию абстрактных типов данных. Если непосредственная проверка не предусмотрена, может оказаться необходимым проверить, не было ли абстрагирование случайно разрушено.

Литература:

Software Engineering: Planning for Change. D. A. Lamb. Prentice-Hall, 1988.

On Design and Development of Program Families. D. L. Parnas. IEEE Trans SE-2, March 1976.

С.2.9 Модульный подход

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.4 и В.9).

Цель: декомпозирование программной системы на небольшие законченные модули с целью сокращения сложности системы.

Описание: модульный подход или модуляризация включает в себя несколько различных правил для этапов проектирования, кодирования и эксплуатации проекта программных средств. Эти правила меняются в соответствии с реализуемым методом проектирования. Большинство методов подчиняются следующим правилам:

- программный модуль должен выполнять одну четко сформулированную задачу или функцию;
- связи между программными модулями должны быть ограничены и строго определены, уровень связности каждого программного модуля должен быть высоким;

- совокупности подпрограмм должны строиться так, чтобы обеспечивать несколько уровней программных модулей;

- размеры подпрограмм следует ограничить некоторыми конкретными значениями, обычно от двух до четырех размеров экрана;

- подпрограммы должны иметь только один вход и один выход;

- программные модули должны взаимодействовать с другими программными модулями через свои интерфейсы, где используются глобальные или общие переменные, которые должны быть хорошо структурированы, доступ к ним должен находиться под контролем, и их использование в каждом конкретном случае должно быть обосновано;

- все интерфейсы программных модулей должны быть полностью документированы;

- все интерфейсы программных модулей должны содержать только необходимые для их функционирования параметры.

Литература:

Structured Design — Fundamentals of a Discipline of Computer Program and Systems Design. E. Yourdon, L. L. Constantine, Prentice-Hall, 1979, ISBN 0-13-854471-9.

С.2.10 Использование доверительных/проверенных программных модулей и их компонентов

Примечания

1 Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.4).

2 Математический аппарат, обеспечивающий числовые оценки данного метода, приведен в приложении D. Аналогичный метод и статистический подход изложены также в В.5.4.

Цель: исключение такого проектирования компонентов программных модулей и аппаратных средств, которое вызывало бы необходимость их интенсивных повторных проверок или перепроектирования для каждого нового применения. Использование преимущества проектов, которые не были формально или строго проверены, но для которых имеется продолжительный опыт эксплуатации.

Описание: данный метод проверяет наличие в программных модулях и компонентах систематических ошибок проектирования и/или эксплуатационных отказов. Только в редких случаях использование доверительных программных модулей и компонентов (то есть проверенных в эксплуатации) достаточно в качестве единственного средства, гарантирующего достижение необходимого уровеня полноты безопасности. Для сложных компонентов со многими возможными функциями (например операционной системы) важно установить, какая из функций достаточно проверена при ее использовании. Например, в случаях использования процедуры самотестирования для обнаружения сбоев аппаратных средств, если в период эксплуатации не появилось отказов аппаратных средств, процедуру самотестирования на обнаружение сбоев нельзя рассматривать как проверенную на практике.

Конкретный компонент или программный модуль может быть достаточно доверительным, если он уже испытан на конкретный уровень полноты безопасности или соответствует следующим критериям:

- спецификация на программный модуль или компонент не менялась;

- программные модули или компоненты эксплуатировались в различных областях применения;

- продолжительность срока эксплуатации — не менее года;

- продолжительность эксплуатации соответствует уровню полноты безопасности или соответствующему числу запросов; демонстрируется частота отказов, не связанная с безопасностью, менее:

10^{-2} на один запрос (в год) с 95 %-ным уровнем доверия, при необходимости 300 эксплуатационных прохождений (в год);

10^{-5} на один запрос (в год) с 99,9 %-ным уровнем доверия, при необходимости 690000 эксплуатационных прохождений (в год);

- весь опыт эксплуатации соотнесен с известным профилем запросов функций программного модуля для гарантии того, что увеличивающийся опыт эксплуатации действительно приводит к увеличению знаний о поведении программного модуля, связанного с соответствующим профилем запроса.

- отказы, не связанные с безопасностью.

Примечание — Отказ, некритичный для безопасности в одном контексте, может быть критичен для безопасности в другом контексте, и наоборот.

Для проверки соответствия критерию компонента или программного модуля должны быть задокументированы:

- точная идентификация каждой системы и ее компонент, включая номера версий (как для программных, так и для аппаратных средств);

- идентификация пользователей и продолжительность их работы;

- продолжительность эксплуатации системы;

- процедура выбора систем, применяемых пользователями, и случаев ее применения;

- процедуры обнаружения и регистрации отказов и устранения сбоев.

Литература:

DIN V VDE 0801 A1: Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

Guidelines for safe automation of chemical processes. CCPS, AIChE, New York, 1993.

С.3 Архитектурное проектирование

С.3.1 Обнаружение и диагностика сбоев

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: обнаружение сбоев в системе, которые могут привести к отказам и тем самым обеспечить основу для контрмер, направленных на минимизацию числа последующих сбоев.

Описание: обнаружение сбоев представляет собой действие по проверке системы на наличие ошибочных состояний [(обусловленных сбоями в проверяемой (под)системе)]. Основная цель обнаружения сбоев состоит в том, чтобы предотвратить появление неверных результатов. Система, действущая в сочетании с параллельно работающими компонентами, останавливающими управление, в случае, если она обнаруживает, что ее собственные результаты некорректны, называется самопроверяемой.

Обнаружение сбоев основывается на принципах избыточности [в основном при обнаружении сбоев аппаратных средств (см. МЭК 61508-2, приложение А)] и разнообразия (программные ошибки). Необходим один из способов голосования для определения корректности результатов. Применимы специальные методы, к которым относятся программирование утверждений, программирование *N*-версий и множественной безопасности. Для аппаратных средств: введение дополнительных сенсоров; контуров регулирования; кодов, проверяющих ошибки, и др.

Обнаружение сбоев может обеспечиваться проверками в области значений или временной области на различных уровнях, особенно на физическом уровне (температура, напряжение и т. п.), логическом (коды, обнаруживающие ошибки), функциональном (утверждения) или внешнем (проверки достоверности). Результаты этих проверок могут быть сохранены и связаны с данными, на которые повлиял сбой с тем, чтобы обеспечить возможность отслеживания отказов.

Сложные системы состоят из подсистем. Эффективность обнаружения ошибок, диагностики и компенсации ошибок зависит от сложности взаимодействия между подсистемами, влияющими на распространение ошибок.

Диагностику ошибок следует применять на уровне самых малых подсистем, поскольку подсистемы меньших размеров допускают более детальную диагностику ошибок (обнаружение ошибочных состояний).

Интегрированные информационные системы уровня предприятия могут обычным способом передавать состояния безопасности системы, в том числе информацию диагностического тестирования, другим управляющим системам. При обнаружении некорректного поведения оно может быть выделено и использовано для запуска корректирующих действий до возникновения опасной ситуации. При появлении инцидента документирование такого некорректного поведения может способствовать его последующему анализу.

Литература:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

С.3.2 Обнаружение и исправление ошибок

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: обнаружение и исправление ошибок в чувствительной к ним информации.

Описание: для информации, состоящей из *n* битов, генерируется закодированный блок из *k* битов, который позволяет обнаруживать и исправлять *r* ошибок. Примерами могут служить код Хэмминга и полиномиальные коды.

Следует отметить, что в системах, связанных с безопасностью, чаще необходимо уничтожить ошибочные данные, чем пытаться исправлять их, поскольку лишь заранее определенная часть ошибок может быть исправлена.

Литература:

The Technology of Error Correcting Codes. E. R. Berlekamp, Proc. IEEE 68 (5), 1980.

A Short Course on Error Correcting Codes. N. J. A. Sloane, Springer Verlag, Wien, 1975.

С.3.3 Программирование с проверкой ошибок

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-2 (таблица А.18) и в МЭК 61508-3 (таблица А.2).

Цель: обнаружение ошибок, оставшихся при проектировании программных средств, в процессе выполнения программ с целью предотвращения критичных для безопасности отказов систем, и продолжение правильного выполнения программы.

Описание: в методе программирования утверждений уже заложена идея проверки предусловий (до выполнения последовательности операторов начальные условия проверяют на соответствие) и постусловий (проверяют результаты после выполнения последовательности операторов). Если предусловия или постусловия не соблюдаются, то выдается сообщение об ошибке.

ПРИМЕР -
assert < pre-condition>;
action 1 ;
.....
action x;
assert < post-condition>;

Литература:

A Discipline of Programming. E. W. Dijkstra, Prentice-Hall, 1976.
The Science of Programming. D. Gries, Springer Verlag, 1981.
Software Development — A Rigorous Approach. C. B. Jones, Prentice-Hall, 1980.

С.3.4 Методы «подушки безопасности»

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: защита от необнаруженных на этапах спецификации и реализации ошибок в программных средствах, которые неблагоприятно влияют на их безопасность.

Описание: метод «подушки безопасности» представляет собой использование внешнего монитора, реализованного на независимом компьютере в другой спецификации. Данный метод касается исключительно гарантии того, чтобы главный компьютер выполнял безопасные, не обязательно корректирующие, действия. «Подушка безопасности» непрерывно контролирует главный компьютер и предотвращает вхождение системы в опасное состояние. Кроме того, если обнаружится, что главный компьютер вошел в потенциально опасное состояние, система должна возвратиться обратно в безопасное состояние с помощью либо «подушки безопасности», либо главного компьютера.

Аппаратные и программные средства «подушки безопасности» следует классифицировать и квалифицировать в соответствии с подходящим SIL.

Литература:

Using AI Techniques to Improve Software Safety. Proc. IFAC SAFECOMP 88, Sarlat, France, Pergamon Press, October 1986.

С.3.5 Многовариантное программирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: обнаружение и наложение маски при выполнении программ на невыявленные на этапах проектирования и реализации ошибки программных средств для предотвращения критичных для безопасности отказов системы и продолжения ее правильной работы.

Описание: при многовариантном программировании заданная программная спецификация проектируется и реализуется различными способами *N* раз. Одни и те же входные значения поступают в *N* версий, и сравниваются результаты, выданные *N* версиями. Если результат определяется как правильный, он поступает на выходы компьютера.

N версий могут выполняться параллельно на различных компьютерах, либо все версии могут выполняться на одном компьютере с последующим сравнением полученных результатов на том же компьютере. Для этих *N* результатов могут быть использованы различные стратегии сравнения и в зависимости от заданных требований применяются следующие стратегии:

- если система находится в безопасном состоянии, можно потребовать полного согласия (все *N* результатов одинаковы), в противном случае используется выходное значение, которое заставит систему перейти в безопасное состояние. Для простых пошаговых систем сравнение может обеспечить безопасность. В этом случае безопасное действие может быть разбито по шагам, если какая-либо версия реализует пошаговые операции. Этот подход обычно используется только для двух версий (*N* = 2);
- для систем, находящихся в опасном состоянии, могут быть реализованы стратегии мажоритарного сравнения. В случаях, если отсутствует общее согласие, могут использоваться вероятностные подходы с тем, чтобы максимизировать вероятность выбора правильного значения, например, принять среднее значение, временно зафиксировать выходы, пока не будет достигнуто согласие и т. п.

Данный метод не устраняет ошибок, не выявленных при проектировании программ, а также ошибок в интерпретации спецификации, однако он является средством для обнаружения и маскирования ошибок, прежде чем они смогут повлиять на безопасность.

Литература:

Dependable Computing: From Concepts to Design Diversity. A. Avizienis and J. C Laprie, Proc. IEEE 74 (5), May 1986.

A Theoretical Basis for Analysis of Multi-version Software subject to Co-incident Failures. D. E. Eckhardt and L. D. Lee, IEEE Trans SE-11 (12), 1985.

Computers can now perform vital functions safely. Otto Berg Von Linde, Railway Gazette International, Vol. 135, No. 11, 1979.

С.3.6 Блоки восстановления

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: повышение вероятности выполнения программой, в конечном счете, своих заданных функций.

Описание: некоторые различные разделы программы, которые часто пишутся независимо, предназначены для выполнения одной требуемой функции. Из таких разделов конструируется окончательная программа. Сначала выполняется первый раздел, называемый первичным. Далее происходит тестирование его результатов. Если проверка проходит успешно, результат принимается и передается следующим разделам программы. Если проверка дает отрицательный результат, то все побочные эффекты первого сбрасываются и выполняется второй раздел, называемый первой альтернативой. Далее следует тестирование второго раздела, которое выполняется аналогично первому. При необходимости могут быть предусмотрены вторая, третья и т. д. альтернативы.

Литература:

System Structure for Software Fault Tolerance. B. Randall. IEEE Trans Software Engineering, Vol. SE-1, No. 2, 1975.

Fault Tolerance — Principles and Practice. T. Anderson, P. A. Lee, Prentice-Hall, 1981.

С.3.7 Восстановление предыдущего состояния

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: обеспечение исправления функциональных операций при наличии одной или нескольких ошибок.

Описание: при обнаружении ошибки система возвращается в первоначальное внутреннее состояние, правильность которого была подтверждена ранее. Данный метод предполагает частое сохранение внутреннего состояния в так называемых четко определенных контрольных точках. Сохранение может быть выполнено глобально (для всей базы данных) или частично (для изменений только между контрольными точками). После этого система должна устранить изменения, произошедшие за это время путем занесения в журнал (аудиторское отслеживание действий) компенсации (все результаты этих изменений аннулируются) или внешнего (ручного) способа.

Литература:

Software Fault Tolerance (Trends in Software, No. 3), M. R. Lyu (ed.), John Wiley & Sons, April 1995, ISBN 0471950688.

С.3.8 Прямое восстановление

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: обеспечение исправления функциональных операций при наличии одного или нескольких сбоев.

Описание: при обнаружении сбоя текущее состояние системы анализируется для достижения состояния, которое через некоторое время будет признано правильным. Данный метод особенно подходит для систем реального времени с небольшой базой данных и высокой скоростью изменения внутреннего состояния. Предполагается, что, по меньшей мере, часть состояний системы может влиять на окружение, и окружение влияет только на часть состояний системы.

Литература:

Software Fault Tolerance (Trends in Software, No. 3), M. R. Lyu (ed.), John Wiley & Sons, April 1995, ISBN 0471950688.

С.3.9 Механизмы повторных попыток парирования сбоя

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: парирование обнаруженного сбоя с помощью механизмов повторных попыток.

Описание: в случае обнаружения сбоя или ошибочного условия предпринимаются попытки парирования сбоя или восстановления ситуации путем повторного выполнения того же кода. Восстановление с помощью повторной попытки может быть полным в виде перезагрузки и повторного пуска процедуры, либо небольшим в виде перепланирования и повторного пуска задачи после выполнения блокировки по времени программы или управляющего действия задачи. Методы повторной попытки широко используются при коммуникационных сбоях или при восстановлении от ошибок, и условия повторной попытки могут быть отделены флажками от ошибки протокола связи (контрольная сумма и т. д.) или от подтверждающего ответа блокировки по времени коммуникации.

Литература:

Reliable Computer Systems: Design and Evaluation, D. P. Siewiorek and R. S. Schwartz, A. K. Peters Ltd., 1998, ISBN 156881092X.

С.3.10 Сохранение достигнутых состояний

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: безопасное прекращение работы программы в случае, если она попытается выполнить неразрешенное действие.

Описание: все соответствующие подробные сведения о каждом выполнении программы документируются. При нормальной работе каждая выполненная операция программы сравнивается с ранее задокументированными сведениями. При обнаружении различий выполняются действия по безопасности.

Документация о выполненных операциях может содержать последовательность индивидуальных шагов «от решения к решению» или последовательность отдельных обращений к массивам, записям или томам, либо к тому и другому.

Допускаются различные методы хранения сведений о последовательностях шагов выполнения программы. Могут быть использованы методы хэш-кодирования для отображения этих последовательностей в виде одного большого числа или последовательности чисел. При нормальной работе перед выполнением выходной операции значения чисел, отображающих последовательности шагов выполнения программы, должны быть сопоставлены со значениями, сохраненными в памяти.

Поскольку возможные комбинации таких последовательностей шагов при выполнении одной программы достаточно велики, может оказаться невозможным рассматривать программы как единое целое. В этом случае данный метод может быть применен на уровне программных модулей.

Литература:

Fail-safe Software — Some Principles and a Case Study. W. Ehrenberger. Proc. SARSS 1987, Altrincham, Manchester, UK, Elsevier Applied Science, 1987.

С.3.11 Постепенное отключение функций

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: обеспечение пригодности наиболее критичных системных функций, несмотря на отказы, путем игнорирования наименее критичных функций.

Описание: данный метод устанавливает приоритеты для различных функций, выполняемых системой. Проект создаваемой системы гарантирует, что в случае недостаточности ресурсов для выполнения всех системных функций функции высшего приоритета будут выполнены в предпочтение функциям более низкого приоритета. Например, функции регистрации ошибки и события могут оказаться задачей более низкого приоритета, чем системные функции управления, и в этом случае управление системой будет продолжаться, даже если аппаратные средства из-за регистрации ошибки окажутся неработоспособными. Более того, если аппаратные средства управления системой окажутся неработоспособными, а аппаратные средства регистрации ошибок останутся работоспособными, то аппаратные средства регистрации ошибок возьмут на себя функцию управления.

Данные соображения относятся в основном к аппаратным средствам, но они применимы также и к системе в целом. Данные соображения должны учитываться, начиная с самых ранних этапов проектирования.

Литература:

Space Shuttle Software. C. T. Sheridan, Datamation, Vol. 24, July 1978.

The Evolution of Fault-Tolerant Computing. Vol. 1 of Dependable Computing and Fault-Tolerant Systems, Edited by A. Avizienis, H. Kopetz and J. C Laprie, Springer Verlag, 1987, ISBN 3-211-81941-X.

Fault Tolerance, Principle and Practices. T. Anderson and P. A. Lee, Vol. 3 of Dependable Computing and Fault-Tolerant Systems, Springer Verlag, 1987, ISBN 3-211-82077-9.

С.3.12 Исправление ошибок методами искусственного интеллекта

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: способностью гибко реагировать на возможные угрозы безопасности, используя сочетания методов и моделей процессов, а также некоторые способы безопасности в режиме «онлайн» и анализа надежности.

Описание: для различных каналов связи системы прогнозирование (вычисление тенденций), исправление ошибок, обслуживание и контролирующие действия могут достаточно эффективно поддерживаться системами, основанными на методах искусственного интеллекта (AI). Правила для таких систем могут быть образованы непосредственно из спецификаций и проверены на соответствие. С помощью методов искусственного интеллекта некоторые ошибки общего характера, попадающие в спецификации, для устранения которых уже существуют некоторые правила проектирования и реализации, могут быть исключены, особенно при представлении комбинаций моделей и методов функциональным или описательным способом.

Методы выбираются так, чтобы ошибки могли быть устранены и влияние отказов минимизировано для обеспечения требуемой полноты безопасности.

Примечание — Предупреждение об исправлении ошибочных данных см.С.3.2 и об отрицательных рекомендациях применения данного метода — МЭК 61508-3 (таблица А.2, пункт 5).

Литература:

Automatic Programming Techniques Applied to Software Development: An approach based on exeption handling. M. Bidoit et al, Proc. 1st Int. Conf. on Applications of Artificial Intelligence to Engineering Problems, Southampton, 165—177, 1986.

Artificial Intelligence and Design of Expert Systems. G. F. Luger and W. A. Stubblefield, Benjamin/Cummings, 1989.

С.3.13 Динамическая реконфигурация

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.2).

Цель: обеспечение функциональности системы, несмотря на внутренний отказ.

Описание: логическая архитектура системы должна быть такой, чтобы ее можно было отобразить в подмножестве доступных ресурсов системы. Логическая архитектура должна быть способна к обнаружению отказа в

физическом ресурсе и дальнейшему повторному преобразованию логической архитектуры в ограниченные ресурсы, остающиеся функционирующими. Несмотря на то, что данный метод в основном традиционно ограничен только восстановлением отказавших модулей аппаратных средств, он применим также к ошибкам в программных средствах при наличии достаточной «избыточности времени прогона» для повторного выполнения программы или при наличии достаточных избыточных данных, которые обеспечат незначительное влияние отдельному и изолированному отказу.

Данный метод должен рассматриваться на первом этапе проектирования системы.

Литература:

Critical Issues in the Design of Reconfigurable Control Computer, H. Schmid, J. Lam, R. Naro and K. Weir, FTCS 14 June 1984, IEEE, 1984.

Assigning Processes to Processors: A Fault-tolerant Approach. G. Kar and C N. Nikolaou, Watson Research Centre, Yorktown, June 1984.

С.4 Инструменты разработки и языки программирования

С.4.1 Строго типизированные языки программирования

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.3).

Цель: снижение вероятности ошибок путем использования языка, который компилятором обеспечивает высокий уровень проверки.

Описание: если скомпилирован строго типизированный язык программирования, то проводится много проверок по использованию типов переменных, например в вызовах процедур и доступе к внешним данным. Компиляция может оказаться безуспешной, и будет выдано сообщение об ошибке при любом использовании типа переменных, которое не соответствует заранее установленным правилам.

Подобные языки обычно позволяют определять установленные пользователем типы данных на основе типов данных базового языка (например целое число, реальное число). Затем эти типы могут быть использованы так же, как и базовый тип. Вводятся строгие проверки, чтобы гарантировать использование правильного типа. Эти проверки проводятся для всей программы, даже если она построена из отдельных скомпилированных модулей. Данные проверки гарантируют также, что число и тип аргументов конкретной процедуры соответствуют числу и типу аргументов в ее вызове, даже если к ней обращаются из отдельно скомпилированных программных модулей.

Строго типизированные языки обычно обеспечивают другие аспекты проверенной на практике технике программного обеспечения, например, легко анализируемые структуры управления (if... then... else..., do... while и т. п.), которые приводят к четко структурированным программам.

Типичными примерами строго типизированных языков являются Pascal [21], Ada [23] и Modula 2 [27].

Литература:

In Search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software. A. Avizienis, M. R. Lyu and W. Schutz. 18th Symposium on Fault-Tolerant Computing, Tokyo, Japan, 27—30 June 1988, IEEE Computer Society Press, 1988, ISBN 0-8186-0867-6.

С.4.2 Подмножество языка

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.3).

Цель: снижение вероятности внесения программных ошибок и повышение вероятности обнаружения оставшихся ошибок.

Описание: язык исследуется для определения программных конструкций, подверженных ошибкам либо сложных для анализа, например, при использовании методов статического анализа. После этого определяется языковое подмножество, которое исключает такие конструкции.

Литература:

Requirements for programming languages in safety and security software standard. B. A. Wichmann. Computer Standards and Interfaces. Vol. 14, pp 433—441, 1992.

Safer C: Developing Software for High-integrity and Safety-critical Systems. L. Hatton, McGraw-Hill, 1994, ISBN 0-07-707640-0.

С.4.3 Сертифицированные средства

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.3).

Цель: помощь разработчику на различных этапах разработки программных средств в использовании необходимых инструментальных средств, которые, где это возможно, должны быть сертифицированы с тем, чтобы обеспечить конкретную степень уверенности в корректности результатов.

Описание: сертификацию инструментальных средств в общем случае допускается проводить независимо, как правило, в национальных органах по сертификации, по независимому набору критериев, находящемуся обычно в национальных или международных стандартах. В идеальном случае инструментальные средства, применяемые на всех стадиях разработки (спецификация, проектирование, кодирование, тестирование и оценка соответствия), и те из них, которые используются в управлении конфигурацией, должны быть сертифицированы.

В настоящее время регулярным процедурам сертификации подвергаются только компиляторы (трансляторы); сертификация проводится национальными органами по сертификации и заключается в проверке компиляторов (трансляторов) на соответствие международным стандартам, например, для языков Ada или Pascal.

Важно отметить, что сертифицированные инструментальные средства и сертифицированные трансляторы обычно сертифицируются только на соответствие стандартам на соответствующий язык или процесс. Обычно они никак не сертифицируются на соответствие стандартам по безопасности.

Литература:

Pascal Validation Suite. UK Distributor: BSI Quality Assurance, PO Box 375, Milton Keynes, MK14 6LL.

Ada Validation Suite. UK Distributor: National Computing Centre (NCC), Oxford Road, Manchester, England.

С.4.4 Инструментальные средства, заслуживающие доверие на основании опыта использования

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.3).

Цель: исключение любых проблем, обусловленных ошибками транслятора, которые могут появиться во время разработки, верификации и эксплуатации программного пакета.

Описание: транслятор используется в тех случаях, когда не очевидно неправильное исполнение многих предыдущих проектов. Если отсутствует опыт эксплуатации трансляторов или в них обнаружны любые известные серьезные ошибки, то от таких трансляторов следует отказаться, если только нет каких-либо других гарантий корректной работы транслятора (см. С.4.4.1).

Если в трансляторе выявлены небольшие недостатки, то соответствующие языковые конструкции фиксируются и в проектах, связанных с безопасностью, не применяются.

Другим вариантом исключения проблем, обусловленных ошибками транслятора, является ограничение языка до его общеиспользуемых конструкций.

Настоящие рекомендации основаны на опыте построения многих проектов. Доказано, что недоработанные трансляторы служат серьезным препятствием в любой программной разработке. Такие трансляторы в общем случае делают невозможным разработку программного обеспечения, связанного с безопасностью.

В настоящее время не существует методов подтверждения корректности всего транслятора или отдельных его частей.

С.4.4.1 Сравнение исходных программ и исполнимых кодов

Цель: убедиться в том, что инструменты, используемые для создания образа PROM, не вносят в него никаких ошибок.

Описание: образ PROM обратно преобразуется в совокупность «объектных» модулей. Эти «объектные» модули обратно преобразуются в скомпонованные файлы языка, которые затем с помощью подходящих методов сравниваются с фактическими исходными файлами, используемыми первоначально для разработки PROM.

Основное преимущество данного метода состоит в том, что инструменты [компиляторы, редакторы связей (компоновщики) и т. п.], используемые для разработки образа PROM, не требуют подтверждения соответствия. Этим методом проверяют правильность преобразования исходного файла, используемого для конкретной системы, связанной с безопасностью.

Литература:

Demonstrating Equivalence of Source Code and PROM Contents. D. J. Pavey and L. A. Winsborrow. The Computer Journal Vol. 36, No. 7, 1993.

Formal demonstration of equivalence of source code and PROM contents: an industrial example. D. J. Pavey and L. A. Winsborrow. Mathematics of Dependable Systems, Ed. C Mitchell and V. Stavridou, Clarendon Press, 1995, ISBN 0-198534-91-4.

Retrospective Formal Verification of Reactor Protection System Software. D. J. Pavey, L. A. Winsborrow, A. R. Lawrence. Proceedings of the Second Safety Through Quality Conference, 1995, ISBN 1-897851-06-5.

Assuring Correctness in a Safety Critical Software Application. L. A. Winsborrow and D. J. Pavey. High Integrity Systems, Vol. 1, No. 5, pp 453—459, 1996.

С.4.5 Библиотека проверенных/верифицированных модулей и компонентов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.3).

Цель: исключение необходимости чрезмерных повторных проверок или перепроектирования компонентов программного обеспечения и аппаратных средств при каждом новом применении. Кроме того, содействовие созданию проектов, которые не были формально или строго проверены, но относительно которых имеется значительная предыстория эксплуатации.

Описание: хорошо спроектированные и структурированные PES строятся из множества компонентов и модулей аппаратных и программных средств, которые четко различаются и которые взаимодействуют друг с другом строго специфицированным способом.

Различные PES, созданные для различных применений, могут содержать большое число одинаковых или очень схожих между собой программных модулей или компонентов. Создание библиотеки таких общеприменимых программных модулей позволяет использовать большую часть ресурсов, необходимых для подтверждения соответствия проекта, сразу для нескольких применений.

Кроме того, использование подобных программных модулей для многих применений дает практическое подтверждение их успешной эксплуатации. Это практическое подтверждение увеличивает доверие пользователей к программным модулям.

Один из подходов, в соответствии с которым программному модулю можно доверять при его практическом использовании, описан в С.2.10.

Литература:

Software Reuse and Reverse Engineering in Practice. P. A. V. Hall (ed.), Chapman & Hall, 1992, ISBN 0-412-39980-6.

DIN V VDE 0801 A1 : Grundsätze für Rechner in Systemen mit Sicherheitsaufgaben (Principles for Computers in Safety-Related Systems). Änderung 1 zu DIN V VDE 0801/01.90. Beuth-Verlag, Berlin, 1994.

С.4.6 Выбор соответствующего языка программирования

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.3).

Цель: обеспечение в максимальной степени требований настоящего стандарта для специального защищающего программирования, строгой типизации, структурного программирования и, возможно, суждений. Выбранный язык программирования должен обеспечить легко верифицируемый код и простые процедуры разработки, верификации и эксплуатации программ.

Описание: язык программирования должен быть полностью и однозначно определен. Язык должен быть ориентирован на пользователя или проблему, а не на процессор или платформу. Широко используемые языки программирования или их подмножества должны быть предпочтительнее языков специального применения [4], [11].

Языки программирования также должны обеспечивать:

- блоковую структуру организации программ;
- проверку времени трансляции;
- печать времени прогона и проверку ограничения массива.

Язык программирования должен включать в себя:

- использование небольших и управляемых программных модулей;
- ограничение доступа к данным в конкретных программных модулях;
- определение поддиапазонов переменных;
- любые другие типы конструкции, ограничивающие ошибки.

Если операции по безопасности системы зависят от ограничений реального времени, то язык программирования должен обеспечивать также обработку исключительных состояний или прерываний.

Желательно, чтобы язык программирования обеспечивался соответствующим транслятором, подходящими библиотеками с заранее созданными программными модулями, отладчиком и инструментами как для управления, так и для разработки.

В настоящее время еще не ясно, будут ли объектно-ориентированные языки программирования предпочтительнее других общепринятых языков.

К свойствам, которые усложняют верификацию и поэтому должны быть исключены, относятся:

- безусловные переходы (за исключением вызовов подпрограмм);
- рекурсии;
- указатели, динамически распределяемые области памяти или любые типы динамических переменных или объектов;
- обработка прерываний на уровне исходного кода;
- множество входов или выходов в циклах, блоках или подпрограммах;
- инициализация или декларация неявных переменных;
- вариантные записи и эквивалентность;
- параметры процедуры.

Языки программирования низкого уровня, в частности ассемблеры, обладают недостатками, связанными с их жесткой ориентацией на процессор машины или на определенную платформу.

Желательным свойством языка программирования является то, что его проектирование и использование должно приводить к созданию программ, выполнение которых предсказуемо. Если используется подходящий конкретный язык программирования, то в нем должно существовать подмножество, которое гарантирует, что выполнение программы предсказуемо. Это подмножество не может быть (в общем случае) статически определено, несмотря на то, что многие статические ограничения помогают гарантировать предсказуемое выполнение. Обычно это может потребовать демонстрации того, что индексы массива находятся в установленных пределах и что числовое переполнение не может возникнуть, и т. п.

Рекомендации по конкретным языкам программирования [21], [23], [25] — [28], [30] — [32] приведены в таблице С.1.

Литература:

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

Таблица С.1 — Рекомендации по конкретным языкам программирования

Язык программирования	SIL1	SIL2	SIL3	SIL4
1 ADA	HR	HR	R	R
2 ADA с подмножеством	HR	HR	HR	HR
3 MODULA-2	HR	HR	R	R
4 MODULA- с подмножеством	HR	HR	HR	HR
5 PASCAL	HR	HR	R	R
6 PASCAL с подмножеством	HR	HR	HR	HR
7 FORTRAN 77	R	R	R	R
8 FORTRAN 77 с подмножеством	HR	HR	HR	HR
9 C	R	—	NR	NR
10 Язык С с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR
11 PL/M	R	—	NR	NR
12 PL/M с подмножеством и стандартом кодирования	HR	R	R	R
13 Ассемблер	R	R	—	—
14 Ассемблер с подмножеством и стандартом кодирования	R	R	R	R
15 Многоступенчатые диаграммы	R	R	R	R
16 Многоступенчатая диаграмма с определенным подмножеством языка	HR	HR	HR	HR
17 Диаграмма функциональных блоков	R	R	R	R
18 Диаграмма функциональных блоков с определенным подмножеством языка	HR	HR	HR	HR
19 Структурированный текст	R	R	R	R
20 Структурированный текст с определенным подмножеством языка	HR	HR	HR	HR
21 Последовательная функциональная диаграмма	R	R	R	R
22 Последовательная функциональная диаграмма с определенным подмножеством языка	HR	HR	HR	HR
23 Список команд	R	—	NR	NR
24 Список команд с определенным подмножеством языка	HR	R	R	R

Примечания

1 Пояснения к рекомендациям R, HR см. в МЭК 61508-3 (приложение А).

2 Системное программное обеспечение включает в себя операционную систему, драйверы, встроенные функции и программные модули, являющиеся частью системы. Программные средства обычно обеспечиваются системой безопасности при поставке. Подмножество языка следует выбирать очень внимательно с тем, чтобы исключить сложные структуры, которые могут образоваться в результате ошибок реализации. Следует проводить проверки для того, чтобы убедиться в правильном использовании подмножества языка программирования.

3 Прикладная программа представляет собой программу, разработанную для конкретного безопасного применения. Во многих случаях такая программа разрабатывается конечным пользователем либо подрядчиком, ориентированным на разработку прикладных программ. В тех случаях, когда ряд языков программирова-

Окончание таблицы С.1

ния поддерживают одни и те же рекомендации, разработчику следует выбрать тот, который повсеместно используется персоналом в конкретной промышленности или отрасли. Подмножество языка программирования следует выбирать с особым вниманием, чтобы исключить сложные структуры, которые могут привести к ошибкам реализации.

4 Если конкретный язык программирования не представлен в настоящей таблице, то это не означает, что он исключен. Этот конкретный язык программирования должен соответствовать требованиям настоящего стандарта.

5 О пунктах 15 — 24 см. МЭК 61131-3.

С.5 Верификация и модификация

С.5.1 Вероятностное тестирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.5, А.7 и А.9).

Цель: получение количественных показателей надежности исследуемой программы.

Описание: количественные показатели могут быть получены с учетом относительных уровней доверия и значимости и должны иметь следующий вид:

- вероятность ошибки при запросе;
- вероятность ошибки в течение определенного периода времени;
- вероятность последствий ошибки.

Из этих показателей могут быть получены другие показатели, например:

- вероятность безошибочной работы;
- вероятность живучести;
- доступность;
- MTBF или частота отказов;
- вероятность безопасного исполнения.

Вероятностные соображения основываются либо на статистических испытаниях, либо на опыте эксплуатации. Обычно число тестовых примеров или наблюдаемых практических примеров очень велико. Обычно тестирование в режиме запросов занимает значительно меньше времени, чем в непрерывном режиме работы.

Для формирования входных данных тестирования и управления выходными данными тестирования обычно используются инструменты автоматического тестирования. Крупные тесты прогоняются на больших центральных компьютерах с имитацией соответствующей периферии. Тестируемые данные выбираются с учетом как систематических, так и случайных ошибок аппаратных средств. Например, общее управление тестированием гарантирует профиль тестируемых данных, тогда как случайный выбор тестируемых данных может управлять отдельными тестовыми примерами более детально.

Как указано выше, индивидуальные средства для тестирования, выполнение тестирования и управление тестированием определяются детализированными целями тестирования. Другие важные условия задаются математическими предпосылками, которые должны быть соблюдены, если оценка тестирования удовлетворяет заданным целям тестирования.

Из опыта эксплуатации также могут быть получены вероятностные представления поведения любого тестируемого объекта. Если соблюдаются одинаковые условия, то к оценкам результатов тестирования может быть применен одинаковый математический аппарат.

Используя эти методы, достаточно сложно продемонстрировать на практике сверхвысокие уровни надежности.

Литература:

Software Testinga via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CI ID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

Validation of ultra high dependability for software based systems. B. Littlewood and L Strigini. Comm. ACM 36 (11), 69—80, 1993.

Handbook of Software Reliability Engineering. M. R. Lyu (ed.). IEEE Computer Society Press, McGraw-Hill, 1995, ISBN 0-07-039400-8.

С.5.2 Регистрация и анализ данных

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.5 и А.8).

Цель: документирование всех данных, решений и разумного обоснования программных проектов с целью обеспечения верификации, подтверждения соответствия, оценки и эксплуатации.

Описание: в процессе всего проектирования разрабатывается подробная документация, в которую входят:

- тестирование, выполняемое на каждом программном модуле;
- решения и их разумные обоснования;
- проблемы и их решения.

В процессе и по завершении проекта эта документация может быть проанализирована на наличие широкого набора информации. В частности, такая информация, использовавшаяся в качестве обоснования при принятии конкретных решений в процессе разработки проекта и очень важная для обслуживания вычислительных систем, не всегда известна инженерам по эксплуатации.

Литература:

Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989, ISBN 1-85166-381-9.

С.5.3 Тестирование интерфейса

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.5).

Цель: обнаружение ошибок в интерфейсах подпрограмм.

Описание: возможны несколько уровней детализации или полноты тестирования. К наиболее важным уровням относится тестирование:

- всех интерфейсных переменных с их предельными значениями;
- всех отдельных интерфейсных переменных с их предельными значениями с другими интерфейсными переменными с их нормальными значениями;
- всех значений предметной области каждой интерфейсной переменной с другими интерфейсными переменными с их нормальными значениями;
- всех значений всех переменных в разных комбинациях (возможно только для небольших интерфейсов);
- при специфицированных условиях тестирования, уместных для каждого вызова каждой подпрограммы.

Эти тестирования особенно важны, если интерфейсы не имеют возможности обнаруживать неправильные значения параметров. Такие тестирования также важны при генерации новых конфигураций ранее существовавших подпрограмм.

С.5.4 Анализ граничных значений

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы В.2, В.3 и В.8).

Цель: обнаружение программных ошибок при предельных и граничных значениях параметров.

Описание: предметная входная область программы разделяется на множество входных классов в соответствии с отношениями эквивалентности (см. С.5.7). Тестирование должно охватывать границы и экстремальные значения классов. Данное тестирование проверяет совпадение границы предметной входной области в спецификации с границами, установленными программой. Использование нулевого значения как в непосредственных, так и в косвенных преобразованиях часто приводит к ошибкам. Особого внимания требуют:

- нулевой делитель;
- знаки пробела ASCII;
- пустой стек или элемент списка;
- заполненная матрица;
- ввод нулевой таблицы.

Обычно границы входных значений напрямую соотносятся с границами выходных значений. Для установления выходных параметров в их предельные значения необходимо записывать специальные тестовые примеры. Следует также по возможности рассмотреть спецификацию такого тестового примера, который побуждает выходное значение превысить установленные спецификацией граничные значения.

Если выходные значения являются последовательностью данных, например таблица, то особое внимание следует уделить первому и последнему элементам, а также спискам, содержащим либо ни одного, либо один, либо два элемента [19].

Литература:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

С.5.5 Предположение ошибок

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы В.2 и В.8).

Цель: исключение ошибки программирования.

Описание: опыт тестирования и интуиция в сочетании со сведениями и заинтересованностью относительно тестируемой системы могут добавить некоторые неклассифицированные тестовые примеры к набору заданных тестовых примеров.

Специальные значения или комбинации значений могут быть подвержены ошибкам. Некоторые вызывающие интерес тестовые примеры могут быть получены из анализа контрольных списков. Следует также рассмотреть, является ли система достаточно устойчивой. Например, следует ли нажимать клавиши на передней панели слишком быстро или слишком часто. Что произойдет, если две клавиши нажать одновременно.

Литература:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

С.5.6 Введение ошибок

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.2).

Цель: подтверждение адекватности набора тестовых примеров.

Описание: некоторые известные типы ошибок вводятся (подсеиваются) в программу, и программа выполняется с тестовыми примерами в режиме тестирования. При обнаружении только некоторых подсеянных ошибок тестовый пример становится неадекватным. Отношение числа найденных подсеянных ошибок к общему числу подсеянных ошибок оценивается как отношение числа найденных реальных ошибок к общему числу реальных ошибок. Это дает возможность оценить число остаточных ошибок и, тем самым, остальную работу по тестированию.

$$\frac{\text{Найденные подсеянные ошибки}}{\text{Общее число подсеянных ошибок}} = \frac{\text{Найденные реальные ошибки}}{\text{Общее число реальных ошибок}}.$$

Обнаружение всех подсеянных ошибок может указывать либо на адекватность тестового примера, либо на то, что подсеянные ошибки было слишком легко найти. Ограничениями данного метода являются: порядок получения любых полезных результатов, типы ошибок. Также необходимо, чтобы позиции подсеивания отражали статистическое распределение реальных ошибок.

Литература:

Software Fault Injection, J. M. Voas и G. McGraw, Wiley 1998.

С.5.7 Разделение входных данных на классы эквивалентности

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы В.2 и В.3).

Цель: адекватное тестирование программных средств с использованием минимума тестируемых данных. Тестируемые данные образуются путем выбора частей входных данных предметной области, требующихся для анализа программных средств.

Описание: данный метод тестирования основывается на отношении эквивалентности входных данных, определяющем разбиение входных данных предметной области.

Тестовые примеры выбираются с целью охвата всех предварительно специфицированных разбиений. Из каждого класса эквивалентности выбирается, по меньшей мере, один тестовый пример.

Существуют следующие основные возможности разбиения входных данных:

- классы эквивалентности, образованные из спецификации, — интерпретация спецификации может быть ориентирована либо на вход, например выбранные значения считаются одинаковыми, либо ориентирована на выход, например, набор значений приводит к одному и тому же функциональному результату;
- классы эквивалентности, образованные в соответствии с внутренней структурой программы, — результаты класса эквивалентности определяются из статического анализа программ, например, набор значений обрабатывается одним и тем же способом.

Литература:

The Art of Software Testing. G. Myers, Wiley & Sons, New York, 1979.

С.5.8 Структурное тестирование

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.2).

Цель: применение тестов, анализирующих определенные подмножества структуры программы.

Описание: на основе анализа программы выбирается набор входных данных так, чтобы мог быть проанализирован достаточно большой (часто с заранее заданным назначением) процент программных кодов. Средства охвата программы, в зависимости от степени требуемой строгости, могут быть различными:

- утверждения — это наименее строгий тест, поскольку можно выполнить все закодированные утверждения без анализа обеих ветвей условного утверждения;
- ветвления — обе стороны каждой ветви следует проверять. Это может оказаться непрактичным для некоторых типов кодов защиты;
- составные условия — анализируется каждое условие в составной условной ветви (связанное оператором И/ИЛИ). См. MCDC (охват условного модифицированного решения, документ DO178B);
- LCSAJ — последовательность линейного кода и переходов представляет собой любую линейную последовательность закодированных утверждений, включая условные утверждения, заканчивающиеся переходом. Многие потенциальные подпоследовательности могут оказаться невыполнимыми благодаря ограничениям, которые налагаются на входные данные в результате выполнения предыдущего кода;
- поток данных — выполняющиеся последовательности выбираются на основе используемых данных; например последовательность, где одна и та же переменная и записывается, и считывается;
- граф вызовов — программа, состоящая из подпрограмм, которые могут быть вызваны из других подпрограмм. Этот граф вызовов представляет собой дерево вызовов подпрограмм в программе. Тесты должны охватывать все вызовы в дереве;
- базовая последовательность — одна из минимального набора конечных последовательностей от начала до конца, когда все дуги охвачены (перекрывающиеся комбинации последовательностей в этом базовом наборе могут сформировать любую последовательность через эту часть программы). Тесты всех базовых последовательностей показали свою эффективность при обнаружении ошибок.

Литература:

Reliability of the Path Analysis Testing Strategy. W. Howden. IEEE Trans Software Engineering, Vol. SE-3, 1976.
Software considerations in airborne systems and equip certification, DO178B, RTCA, December 1992.
Structure testing, McCabe; NBS Special Publication 500-99, 1982.
A software reliability study, Walsh [USA] National Computer Conference, 1979.

С.5.9 Анализ потоков управления

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.8).

Цель: обнаружение низкокачественных и потенциально некорректных структур программ.

Описание: анализ потока управления представляет собой метод статического тестирования для нахождения подозреваемых областей программы, которые не соответствуют оправдавшей себя практике программирования. Программа анализируется, формируя направленный граф, который может быть проанализирован на наличие:

- недоступных фрагментов программы, например безусловных переходов, которые делают фрагменты программы недостижимыми;
- запутанных кодов. Хорошо структурированный код имеет управляющий граф, допускающий сокращение путем последовательного сокращения графа до одного узла. В отличие от этого плохо структурированный код может быть сокращен только до группы, состоящей из нескольких узлов.

Литература:

Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACM Trans. on Prog. Lang. and Syst., 1985.

С.5.10 Анализ потока данных

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.8).

Цель: обнаружение низкокачественных и потенциально некорректных структур программ.

Описание: анализ потока данных представляет собой метод статического тестирования, объединяющий информацию, полученную из анализа потока управления, с информацией о том, какие переменные считываются или записываются в различных частях кода. Данный метод может проверять:

- переменные, которые могут быть считаны до присвоения им значений. Такую ситуацию можно исключить, если всегда присваивать значение при объявлении новой переменной;
- переменные, записанные несколько раз, но не считанные. Такая ситуация может указывать на пропущенный код;
- переменные, которые записаны, но никогда не считываются. Такая ситуация может указывать избыточный код.

Аномальный поток данных не всегда непосредственно соответствует программным ошибкам, но если аномалии исключены, то маловероятно, что код будет содержать ошибки.

Литература:

Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACM Trans. on Prog. Lang. and Syst., 1985.

С.5.11 Выявление скрытых схем исполнения

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.8).

Цель: обнаружение неожидаемых путей или логических потоков в системе, в конкретных условиях инициирующих нежелательные или запрещающих необходимые функции.

Описание: путь паразитной схемы может содержать аппаратные, программные средства, операторы действий или комбинации этих элементов. Паразитные схемы не являются результатом неисправностей аппаратных средств, а представляют собой скрытые условия невнимательного проектирования системы или кодирования прикладных программ, что при определенных условиях может привести к неправильному функционированию системы.

Паразитные схемы разделяют на следующие категории:

- паразитные пути, вызывающие потоки тока, энергии или логических последовательней по неожидаемому пути или в незаданном направлении;
- паразитная синхронизация, при которой события происходят в неожидаемой или противоречивой последовательности;
- паразитная индикация, вызывающая неоднозначные или ложные изображения условий эксплуатации системы, что может привести к нежелательным действиям оператора;
- паразитные метки, некорректно или неточно размечающие системные функции, например системные входы, управления, изображения, шины и т. д., что может ввести в заблуждение оператора, который может выполнить в системе некорректные действия.

Анализ паразитных схем основывается на распознавании базовых топологических комбинаций в аппаратной или программной структуре (например, шесть базовых комбинаций были предложены для программных средств). Анализ осуществляется с помощью контрольного списка вопросов об использовании базовых топологических компонентов и отношениях между ними.

Литература:

Sneak Analysis and Software Sneak Analysis. S. G. Godoy and G. J. Engels. J. Aircraft Vol. 15, No. 8, 1978.

Sneak Circuit Analysis. J. P. Rankin, Nuclear Safety, Vol. 14, No. 5, 1973.

С.5.12 Тестирование на символьном уровне

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.8).

Цель: показать соответствие между исходным кодом и спецификацией.

Описание: переменные программы оцениваются после замены во всех операторах присваивания левой его части на правую. Условные ветви и циклы преобразуются в булевские выражения. Окончательный результат представляет собой символьное выражение для каждой переменной программы. Оно может быть проверено относительно предполагаемого символьного выражения.

Литература:

Formal Program Verification using Symbolic Execution. R. B. Dannenberg and G. W. Ernst. IEEE Transactions on Software Engineering, Vol. SE-8, No. 1, 1982.

Symbolic Execution and Software Testing. J. C King, Communications of ACM, Vol. 19, No. 7, 1976.

С.5.13 Формальное доказательство

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.9).

Цель: подтверждение корректности программ или спецификаций без их исполнения, используя теоретические и математические модели и правила.

Описание: ряд утверждений устанавливается в различных точках программы, и они используются в качестве предусловий и постусловий для различных путей программы. Доказательство демонстрирует, что программа преобразует предусловия в постусловия в соответствии с набором логических правил и завершается.

В настоящем стандарте описаны различные формальные методы, например CCS, CSP, HOL, LOTOS, OBJ, временная логика, VDM и Z (см. С.2.4).

Альтернативным методу формального доказательства является «строгий аргумент». Подготавливается процедура формального доказательства, в которой представлены основные этапы, но включены не все математические подробности. Метод «строгий аргумент» является более слабым методом верификации, устанавливающим, что доказательство было бы возможным, если бы к этому были предприняты попытки.

Литература:

Software Development — A Rigorous Approach. C. B. Jones. Prentice-Hall, 1980.

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall, 2nd Edition, 1990.

С.5.14 Метрики сложности программного обеспечения

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.9 и А.10).

Цель: прогнозирование характеристик программ, исходя из свойств самих программ или их разработки, либо предысторий тестирования.

Описание: данные методы оценивают некоторые структурные свойства программных средств и их отношения к требуемым характеристикам, например надежность или сложность. Для оценки большинства средств требуются программные инструменты. Некоторые применяющиеся метрики перечислены ниже:

- теоретическая сложность графа. Эта метрика может быть применена на раннем этапе жизненного цикла для оценки компромиссных решений и основана на величине сложности графа управления программы, представленной ее цикломатическим числом;
- число способов активизации некоторых программных модулей (доступность) – чем больше программных модулей может быть доступно, тем должна быть большая вероятность их отладки;
- теория метрик Холстеда. При помощи этих средств вычисляют длину программы путем подсчета числа операторов и операндов; данная метрика дает меру сложности и размеры, которые формируют основу для сравнений при оценке будущих разрабатываемых ресурсов;
- число входов и выходов на программный модуль. Сведение к минимуму числа точек входов/выходов является ключевой особенностью методов структурного проектирования и программирования.

Литература:

Software Metrics: A Rigorous and Practical Approach. N. E. Fenton, International Thomson Computer Press, 1996, ISBN 1-85032-275-9, 2nd Edition.

A Complexity Measure. T. J. McCabe. IEEE Trans on Software Engineering, Vol. SE-2, No. 4, December 1976.

Models and Measurements for Quality Assessments of Software. S. N. Mohanty. ACM Computing Surveys, Vol. 11, No. 3, September 1979.

Elements of Software Science. M. H. Halstead. Elsevier, North Holland, New York, 1977.

С.5.15 Проверка разработки программ

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.8).

Цель: обнаружение ошибок на всех этапах разработки программ.

Описание: «формальный» аудит гарантирующих качество документов, направленный на отыскание ошибок. Процедура проверки состоит из пяти этапов: планирование, подготовка, исследование, анализ и учет. Каждый из этих этапов имеет свои конкретные цели. Должна быть проанализирована вся разработка системы (спецификация, проектирование, кодирование и тестирование).

Литература:

Design and Code Inspections to Reduce Errors in Program Development. M. E. Fagan, IBM Systems Journal, No. 3, 1976.

С.5.16 Сквозной контроль/Анализ проектов

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.8).

Цель: обнаружение ошибок в различных частях проекта с высокой оперативностью и экономичностью.

Описание: МЭК опубликовал руководство по общему представлению формального анализа проектов, которое содержит общее описание представления формального анализа проектов, его цели, подробные сведения о различных типах анализа проекта, состав группы анализа проекта и относящиеся к ним обязанности и ответственности. Это руководство содержит также общие руководящие материалы по планированию и выполнению формального анализа проектов, а также конкретные подробные сведения, относящиеся к роли независимых специалистов в группе по анализу проекта [12]. Например, помимо прочего в функции специалистов входят надежность, поддержка обслуживания и доступность.

В упомянутом выше руководстве МЭК рекомендует, чтобы формальный анализ проекта проводился для всех новых изделий/процессов, применений и при пересмотрах существующих изделий и производственных процессов, влияющих на функции, производительность, безопасность, надежность, способность анализировать обслуживание, доступность, способность к экономичности и другие характеристики, влияющие на конечные изделия/процессы, пользователей или наблюдателей.

Закодированный сквозной контроль состоит из группы сквозного контроля, выбирающей небольшой набор изложенных на бумаге тестовых примеров, представляющих наборы входных данных и соответствующие предполагаемые выходы для программы. После этого тестовые данные вручную трассируются через логику программы.

Литература:

Software Inspection. T. Gilb, D. Graham, Addison-Wesley, 1993, ISBN 0-201-63181-4.

С.5.17 Макетирование/анимация

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы В.3 и В.5).

Цель: проверка возможности реализации системы при наличии заданных ограничений. Увязка интерпретации разработчика спецификации системы с ее потребителем для исключения непонимания между ними.

Описание: выделяются подмножество системных функций, ограничения и требования к рабочим параметрам. С помощью инструментов высокого уровня строится макет. На данном этапе не требуется рассматривать ограничения, например используемый компьютер, язык реализации, объем программ, обслуживание, надежность и доступность. Макет оценивается по критериям потребителя, и системные требования могут быть модифицированы в свете этой оценки.

Литература:

The emergence of rapid prototyping as a real-time software development tool. J. E. Cooling, T. S. Hughes, Proc. 2nd Int. Conf. on Software Engineering for Real-time Syatems, Cirencester, UK, IEE, 1989.

Software evolution through rapid prototyping. Luqi, IEEE Computer 22 (5), 13—27, May 1989.

Approaches to Prototyping. R. Budde et al, Springer Verlag, 1984, ISBN 3-540-13490-5.

Proc. Working Conference on Prototyping. Namur, October 1983, Budde et al, Springer Verlag, 1984.

Using an executable specification language for an information system. S. Urban et al. IEEE Trans Software Engineering, Vol. SE-11 No. 7, July 1985.

С.5.18 Моделирование процесса

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.3).

Цель: тестирование функции программной системы вместе с ее интерфейсами во внешнем окружении, не допуская модификации реального окружения.

Описание: создание системы только для целей тестирования, имитирующей поведение управляемого оборудования (EUC).

Имитация может осуществляться только программными средствами либо сочетанием программных и аппаратных средств. Она должна:

- обеспечить входы, эквивалентные входам, которые могут иметь место при фактической установке EUC;
- реагировать на выходные результаты тестирования программных средств способом, точно отражающим объект управления;
- обладать средствами для входных данных оператора, обеспечивающими любые нарушения, с которыми должна справиться тестируемая система.

Когда программные средства протестированы, созданная система может тестировать аппаратные средства с их входами и выходами.

Литература:

Software Testing via Environmental Simulation (CONTESSE Report). Available until December 1998 from: Ray Browne, CIID, DTI, 151 Buckingham Palace Road, London, SW1W 9SS, UK, 1994.

С.5.19 Требования к реализации

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.6).

Цель: установление демонстрируемых требований к функционированию системы программных средств.

Описание: выполняется анализ как системы, так и спецификаций требований программного обеспечения с целью спецификации всех общих и конкретных, явных и неявных требований к функционированию.

Каждое требование к функционированию анализируется по очереди для того, чтобы определить:

- критерии успешности результата, который следует получить;
- возможность получения меры критерия успешности;
- потенциальную точность таких результатов измерения;
- этапы проектирования, на которых эти результаты измерения могут быть оценены и
- этапы проектирования, на которых могут быть получены эти результаты измерений.

Затем анализируется целесообразность каждого требования к функционированию для получения списка требований к функционированию, критериев успешности результата и возможных результатов измерений. Основными целями являются:

- связь каждого требования к функционированию, по крайней мере, с одной мерой;
- выбор (где это возможно) точных и эффективных мер, которые могут быть использованы на самых ранних стадиях разработки;
- спецификация важных и факультативных требований к функционированию и критериев успешности результата;
- использование (по возможности) преимуществ применения одной меры для нескольких требований к функционированию.

С.5.20 Моделирование реализации

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.5, В.2 и В.5).

Цель: достаточная для удовлетворения специфицированных требований рабочая производительность системы.

Описание: спецификация требований включает в себя требования к пропускной способности и реакции конкретных функций, возможно, объединенных с ограничениями на использование общих системных ресурсов. Предложенный проект системы сравнивается с установленными требованиями путем:

- создания модели процессов системы и их взаимодействий;
- определения, используемых каждым процессом, ресурсов (время процессора, полоса пропускания канала связи, объем памяти и т. п.);
- определения распределения запросов, выдаваемых системе при средних и наихудших условиях;
- вычисления средних и наихудших случаев значений величин пропускной способности и времени ответа для конкретных функций системы.

Для простых систем может оказаться достаточным аналитическое решение, тогда как для более сложных систем более подходящей для получения точных результатов является создание модели системы.

Перед детальным моделированием может быть использована более простая проверка «бюджета ресурсов», которая суммирует требования к ресурсам всех процессов. Если сумма этих требований к системе превышает возможности спроектированной системы, проект считается нереализуемым. Даже в случае, если проект проходит эту простую проверку, моделирование выполнения может показать, что слишком большие задержки и времена ответов происходят из-за недостатка ресурсов. Для исключения такой ситуации инженеры часто проектируют системы, использующие только часть (например 50 %) общих ресурсов для уменьшения вероятности нехватки ресурсов.

Литература:

The Design of Real-time Systems: From Specification to Implementation and Verification. H. Kopetz et al, Software Engineering Journal 72 — 82, 1991.

С.5.21 Проверка на критические и напряженные нагрузки

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.6).

Цель: подвергнуть тестируемый объект исключительно высокой нагрузке, чтобы показать, что тестируемый объект будет легко выдерживать нормальную рабочую нагрузку.

Описание: существует множество тестов для проверки на критические и напряженные нагрузки, например:

- если работа объекта происходит в режиме упорядоченного опроса, то объект тестирования подвергается в единицу времени гораздо большим входным изменениям, чем при нормальных условиях;
- если работа объекта происходит по запросам, то число запросов в единицу времени для тестируемого объекта увеличивается относительно нормальных условий;

- если объем базы данных играет важную роль, то этот объем увеличивается относительно ее объема при нормальных условиях;

- имеющие решающее влияние устройства настраиваются на свои максимальные скорости или самые малые скорости соответственно;

- для экстремальных тестов все факторы, имеющие решающее влияние, по возможности вводятся одновременно в граничные условия.

Для указанных выше тестов может быть оценено поведение во времени тестируемого объекта. Можно также исследовать изменения нагрузки и проверить размер внутренних буферов или динамических переменных, стеков и т. п.

С.5.22 Ограничения на время реакции и объем памяти

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.6).

Цель: обеспечение соответствия системы требованиям к параметрам времени и памяти.

Описание: спецификация требований к системе и программному обеспечению включает в себя требования к памяти и времени выполнения системой конкретных функций, возможно, объединенных с ограничениями на использование общих системных ресурсов.

Данный метод выполняется для определения распределения запросов при средних и наихудших условиях. Рассматриваемый метод требует оценки используемых ресурсов и затраченного времени каждой функцией системы. Такие оценки могут быть получены различными способами, например сравнением с существующей системой или макетированием и дальнейшим сравнением времени реакции с критическими системами.

С.5.23 Анализ влияния

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.8).

Цель: определение влияния, изменяющего или расширяющего программную систему, которому могут подвергаться также и другие программные модули в данной программной системе, а также другие системы.

Описание: перед выполнением модификации или расширения программного обеспечения следует определить влияние модификаций или расширений на программное обеспечение, а также определить, на какие программные системы и программные модули это повлияет.

Далее принимается решение о повторной верификации программной системы. Это зависит от числа, подвергнувшихся воздействию программных модулей, их критичности и характера изменений. Возможными решениями могут быть:

- повторная проверка только изменений программного модуля;
- повторная проверка всх, подвергнувшихся воздействию, программных модулей;
- повторная проверка всей системы.

Литература:

Dependability of Critical Computer Systems 2. F. J. Redmill, Elsevier Applied Science, 1989. ISBN 1-85166-381-9.

С.5.24 Управление конфигурацией программного обеспечения

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.8).

Цель: обеспечение согласованности результатов работы групп поставщиков проекта, а также изменений в этих поставках. В общем случае управление конфигурацией применимо к разработке как аппаратных, так и программных средств.

Описание: управление конфигурацией программных средств представляет собой метод, используемый в течение всей разработки. В сущности он требует документирования разработки каждой версии каждой значимой ее поставки и каждой взаимосвязи между различными версиями разработки различных поставщиков. Полученная документация позволяет разработчику определять, как влияет на другие поставки изменение в первой поставке (особенно одного из его компонентов). В частности, системы или подсистемы могут надежно компоноваться (конфигурироваться) из согласованных наборов версий компонентов.

Литература:

Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs. MIL-STD-483.

Software Configuration Management. J. K. Buckle. Macmillan Press, 1982. Software Configuration Management. W. A. Babich. Addison-Wesley, 1986.

Configuration Management Requirements for Defence Equipment. UK Ministry of Defence Standards 05-57 Issue 3, July 1993.

С.6 Оценка функциональной безопасности

Примечание — Соответствующие методы и средства см. также в В.6.

С.6.1 Таблицы решений и таблицы истинности

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблицы А.10 и В.7).

Цель: обеспечение ясных и согласующихся спецификаций и анализа сложных логических комбинаций и их отношений.

Описание: данный метод использует бинарные таблицы для точного описания логических отношений между булевыми переменными программы.

Использование таблиц и точность метода позволили применить его в качестве средства анализа сложных логических комбинаций, выраженных в бинарных кодах.

Рассматриваемый метод достаточно легко автоматизируется, поэтому его можно использовать в качестве средства спецификации систем.

С.6.2 Исследование опасности и работоспособности (HAZOP)

Цель: определение угрозы безопасности в предлагаемой или существующей системе, их возможных причин и последствий и рекомендуемых действий по минимизации вероятности их появления.

Описание: группа специалистов в области создаваемой системы принимает участие в структурном анализе проекта системы путем ряда запланированных совещаний. Они рассматривают как реализацию функций проекта системы, так и способы работы системы на практике (включая действия персонала и процедуры эксплуатации системы). Руководитель группы специалистов инициирует ее участников создавать потенциальные опасности и управляет этой процедурой, описывая каждую часть системы в сочетании с отдельными ключевыми словами: «отсутствует», «более», «менее», «часть целого», «больше чем» (или «так же как и») и «иначе чем». Каждое применимое условие или режим отказа рассматривается с точки зрения реализуемости, причин возникновения, возможных последствий (появляется ли опасность), способа устранения и, в случае устранения, выбора наиболее целесообразного метода.

Затем часто возникает необходимость провести дальнейшее исследование опасностей (методом вероятностной или количественной оценки риска) с целью их более подробного рассмотрения.

Исследование опасностей может выполняться на разных стадиях разработки проекта, однако наиболее эффективным такое исследование может быть на начальных стадиях с тем, чтобы как можно раньше повлиять на основные решения по проектированию и работоспособности. Полезно в графике выполнения проекта определить фиксированное время для совещаний длительностью не менее половины дня и не более четырех раз в неделю с тем, чтобы рассматривать весь поток сопроводительной документации. Сопроводительная документация, выработанная на совещаниях, должна составлять существенную часть досье об опасности/безопасности системы.

Метод HAZOP создавался для производственных процессов и без модификации его сложно применить к программным элементам PES. Были предложены различные производные методы PES HAZOP (или Computer HAZOPs — «CHAZOPs»), которые сопровождались новыми руководящими материалами и/или реализовывали способы систематического охвата системной и программной архитектур.

Литература:

Draft Interim Defence Standard 00-58/1: «A Guide to HAZOP Studies on Systems which Incorporate a Programmable Electronic System». Ministry of Defence (UK). March 1995.

Hazard and Operability (HAZOP) studies applied to computer-controlled process plants. P. Chung and E. Broomfield. In «Computer Control and Human Error by T. Kletz, Institution of Chemical Engineers, 165—189 Railway Terrace, Rugby, CV1 3HQ, UK, 1995, ISBN 0-85295-362-3.

Reliability and Hazard Criteria for Programmable Electronic Systems in Chemical Industry. E. Johnson. Proc. of Safety and Reliability of PES, PES 3 Safety Symposium, B. K. Daniels (ed.), 28—30 May 1986, Guernsey Channel Islands, Elsevier Applied Science, 1986.

HAZOP and HAZAN. T. A. Kletz. Institution of Chemical Engineers, 165 — 189 Railway Terrace, Rugby, CV1 3HQ, UK, 3rd Edition, 1992, ISBN 0-85295-285-6.

A Guide to HAZOPS. Chemical Industries Association Ltd, 1977.

Reliability Engineering and Risk Assessment. E. J. Henlty and H. Kumamoto, Prentice-Hall, 1981.

Systems Reliability and Risk Analysis (Engineering Application of Systems Reliability and Risk Analysis), E. G. Frenkel, Kluwer Academic Pub., May 1988, ISBN 90-2473-665X.

Control Hazard Studies for Process Plants. K. Walters, in Integrated Risk Assessment-Current Practice and New Directions, edited by R. E. Melchers and M. G. Stewart, The University of Newcastle, NSW Australia. A. A. Balkema Publishers, Rotterdam Netherlands 1995, ISBN 90-5410-5550.

С.6.3 Анализ отказов по общей причине

Примечания

1 Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.10).

2 См. также МЭК 61508-6 (приложение D).

Цель: определение возможных отказов в нескольких системах или нескольких подсистемах, которые могут свести к нулю преимущества избыточности из-за одновременного появления одних и тех же отказов во многих частях системы.

Описание: системы, ориентированные на безопасность объекта, часто используют избыточность аппаратных средств и мажоритарный принцип голосования. Этот подход исключает случайные отказы в компонентах или подсистемах аппаратных средств, которые могут помешать корректной обработке данных.

Однако некоторые отказы могут оказаться общими для нескольких компонентов или подсистем. Например, если система установлена в одном помещении, то недостатки вентиляции могут снизить преимущества избы-

точности. Это может оказаться верным и для других внешних влияний на систему (например пожар, затопление, электромагнитные влияния, трещины в панелях и землетрясение). Система может быть также подвержена воздействиям, относящимся к ее функционированию и эксплуатации. Поэтому важно, чтобы в рабочих инструкциях были предусмотрены адекватные и хорошо задокументированные процедуры по функционированию и эксплуатации системы, а обслуживающий персонал был хорошо обучен.

Внутренние причины также вносят большой вклад в общее число отказов. Их основой могут являться ошибки проектирования общих или идентичных компонентов и их интерфейсов, в том числе и устаревших компонентов. Анализ отказов по общей причине должен отыскивать также общие дефекты в системе. К методам анализа отказов по общей причине относятся: общее управление качеством; анализ проектов; верификация и тестирование независимой группой; анализ реальных ситуаций, полученных из опыта работы аналогичных систем. Однако область применения такого анализа выходит за рамки только аппаратных средств. Даже если разные программы используются в разных каналах избыточных систем, возможна некоторая общность в программных подходах, которая может привести к росту отказов по общей причине (например ошибки в общей спецификации).

Если отказы по общей причине не появляются точно в одно и то же время, то должны быть предприняты меры предосторожности путем сравнения методов, применяемых в различных каналах. При этом применение каждого метода должно обнаруживать отказ до того, как он окажется общим для всех каналов. Анализ отказов по общей причине должен использовать этот подход.

Литература:

Review of Common Cause Failures. I. A. Watson, UKAEA, Centre for Systems Reliability, Wigshaw Lane, WA3 4NE, England, NCSR R 27, July 1981.

Common-Mode Failures. in Redundancy Systems. I. A. Watson and G. T. Edwards. Nuclear Technology Vol. 46, December 1979.

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office, London, 1987.

С.6.4 Модели Маркова

Примечание — Краткое сравнение данного метода с методом, основанным на блок-схемах надежности, при анализе полноты безопасности аппаратных средств см. в МЭК 61508-6 (приложение В.1).

Цель: оценка надежности, безопасности и доступности систем.

Описание: строится граф системы, представляющий состояния системы, связанные с ее отказами (состояния отказов представляются узлами графов). Связи между узлами, представляющие собой события-отказы или события-восстановления, имеют весовые коэффициенты, соответствующие частотам отказов или частотам восстановлений. Предполагается, что переход из состояния N в последующее состояние $N + 1$ не зависит от предыдущего состояния $N - 1$. Следует заметить, что события, состояния и частоты отказов могут быть детализированы так, что может быть получено точное описание системы, например обнаруженные или необнаруженные отказы, обнаружение наибольшего отказа и т. п.

Метод Маркова подходит для моделирования многих систем, уровень избыточности которых изменяется со временем вследствие нахождения компонента в состоянии отказа или восстановления [15]. Другие классические методы, например FMEA и FTA, не могут быть адаптированы к моделированию влияний отказов в течение жизненного цикла системы, поскольку не существует простой комбинаторной формулы для вычисления соответствующих вероятностей.

В простейших случаях такую формулу, описывающую вероятности системы, можно найти в литературе или вывести самостоятельно. В более сложных случаях существуют методы упрощения (то есть сокращение числа состояний). Для очень сложных случаев результаты могут быть вычислены с помощью моделирования графа на компьютере.

Литература:

The Theory of Stochastic Processes. R. E. Cox and H. D. Miller, Methuen and Co. Ltd., London, UK, 1963.

Finite MARKOV Chains. J. G. Kemeny and J. L. Snell. D. Van Nostrand Company Inc, Princeton, 1959.

Reliability Handbook. B. A. Koslov and L. A. Usnakov, Holt Rinehart and Winston Inc, New York, 1970.

The Theory and Practice of Reliable System Design. D. P. Siewiorek and R. S. Swarz, Digital Press, 1982.

С.6.5 Структурные схемы надежности

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица А.10) и метод также использован в МЭК 61508-6 (приложение В).

Цель: моделирование в форме диаграмм набора событий, которые должны происходить, и условий, которые должны быть удовлетворены, для успешного выполнения операций системы или задач.

Описание: данный метод позволяет сформировать успешный маршрут, состоящий из блоков, линий и логических переходов. Такой успешный маршрут начинается от одной стороны диаграммы и проходит через блоки и логические переходы до другой стороны диаграммы. Блок представляет собой условие или событие, маршрут проходит через него, если условие истинно или событие произошло. Когда маршрут подходит к логическому переходу, то он продолжается, если критерий логического перехода выполняется. Если маршрут достигает какой-либо

вершины, то он может продолжаться по всем исходящим из нее путям. Если существует, по меньшей мере, один успешный маршрут через всю диаграмму, то цель анализа считается достигнутой [10].

Литература:

System Reliability Engineering Methodology: A Division of the State of the Art. J. B. Fussel and J. S. Arend, Nuclear Safety 20 (5), 1979.

Fault Tree Handbook. W. E. Vesely et al, NUREG-0942, Division of System Safety Office at Nuclear Reactor Regulation, US Nuclear Regulatory Commission, Washington, DC 20555, 1981.

С.6.6 Моделирование методом Монте-Карло

Примечание — Ссылка на данный метод/средство приведена в МЭК 61508-3 (таблица В.4).

Цель: моделирование ситуаций реального мира с помощью программных средств методом генерации случайных чисел.

Описание: моделирование методом Монте-Карло используется для решения двух классов проблем:

- вероятностных, в которых для генерации стохастических ситуаций используются случайные числа;
- детерминистических, которые математически преобразуются в эквивалентную вероятностную форму.

Метод Монте-Карло формирует потоки случайных чисел с тем, чтобы генерировать шум при анализе сигналов или добавлять их в случайные смещения или допуски.

Данный метод гарантированно обеспечивает нахождение смещений, допусков или шума в приемлемых диапазонах.

Общие принципы моделирования методом Монте-Карло заключаются в переформулировании проблемы так, чтобы полученные результаты были как можно более точными, что позволяет отказаться от решения проблемы в ее исходной постановке.

Литература:

Monte Carlo Methods. J. M. Hammersley, D. C Handscomb, Chapman & Hall, 1979.

Приложение D
(справочное)

Вероятностный подход определения полноты безопасности предварительно разработанных программных средств

D.1 Общие положения

Настоящее приложение содержит исходные руководящие материалы по использованию вероятностного подхода к определению полноты безопасности программных средств для предварительно разработанных программ на основе их опыта эксплуатации. Вероятностный подход является наиболее подходящим для оценки операционных систем, библиотечных компонентов, компиляторов и других программных систем. Настоящее приложение также содержит описание возможностей вероятностного подхода, однако его следует использовать только специалистам, компетентным в статистическом анализе.

Примечание — В настоящем приложении используется термин «уровень доверия», который описан в IEEE 352:1987. Эквивалентный термин "уровень значимости" приведен в [14].

Предложенные в настоящем приложении методы могут быть также использованы для демонстрации роста уровня полноты безопасности программных средств, которые некоторое время успешно эксплуатировались. Например, программные средства, созданные в соответствии с требованиями МЭК 61508-3 для SIL1, после соответствующего периода успешной работы в большом числе применений могут продемонстрировать соответствие уровню полноты безопасности SIL2.

Число запросов без отказов при испытании или число часов, необходимое для работы без отказов, для определения конкретного уровня полноты безопасности представлено в таблице D.1. В таблице D.1 также обобщены результаты, приведенные в D.2.1 и D.2.3.

Опыт эксплуатации может быть выражен математически, как показано в D.2, для дополнения или замены статистического тестирования, а опыт эксплуатации, полученный из нескольких мест эксплуатации, может быть объединен (путем добавления конкретного числа обработанных запросов или часов работы в течение эксплуатации), но только в случае, если:

- программная версия, подлежащая использованию в системе E/E/PE, связанной с безопасностью, будет идентична версии, для которой предъявлен результат опыта ее эксплуатации;
- эксплуатационный профиль входного пространства очень близок друг другу;

- существует эффективная система уведомлений и документирования отказов;
- справедливы принятые в D.2 предположения.

Таблица D.1 — Необходимая предыстория для определения уровня полноты безопасности

SIL	Режим работы с низкой интенсивностью запросов (вероятность отказа при выполнении планируемых функций по запросу)	Число реальных запросов		Режим с высокой интенсивностью запросов или непрерывный режим работы (вероятность опасного отказа в час)	Общее число часов эксплуатации	
		$1-\alpha = 0{,}99$	$1-\alpha = 0{,}95$		$1-\alpha = 0{,}99$	$1-\alpha = 0{,}95$
4	$\geq 10^{-5}$ до $< 10^{-4}$	$4{,}6 \times 10^5$	3×10^5	$\geq 10^{-9}$ до $< 10^{-8}$	$4{,}6 \times 10^9$	3×10^9
3	$\geq 10^{-4}$ до $< 10^{-3}$	$4{,}6 \times 10^4$	3×10^4	$\geq 10^{-8}$ до $< 10^{-7}$	$4{,}6 \times 10^8$	3×10^8
2	$\geq 10^{-3}$ до $< 10^{-2}$	$4{,}6 \times 10^3$	3×10^3	$\geq 10^{-7}$ до $< 10^{-6}$	$4{,}6 \times 10^7$	3×10^7
1	$\geq 10^{-2}$ до $< 10^{-1}$	$4{,}6 \times 10^2$	3×10^2	$\geq 10^{-6}$ до $< 10^{-5}$	$4{,}6 \times 10^6$	3×10^6

Примечания

1 Величина $1-\alpha$ представляет собой уровень доверия.

2 Предпосылки и описание процедур получения числовых значений в настоящей таблице см. в D.2.1 и D.2.3.

D.2 Формулы статистического тестирования и примеры их использования

D.2.1 Простой статистический тест для режима работы с низкой интенсивностью запросов

D.2.1.1 Исходные предпосылки

a) Распределение тестовых данных равно распределению запросов при выполнении операций в режиме «онлайн».

b) Прохождения тестов статистически не зависят друг от друга в отношении причины отказа.

c) Для обнаружения любых отказов, которые могут появиться, существует адекватный механизм.

d) Число тестовых примеров $n > 100$.

e) Во время прогона n тестовых примеров отказы отсутствуют.

D.2.1.2 Результаты

Вероятность отказа p (на один запрос) при уровне доверия $1-\alpha$ определяется из выражения

$$p \leq 1 - \sqrt[n]{\alpha} \quad \text{или} \quad n \geq -\frac{\ln \alpha}{p}.$$

D.2.1.3 Пример

Таблица D.2 — Вероятности отказа режима работы с низкой интенсивностью запросов

$1-\alpha$	P
0,95	$3/n$
0,99	$4{,}6/n$

Для вероятности отказа при запросе для уровня полноты безопасности SIL3 при 95 %-ном уровне доверия применение указанной формулы дает 30000 тестовых примеров при выполнении условий принятых предпосылок. Результаты для каждого уровня полноты безопасности объединены в таблице D.1.

D.2.2 Тестирование входного массива (предметной области) для режима работы с низкой интенсивностью запросов

D.2.2.1 Исходные предпосылки

Единственная исходная предпосылка состоит в том, что тестируемые данные выбираются так, чтобы обеспечить случайное унифицированное распределение по входному массиву (предметной области).

D.2.2.2 Результаты

Необходимо определить число тестов n, которые требуются, исходя из порога точности δ, входов для тестируемой функции с низкой интенсивностью запросов (например безопасное отключение).

Таблица D.3 — Средние расстояния между двумя точками тестирования

Размер предметного пространства	Среднее расстояние между двумя точками тестирования в произвольном направлении
1	$\delta = 1/n$
2	$\delta = \sqrt[2]{1/n}$
3	$\delta = \sqrt[3]{1/n}$
K	$\delta = \sqrt[K]{1/n}$
Примечание — K может быть любым положительным целым числом. Значения 1, 2 и 3 приведены только в качестве примеров.	

D.2.2.3 *Пример*

Рассмотрим безопасное отключение, которое зависит только от двух переменных A и B. Если проверкой было установлено, что пороговые значения, которые разделяют входную пару переменных A и B, определены с точностью до 1 % от диапазона измерения A или B, то число равномерно распределенных тестовых примеров, требуемое в области A и B, будет равно

$$n = 1/\delta^2 = 10^4.$$

D.2.3 Простой статистический тест для режима с высокой интенсивностью запросов или непрерывного режима работы

D.2.3.1 Исходные предпосылки

a) Распределение данных такое же, как и распределение при выполнении операций в режиме «онлайн».

b) Относительное уменьшение вероятности отсутствия отказа пропорционально длительности рассматриваемого интервала времени и постоянно в противном случае.

c) Для обнаружения любых отказов, которые могут появиться, существует адекватный механизм.

d) Тест выполняется в течение времени тестирования t.

e) Во время тестирования t никаких отказов не происходит.

D.2.3.2 Результаты

Соотношение между интенсивностью отказов λ, уровнем доверия $1 - \alpha$ и временем тестирования t имеет вид

$$\lambda = -\frac{\ln\alpha}{t}.$$

Интенсивность отказов обратно пропорциональна среднему времени наработки на отказ:

$$\lambda = \frac{1}{MTBF}.$$

Примечание — Настоящий стандарт не делает различий между интенсивностью отказов в час и частотой отказов в час. Строго говоря, вероятность отказа F связана с частотой отказов f выражением $F = 1 - e^{-ft}$, однако область применения настоящего стандарта охватывает частоту отказов менее 10^{-5} 1/ч, а для небольших значений частоты справедливо $F \cong f \cdot t$.

D.2.3.3 Пример

Таблица D.4 — Вероятности отказа для режима с высокой интенсивностью запросов или непрерывного режима работы

$1 - \alpha$	γ
0,95	$3t$
0,99	$4,6/t$

Для подтверждения того, что среднее время наработки на отказ составляет, по меньшей мере, 10^8 ч с уровнем доверия 95 %, требуется время тестирования 3×10^8 ч и должны быть соблюдены исходные предпосылки. Число тестов, необходимое для каждого уровня полноты безопасности,— в соответствии с таблицей D.1.

D.2.4 Полное тестирование

Программу можно рассматривать как урну, содержащую N шаров. Каждый шар представляет собой конкретное свойство программы. Шары извлекаются случайно и заменяются после проверки. Полное тестирование достигается, если все шары извлечены.

D.2.4.1 Исходные предпосылки

a) Распределение тестируемых данных таково, что каждое из N свойств программы тестируется с равной вероятностью.

b) Тесты проводятся независимо друг от друга.

c) Каждый появляющийся отказ обнаруживается.

d) Число тестовых примеров $n >> N$.

e) Во время n тестовых примеров отказы не появляются.

f) Каждый прогон теста контролирует одно свойство программы (свойство программы - это то, что может быть протестировано во время одного прогона теста).

D.2.4.2 Результаты

Вероятность тестирования всех свойств программы p определяется выражением

$$p = \sum_{j=0}^{N-1} (-1)^j \left(\frac{N}{j}\right)\left(\frac{N-j}{N}\right)^n \quad \text{или} \quad p = 1 + \sum_{j=1}^{N} (-1)^j C_{j,N}\left(\frac{N-j}{N}\right)^n,$$

где $C_{j,N} = \dfrac{N(N-1)....(N-j+1)}{j!}$.

При оценке этого выражения обычно только первые его члены имеют значение, поскольку в реальных условиях выполняется соотношение $n >> N$, что делает все члены этого выражения при большом j несущественными. Это видно из таблицы D.5.

D.2.4.3 *Пример*

Рассмотрим программу, которая имела несколько инсталляций в течение нескольких лет. За это время она выполнялась, по меньшей мере, $7{,}5 \times 10^6$ раз. Предположим, что каждое сотое выполнение программы соответствует перечисленным выше исходным предпосылкам (см. D.2.4.1). Поэтому для статистической оценки могут быть приняты $7{,}5 \times 10^4$ выполнений программы. Если предположить, что 4000 тестовых прохождений программы могут выполнить исчерпывающее тестирование, считая такую оценку консервативной, то в соответствии с таблицей D.5 вероятность того, что не все будет протестировано, составляет $2{,}87 \times 10^{-5}$.

При $N = 4000$ значения первых членов в зависимости от n представлены в таблице D.5.

Таблица D.5 — Вероятность тестирования всех свойств программы

n	p
5×10^4	$1 - 1{,}9 \times 10^{-2} + 1{,}10 \times 10^{-4} - ...$
$7{,}5 \times 10^4$	$1 - 2{,}87 \times 10^{-5} + 4 \times 10^{-10} - ...$
1×10^5	$1 - 5{,}54 \times 10^{-8} + 1{,}52 \times 10^{-15} - ...$
2×10^5	$1 - 7{,}67 \times 10^{-19} + 2{,}9 \times 10^{-37} - ...$

На практике такие оценки должны быть консервативными [14].

D.3 Литература

Более подробную информацию по указанным выше методам можно найти в IEEE 352:1987 и следующих документах:

Verification and Validation of Real-Time Software, Chapter 5. W. J. Quirk (ed.). Springer Verlag, 1985, ISBN 3-540-15102-8.

Combining Probabilistic and Deterministic Verification Efforts. W. D. Ehrenberger, SAFECOMP 92, Pergamon Press, ISBN 0-08-041893-7.

Ingenieurstatistik. Heinhold/Gaede, Oldenburg, 1972, ISBN 3-486-31743-1.

Приложение F
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам Российской Федерации

Таблица F.1

Обозначение ссылочного международного стандарта	Обозначение и наименование соответствующего национального стандарта Российской Федерации
ИСО/МЭК Руководство 51:1999	ГОСТ Р 51898 — 2002 Аспекты безопасности. Правила включения в стандарты
МЭК Руководство 104:1997	*
МЭК 61508-1:1998	ГОСТ Р МЭК 61508-1—2006 (МЭК 61508-1—1998) Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 1. Общие требования
МЭК 61508-2:2000	ГОСТ Р МЭК 61508-2—2007 (МЭК 61508-2—2000) Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 2. Требования к системам
МЭК 61508-3:1998	ГОСТ Р МЭК 61508-3—2006 (МЭК 61508-3—1998) Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 3. Требования к программному обеспечению
МЭК 61508-4:1998	ГОСТ Р МЭК 61508-4—2006 (МЭК 61508-4—1998) Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 4. Термины и определения
МЭК 61508-5:1998	ГОСТ Р МЭК 61508-5—2006 (МЭК 61508-5—1998) Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 5. Рекомендации по применению методов определения уровней полноты безопасности
МЭК 61508-6:2000	ГОСТ Р МЭК 61508-6—2007 (МЭК 61508-6—2000) Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 6. Руководство по применению МЭК 61508-2:2000 и МЭК 61508-3:1998
IEEE 352:1987	*

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Библиография

[1] IEC 60068-1:1988 Environmental testing — Part 1: General and guidance

[2] IEC 60529:1989 Degrees of protection provided by enclosures (IP Code)

[3] IEC 60812:1985[1)] Analysis techniques for system reliability — Procedure for failure mode and effects analysis (FMEA)

[4] IEC 60880:1986[2)] Software for computers in the safety systems of nuclear power stations

[5] IEC 61000-4-1:1992[3)] Electromagnetic compatibility (EMC) — Part 4: Testing and measurement techniques — Section 1: Overview of immunity tests. Basic EMC publication

[6] IEC 61000-4-5:1995[4)] Electromagnetic compatibility (EMC) — Part 4: Testing and measurement techniques — Section 5: Surge immunity test

[7] IEC 61000-5-2:1997 Electromagnetic compatibility (EMC) — Part 5: Installation and mitigation guidelines — Section 2: Earthing and cabling

[8] IEC 61025:1990[5)] Fault tree analysis (FTA)

[9] IEC 61069-5:1994 Industrial-process measurement and control — Evaluation of system properties for the purpose of system assessment — Part 5: Assessment of system dependability

[10] IEC 61078:1991[6)] Analysis techniques for dependability — Reliability block diagram method

[11] IEC 61131-3:1993[7)] Programmable controllers — Part 3: Programming languages

[12] IEC 61160:1992[8)] Formal design review Amendment 1 (1994)

[13] IEC 61163-1:1995[9)] Reliability stress screening — Part 1: Repairable items manufactured in lots

[14] IEC 61164:1995[10)] Reliability growth — Statistical test and estimation methods

[15] IEC 61165:1995[11)] Application of Markov techniques

[16] IEC 61346-1:1996 Industrial systems, installations and equipment and industrial products — Structuring, principles and reference designation — Part 1: Basic rules

[17] IEC 61506:1997 Industrial-process measurement and control — Documentation of application software

[18] IEC 61511-SER Functional safety — Safety instrumented systems for the process industry sector — ALL PARTS

[19] IEC 61704[12)] Guide to the selection of software test methods for reliability assessment

[20] ISO 5807:1985 Information processing — Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts

[21] ISO/IEC 7185:1990 Information technology — Programming languages — Pascal

[22] ISO/IEC 8631:1989 Information technology — Program constructs and conventions for their representation

[23] ISO/IEC 8652:1995 Information technology — Programming languages — Ada

[24] ISO 8807:1989 Information processing systems — Open Systems Interconnection — LOTOS — A formal description technique based on the temporal ordering of observational behaviour

[25] ISO/IEC 9899:1990[13)] Programming languages — C

[26] ISO/IEC 10206:1991 Information technology — Programming languages — Extended Pascal

[27] ISO/IEC 10514-1:1996 Information technology — Programming languages — Part 1: Modula-2, Base Language

---

[1)] В настоящее время действует IEC 60812:2006 Analysis techniques for system reliability — Procedure for failure mode and effects analysis (FMEA).

[2)] В настоящее время действует IEC 60880:2006 Software for computers in the safety systems of nuclear power stations.

[3)] В настоящее время действует IEC 61000-4-1:2000 Electromagnetic compatibility (EMC) — Part 4: Testing and measurement techniques — Section 1: Overview of immunity tests. Basic EMC publication.

[4)] В настоящее время действует IEC 61000-4-5:2005 Electromagnetic compatibility (EMC) — Part 4: Testing and measurement techniques — Section 5: Surge immunity test.

[5)] В настоящее время действует IEC 61025:2006 Fault tree analysis (FTA).

[6)] В настоящее время действует IEC 61078:2006 Analysis techniques for dependability — Reliability block diagram method.

[7)] В настоящее время действует IEC 61131-3:2003 Programmable controllers — Part 3: Programming languages.

[8)] В настоящее время действует IEC 61160:2005 Formal design review.

[9)] В настоящее время действует IEC 61163-1:2006 Reliability stress screening — Part 1: Repairable items manufactured in lots.

[10)] В настоящее время действует IEC 61164:2004 Reliability growth — Statistical test and estimation methods.

[11)] В настоящее время действует IEC 61165:2006 Application of Markov techniques.

[12)] В стадии разработки.

[13)] В настоящее время действует ISO/IEC 9899:1999 Programming languages – C.

[28] ISO/IEC 10514-3:1998 Information technology — Programming languages — Part 3: Object Oriented Modula-2

[29] ISO/IEC 13817-1:1996 Information technology — Programming languages, their environments and system software interfaces — Vienna Development Method — Specification Language — Part 1: Base language

[30] ISO/IEC 14882:1998[1] Programming languages — C++

[31] ISO/IEC 1539-1:1997[2] 2004 Information technology — Programming languages — Fortran — Part 1: Base language

[32] ISO/IEC/TR 15942[3] Guidance for the use of the Ada programming language in high integrity systems

[1] В настоящее время действует ISO/IEC 14882:2003 Programming languages – C++.

[2] В настоящее время действует ISO/IEC 1539-1:2004 Information technology — Programming languages — Fortran — Part 1: Base language.

[3] В стадии разработки.

УДК 62-783:614.8:331.454:006.354 ОКС 13.110 Т51

Ключевые слова: функциональная безопасность; жизненный цикл систем; электрические компоненты; электронные компоненты; программируемые электронные компоненты и системы; системы, связанные с безопасностью; случайные отказы оборудования; систематические отказы; планирование функциональной безопасности; методы и средства; полнота безопасности

Редактор *В. Н. Копысов*
Технический редактор *В. Н. Прусакова*
Корректор *Е. Ю. Митрофанова*
Компьютерная верстка *З. И. Мартыновой*

Сдано в набор 10.04.2008. Подписано в печать 23.09.2008. Формат $60\times84^{1}/_{8}$. Бумага офсетная. Гарнитура Ариал. Печать офсетная. Усл. печ. л. 8,37. Уч.-изд. л. 8,70. Тираж 313 экз. Зак. 935.

ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
www.gostinfo. ru info@gostinfo. ru
Набрано и отпечатано в Калужской типографии стандартов, 248021 Калуга, ул. Московская, 256.