ГОСУДАРСТВЕННЫЙ СТАНДАРТ
РЕСПУБЛИКИ БЕЛАРУСЬ

СТБ IEC 62279-2011

УДК 629.4.067:004.422.8(083.74)(476) МКС 45.060 КП 06 IDT

Ключевые слова: программное обеспечение, уровень полноты безопасности программного обеспечения, тестирование программного обеспечения, жизненный цикл программного обеспечения, архитектура программного обеспечения, верификация программного обеспечения

Предисловие

Цели, основные принципы, положения по государственному регулированию и управлению в области технического нормирования и стандартизации установлены Законом Республики Беларусь «О техническом нормировании и стандартизации».

1 ПОДГОТОВЛЕН научно-производственным республиканским унитарным предприятием «Белорусский государственный институт стандартизации и сертификации» (БелГИСС)

ВНЕСЕН Госстандартом Республики Беларусь

2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ постановлением Госстандарта Республики Беларусь от 27 апреля 2011 г. № 19

3 Настоящий стандарт идентичен международному стандарту IEC 62279:2002 Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems (Железные дороги. Системы связи, сигнализации и обработки данных. Программное обеспечение для систем управления и защиты на железной дороге).

Международный стандарт разработан техническим комитетом по стандартизации IEC/TC 9 «Электрическое оборудование и системы для железных дорог» Международной электротехнической комиссии (IEC).

Перевод с английского языка (en).

Официальные экземпляры международного стандарта, на основе которого подготовлен настоящий государственный стандарт, и международных и европейского стандартов, на которые даны ссылки, имеются в Национальном фонде ТНПА.

В разделе «Нормативные ссылки» и тексте стандарта ссылки на международные и европейский стандарты актуализированы.

Сведения о соответствии государственных стандартов ссылочным международным стандартам приведены в дополнительном приложении Д.А.

Степень соответствия – идентичная (IDT)

4 ВВЕДЕН ВПЕРВЫЕ

Издан на русском языке

Содержание

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ

Железные дороги
СИСТЕМЫ СВЯЗИ, СИГНАЛИЗАЦИИ И ОБРАБОТКИ ДАННЫХ
Программное обеспечение для систем управления и защиты на железных дорогах

Чыгунка
СІСТЭМЫ СУВЯЗІ, СІГНАЛІЗАЦЫІ І АПРАЦОЎКІ ДАНЫХ
Праграмнае забеспячэнне для сістэм кіравання і засцярогі на чыгунцы

Railway applications
Communications, signalling and processing systems
Software for railway control and protection systems

Дата введения 2011-08-01

1 Область применения

1.1 Настоящий стандарт устанавливает технические требования и порядок разработки программируемых электронных систем, использующихся на железнодорожных дорогах в целях управления и защиты. Он распространяется на все области, связанные с безопасностью: от критически важных, таких как аварийно-защитная сигнализация, до менее критичных, таких как системы управления информацией. Данные системы могут быть реализованы на базе специализированных микропроцессоров, программируемых логических контроллеров, многопроцессорных распределительных систем, крупномасштабных систем с центральным процессором или другой архитектуры.

1.2 Настоящий стандарт распространяется исключительно на программное обеспечение (ПО) и взаимодействие между ПО и системой, частью которой оно является.

1.3 Уровни полноты безопасности ПО выше нуля устанавливаются для систем, последствия отказа которых способны повлечь за собой опасность для жизни людей. Экономические или экологические факторы также могут служить основанием для использования более высоких уровней полноты безопасности ПО.

1.4 Настоящий стандарт распространяется на все ПО, используемое при разработке и внедрении железнодорожных систем управления и защиты, включая:

– прикладное программирование;
– операционные системы;
– средства поддержки;
– встроенное ПО.

Прикладное программирование включает высоко- и низкоуровневое программирование и специальное программирование (например, для программируемых логических контроллеров).

1.5 В настоящем стандарте также описано использование стандартного готового коммерческого ПО.

1.6 Настоящий стандарт также устанавливает требования к системам, конфигурируемым на основе прикладных данных.

1.7 Настоящий стандарт не предназначен для решения коммерческих задач. Они должны рассматриваться как неотъемлемая часть любого договора. Все разделы настоящего стандарта необходимо тщательно рассматривать для конкретной коммерческой ситуации.

1.8 Настоящий стандарт не распространяется на ПО и системы, разработанные ранее. Поэтому он применяется к новым разработкам и к существующим системам, если они подвергаются значительным изменениям. В случае незначительных изменений применяется раздел 16.

2 Нормативные ссылки

Для применения настоящего стандарта необходимы следующие ссылочные стандарты. Для датированных ссылок применяют только указанное издание ссылочного стандарта. Для недатированных ссылок применяют последнее издание ссылочного стандарта (включая все его изменения).

IEC 62278:2002 Железные дороги. Технические условия и демонстрация надежности, эксплуатационной готовности, ремонтопригодности и удобства обслуживания

Издание официальное

IEC 62280-1:2002 Железные дороги. Системы связи, сигнализации и обработки данных. Часть 1. Экстренная связь в закрытых системах передачи

IEC 62280-2:2002 Железные дороги. Системы связи, сигнализации и обработки данных. Часть 2. Экстренная связь в открытых системах передачи

ISO 9000:2000 Системы менеджмента качества. Основные положения и словарь

ISO 9000-3:1997 Стандарты в области управления качеством и обеспечения качества. Часть 3. Руководящие указания по применению стандарта ИСО 9001:1994 для разработки, поставки, установки и обслуживанию программного обеспечения

ISO 9001:1994 Система качества. Модель для обеспечения качества при проектировании, разработке, производстве, монтаже и обслуживании

EN 50129:2003 * Железные дороги. Системы связи, сигнализации и обработки данных. Безопасность электронных систем сигнализации

3 Термины и определения

В настоящем стандарте применяют следующие термины с соответствующими определениями. Для терминов, не установленных в настоящем разделе, необходимо в порядке приоритета принимать во внимание следующие стандарты:

IEC 9000:2000 Системы менеджмента качества. Основные положения и словарь

IEC 60050-191:1990 Международный электротехнический словарь. Глава 191. Надежность и качество услуг

IEEE STD 610.12 Глоссарий терминов технологий программирования

ISO/IEC 2382 (все части) Информационные технологии. Словарь

ISO/IEC 9126 (все части) Разработка программного обеспечения. Качество продукции

3.1 оценка (assessment): Процесс анализа, целью которого является установление, создан ли отделом разработок и лицом, ответственным за валидацию, продукт, соответствующий установленным требованиям, и формирование решения относительно пригодности изделия для применения его по назначению.

3.2 эксперт (assessor): Лицо или представитель, назначенный для проведения оценки.

3.3 пригодность к эксплуатации (availability): Способность изделия быть в состоянии выполнять требуемую функцию при заданных условиях в заданный момент времени при наличии необходимых внешних ресурсов.

3.4 готовое коммерческое ПО (COTS, commercial off-the-shelf software): Программное обеспечение, разработанное в соответствии с потребностями рынка, доступное на платной основе, пригодность которого к использованию по назначению подтверждается широким кругом покупателей.

3.5 отдел разработок (design authority): Организация, орган или подразделение, ответственные за разработку проектного решения для выполнения установленных требований, контроль за дальнейшей разработкой системы и вводом ее в эксплуатацию в конкретной среде.

3.6 разработчик (designer): Одно или несколько лиц, назначенных отделом разработок для анализа установленных требований и их реализации в соответствующих проектных решениях, обеспечивающих необходимую полноту безопасности.

3.7 элемент (element): Часть изделия, которая была определена как базовая единица или структурный элемент. Элемент может быть простым или сложным.

3.8 ошибка (error): Отклонение от заданного проектного решения, которое может привести к непредсказуемому поведению или отказу системы.

3.9 отказ (failure): Отклонение от заданных рабочих характеристик системы, представляющее собой последствие ошибки или сбоя в системе.

3.10 сбой (fault): Ненормальный режим, который может привести к ошибке или отказу системы. Сбой может быть случайным или систематическим.

3.11 предотвращение сбоев (fault avoidance): Применение методов разработки, целью которых является предотвращение появления сбоев в ходе разработки и построения системы.

3.12 устойчивость к сбоям (fault tolerance): Предусмотренная способность системы к обеспечению постоянного корректного предоставления услуги согласно установленным требованиям при ограниченном количестве сбоев программного обеспечения и аппаратных средств.

---

* Действует взамен ENV 50129:1998.

3.13 встроенное ПО (firmware): Упорядоченный набор инструкций и связанных с ними данных, которые хранятся в месте, функционально независимом от основного запоминающего устройства (как правило, в постоянном запоминающем устройстве).

3.14 общее ПО (generic software): Программное обеспечение, которое может быть использовано для инсталляции на различных объектах исключительно посредством применения прикладных данных.

3.15 исполнитель (implementer): Одно или несколько лиц, назначенных отделом разработок для физической реализации конкретных решений.

3.16 изделие (product): Множество элементов, связанных между собой для формирования системы, подсистемы или единицы оборудования, таким образом, чтобы соответствовать установленным требованиям. В настоящем стандарте изделие может рассматриваться как состоящее полностью из элементов программного обеспечения или документации.

3.17 программируемый логический контроллер; ПЛК (programmable logic controller; PLC): Система управления на полупроводниковых приборах с программируемой пользователем памятью для хранения инструкций по реализации определенных функций.

3.18 надежность (reliability): Способность изделия выполнять требуемую функцию при определенных условиях в течение определенного времени.

3.19 прослеживаемость требований (requirements traceability): Обеспечение того, чтобы требования могли быть представлены как выполненные должным образом.

3.20 риск (risk): Комбинация частоты, или вероятности, и последствий определенного опасного события.

3.21 безопасность (safety): Отсутствие неприемлемых уровней риска.

3.22 отдел обеспечения безопасности (safety authority): Организация, орган или подразделение, ответственные за оценку системы безопасности в отношении ее соответствия функциональному назначению и правовым требованиям безопасности.

3.23 ПО, связанное с безопасностью (safety-related software):Программное обеспечение, выполняющее функции обеспечения безопасности.

3.24 программное обеспечение (software): Интеллектуальный продукт, включающий программы, процедуры, правила и любую связанную с ними документацию, имеющие отношение к функционированию системы.

Примечание – ПО не зависит от используемых носителей информации.

3.25 жизненный цикл ПО (software life cycle): Деятельность, выполняемая в течение периода времени, который начинается от формирования требований к созданию программного обеспечения и заканчивается тогда, когда программное обеспечение больше непригодно к эксплуатации. Жизненный цикл программного обеспечения, как правило, включает стадии установления требований, разработки, тестирования, интеграции, установки и обслуживания.

3.26 ремонтопригодность ПО (software maintainability): Возможность внесения изменений в систему для устранения сбоев, улучшения функциональных характеристик и других параметров или адаптации к различным средам.

3.27 обслуживание ПО (software maintenance): Действие или комплекс действий, проводимых с программным обеспечением после его приемки конечным пользователем, целью которых является усовершенствование, расширение и/или исправление его функциональных характеристик.

3.28 уровень полноты безопасности ПО (software safety integrity level): Классификационный номер, который определяет методы и меры, которые должны быть применены для снижения потенциально возможных сбоев программного обеспечения до надлежащего уровня.

3.29 уровень полноты безопасности системы (system safety integrity level): Число, которое указывает степень достоверности соответствия системы установленным требованиям безопасности.

3.30 прослеживаемость (traceability): Степень связи между двумя или более изделиями в процессе разработки, особенно между теми, которые состоят в отношениях предыдущего с последующим или главного со второстепенным.

3.31 валидация (validation): Подтверждение посредством проведения анализа и тестирования того, что изделие во всех отношениях соответствует установленным требованиям.

3.32 лицо, ответственное за валидацию (validator): Лицо или представитель, назначенные для проведения валидации.

3.33 верификация (verification): Подтверждение посредством проведения анализа и тестирования того, что результат каждой стадии жизненного цикла соответствует требованиям предыдущей стадии.

3.34 лицо, ответственное за верификацию (verifier): Лицо или представитель, назначенные для проведения верификации.

4 Цели и соответствие

4.1 В каждом из следующих разделов подробно описаны соответствующие цели и требования.

4.2 Для обеспечения соответствия настоящему стандарту должно быть продемонстрировано, что каждое из требований выполнено в соответствии с установленным уровнем полноты безопасности ПО и, как следствие, цель раздела достигнута.

4.3 Если требование сопровождается словами «в степени, требуемой уровнем полноты безопасности ПО», это означает, что для выполнения этого требования может быть применен ряд методов и мер.

4.4 Если применимы требования 4.3, должны быть использованы приведенные в настоящем стандарте таблицы для выбора методов и мер, соответствующих уровню полноты безопасности ПО.

4.5 Если метод или мера в таблицах классифицированы как строго рекомендуемые, отказ от применения данного метода или меры должен быть подробно обоснован и зафиксирован в плане обеспечения качества ПО или в другом документе, на который дается ссылка в плане обеспечения качества ПО. В этом нет необходимости при применении установленной в настоящем стандарте комбинации методов, приведенных в соответствующей таблице.

4.6 Если предложенные для применения метод или мера не приведены в таблицах, то их эффективность и соответствие конкретному требованию и общей цели раздела должны быть обоснованы и зарегистрированы в плане обеспечения качества ПО или в другом документе, на который дается ссылка в плане обеспечения качества ПО.

4.7 Соблюдение требований конкретного раздела и применение соответствующих методов и мер, приведенных в таблицах, должны быть оценены посредством проведения проверки документов, установленных настоящим стандартом, других объективных данных, аудита и испытаний при освидетельствовании.

4.8 В соответствии с настоящим стандартом необходимо использовать комплекс методов, а также правильно их применять. Эти методы установлены в таблицах и уточнены в библиографии.

5 Уровни полноты безопасности ПО

5.1 Цель

Описание присвоения уровней полноты безопасности ПО.

5.2 Требования

5.2.1 Согласно требованиям IEC 62278 и EN 50129 необходимо разработать:

– спецификацию требований к системе;
– спецификацию требований безопасности системы;
– описание архитектуры системы;
– план обеспечения безопасности системы,

что включает:

– функции безопасности;
– конфигурацию или архитектуру системы;
– требования надежности аппаратных средств;
– требования полноты безопасности.

Уровень полноты безопасности ПО должен быть установлен в ходе общего процесса определения уровня полноты безопасности согласно IEC 62278.

5.2.2 Необходимый уровень полноты безопасности ПО определяется на основе уровня риска, связанного с использованием ПО в системе, и уровня полноты безопасности самой системы.

5.2.3 Без дополнительных мер предосторожности уровень полноты безопасности ПО должен быть как минимум идентичен уровню полноты безопасности системы. Однако если существуют механизмы, предотвращающие приведение системы в небезопасное состояние по причине сбоев модуля ПО, то уровень полноты безопасности модуля ПО может быть снижен.

5.2.4 Должны быть учтены риски, влекущие:

– гибель человека или людей;
– травмы или заболевания людей;
– экологическое загрязнение;
– утрату или ущерб имуществу.

5.2.5 Риски можно оценить в числовом выражении, однако оценить полноту безопасности ПО таким же образом не представляется возможным. Поэтому в настоящем стандарте устанавливаются следующие пять уровней полноты безопасности ПО:

Уровень полноты безопасности ПО	Описание полноты безопасности ПО
4	Очень высокая
3	Высокая
2	Средняя
1	Низкая
0	Не связана с обеспечением безопасности

5.2.6 Уровень полноты безопасности ПО должен быть установлен в спецификации требований к ПО (раздел 8). Если разные компоненты ПО имеют разные уровни полноты безопасности ПО, то они должны быть установлены в спецификации архитектуры ПО (раздел 9).

6 Персонал и ответственность

6.1 Цель

Обеспечить, чтобы весь персонал, ответственный за ПО, был компетентен для выполнения своих обязанностей.

6.2 Требования

6.2.1 Поставщик и/или разработчик и потребитель как минимум должны соблюдать требования соответствующих частей ISO 9001 согласно руководящим указаниям, приведенным в ISO 9000-3.

6.2.2 Реализация процесса обеспечения безопасности должна осуществляться под контролем соответствующей организации по безопасности, которая отвечает требованиям подраздела «Организация по безопасности» раздела «Обеспечение управления безопасностью» EN 50129. Данное требование не применяется, если ПО имеет уровень полноты безопасности 0.

6.2.3 Персонал, задействованный на всех стадиях жизненного цикла ПО, включая управленческую деятельность, должен пройти обучение, иметь соответствующее образование, опыт и квалификацию.

6.2.4 Настоятельно рекомендуется, чтобы образование, опыт и квалификация персонала, привлекаемого на любой стадии жизненного цикла ПО, включая управленческую деятельность, были подтверждены в соответствующей области деятельности. Данное требование не применяется, если ПО имеет уровень полноты безопасности 0.

6.2.5 Обоснование, описанное в 6.2.4, должно быть зафиксировано в плане обеспечения качества ПО и включать подтверждение компетентности в следующих областях:

i) технические знания, соответствующие сфере деятельности;

ii) разработка ПО;

iii) конструирование компьютерных систем;

iv) технические знания в области средств обеспечения безопасности;

v) правовая и нормативная база.

6.2.6 Должен быть назначен независимый эксперт ПО (см. также 6.2.10 и 14.4.1).

6.2.7 Эксперт должен быть уполномочен проводить оценку ПО.

6.2.8 На протяжении жизненного цикла ПО заинтересованные стороны должны обладать независимостью в степени, требуемой уровнем полноты безопасности ПО, согласно рисунку 5, который трактуется следующим образом.

На всех уровнях полноты безопасности ПО эксперт утверждается отделом обеспечения безопасности и должен быть независимым от поставщика, за исключением случаев, оговоренных в 6.2.10.

Разработчик/исполнитель, лицо, ответственное за верификацию, и лицо, ответственное за валидацию, могут быть работниками одной организации, но при этом должны соблюдаться установленные ниже минимальные требования независимости.

На уровне полноты безопасности ПО 0 ограничений нет – разработчик/исполнитель, лицо, ответственное за верификацию, и лицо, ответственное за валидацию, могут быть одним и тем же лицом.

На уровнях полноты безопасности ПО 1 и 2 лицо, ответственное за верификацию, и лицо, ответственное за валидацию, могут быть одним и тем же лицом, но они не могут быть разработчиком/исполнителем. Однако все перечисленные лица могут находиться в подчинении менеджера проекта.

На уровнях полноты безопасности ПО 3 и 4 существует два варианта:

a) Лицо, ответственное за верификацию, и лицо, ответственное за валидацию, могут быть одним и тем же лицом, но они не могут быть разработчиком/исполнителем. Кроме того, лицо, ответственное за верификацию, и лицо, ответственное за валидацию, не должны находиться в подчинении менеджера проекта, как разработчик/исполнитель, и они должны иметь полномочия останавливать выпуск изделия.

b) Разработчик/исполнитель, лицо, ответственное за верификацию, и лицо, ответственное за валидацию, должны быть разными лицами. Разработчик/исполнитель и лицо, ответственное за верификацию, могут подчиняться менеджеру проекта, в то время как лицо, ответственное за валидацию, не должно ему подчиняться, и, кроме того, лицо, ответственное за валидацию, уполномочено останавливать выпуск изделия.

6.2.9 За выполнение требований разделов настоящего стандарта ответственны:

– спецификация требований к ПО (раздел 8) – разработчик;
– спецификация требований к тестированию ПО (раздел 8) – лицо, ответственное за валидацию;
– архитектура ПО (раздел 9) – разработчик;
– проектирование и разработка ПО (раздел 10) – разработчик;
– верификация и тестирование ПО (раздел 11) – лицо, ответственное за верификацию;
– интеграция ПО и аппаратных средств (раздел 12) – разработчик;
– валидация ПО (раздел 13) – лицо, ответственное за валидацию;
– оценка ПО (раздел 14) – эксперт.

6.2.10 По решению отдела обеспечения безопасности эксперт может назначаться от организации поставщика или организации потребителя, но в таких случаях эксперт должен:

– быть уполномочен отделом обеспечения безопасности;
– быть полностью независимым от группы, работающей над проектом;
– напрямую подчиняться отделу обеспечения безопасности.

7 Жизненный цикл и документация

7.1 Цели

7.1.1 Структурирование разработки ПО на определенные фазы и виды деятельности.

7.1.2 Документальное оформление всей информации, относящейся к ПО, на протяжении жизненного цикла ПО.

7.2 Требования

7.2.1 Должна быть выбрана модель жизненного цикла для разработки ПО. Она должна быть подробно описана в плане обеспечения качества ПО согласно разделу 15. В качестве примера две возможные модели жизненного цикла приведены на рисунках 3 и 4.

7.2.2 Процедуры обеспечения качества должны проводиться параллельно с деятельностью на стадиях жизненного цикла, при этом должна применяться аналогичная терминология.

7.2.3 Все виды деятельности на конкретной стадии жизненного цикла ПО должны быть определены до ее начала. Каждая стадия жизненного цикла ПО должна делиться на простые задачи с четко определенными входными и выходными данными и видом деятельности для каждой из задач.

7.2.4 В плане обеспечения качества ПО должны быть описаны все необходимые действия по верификации и отчеты.

7.2.5 Все документы должны быть структурированы для обеспечения постоянного расширения параллельно с процессом разработки.

7.2.6 Прослеживаемость документов должна обеспечиваться посредством присвоения каждому документу уникального ссылочного номера и определения и документального оформления связи его с другими документами. Каждый термин, сокращение или аббревиатура должны быть однозначными в пределах одного документа. Если по имевшим место ранее причинам это не представляется возможным, разнящиеся значения должны быть перечислены с указанием ссылок.

Кроме того, каждый документ, за исключением документов по готовому коммерческому ПО (см. 9.4.5) или ранее разработанному ПО (см. 9.4.6), оформляется согласно следующим правилам:

a) он должен содержать или реализовывать все применимые условия и требования предшествующего документа, с которым он состоит в иерархической связи;

b) он не должен противоречить предшествующему документу;

c) каждый термин, сокращение или аббревиатура должны быть однозначными в пределах одного документа;

d) ссылка на любой пункт или понятие делается с использованием одинакового наименования или описания в каждом документе.

Результат процесса прослеживаемости является предметом официального управления конфигурацией.

7.2.7 Содержание всех документов должно быть изложено по форме, пригодной для работы, обработки и хранения.

7.2.8 Документы, перечисленные в таблице документальной перекрестной ссылки (см. ниже), разрабатываются при необходимости в степени, требуемой уровнем полноты безопасности ПО.

7.2.9 Количество документов, которые требуется разработать, будет варьироваться в зависимости от объема, сложности и жизненного цикла разрабатываемого ПО. Некоторые документы могут быть объединены (при условии, что необходимые подробности не будут упущены). Для крупных проектов может потребоваться разделение перечисленной документации (по иерархическому принципу) на ряд более удобных в управлении документов меньшего объема. Документы, которые были разработаны независимыми группами или органами, не должны объединяться в единый документ.

7.2.10 Связь между различными документами, описанными в настоящем разделе, может быть также определена через таблицу документальной перекрестной ссылки. Для каждого из документов, перечисленных в графе «Документы», стадия и раздел, связанные с его созданием, указаны по горизонтали и вертикали относительно ячейки, содержащей символ «■». Стадии, на которых он используется, указаны по вертикали относительно ячеек, отмеченных символом «♦». Номер раздела или другая ссылка на определение документа указаны в графе «Где определено». Если указан номер раздела, следующие разделы также необходимо проверить, так как они могут содержать более подробную информацию. Следует отметить, что ссылка на план управления конфигурацией ПО приведена в скобках, так как в данном разделе просто дана ссылка на ISO 9001.

Таблица документальной перекрестной ссылки

Раздел	8	9	10	11	12	13	14	15	16	Документы	Где определено
Заголовок	SRS	SA	SDD	SVer	S/H I	SVal	Ass	Q	Ma		
Стадии (*) = параллельно с другими стадиями											
Входы системы (вход из разработки)	♦			♦	♦					Спецификация требований к системе	EN 50129 (B.2.3)
	♦	♦		♦	♦	♦	♦			Спецификация требований безопасности системы	EN 50129 (B.2.4)
	♦				♦					Описание архитектуры системы	EN 50129 (B.2.1)
										План обеспечения безопасности системы	EN 50129 IEC 62278
Планирование ПО (*)	♦	♦	♦	♦		♦	♦	■		План обеспечения качества ПО	15.4.3
						♦	♦	■		План управления конфигурацией ПО	15.4.2
				■		♦	♦			План верификации ПО	11.4.1
				■		♦	♦			План интеграции ПО	11.4.5
					■	♦	♦			План интеграции ПО и аппаратных средств	12.4.1
						■	♦			План валидации ПО	13.4.3

Раздел	8	9	10	11	12	13	14	15	16	Документы	Где определено
Заголовок	SRS	SA	SDD	SVer	S/H I	SVal	Ass	Q	Ma		
Стадии (*) = параллельно с другими стадиями											
							♦		■	План сопровождения ПО	16.4.3
										План подготовки данных	17.4.2.1
										План проверки данных	17.4.2.4
Требования к ПО (*)	■	♦	♦	♦	♦	♦	♦			Спецификация требований к ПО	8.4.1
										Спецификация прикладных требований	17.4.1.1
	■			♦	♦	♦	♦			Спецификация требований к тестированию ПО	8.4.13
				■						Отчет о верификации требований к ПО	11.4.1
Разработка ПО		■	♦	♦	♦	♦	♦			Спецификация архитектуры ПО	9.4.1
			■	♦	♦	♦	♦			Спецификация проекта ПО	10.4.3
				■						Отчет о верификации архитектуры и проекта ПО	11.4.12
Разработка модуля ПО			■	♦	♦	♦	♦			Спецификация проекта модуля ПО	10.4.3
			■	♦	♦	♦	♦			Спецификация испытаний модуля ПО	10.4.14
				■						Отчет о верификации модуля ПО	11.4.13
Кодирование			■	♦	♦	♦	♦			Исходный код ПО	
				■		♦	♦			Отчет о верификации исходного кода ПО	11.4.14
Тестирование модуля			■	♦						Отчет об испытаниях модуля ПО	10.4.14
Интеграция ПО				■						Отчет об испытаниях интеграции ПО	11.4.15
										Отчет об испытаниях данных	17.4.2.4
Интеграция ПО и аппаратных средств					■					Отчет об испытаниях интеграции ПО и аппаратных средств	12.4.8
Валидация (*)						■				Отчет о валидации ПО	13.4.10

Раздел	8	9	10	11	12	13	14	15	16	Документы	Где определено
Заголовок	SRS	SA	SDD	SVer	S/H I	SVal	Ass	Q	Ma		
Стадии (*) = параллельно с другими стадиями											
Оценка (*)							■			Отчет об оценке ПО	14.4.9
Сопровождение									■	Записи об изменениях	16.4.9
									■	Записи об обслуживании ПО	16.4.8

8 Спецификация требований к ПО

8.1 Цель

8.1.1 Описание документа, в котором определен полный набор требований для ПО, с соблюдением всех требований к системе в степени, требуемой уровнем полноты безопасности ПО. Он является всеобъемлющим документом для каждого участника разработки ПО, которому, таким образом, не требуется изучать все документы в поисках требований.

8.1.2 Описание спецификации требований к тестированию ПО.

8.2 Входные документы

1) Спецификация требований к системе.
2) Спецификация требований безопасности системы.
3) Описание архитектуры системы.
4) План обеспечения качества ПО.

8.3 Выходные документы

1) Спецификация требований к ПО.
2) Спецификация требований к тестированию ПО.

8.4 Требования

8.4.1 В спецификации требований к ПО должны быть установлены требования к свойствам разрабатываемого ПО, а не к порядку его разработки. Эти свойства определены в ISO/IEC 9126 (за исключением безопасности) и включают:

– функциональность (включая производительность и допустимое время отклика);
– надежность и ремонтопригодность;
– безопасность (включая функции безопасности и связанные с ними уровни полноты безопасности ПО);
– эффективность;
– удобство использования;
– компактность.

Уровень полноты безопасности ПО определяется согласно разделу **5** и фиксируется в спецификации требований к ПО.

8.4.2 В степени, требуемой уровнем полноты безопасности ПО, спецификация требований к ПО должна быть выражена и построена так, чтобы эти требования были:

i) полными, четкими, точными, однозначными, пригодными для верификации, тестирования и обслуживания, и выполнимыми;

ii) прослеживаемыми по отношению ко всем документам, указанным в 8.2.

8.4.3 Спецификация требований к ПО должна включать способы выражения и описания, понятные ответственному персоналу, задействованному на протяжении всего жизненного цикла системы.

8.4.4 В спецификации требований к ПО должны быть идентифицированы и зарегистрированы все взаимодействия с любыми другими системами, внутри или вне рассматриваемого оборудования, включая операторов, если такая связь существует или предполагается.

8.4.5 Все возможные режимы функционирования должны быть подробно описаны в спецификации требований к ПО.

8.4.6 В спецификации требований к ПО должны быть подробно описаны все возможные режимы работы программируемых электронных устройств, в частности работа при отказе.

8.4.7 В спецификации требований к ПО должны быть идентифицированы и зарегистрированы любые ограничения между аппаратными средствами и ПО.

8.4.8 В спецификации требований к ПО должна быть указана степень самопроверки ПО и заданный уровень программной проверки аппаратных средств. Самопроверка ПО включает обнаружение и передачу сообщений, производимых ПО, в отношении собственных отказов и ошибок.

8.4.9 В спецификации требований к ПО должны быть установлены требования к периодическому тестированию функций в степени, требуемой спецификацией требований безопасности системы.

8.4.10 В спецификации требований к ПО должны быть установлены требования относительно обеспечения возможности тестирования функций безопасности во время эксплуатации всей системы в степени, требуемой спецификацией требований безопасности системы.

8.4.11 Если требуется, чтобы ПО выполняло определенные функции, особенно связанные с достижением требуемого уровня полноты безопасности системы, то такие функции должны быть четко определены в спецификации требований к ПО.

8.4.12 Если требуется, чтобы ПО выполняло небезопасные функции, то такие функции должны быть четко определены в спецификации требований к ПО.

8.4.13 Спецификация требований к тестированию ПО разрабатывается на основе спецификаций требований к ПО. Спецификация тестирования используется для верификации всех требований, описанных в спецификации требований к ПО, а также в качестве описания тестов, которые должны быть проведены с завершенным ПО.

8.4.14 В спецификации требований к тестированию ПО должен устанавливаться набор тестовых данных для каждой требуемой функции, включая:

i) требуемые входные сигналы с их последовательностью и значениями;

ii) ожидаемые выходные сигналы с их последовательностью и значениями;

iii) критерии приемки, в том числе рабочие характеристики и аспекты качества.

8.4.15 Прослеживаемость требований является важным критерием при валидации системы, связанной с безопасностью. Должны быть предоставлены средства, позволяющие демонстрировать прослеживаемость требований на всех стадиях жизненного цикла.

8.4.16 Любые непрослеживаемые данные должны быть отмечены, чтобы они не оказывали влияния на безопасность или полноту системы.

9 Архитектура ПО

9.1 Цели

9.1.1 Разработка архитектуры ПО, удовлетворяющей спецификации требований к ПО в степени, соответствующей уровню полноты безопасности ПО.

9.1.2 Анализ требований, установленных для ПО архитектурой системы.

9.1.3 Определение и оценка важности взаимодействия между аппаратным оборудованием и ПО в части безопасности.

9.1.4 Выбор метода разработки, если он не был определен ранее.

9.2 Входные документы

1) Спецификация требований к ПО.
2) Спецификация требований безопасности системы.
3) Описание архитектуры системы.
4) План обеспечения качества ПО.

9.3 Выходные документы

Спецификация архитектуры ПО.

9.4 Требования

9.4.1 Предполагаемая архитектура ПО должна быть определена поставщиком и/или разработчиком ПО и подробно описана в спецификации архитектуры ПО.

9.4.2 В спецификации архитектуры ПО должна быть учтена возможность выполнения спецификации требований к ПО при обеспечении заданного уровня полноты безопасности ПО.

9.4.3 В спецификации архитектуры ПО должна быть установлена, оценена и детально описана значимость взаимодействия аппаратных средств и ПО. Согласно требованиям IEC 62278 и EN 50129, результаты предварительных исследований взаимодействия между аппаратными средствами и ПО должны содержаться в спецификации требований безопасности системы.

9.4.4 В спецификации архитектуры ПО должны быть установлены все элементы ПО и для этих элементов определено:

i) являются ли эти элементы новыми, существующими или лицензионными;

ii) проводилась ли валидация этих элементов ранее и каковы были условия валидации;

iii) уровень полноты безопасности элемента ПО.

9.4.5 На использование готового коммерческого ПО накладываются следующие ограничения:

i) при уровне полноты безопасности ПО 0 готовое коммерческое ПО должно применяться без дальнейших мер предосторожности;

ii) при уровне полноты безопасности ПО 1 или 2 готовое коммерческое ПО должно включаться в процесс валидации ПО;

iii) при уровне полноты безопасности ПО 3 или 4 должны быть приняты следующие меры предосторожности:

a) готовое коммерческое ПО должно быть подвергнуто тестированию при валидации;

b) должен быть проведен анализ возможных отказов готового коммерческого ПО;

c) должна быть разработана стратегия выявления отказов готового коммерческого ПО и защиты системы от этих отказов;

d) стратегия защиты должна быть подвергнута тестированию при валидации;

e) должны быть созданы и оценены журналы регистрации ошибок;

f) должны быть по возможности использованы только простейшие функции готового коммерческого ПО.

9.4.6 Если как часть проекта необходимо использовать разработанное ранее ПО, то оно должно быть четко идентифицировано и документально оформлено. В спецификации архитектуры ПО должно быть приведено подтверждение соответствия ПО спецификации требований ПО и уровню полноты безопасности. Влияние любых изменений ПО на оставшуюся часть системы должно быть тщательно изучено для того, чтобы определить, требуют ли они повторной проверки и оценки. Должно быть доказано, что выполняются требования спецификации интерфейсов с другими модулями, которые не проходят повторную верификацию, валидацию и оценку.

9.4.7 В проекте по мере возможности должны использоваться прошедшие верификацию существующие модули ПО, разработанные в соответствии с требованиями настоящего стандарта.

9.4.8 В архитектуре ПО должна быть минимизирована та его часть, которая влияет на безопасность.

9.4.9 Если ПО состоит из элементов с разными уровнями полноты безопасности ПО, то всем элементам ПО должен быть присвоен высший уровень полноты безопасности ПО, за исключением случаев, когда имеется подтверждение независимости между элементами высокого и низкого уровней полноты безопасности ПО. Данное свидетельство регистрируется в спецификации архитектуры ПО.

9.4.10 В спецификации архитектуры ПО должны быть определена стратегия разработки ПО в степени, требуемой уровнем полноты безопасности ПО. Спецификация архитектуры ПО должна быть изложена и структурирована таким образам, чтобы она была:

i) полной, четкой, точной, однозначной, пригодной для верификации, тестирования, обслуживания, и выполнимой;

ii) прослеживаемой по отношению к спецификации требований к ПО.

9.4.11 В спецификации архитектуры ПО должен быть обоснован баланс между стратегиями предотвращения сбоев и их устранением.

9.4.12 В спецификации архитектуры ПО должны быть обоснованы методы и меры, выбранные из набора, соответствующего спецификации требований к ПО при заданном уровне полноты безопасности ПО.

10 Разработка и внедрение ПО

10.1 Цели

10.1.1 Разработка и внедрение ПО с заданным уровнем полноты безопасности согласно спецификации требований к ПО и спецификации архитектуры ПО.

10.1.2 Разработка ПО, пригодного для анализа, тестирования, верификации и обслуживания. Тестирование модуля ПО также проводится на этой стадии. Так как верификация и тестирование являются критичными факторами для валидации, им необходимо уделить особое внимание в ходе проектирования и разработки с целью обеспечения пригодности разрабатываемой системы и ее ПО к тестированию с самого начала.

10.1.3 Подбор с учетом требуемого уровня полноты безопасности ПО подходящего набора средств, включая языки и компиляторы, которые будут содействовать верификации, валидации, оценке и обслуживанию на протяжении всего жизненного цикла ПО.

10.1.4 Осуществление интеграции ПО.

10.2 Входные документы

1) Спецификация требований к ПО.
2) Спецификация архитектуры ПО.
3) План обеспечения качества ПО.

10.3 Выходные документы

1) Спецификация проекта ПО.
2) Спецификация проекта модуля ПО.
3) Спецификация тестирования модуля ПО.
4) Исходный код ПО и сопроводительная документация.
5) Отчет о тестировании модуля ПО.

10.4 Требования

10.4.1 До начала разработки должны быть доступны, даже в неокончательной редакции, спецификация требований к ПО и спецификация архитектуры ПО.

10.4.2 Объем и сложность разрабатываемого ПО должны быть сведены к минимуму.

10.4.3 В спецификации проектирования ПО должны быть установлены требования к проекту ПО на основе деления его на модули, на каждый из которых имеется спецификация проекта модуля ПО и спецификация тестирования модуля ПО.

10.4.4 Спецификация проекта ПО должна включать описание:

i) элементов ПО, установленных в архитектуре ПО и требованиях к их уровню полноты безопасности;

ii) внешних интерфейсов элементов ПО;

iii) интерфейсов между элементами ПО;

iv) структуры данных;

v) деления требований на элементы;

vi) основных алгоритмов и упорядочивания;

vii) диаграмм.

10.4.5 В спецификации проекта модуля ПО должны быть описаны (одна спецификация проекта модуля ПО на каждый модуль):

i) идентификация всех низших элементов ПО (в настоящем стандарте называемых модулями), прослеживаемых к верхнему уровню;

ii) их детальные интерфейсы со средой и другими модулями с детализацией вводов и выводов;

iii) их уровень полноты безопасности;

iv) детальные алгоритмы и структуры данных.

Каждая спецификация проекта модуля ПО должна быть независимой и позволять кодировку соответствующего модуля.

10.4.6 Каждый модуль ПО должен быть читаемым, понятным и пригодным к тестированию.

10.4.7 Подходящий набор средств, включая методы, языки и компиляторы, должны выбираться для требуемого уровня полноты безопасности ПО на протяжении всего жизненного цикла ПО.

10.4.8 При необходимости должны использоваться средства автоматизированного тестирования и интегрированные средства разработки. При этом следует принимать во внимание потребности лиц, ответственных за верификацию и валидацию.

10.4.9 В степени, требуемой уровнем полноты безопасности ПО, выбранный язык программирования должен иметь транслятор/компилятор, имеющий одно из нижеследующего:

i) сертификат соответствия признанному государственному/международному стандарту;

ii) отчет об оценке с подробным описанием его пригодности к применению по назначению;

iii) механизм избыточного контроля, который обеспечивает выявление ошибок трансляции.

10.4.10 Выбранный язык должен отвечать следующим требованиям:

i) иметь характеристики, позволяющие определять ошибки программирования;

ii) иметь характеристики, подходящие для метода разработки.

10.4.11 Если требования, установленные в 10.4.10, выполнить невозможно, то обоснование выбора любого альтернативного языка с подробным описанием его пригодности к применению по назначению должно быть отражено в спецификации архитектуры ПО или в плане обеспечения качества ПО.

10.4.12 Для разработки всего ПО должны быть разработаны и использоваться стандарты кодирования. На них должна быть дана ссылка в плане обеспечения качества ПО (см. 15.4.5).

10.4.13 Стандарты кодирования должны устанавливать правильные методы программирования, запрещать использование небезопасных свойств языка и описывать процедуры документального оформления исходного кода. Каждый модуль ПО должен содержать в исходном коде как минимум следующую информацию:

– автор;

– история конфигурации;

– краткое описание.

Для данной информации рекомендуется использовать стандартную форму. Она должна быть единой для всех модулей.

10.4.14 Тестирование модуля ПО: каждый модуль должен иметь спецификацию тестирования модуля ПО, согласно которой проводится тестирование. Это тестирование устанавливает, что каждый модуль выполняет назначенную ему функцию. В спецификации тестирования модуля ПО должна быть определена требуемая степень эффективности тестирования.

Должен быть создан отчет о тестировании модуля ПО, который должен соответствовать следующим требованиям:

i) должен содержать справку о результатах тестирования и информацию о том, соответствует ли каждый модуль требованиям спецификации проектирования модуля ПО;

ii) должен содержать справку об условиях тестирования, которая создается по каждому модулю и показывает, что все инструкции исходного кода выполнялись хотя бы раз;

iii) должен быть выполнен в форме, удобной для проверки;

iv) набор данных тестирования и его результаты должны быть оформлены в машиночитаемой форме для последующего анализа; тесты должны быть воспроизводимыми и проводиться автоматическими средствами, если это осуществимо.

Проверка соответствия модуля спецификации тестирования проводится при верификации (см. раздел 11).

10.4.15 Согласно уровню полноты безопасности ПО, выбранный метод проектирования должен обладать свойствами, способствующими:

i) обобщению, модульности и другим свойствам, которые определяют сложность;

ii) четкому и точному выражению:

– функциональности;

– информационного потока между элементами;

– упорядоченной и привязанной ко времени информации;

– согласованности;

– структуры и свойств данных;

iii) восприятию людьми;

iv) верификации и валидации.

10.4.16 Выбранный метод разработки должен обладать свойствами, способствующими обслуживанию ПО. Такие свойства должны включать модульность, сокрытие информации и инкапсуляцию.

10.4.17 Интеграция модулей ПО должна являться процессом постепенного объединения отдельных и ранее опробованных модулей ПО в единое целое (или ряд составляющих подсистем), чтобы интерфейсы модулей и объединенное ПО могли быть в достаточной мере опробованы перед интеграцией и тестированием системы.

10.4.18 В контексте настоящего стандарта и в степени, соответствующей заданному уровню полноты безопасности ПО, прослеживаемость должна в особенности затрагивать:

i) прослеживаемость требований к проектированию или другим объектам, соответствующим им;

ii) прослеживаемость объектов проектирования к объектам программирования, которые реализуют их.

11 Верификация и тестирование ПО

11.1 Цель

Проведение тестирования и оценивания изделий на конкретной стадии в степени, требуемой уровнем полноты безопасности ПО, для обеспечения корректности и согласованности с входными для данной стадии изделиями и стандартами.

11.2 Входные документы

1) Спецификация требований к системе.
2) Спецификация требований безопасности ПО.
3) Спецификация требований к ПО.
4) Спецификация требований к тестированию ПО.
5) Спецификация архитектуры ПО.
6) Спецификация проектирования ПО.
7) Спецификации проекта модуля ПО.
8) Спецификации тестирования модуля ПО.
9) Исходный код ПО и сопроводительная документация.
10) План обеспечения качества ПО.
11) Отчет тестирования модуля ПО.

11.3 Выходные документы

1) План верификации ПО.
2) Отчет о верификации требований к ПО.
3) Отчет о верификации архитектуры и проекта ПО.
4) Отчет о верификации модуля ПО.
5) Отчет о верификации исходного кода ПО.
6) План тестирования интеграции ПО.
7) Отчет о тестировании интеграции ПО.

11.4 Требования

11.4.1 Чтобы деятельность по верификации могла быть должным образом установлена, должен быть разработан план верификации ПО, который также может быть предусмотрен для конкретной разработки или других нужд верификации. В процессе разработки (и в зависимости от размера системы) план может быть разделен на ряд более мелких документов и естественным образом дополняться по мере того, как потребности верификации становятся более ясными.

11.4.2 В плане верификации ПО должны быть зарегистрированы все критерии, методы и средства, подлежащие применению в ходе верификации на данной стадии.

11.4.3 В плане верификации ПО должна описываться деятельность, целью которая является гарантирование правильности и последовательности по отношению к изделиям и стандартам, являющимся входными на данной стадии.

11.4.4 В плане верификации ПО должны быть описаны:

i) выбор стратегии и методов верификации. Во избежание чрезмерной сложности при оценке верификации и тестирования предпочтение должно быть отдано выбору тестовых данных, методов и т. д., которые сами по себе могут быть легко подвергнуты анализу;

ii) выбор и использование испытательного оборудования для тестирования ПО;

iii) выбор и документальное оформление верификации;

iv) оценивание полученных результатов верификации;

v) оценивание требований надежности;

vi) роли и ответственность лиц, вовлеченных в процесс тестирования;

vii) уровень тестового покрытия, который необходимо достичь.

11.4.5 В плане тестирования интеграции ПО должны быть указаны:

i) тестовые данные и результаты тестов;

ii) типы тестов, подлежащих проведению;

iii) условия тестирования, средства, конфигурация и программы;

iv) критерии тестирования, на основе которых будет определяться завершение тестирования.

11.4.6 На каждой стадии разработки необходимо удостовериться, что выполняются требования к функциональности, надежности, рабочим характеристикам и безопасности.

11.4.7 Верификация должна проводиться независимой стороной в степени, требуемой уровнем полноты безопасности ПО, как показано на рисунке 5.

11.4.8 Тестирование, которое было проведено разработчиком до верификации и не было документально оформлено в полной мере, не должно рассматриваться как часть верификации.

11.4.9 Результаты каждой верификации должны сохраняться в форме, которая определена или на которую дана ссылка в плане верификации ПО, чтобы данная информация могла быть проверена.

11.4.10 После каждой верификации должен создаваться отчет о верификации, содержащий подтверждение того, что ПО прошло верификацию, или причины получения отрицательного результата данной верификации. Отчеты о верификации должны содержать описание:

i) элементов, которые не соответствуют спецификации требований к ПО, спецификации проектирования ПО или спецификации проектирования модуля ПО;

ii) элементов, которые не соответствуют плану обеспечения качества ПО;

iii) модулей, данных, структур и алгоритмов, плохо адаптированных к проблеме;

iv) выявленных ошибок или недостатков;

v) подлинности и конфигурации верифицированных элементов.

11.4.11 Верификация требований к ПО: после утверждения спецификации требований к ПО при верификации должны рассматриваться:

i) соответствие спецификации требований к ПО в части выполнения требований, установленных в спецификации требований к системе, спецификации требований безопасности системы и плане обеспечения качества ПО;

ii) соответствие спецификации тестирования требований ПО как тестирование спецификации требований к ПО;

iii) внутренняя непротиворечивость спецификации требований к ПО.

Результаты должны быть отражены в отчете о верификации требований к ПО.

11.4.12 Верификация архитектуры и проектирования ПО: после утверждения спецификации архитектуры ПО и спецификации проектирования ПО при верификации должны рассматриваться:

i) соответствие спецификации архитектуры ПО и спецификации проектирования ПО спецификации требований к ПО;

ii) соответствие спецификации проектирования ПО спецификации требований к ПО в части непротиворечивости и завершенности;

iii) соответствие плана тестирования интеграции ПО в качестве набора тестовых данных спецификации архитектуры ПО и спецификации проектирования ПО;

iv) внутренняя непротиворечивость спецификации архитектуры ПО и спецификации проектирования ПО.

Результаты должны быть отражены в отчете о верификации архитектуры ПО и проектирования ПО.

11.4.13 Верификация модуля ПО: после утверждения каждой спецификации проектирования модуля ПО при верификации должны рассматриваться:

i) соответствие спецификации проектирования модуля ПО спецификации проектирования ПО;

ii) соответствие спецификации тестирования модуля ПО как набора тестовых данных спецификации проектирования модуля ПО;

iii) разделение спецификации проектирования ПО на модули ПО и спецификации проектирования модулей ПО с учетом:

– выполнимости требуемых действий;

– контролепригодности для дальнейшей верификации;

– удобочитаемости для группы разработки и верификации;

– удобства сопровождения для дальнейшего развития;

iv) соответствие отчетов о тестировании модулей ПО как записей тестирования, проведенных согласно спецификации тестирования модуля ПО.

Результаты должны быть отражены в отчете о верификации модуля ПО.

11.4.14 Верификация исходного кода ПО: в степени, требуемой уровнем полноты безопасности ПО, исходный код ПО должен проверяться для обеспечения соответствия спецификации проектирования модуля ПО и плану обеспечения качества ПО. Данная деятельность включает проверку для выявления того, корректно ли были применены стандарты кодирования.

Результаты должны быть отражены в отчете о верификации исходного кода ПО.

11.4.15 Отчет о тестировании интеграции ПО должен соответствовать следующим требованиям:

i) отчет о тестировании интеграции ПО должен создаваться с указанием результатов тестирования и подтверждением того, были ли выполнены задачи и критерии плана тестирования интеграции ПО. В случае невыполнения должны быть зарегистрированы причины;

ii) отчет о тестировании интеграции ПО должен быть оформлен в форме, удобной для проверки;

iii) набор тестовых данных и результаты тестирования должны быть оформлены в машиночитаемой форме для последующего анализа;

iv) тестирование должно быть воспроизводимым и по возможности проводиться автоматическими средствами;

v) подлинность и конфигурация верифицированных элементов.

11.4.16 Интеграция ПО и аппаратных средств описана в 12.4.8.

12 Интеграция ПО и аппаратных средств

12.1 Цели

12.1.1 Убедиться, что ПО и аппаратные средства корректно взаимодействуют при выполнении предназначенных им функций.

12.1.2 Объединение ПО и аппаратных средств с обеспечением их совместимости для выполнения спецификации требований безопасности системы и требований заданного уровня полноты безопасности ПО.

12.2 Входные документы

1) Спецификация требований к системе.
2) Спецификация требований безопасности системы.
3) Описание архитектуры системы.
4) Спецификация требований к ПО.
5) Спецификация требований тестирования ПО.
6) Спецификация архитектуры ПО.
7) Спецификация проектирования ПО.
8) Спецификации проектирования модуля ПО.
9) Спецификации тестирования модуля ПО.
10) Исходный код ПО и сопроводительная документация.
11) Документация на аппаратные средства.

12.3 Выходные документы

1) План тестирования интеграции ПО и аппаратных средств.
2) Отчет о тестировании интеграции ПО и аппаратных средств.

12.4 Требования

12.4.1 План тестирования интеграции ПО и аппаратных средств для уровней полноты безопасности ПО выше нуля создается на раннем этапе жизненного цикла разработки для того, чтобы деятельность по интеграции могла быть соответствующим образом урегулирована, план также может быть предусмотрен для конкретного проектирования или других интеграционных нужд. В процессе разработки (и в зависимости от размера системы) план может быть разделен на ряд более мелких документов и дополняться по мере развития проектирования аппаратных средств и ПО, когда детальные потребности интеграции станут более ясными.

12.4.2 План тестирования интеграции ПО и аппаратных средств содержит:

i) наборы тестовых данных и результатов тестирования;

ii) типы тестов, подлежащих проведению;

iii) условия тестирования, включая инструменты, описание вспомогательного ПО и конфигурации;

iv) критерии тестирования, на основе которых будет определяться завершение тестирования.

12.4.3 План тестирования интеграции ПО и аппаратных средств должен различать деятельность, которую разработчик может выполнить в своем помещении, и деятельность, требующую доступа к стороне пользователя.

12.4.4 План тестирования интеграции ПО и аппаратных средств должен различать следующие деятельности:

i) объединение ПО с целевыми аппаратными средствами;

ii) интеграция системы.

12.4.5 Инструменты и оборудование, определенные в плане тестирования интеграции ПО и аппаратных средств, должны быть доступны как можно раньше.

12.4.6 Во время интеграции ПО и аппаратных средств любая модификация или изменение интегрируемой системы должны стать предметом немедленного изучения с целью определения всех модулей, на которые вследствие этих изменений было оказано влияние, и необходимой деятельности по повторной верификации.

12.4.7 Набор тестовых данных и результаты тестирования желательно оформлять в машиночитаемой форме для последующего анализа.

12.4.8 Должен быть создан отчет о тестировании интеграции ПО и аппаратных средств, соответствующий следующим требованиям:

i) отчет о тестировании интеграции ПО и аппаратных средств должен содержать результаты тестирования и устанавливать, были ли выполнены задачи и критерии плана тестирования интеграции ПО и аппаратных средств. Если нет, то это должно быть указано;

ii) отчет о тестировании интеграции программного и аппаратного обеспечения должен быть выполнен в форме, пригодной для проведения проверки;

iii) набор данных тестирования и его результаты регистрируются, желательно в машиночитаемой форме, для последующего анализа;

iv) отчет о тестировании интеграции ПО и аппаратных средств также должен содержать свидетельство того, что лицо, ответственное за верификацию, удовлетворено отчетом в отношении регистрации выполненных тестов в соответствии с планом интеграции ПО и аппаратных средств;

v) отчет о тестировании интеграции ПО и аппаратных средств должен документировать подлинность и конфигурацию всех включенных элементов.

13 Валидация ПО

13.1 Цель

Проведение анализа и тестирования интегрированной системы на соответствие спецификации требований к ПО с особым акцентом на функциональность и безопасность согласно уровню полноты безопасности ПО.

13.2 Входные документы

1) Спецификация требований к ПО.
2) Вся документация по аппаратным средствам и ПО.
3) Спецификация требований безопасности системы.

13.3 Выходные документы

1) План валидации ПО.
2) Отчет о валидации ПО.

13.4 Требования

13.4.1 Анализ и тестирование являются основной деятельностью валидации.

13.4.2 Имитация и моделирование могут быть использованы для дополнения процесса валидации.

13.4.3 План валидации ПО должен быть установлен и детализирован в соответствующей документации.

13.4.4 План валидации ПО должен быть разработан и выполнен, а его результаты – оценены независимой стороной в степени, требуемой уровнем полноты безопасности ПО.

13.4.5 Если это требуется уровнем полноты безопасности ПО, охват и содержание плана валидации ПО должны быть согласованы с экспертом. Данное согласование должно сопровождаться документальным подтверждением присутствия эксперта при тестировании.

13.4.6 План валидации ПО должен включать краткое обоснование выбора стратегии валидации. Обоснование согласно требуемому уровню полноты безопасности ПО должно включать рассмотрение применения:

i) ручных или автоматических методов (или тех и других);

ii) статических или динамических методов (или тех и других);

iii) аналитических или статистических методов (или тех и других).

13.4.7 В плане валидации ПО должны быть определены шаги, необходимые для демонстрации соответствия:

– спецификации требований к ПО;
– спецификации архитектуры ПО;
– спецификации проектирования ПО;

– спецификаций проектирования модулей ПО

в реализации требований безопасности, установленных в спецификациях требований безопасности системы. Лицо, ответственное за валидацию, должно проверить, завершен ли процесс верификации.

13.4.8 Измерительное оборудование, использованное при валидации, подлежит поверке или калибровке. Должна быть продемонстрирована пригодность любых средств, ПО и аппаратных средств, используемых при валидации, для данной цели.

13.4.9 ПО должно тестироваться посредством симуляции входных данных, существующих при обычной эксплуатации, ожидаемых нарушений и нежелательных условий, требующих реакции со стороны системы.

13.4.10 Результаты валидации должны быть документально оформлены в отчете о валидации ПО в форме, пригодной для проверки.

13.4.11 После окончания интеграции аппаратных средств и ПО должен быть создан отчет о валидации ПО, соответствующий следующим требованиям:

i) должно быть установлено, выполнены ли задачи и критерии плана валидации ПО. Если нет, то это должно быть указано;

ii) должны приводиться результаты тестирования и заключение о том, соответствуют ли все ПО на своем целевом оборудовании требованиям, установленным в спецификации требований к ПО;

iii) должна быть предоставлена оценка тестового покрытия в соответствии с требованиями спецификации требований к ПО;

iv) отчет о валидации ПО должен быть выполнен в форме, пригодной для проверки;

v) набор тестовых данных и его результаты регистрируются в машиночитаемой форме для последующего анализа;

vi) тестирование должно быть воспроизводимым и выполняться по возможности автоматическими средствами.

13.4.12 Отчет о валидации ПО должен содержать идентичность и конфигурацию:

i) использованных аппаратных и программных средств;

ii) использованного оборудования;

iii) калибровки оборудования;

iv) использованных моделей симуляции;

v) обнаруженных несоответствий;

vi) выполненных корректирующих мероприятий.

13.4.13 Любые обнаруженные несоответствия, включая найденные ошибки, должны быть четко идентифицированы в отдельном разделе отчета о валидации ПО и включены в сопроводительную документацию на поставленное ПО.

13.4.14 ПО должно тестироваться согласно спецификации тестирования ПО. Это тестирование должно установить, что все требования спецификации требований к ПО были выполнены корректно.

Результаты должны регистрироваться в отчете о валидации ПО.

14 Оценка ПО

14.1 Цель

Определить, соответствует ли ПО установленному уровню полноты безопасности и пригодно ли оно для применения по назначению на основе оценки процессов жизненного цикла и полученных продуктов.

14.2 Входные документы

1) Спецификация требований безопасности системы.

2) Вся документация по аппаратным и программным средствам.

14.3 Выходные документы

Отчет об оценке ПО.

14.4 Требования

14.4.1 Для оборудования, имеющего уровень полноты безопасности ПО 0:

i) эксперт должен подтвердить, что этот уровень полноты безопасности приемлем для данного ПО;

ii) возможно, также потребуется согласовать данный уровень полноты безопасности между поставщиком и пользователем при проведении тендера.

14.4.2 ПО с отчетом об оценке ПО другого эксперта не должно быть объектом полностью новой оценки. Второй эксперт должен проверить ПО на соответствие требуемому уровню полноты безопасности и пригодности для применения на предназначенном для этого компьютере.

14.4.3 Эксперт должен иметь доступ к процессу проектирования и разработки и ко всей документации, связанной с проектом.

14.4.4 Оценка ПО должна выполняться экспертом, который независим от группы разработчиков.

14.4.5 Эксперт должен оценивать ПО системы на пригодность к применению по назначению и правильность выполнения требований спецификации требований безопасности системы.

14.4.6 В степени, требуемой уровнем полноты безопасности ПО, эксперт должен определить, были ли выбраны и применены соответствующие методы на каждой стадии жизненного цикла ПО.

14.4.7 Если это требуется уровнем полноты безопасности ПО, эксперт должен согласовать объем и содержание плана валидации ПО. Это согласование должно содержать информацию о присутствии эксперта при тестировании.

14.4.8 Эксперт может потребовать выполнения дополнительной работы по верификации и валидации по его усмотрению.

14.4.9 Эксперт должен составить отчеты по каждой проверке, в которых детализируются результаты выполненной им оценки.

14.4.10 Если по мнению эксперта ПО пригодно к применению по назначению, то в отчете об оценке ПО должно содержаться заключение относительно уровня полноты безопасности, обеспеченного ПО.

14.4.11 Если ПО непригодно к применению по назначению либо не обеспечен требуемый уровень полноты безопасности ПО, эксперт должен только сообщить о несоответствиях в отчете оценки ПО, не давая указаний относительно каких-либо технических решений.

15 Обеспечение качества ПО

15.1 Цели

15.1.1 Идентификация, мониторинг и управление всей деятельностью, как технической, так и организационной, необходимой для обеспечения требуемого качества ПО. Это требуется для того, чтобы обеспечить качественную защиту от систематических ошибок и возможность проведения контрольных проверок с целью эффективного выполнения работ по верификации и валидации.

15.1.2 Обеспечение свидетельств проведения вышеуказанных работ.

15.2 Входные документы

Все документы каждого этапа жизненного цикла.

15.3 Выходные документы

1) План обеспечения качества ПО.

2) План управления конфигурацией ПО.

Все вышеуказанные планы должны создаваться в начале проекта и актуализироваться на протяжении всего жизненного цикла.

15.4 Требования

15.4.1 Поставщик и/или разработчик должен иметь и применять систему менеджмента качества, соответствующую ISO 9001:2001, в целях выполнения требований настоящего стандарта. Настоятельно рекомендуется аккредитация по ISO/IEC 17025.

15.4.2 Поставщик и/или разработчик и заказчик должны применять в отношении ПО соответствующие требования ISO 9001 согласно руководящим указаниям, приведенным в ISO 9000-3.

15.4.3 Поставщик и/или разработчик должны составить и документально оформить по каждому проекту план обеспечения качества ПО для выполнения требований 15.4.1 и 15.4.2, которые должны выражаться в измеримых показателях, если это возможно.

15.4.4 В плане обеспечения качества ПО должен быть раздел, в котором подробно описывается процесс актуализации на протяжении всего проекта: частота, ответственность и метод.

15.4.5 Все действия, мероприятия, документы и т. д., необходимые в соответствии с ISO 9000-3 и настоящим стандартом, должны быть описаны или снабжаться ссылками в плане обеспечения качества ПО и привязываться к конкретной разработке.

За исключением нулевого уровня полноты безопасности ПО, в плане обеспечения качества ПО как минимум должны быть указаны или обеспечены ссылками нижеследующие элементы.

Это необходимо для обеспечения учета всех аспектов безопасности в ПО в соответствии с его уровнем полноты безопасности.

Настоящий перечень не является исчерпывающим:

i) определение модели для всего жизненного цикла, определение каждого этапа, в том числе:

– работы и элементарные задачи;

– входные и выходные критерии;

– входные и выходные данные каждого этапа;

– основные работы по обеспечению качества на каждом этапе;

– штатное подразделение, отвечающее за отдельную работу и элементарную задачу;

ii) прослеживаемость требований;

iii) прослеживаемость структуры документации;

iv) документация, связанная с разработкой, верификацией и валидацией, эксплуатацией и обслуживанием ПО;

v) процедуры интеграции системы;

vi) применяемые стандарты кодирования;

vii) оценка предыдущих проверок пригодности;

viii) определение системы метрик (количественные показатели) как по продукции, так и по процессу. Для системы метрик программного продукта необходимо делать ссылку на качественные характеристики и инструкции по проведению оценки в соответствии с ISO/IEC 9126.

15.4.6 Управление конфигурацией должно осуществляться в соответствии с инструкциями ISO 9001.

Каждый программный документ должен быть подвергнут контролю конфигурации, прежде чем будет издана его первая утвержденная редакция. Исходный код ПО должен быть передан под контроль конфигурации до начала документируемого тестирования модуля.

Нельзя вносить несанкционированные изменения в любой элемент, находящийся под управляемым контролем конфигурации. Необходимо принять меры предосторожности в целях предотвращения и обнаружения ошибок, возникающих в машиночитаемом коде во время хранения, пересылки, перемещения или копирования.

Управление конфигурацией не должно быть ограничено лишь разработкой и обслуживанием продукта, оно также должно учитывать и используемую в течение всего жизненного цикла среду. Такое расширение необходимо для воспроизводимости разработки и обслуживания, что должно включать конфигурационные файлы, ассемблеры, компиляторы, отладочные программы и все другие используемые средства.

15.4.7 Адекватность и результаты плана верификации ПО должны быть проверены.

15.4.8 Поставщик и/или разработчик должен создать, документально оформить и выполнить процедуры контроля сторонних поставщиков, в том числе:

– методы, обеспечивающие соответствие установленным требованиям ПО, предоставленного сторонними поставщиками. Ранее разработанное ПО должно соответствовать требованиям уровня полноты безопасности ПО и надежности. Новое ПО должно быть разработано и обслуживаться в соответствии с планом обеспечения качества ПО поставщика либо с конкретным планом обеспечения качества ПО, составленным сторонним поставщиком согласно плану обеспечения качества ПО поставщика;

– методы, обеспечивающие достаточность и полноту требований, выдвигаемых стороннему поставщику ПО.

15.4.9 Поставщик и/или разработчик должен создать, документально оформить и выполнить процедуры предоставления отчетности по проблемным вопросам и принятия корректирующих мер. Такие процедуры, как часть системы обеспечения качества, должны отвечать требованиям соответствующих разделов ISO 9001, определяя в частности:

– документацию, необходимую для предоставления отчетности по проблемным вопросам и/или принятия корректирующих мер, с целью обратного влияния на управление;

– как должна анализироваться информация, собираемая в отчетах по проблемам с целью выявления их причин;

– практику, которой необходимо следовать при отчетности, прослеживании и решении проблем, выявленных как на этапе разработки, так и при обслуживании ПО;

– профилактические действия для решения проблем на уровне, соответствующем требуемому уровню полноты безопасности ПО;

– конкретные организационные обязанности в отношении разработки и обслуживания ПО;

– пути применения средств контроля для обеспечения принятия корректирующих мер и их эффективности;

– применяемые формы отчетности;

– требования для повторного проведения тестирования, верификации, валидации и оценки.

Как минимум, организация процесса отчетности по проблемным вопросам и корректирующим мерам должна применяться в течение жизненного цикла ПО, начиная сразу после интеграции ПО и до начала процесса официального утверждения ПО, охватывая также весь этап обслуживания ПО.

16 Обслуживание ПО

16.1 Цель

Обеспечение надлежащего функционирования ПО с сохранением уровня полноты безопасности ПО и надежности при внесении изменений, расширении функциональных возможностей или адаптации ПО.

16.2 Входные документы

Все документы.

16.3 Выходные документы

1) План обслуживания ПО.
2) Записи об изменении ПО.
3) Записи об обслуживании ПО.

16.4 Требования

16.4.1 Как минимум, обслуживание должно осуществляться в соответствии с инструкциями ISO 9000.

Помимо этого, необходимо выполнять нижеследующие требования по обслуживанию ПО.

16.4.2 Пригодность к обслуживанию должна быть заложена в системе ПО, в частности, за счет соблюдения требований раздела 10. Для установления требований и осуществления проверки минимального уровня пригодности к обслуживанию необходимо также руководствоваться ISO/IEC 9126-1.

16.4.3 Процедуры по обслуживанию ПО должны быть установлены и указаны в плане обслуживания ПО. Эти процедуры также включают:

i) контроль отчетности об ошибках, журналов регистрации ошибок, записей об обслуживании, изменений авторизации и конфигурации ПО/системы;

ii) верификацию, валидацию и оценку;

iii) определение уполномоченного органа, утверждающего измененное ПО.

16.4.4 Работы по обслуживанию должны подвергаться проверке согласно плану обслуживания ПО с периодичностью, указанной в плане обеспечения качества ПО.

16.4.5 Обслуживание должно осуществляться с тем же уровнем оценки, средств, документации, планирования и организации, как и при первоначальной разработке системы. Это распространяется также на управление конфигурацией, управление изменениями, документооборот и независимость вовлеченных сторон.

16.4.6 Настоящий стандарт не имеет обратной силы. Поэтому он распространяется главным образом на новые разработки и полностью применяется для существующих систем, если они подвергаются значительным модификациям. Для 3-го или 4-го уровня полноты безопасности ПО подрядные организации, прежде чем приступить к работе по внесению изменений, должны принять решение о том, рассматриваются ли работы по обслуживанию как значительные или незначительные или же имеющиеся методы обслуживания пригодны для системы. Поставщику ПО для уровней полноты безопасности 0, 1 и 2 необходимо принять аналогичное решение.

16.4.7 Организация контроля стороннего поставщика, отчетности по проблемным вопросам и корректирующих мер должна осуществляться на основании критериев, указанных в соответствующих пунктах раздела «Обеспечение качества ПО».

16.4.8 Записи об обслуживании ПО должны осуществляться для каждого программного элемента с момента, предваряющего его первый выпуск, с последующим их ведением. Помимо требований соответствующих пунктов раздела «Отчеты и записи о сопровождении» ISO 9000-3 такие записи должны включать:

i) ссылку на все записи об изменении ПО для данного программного элемента;

ii) информацию о последовательных изменениях;

iii) тестовые данные для компонентов, включая повторную валидацию и данные регрессивного тестирования;

iv) историю конфигурации ПО.

16.4.9 Для каждой деятельности по обслуживанию должны устанавливаться записи об изменениях ПО. Такие записи должны включать:

i) запрос на внесение изменения или видоизменения;

ii) анализ влияния работ по обслуживанию на систему в целом, в том числе аппаратные средства, ПО, взаимодействие с пользователем, средой и другое возможное взаимодействие;

iii) подробное описание модификации или изменения;

iv) повторную валидацию, регрессивное тестирование и повторную оценку модификации или изменения, насколько этого требует уровень полноты безопасности ПО. Ответственность за повторную валидацию может варьироваться по проектам в зависимости от уровня полноты безопасности ПО. Помимо этого, влияние модификации или изменения на процесс повторной валидации может быть ограничено различными уровнями систем (только измененные модули, все идентифицированные затронутые модули, система целиком). Поэтому план валидации ПО должен рассматривать обе проблемы согласно уровню полноты безопасности ПО. Степень независимости повторной валидации должна быть такой же, как для первичной валидации.

17 Системы, конфигурируемые на основе прикладных данных

17.1 Цели

17.1.1 Характерной особенностью железнодорожных систем управления и защиты является необходимость проектирования каждой системы с соблюдением индивидуальных требований для конкретного применения. Системы, конфигурированные на основе прикладных данных, позволяют использовать универсальное ПО, одобренное как тип, с индивидуальными требованиями для каждого случая, определяемыми как данные (конкретные прикладные данные). Эти данные обычно представлены в виде таблиц или на конкретном прикладном языке, которые интерпретируются универсальным ПО.

17.1.2 Для систем, критичных к безопасности, требуется высокий уровень ресурсов для разработки ПО, обеспечивающего необходимый уровень полноты безопасности системы, что делает весьма привлекательным выбор системы, конфигурированной на основе прикладных данных, так как позволяет повторно использовать универсальное ПО. Тем не менее, поскольку безопасная работа системы, по всей вероятности, будет зависеть от правильности данных, то процедуры, применяемые для выработки данных, должны соответствовать уровню полноты безопасности системы.

17.1.3 Нижеследующие подразделы описывают требования настоящего стандарта для первоначальной разработки универсального ПО для систем, конфигурированных на основе прикладных данных, а также для последующей разработки каждого набора данных специфики применения.

17.2 Входные документы

1) Спецификация требований к ПО.
2) Спецификация архитектуры ПО.

17.3 Выходные документы

1) Спецификация требований по применению.
2) План подготовки данных.
3) План тестирования данных.
4) Отчет о тестировании данных.

17.4 Требования

17.4.1 Жизненный цикл подготовки данных

На рисунке 6 показан жизненный цикл приложения, в котором используется аппаратное и универсальное ПО вместе с конкретными прикладными данными. Ниже следуют этапы жизненного цикла.

17.4.1.1 Спецификация требований приложения

К приложению должны быть установлены требования. Они также должны содержать конкретные требования к отдельной установке (например, схема путей, расположение средств сигнализации, ограничения скорости), а также стандарты, которым приложение должно соответствовать (например, правила сигнализации, уровни полноты безопасности системы).

17.4.1.2 Общий рабочий проект

Должна быть определена архитектура системы, а также указано количество и тип применяемых универсальных компонентов. Те компоненты, которые содержат ПО, должны быть разработаны в соответствии с настоящим стандартом. Функции, указанные в требованиях, должны быть распределены по компонентам, а также определено физическое нахождение каждого компонента.

17.4.1.3 Подготовка данных

Процесс подготовки данных должен включать выработку конкретной информации (например, управляющие таблицы), разработку исходного кода данных и его компиляцию, проверку и другие действия по верификации, а также тестирование прикладных данных.

17.4.1.4 Интеграция и приемка

Для некоторых систем прикладные данные будут интегрированы с общим аппаратным и программным обеспечением для тестирования разработчиком, прежде чем они будут установлены на месте работы. Этого может не понадобиться, если достаточную степень достоверности можно получить другим способом. Затем оборудование устанавливается на месте работ и на новом оборудовании выполняется тестирование интеграции. В конечном итоге система вводится в эксплуатацию как полностью готовая, процесс окончательной приемки осуществляется на укомплектованном оборудовании.

17.4.1.5 Валидация и оценка

Деятельность по валидации и оценке должна быть направлена на проверку работы на каждом этапе жизненного цикла.

17.4.2 Средства и методы подготовки данных

Для каждого нового типа системы, конфигурируемой на основе прикладных данных, должны быть разработаны конкретные методы подготовки данных и средств, чтобы жизненный цикл подготовки данных, указанный в 17.4.1, мог быть применен для установки новой системы. Разработка таких методов и средств должна выполняться в соответствии с настоящим стандартом, параллельно с разработкой общего программного и аппаратного обеспечения для системы. Цель работ по верификации, валидации и оценке – обеспечение совместимости средств подготовки данных и универсального ПО.

17.4.2.1 На этапе проектирования ПО для системы, конфигурируемой на основе прикладных данных, должен быть выработан план подготовки данных, чтобы определить структуру документации для процесса подготовки данных. Эти документы должны быть связаны с моделью жизненного цикла подготовки данных, описанной в 17.4.1. В плане должны быть указаны методы и средства подготовки данных, которые применяются в соответствии с уровнем полноты безопасности системы. План должен содержать требования к независимости между персоналом, выполняющим верификацию, валидацию и задания по разработке.

17.4.2.2 План подготовки данных должен выделять уровень полноты безопасности для любых средств аппаратного и программного обеспечения, используемых в жизненном цикле подготовки данных. Такой уровень полноты безопасности является производной от требуемого уровня полноты безопасности и степени, до которой выходные данные каждого средства проверяются другими ручными или автоматизированными методами.

17.4.2.3 При возможности план подготовки данных должен предусматривать систему обозначений по описанию требований и проектов, которые знакомы инженерам-прикладникам, например стандартные планы сигналов и управляющие таблицы. При введении новых обозначений должна быть предоставлена необходимая документация пользователя, а также проведено обучение, если это необходимо.

17.4.2.4 Отчеты по верификации, тестированию, валидации и оценке, подтверждающие, что подготовка данных выполнена в соответствии с планом, могут быть стандартизированы в виде проверочных листов в целях минимизации рабочей нагрузки при составлении документации по каждой установке системы. Эта информация должна содержаться в плане тестирования данных, а результаты отмечены в отчете тестирования данных.

17.4.2.5 Все данные и соответствующая документация должны соответствовать требованиям по управлению конфигурацией раздела 15 настоящего стандарта. Записи по управлению конфигурацией должны содержать версии универсального ПО, для которого были разработаны данные, а также версии средств, примененных в процессе подготовки данных.

17.4.3 Разработка ПО

Разработка универсального ПО для использования в системе, конфигурируемой на основе прикладных данных, должна соответствовать требованиям разделов 1 – 16.

17.4.3.1 На этапе составления спецификации требований к ПО должны быть определены те функции, которые используют прикладные данные в каждой системе и подсистеме. Уровень полноты безопасности системы, выделенный для каждой подсистемы, будет определять стандарты, применяемые к последующей разработке данных для всех установок системы.

17.4.3.2 На этапе проектирования ПО должны быть подробно определены интерфейсы между универсальным ПО и прикладными данными, если это уже не было определено на предшествующих этапах жизненного цикла, например, как результат требований по использованию существующего конкретного языка.

17.4.3.3 На этапе проектирования программных модулей должно быть обеспечено строгое разделение между кодом и данными программы, то есть должна существовать возможность повторно компилировать и обновлять либо универсальное ПО, либо данные без необходимости обновлять другое, если не было внесено изменение в установленном интерфейсе между ПО и данными. Также конкретные прикладные данные следует отделять от других данных.

17.4.3.4 На этапе обслуживания ПО методы контроля изменений должны обеспечить возможность внесения изменения в универсальное ПО только после того, как будет установлено, что либо измененное ПО совместимо с первоначальными данными, либо данные были пересмотрены, как это требуется.

17.4.3.5 В процессе верификации и на стадии валидации ПО должно быть обращено особое внимание на то, что все значимые комбинации данных учтены.

17.4.3.6 Универсальное ПО должно быть разработано так, чтобы оно могло обнаруживать поврежденные данные конфигурации, где это возможно.

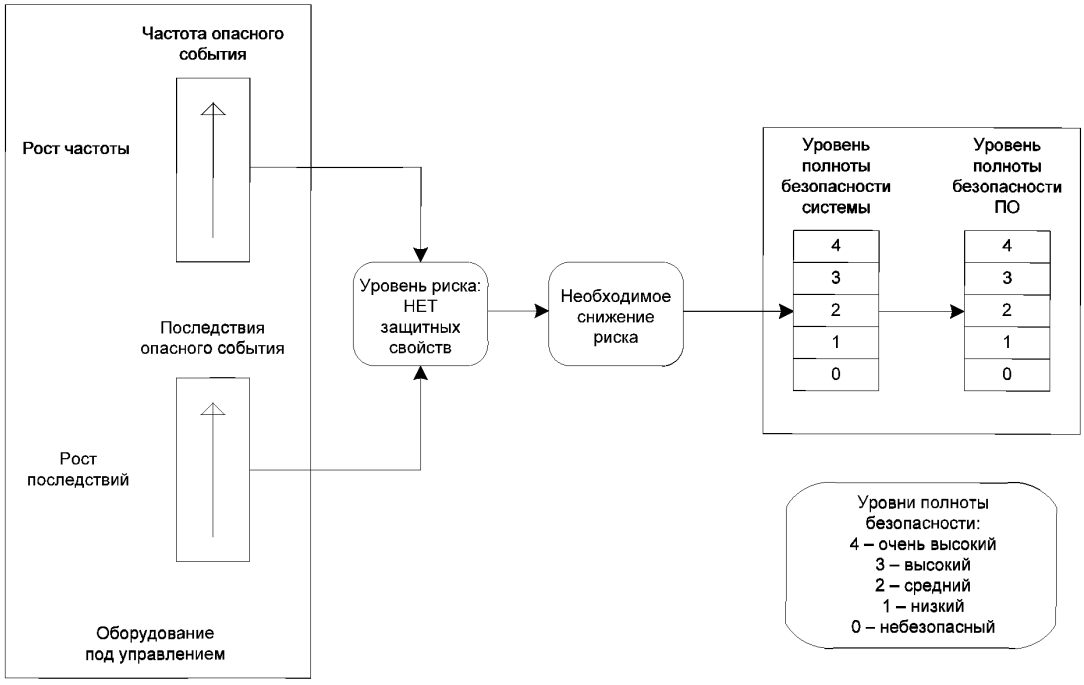

Рисунок 1 – Уровни полноты безопасности систем, связанных с обеспечением безопасности

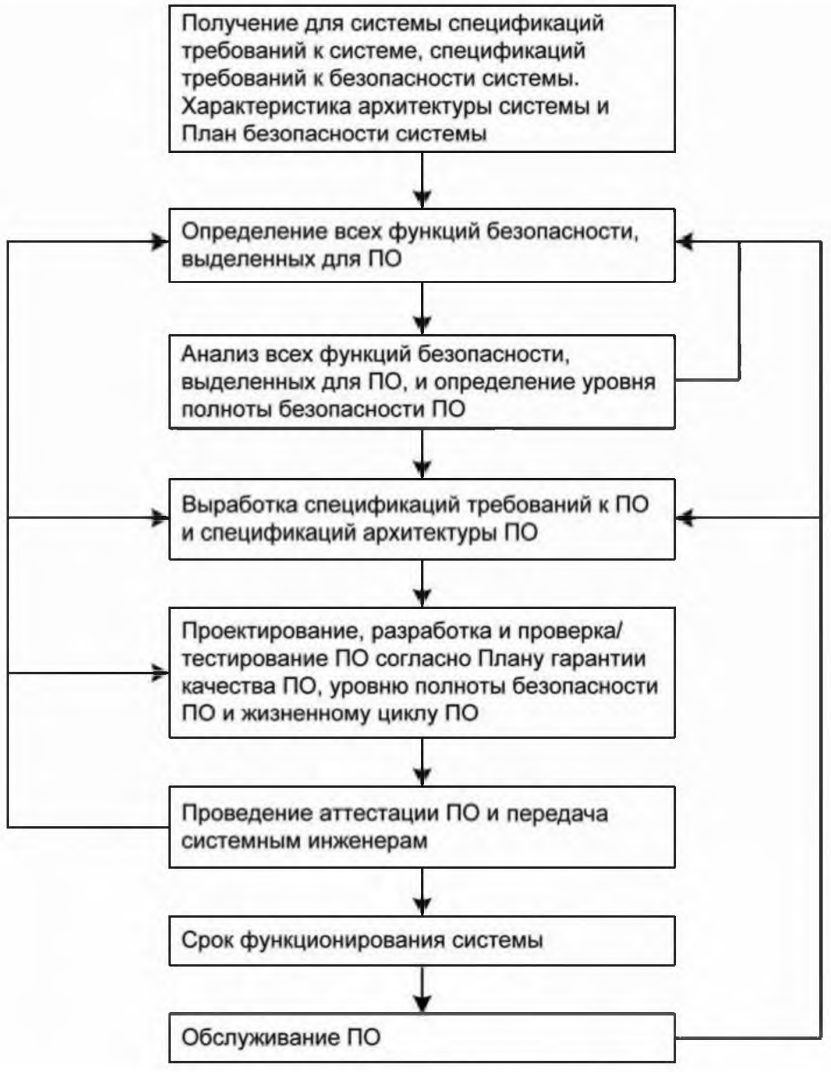

Рисунок 2 – Схема безопасности ПО

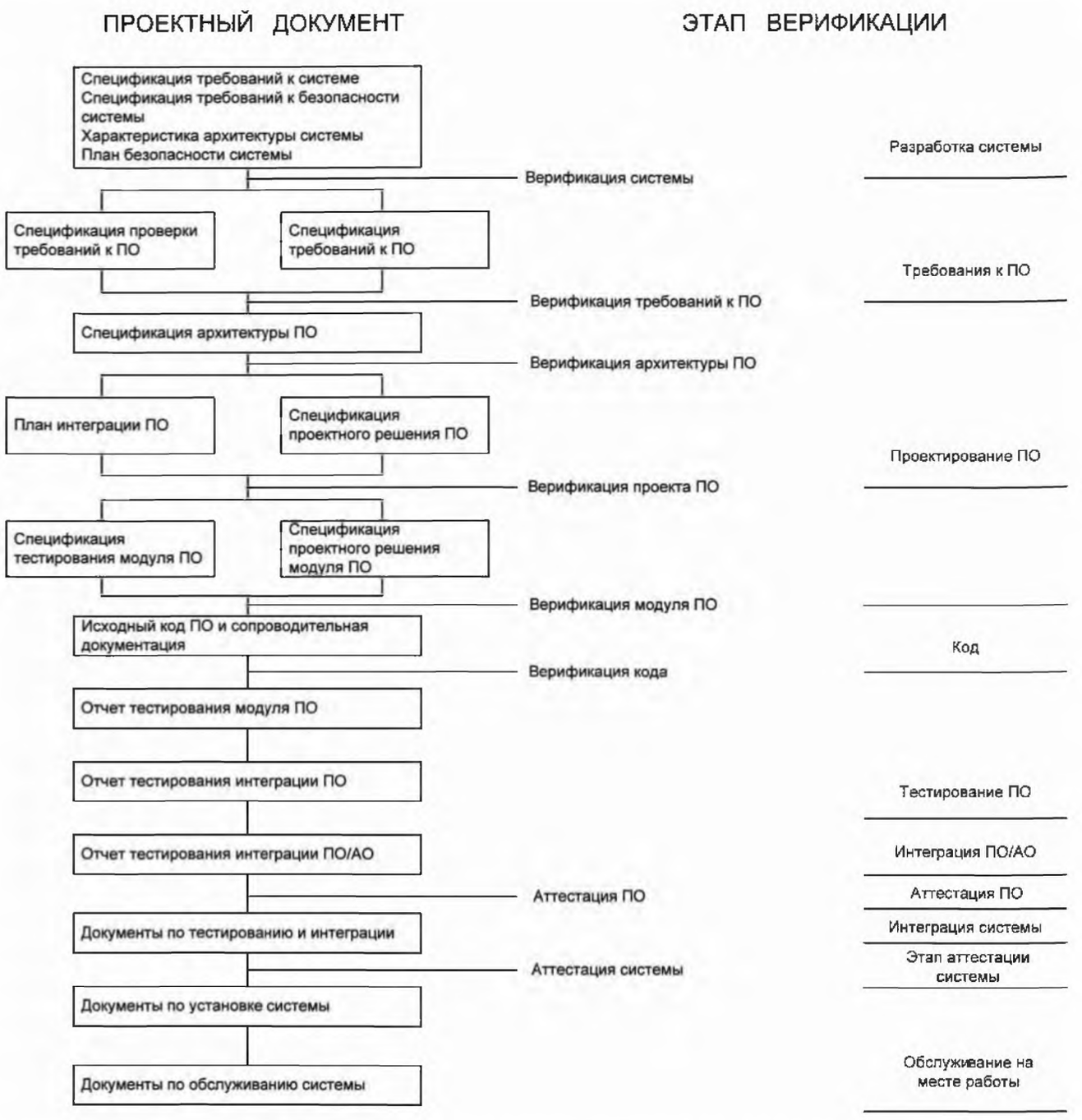

Рисунок 3 – Развитие жизненного цикла разработки 1

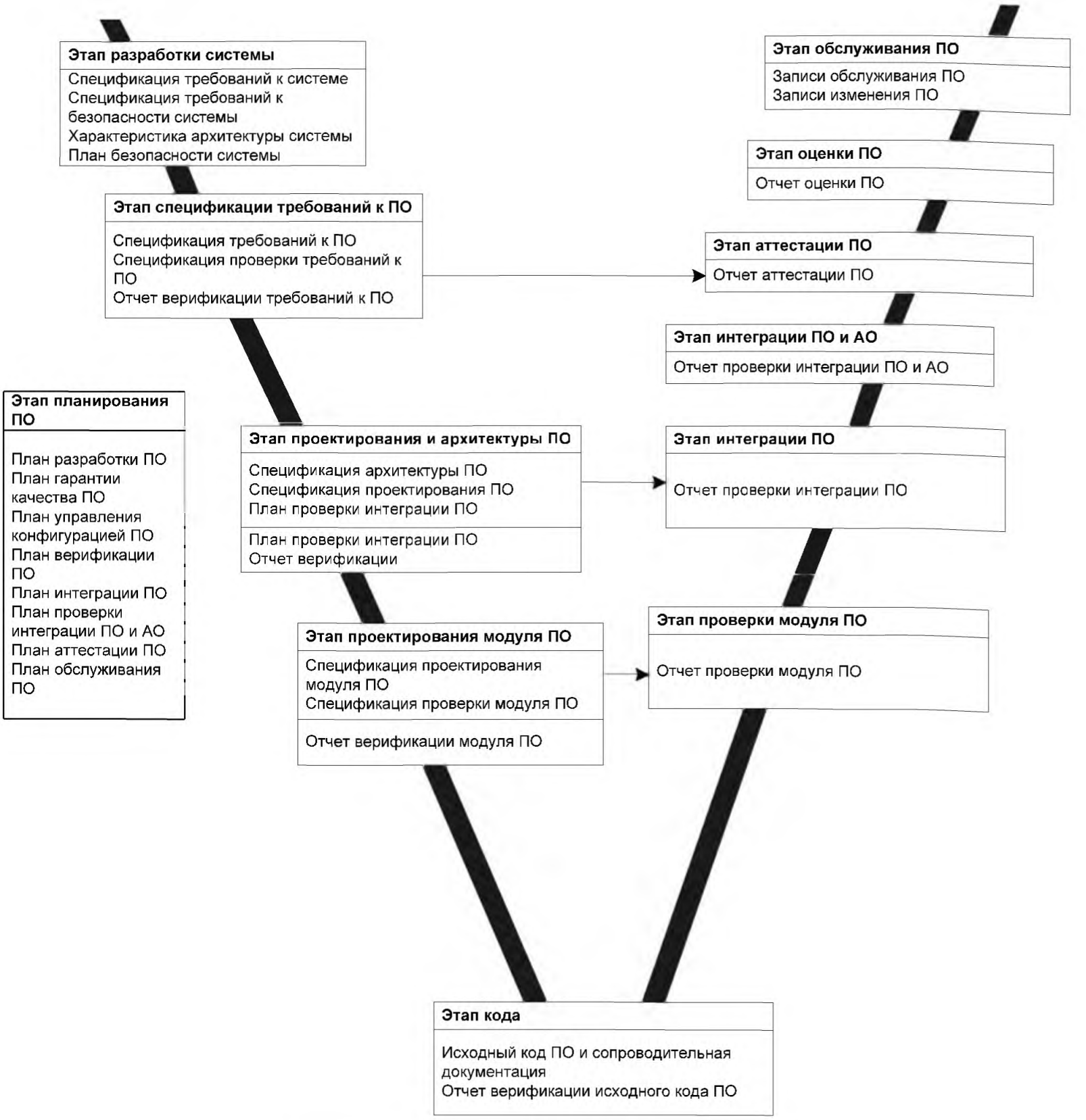

Рисунок 4 – Развитие жизненного цикла разработки 2

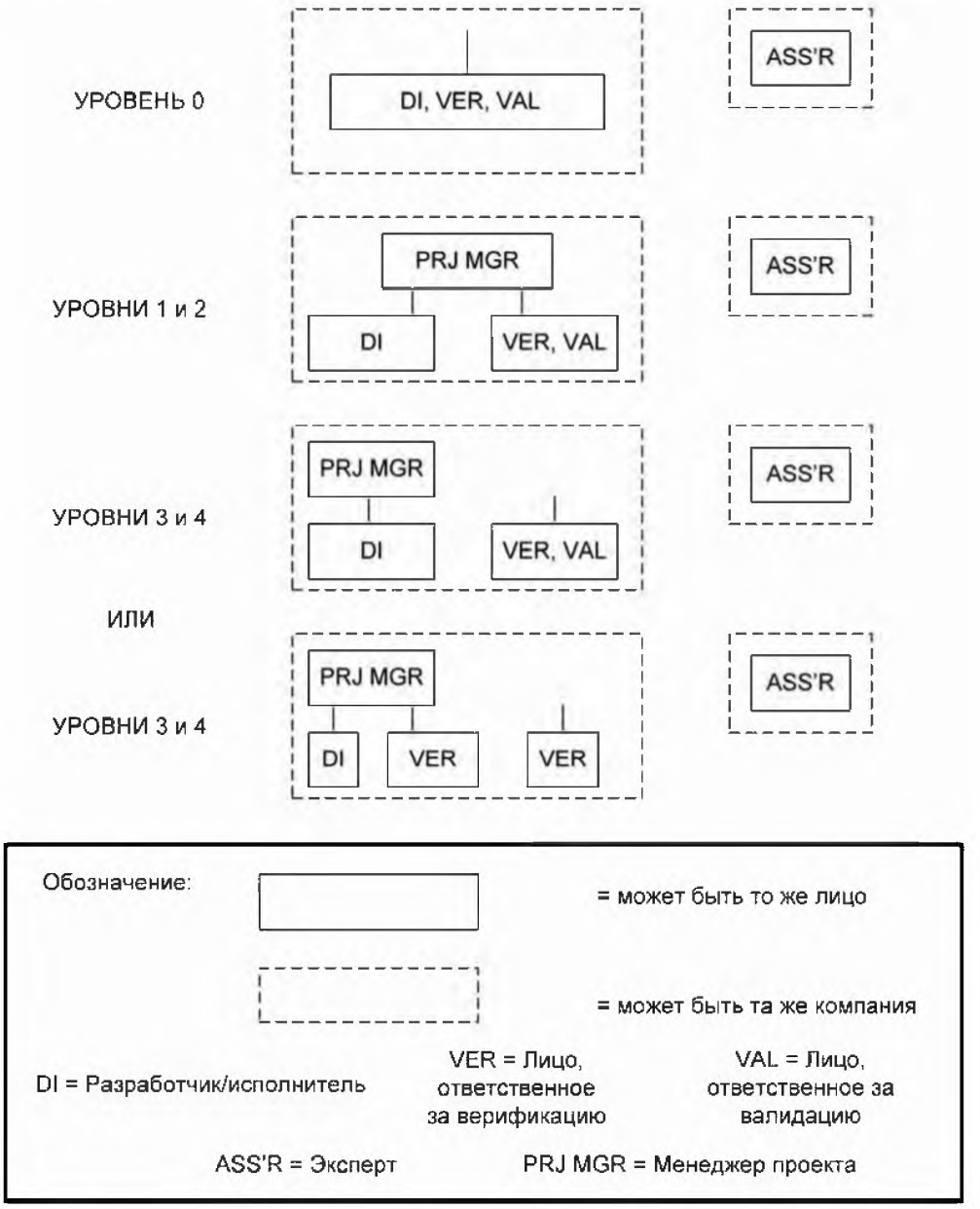

Рисунок 5 – Независимость персонала при различных уровнях полноты безопасности ПО

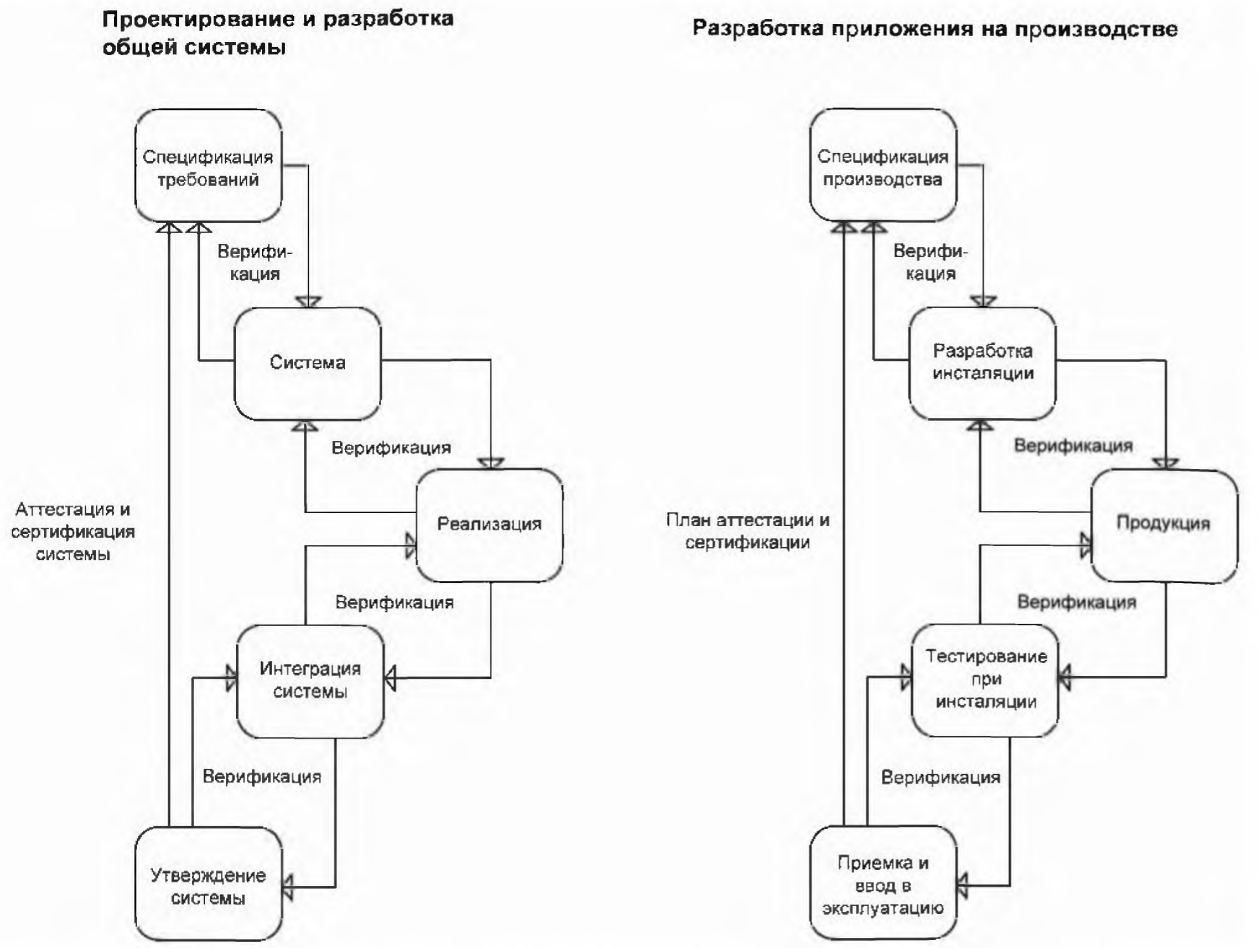

Рисунок 6 – Связь между разработкой общей системы и разработкой приложения

Приложение А
(обязательное)

Критерии выбора методов и мероприятий

Каждый из разделов 7 – 16 имеет соответствующую таблицу, в которой указаны средства достижения соответствия. Есть таблицы более низкого уровня, детальные таблицы (dt), в которых записи раскрываются более подробно. Например, полуформальные методы подробно описываются в детальной таблице dt7 (таблица А.18). Также имеется справочное приложение В, на которое есть ссылка в таблицах к разделам.

Для каждого метода или мероприятия, указанного в таблице, существует требование в отношении каждого уровня полноты безопасности ПО от 1 до 4, а также для уровня 0, не связанного с обеспечением безопасности. В настоящем стандарте требования для уровней полноты безопасности ПО 1 и 2 одинаковые для каждого метода. Аналогично, для каждого метода одинаковые требования безопасности для уровней полноты безопасности ПО 3 и 4. Такими требованиями могут быть:

– «О» – данный символ означает, что применение метода является обязательным;

– «СР» – данный символ означает, что метод или мероприятие строго рекомендованы для указанного уровня полноты безопасности. Если этот метод или мероприятие не применяется, то необходимо предоставить подробное обоснование этого в плане обеспечения качества ПО или ином документе, на который приведена ссылка в плане обеспечения качества ПО;

– «Р» – данный символ означает, что метод или мероприятие рекомендованы для указанного уровня полноты безопасности. Это более низкий уровень рекомендации, чем «СР», и такие методы могут совмещаться, образовывая часть пакета;

– «–» – данный символ означает, что метод или мероприятие не имеют рекомендации по применению или неприменению;

– «НР» – данный символ означает, что метод или мероприятие однозначно не рекомендованы для указанного уровня полноты безопасности. Если метод или мероприятие применяются, то необходимо предоставить подробное обоснование этого в плане обеспечения качества ПО или ином документе, на который приведена ссылка в плане обеспечения качества ПО.

Сочетания методов или мероприятий должны быть указаны в плане обеспечения качества ПО (или ином документе, на который приведена ссылка в плане обеспечения качества ПО) с одним или несколькими выбранными методами или мероприятиями, если примечание к таблице не содержит иных требований. Эти примечания могут содержать ссылки на одобренные методы или одобренные сочетания методов. Если такие методы или сочетания методов применяются, то эксперт принимает их как правильные методы и должен интересоваться исключительно корректностью их применения. Если применяется другой набор методов, который надлежащим образом обоснован, то эксперт может признать его приемлемым.

Таблицы к разделам

Таблица А.1 – Вопросы и документация жизненного цикла (раздел 7)

Документация	Уровень полноты безопасности ПО				
	0	1	2	3	4
1. Документы по планированию ПО	Р	СР	СР	СР	СР
2. Документы по требованиям к ПО	Р	СР	СР	СР	СР
3. Документы по проектированию ПО	–	СР	СР	СР	СР
4. Документы по модулю ПО	–	СР	СР	СР	СР
5. Документация и исходный код	Р	СР	СР	СР	СР
6. Отчеты о тестировании ПО	–	СР	СР	СР	СР
7. Отчеты об интеграции ПО и АО	–	СР	СР	СР	СР
8. Отчет о валидации ПО	Р	СР	СР	СР	СР
9. Отчет об оценке ПО	–	СР	СР	СР	СР
10. Записи об обслуживании ПО	Р	СР	СР	СР	СР
Требование – Соответствие ISO 9000 подразумевает разработку документации для всех уровней полноты безопасности ПО. Для уровня 0 разработчик должен выбрать подходящий тип документа.					

Таблица А.2 – Спецификация требований к ПО (раздел 8)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Формальные методы, в том числе, например, CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z и B	B.30	–	Р	Р	СР	СР
2. Полуформальные методы	dt7	Р	Р	Р	СР	СР
3. Структурная методология, в том числе, например, JSD, MASCOT, SADT, SDL, SSADM и Yourdon B.60	B.60	Р	СР	СР	СР	СР
Требования 1 Спецификация требований к ПО всегда требует описания проблемы на естественном языке и необходимых математических выражений, которые отражают применение. 2 Таблица отражает дополнительные требования по четкому и ясному определению спецификации. Один или несколько из этих методов должны быть выбраны для соответствия применяемому уровню полноты безопасности ПО.						

Таблица А.3 – Архитектура ПО (раздел 9)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Защитное программирование	B.15	–	Р	Р	СР	СР
2. Выявление сбоев и их диагностика	B.27	–	Р	Р	СР	СР
3. Коды, устраняющие ошибки	B.20	–	Р	Р	СР	СР
4. Коды, обнаруживающие ошибки	B.20	–	Р	Р	СР	СР
5. Программирование с регистрацией отказов	B.25	–	Р	Р	СР	СР
6. Метод мониторинга безопасности	B.54	–	Р	Р	Р	Р
7. Многовариантное программирование	B.17	–	Р	Р	СР	СР
8. Блок восстановления	B.50	–	Р	Р	Р	Р
9. Восстановление предыдущего состояния файла	B.5	–	НР	НР	НР	НР
10. Восстановление без возврата	B.32	–	НР	НР	НР	НР
11. Механизмы восстановления после сбоев путем повторного запуска	B.53	–	Р	Р	Р	Р
12. Запоминание решенных проблем	B.39	–	Р	Р	СР	СР
13. Искусственный интеллект. Устранение сбоев	B.1	–	НР	НР	НР	НР

Окончание таблицы А.3

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
14. Динамическая реконфигурация	B.18	–	HP	HP	HP	HP
15. SEEA – анализ последствий ошибки программного обеспечения	B.26		P	P	CP	CP
16. Анализ дерева сбоев	B.28		P	P	CP	CP

Требования

1 Утвержденные сочетания методов для уровней полноты безопасности ПО 3 и 4 должны быть следующими:

a) 1, 7 и один из 4, 5 или 12;

b) 1, 4 и 5;

c) 1, 4 и 12;

d) 1, 2 и 4;

e) 1 и 4 и один из 15 и 16.

2 Один или несколько из этих методов, кроме 3, 9, 10, 13 и 14, должны быть выбраны для выполнения требований для уровней полноты безопасности ПО 1 и 2.

3 Некоторые из этих вопросов могут быть определены на уровне системы.

4 Коды, корректирующие ошибки, могут быть использованы в соответствии с требованиями IEC 62280-1 и IEC 62280-2.

Таблица А.4 – Проектирование и внедрение ПО (раздел 10)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Формальные методы, в том числе, например, CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z и B	B.30	–	P	P	CP	CP
2. Полуформальные методы	dt7	P	CP	CP	CP	CP
3. Структурная методология, в том числе, например, JSD, MASCOT, SADT, SDL, SSADM и Yourdon	B.60	P	CP	O	O	O
4. Модульный подход	dt9	CP	O	O	O	O
5. Стандарты проектирования и кодирования	dt1	CP	CP	CP	O	O
6. Поддающиеся анализу программы	B.2	CP	CP	CP	CP	CP
7. Языки программирования со строгим контролем типов	B.57	P	CP	CP	CP	CP
8. Структурное программирование	B.61	P	CP	CP	CP	CP
9. Язык программирования	dt4	P	CP	CP	CP	CP
10. Подмножество языка	B.38	P	–	–	CP	CP
11. Сертифицированные средства и трансляторы	B.7	P	CP	CP	CP	CP
12. Транслятор, испытанный практическим применением	B.65	CP	CP	CP	CP	CP
13. Библиотека надежных/проверенных модулей и компонентов	B.40	P	P	P	P	P
14. Функциональное тестирование/тестирование методом «черного ящика»	dt3	CP	CP	CP	M	M
15. Тестирование эксплуатационных параметров	dt6	–	CP	CP	CP	CP
16. Тестирование интерфейса	B.37	CP	CP	CP	CP	CP
17. Записи данных и анализ	B.13	CP	CP	CP	O	O
18. Нечеткая логика	B.67	–	–	–	–	–

Окончание таблицы А.4

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
19. Объектно ориентированное программирование	B.68	–	P	P	P	P
Требования 1 Пригодный набор методов должен быть выбран согласно уровню полноты безопасности ПО. 2 При уровне полноты безопасности ПО 3 или 4 утвержденный набор методов должен включать один из методов 1, 2 или 3 вместе с одним из методов 11 или 12. Остальные методы рассматриваются согласно своим рекомендациям.						

Таблица А.5 – Верификация и тестирование (раздел 11)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Нормальное	B.31	–	P	P	CP	CP
2. Вероятностное тестирование	B.47	–	P	P	CP	CP
3. Статистический анализ	dt8	–	CP	CP	CP	CP
4. Динамический анализ и тестирование	dt2	–	CP	CP	CP	CP
5. Система показателей	B.42	–	P	P	P	P
6. Прослеживаемость	B.69	–	P	P	CP	CP
7. SEEA – анализ последствий ошибки программного обеспечения	B.26	–	P	P	CP	CP
Требования 1 Для уровней полноты безопасности ПО 3 или 4 утвержденные сочетания методов должны быть: a) 1 и 4; b) 3 и 4; или c) 4, 6 и 7. 2 Для уровней полноты безопасности ПО 1 или 2 утвержденные сочетания методов должны быть: a) 1; или b) 3 и 4; или c) 4.						

Таблица А.6 – Интеграция ПО и аппаратных средств (раздел 12)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Функциональное тестирование/тестирование методом «черного ящика»	dt3	CP	CP	CP	CP	CP
2. Проверка эксплуатационных параметров	dt6	–	P	P	CP	CP
Требования 1 Для уровня полноты безопасности ПО 0 метод 1 должен быть утвержденным методом. 2 Для уровня полноты безопасности ПО 1, 2, 3 или 4 утвержденным должно быть сочетание методов 1 и 2.						

Таблица А.7 – Валидация ПО (раздел 13)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Вероятностное тестирование	B.47	–	P	P	CP	CP
2. Проверка эксплуатационных параметров	dt6	–	CP	CP	O	O
3. Функциональное тестирование/тестирование методом «черного ящика»	dt3	CP	CP	CP	O	O
4. Моделирование	dt5	–	P	P	P	P
Требование – Для уровня полноты безопасности ПО 1, 2, 3 или 4 утвержденным должно быть сочетание методов 2 и 3.						

Таблица А.8 – Разделы, подлежащие оценке

Разделы, подлежащие оценке	Раздел	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Уровни полноты безопасности ПО	5	СР	СР	СР	СР	СР
2. Персонал и ответственность	6	–	Р	Р	СР	СР
3. Жизненный цикл и документация	7	–	СР	СР	СР	СР
4. Спецификация требований к ПО	8	Р	СР	СР	СР	СР
5. Архитектура ПО	9	–	Р	Р	СР	СР
6. Разработка и внедрение ПО	10	–	СР	СР	СР	СР
7. Верификация и тестирование ПО	11	–	СР	СР	СР	СР
8. Интеграция ПО и аппаратных средств	12	–	Р	Р	СР	СР
9. Валидация ПО	13	–	СР	СР	СР	СР
10. Обеспечение качества ПО	15	–	СР	СР	СР	СР
11. Обслуживание ПО	16	–	СР	СР	СР	СР

Таблица А.9 – Оценка ПО (раздел 14), методы оценки

Методы оценки	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Проверочные списки	B.8	СР	СР	СР	СР	СР
2. Статистический анализ	B.14 B.42 dt8	Р	СР	СР	СР	СР
3. Динамический анализ ПО	dt2 dt3	–	Р	Р	СР	СР
4. Причинно-следственные диаграммы	B.6	Р	Р	Р	Р	Р
5. Анализ дерева событий	B.23	–	Р	Р	Р	Р
6. Анализ дерева сбоев	B.28	Р	Р	Р	СР	СР
7. SEEA – анализ последствий ошибки программного обеспечения	B.26	–	Р	Р	СР	СР
8. Анализ отказов по общим причинам	B.10	–	Р	Р	СР	СР
9. Модели Маркова	B.41	–	Р	Р	Р	Р
10. Блок-схема надежности	B.51	–	Р	Р	Р	Р
11. Полевое испытание перед приемкой	–	Р	СР	СР	СР	СР
Требование – Один или несколько из этих методов должны быть выбраны для соответствия применяемому уровню полноты безопасности ПО.						

Таблица А.10 – Обеспечение качества ПО (раздел 15)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Аккредитация по ISO 9001	15	Р	СР	СР	СР	СР
2. Соответствие ISO 9000-3	15	О	О	О	О	О
3. Система качества организации	15	О	О	О	О	О
4. Управление конфигурацией	B.56	О	О	О	О	О

Таблица А.11 – Обслуживание ПО (раздел 16)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Анализ последствий	B.35	Р	СР	СР	О	О
2. Записи данных и анализ	B.13	СР	СР	СР	О	О

Детальные таблицы (dt)

Таблица А.12 (dt1) – Стандарты проектирования и кодирования (раздел 10)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Наличие стандарта кодирования	B.16	СР	СР	СР	СР	СР
2. Руководство по стилю кодирования	B.16	СР	СР	СР	СР	СР
3. Отсутствие динамических объектов	B.16	–	Р	Р	СР	СР
4. Отсутствие динамических переменных	B.16	–	Р	Р	СР	СР
5. Ограниченное использование указателей	B.16	–	Р	Р	Р	Р
6. Ограниченное использование рекурсии	B.16	–	Р	Р	СР	СР
7. Отсутствие безусловных переходов	B.16	–	СР	СР	СР	СР

Требования

1 Принимается, что методы 3, 4 и 5 могут присутствовать как часть утвержденного компилятора или транслятора.

2 Подходящий набор методов должен быть выбран согласно уровню полноты безопасности ПО.

Таблица А.13 (dt2) – Динамический анализ и тестирование (разделы 11 и 14)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Анализ граничных значений (тестирование)	B.4	–	СР	СР	СР	СР
2. «Провоцирование» ошибок (тестирование)	B.21	Р	Р	Р	Р	Р
3. Подсев ошибок (тестирование)	B.22	–	Р	Р	Р	Р
4. Моделирование функционирования	B.45	–	Р	Р	СР	СР
5. Классы эквивалентности и тестирование сегмента входа	B.19	–	Р	Р	СР	СР
6. Структурное тестирование	B.58	–	Р	Р	СР	СР

Требования

1 Анализ тестовых данных находится на уровне подсистемы и основан на спецификации и/или спецификации и коде.

2 Пригодный набор методов должен быть выбран согласно уровню полноты безопасности ПО.

Таблица А.14 (dt3) – Функциональное тестирование/тестирование методом «черного ящика» (разделы 10, 12, 13 и 14)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Причинно-следственные диаграммы (тестирование)	B.6	–	–	–	Р	Р
2. Создание прототипа/анимация	B.49	–	–	–	Р	Р
3. Анализ граничных значений	B.4	Р	СР	СР	СР	СР
4. Классы эквивалентности и тестирование сегмента входа	B.19	Р	СР	СР	СР	СР
5. Имитация работы	B.48	Р	Р	Р	Р	Р

Требования

1 Полнота моделирования будет зависеть от диапазона уровня полноты безопасности ПО, сложности и вида применения.

2 Подходящий набор методов должен быть выбран согласно уровню полноты безопасности ПО.

Таблица А.15 (dt4) – Языки программирования (раздел 10)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. ADA	B.62	P	CP	CP	P	P
2. MODULA-2	B.62	P	CP	CP	P	P
3. PASCAL	B.62	P	CP	CP	P	P
4. Fortran 77	B.62	P	P	P	P	P
5. C или C++ (не ограниченный)	B.62	P	P	P	HP	HP
6. Подмножество C или C++ со стандартами кодирования	B.62 B.38	P	P	P	P	P
7. PL/M	B.62	P	P	P	HP	HP
8. BASIC	B.62	P	HP	HP	HP	HP
9. Assembler	B.62	P	P	P	–	–
10. Ladder Diagrams	B.62	P	P	P	P	P
11. Functional Blocks	B.62	P	P	P	P	P
12. Statement List	B.62	P	P	P	P	P

Требования

1 При уровнях полноты безопасности ПО 3 и 4, когда применяется подмножество языков 1, 2, 3 и 4, рекомендация меняется на CP.

2 Для некоторых случаев применения только языки 7 и 9 могут быть единственно доступными. При уровнях полноты безопасности ПО 3 и 4, где нет строго рекомендованного варианта, строго рекомендуется, чтобы было языковое подмножество, чтобы поднять рекомендацию до «P», а также должен быть точный набор стандартов кодирования.

3 Для получения информации по оценке соответствия языка программирования см. ссылку в библиографии на «соответствующий язык программирования» (B.62).

4 Если в таблице отсутствует конкретный язык, то он не исключается автоматически. Тем не менее он должен соответствовать B.62.

Таблица А.16 (dt5) – Моделирование (раздел 13)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Диаграммы потока данных	B.12	–	P	P	P	P
2. Конечные автоматы/диаграммы состояний	B.29	–	CP	CP	CP	CP
3. Формальные методы	B.30	–	P	P	CP	CP
4. Моделирование функционирования	B.45	–	P	P	CP	CP
5. Временные сети Петри	B.64	–	CP	CP	CP	CP
6. Создание прототипа/анимация	B.49	–	P	P	P	P
7. Структурные диаграммы	B.59	–	P	P	CP	CP
Требование – Подходящий набор методов должен быть выбран согласно уровню полноты безопасности ПО.						

Таблица А.17 (dt6) – Проверка качества функционирования (разделы 10, 12 и 13)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Лавинное/нагрузочное тестирование	B.3	–	P	P	CP	CP
2. Ограничения времени реакции и памяти	B.52	–	CP	CP	CP	CP
3. Требования к функционированию	B.46	–	CP	CP	CP	CP
Требование – Подходящий набор методов должен быть выбран согласно уровню полноты безопасности ПО.						

Таблица А.18 (dt7) – Полуформальные методы (разделы 8 и 10)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Логические диаграммы/диаграммы функциональных блоков	–	Р	Р	Р	СР	СР
2. Диаграммы последовательности	–	Р	Р	Р	СР	СР
3. Диаграммы потока данных	В.12	Р	Р	Р	Р	Р
4. Конечные автоматы/диаграммы состояний	В.29	–	Р	Р	СР	СР
5. Временные сети Петри	В.64	–	Р	Р	СР	СР
6. Таблицы истинности	В.14	Р	Р	Р	СР	СР
Требование – Подходящий набор методов должен быть выбран согласно уровню полноты безопасности ПО.						

Таблица А.19 (dt8) – Статический анализ (разделы 11 и 14)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Анализ граничных значений	В.4	–	Р	Р	СР	СР
2. Проверочные списки	В.8	–	Р	Р	Р	Р
3. Анализ потока управления	В.9	–	СР	СР	СР	СР
4. Анализ потока данных	В.11	–	СР	СР	СР	СР
5. «Провоцирование» ошибок	В.21	–	Р	Р	Р	Р
6. Проверки по Фагану	В.24	–	Р	Р	СР	СР
7. Анализ ложной цепи	В.55	–	–	–	Р	Р
8. Символическое выполнение	В.63	–	Р	Р	СР	СР
9. Прогоны/обзоры разработки	В.66	СР	СР	СР	СР	СР
Требование – Подходящий набор методов должен быть выбран согласно уровню полноты безопасности ПО.						

Таблица А.20 (dt9) – Модульный принцип (раздел 10)

Метод/мероприятие	Ссылка	Уровень полноты безопасности ПО				
		0	1	2	3	4
1. Ограниченный размер модуля	В.43	СР	СР	СР	СР	СР
2. Скрытие информации/инкапсуляция	В.36	Р	СР	СР	СР	СР
3. Ограничение числа параметров	В.43	Р	Р	Р	Р	Р
4. Одна точка входа и одна точка выхода в подпрограммах и функциях	В.43	Р	СР	СР	СР	СР
5. Полностью определенный интерфейс	В.43	СР	СР	СР	О	О
Требование – Подходящий набор методов должен быть выбран согласно уровню полноты безопасности ПО.						

Приложение В
(справочное)

Библиография методов

В.1 Искусственный интеллект. Устранение сбоев

(Ссылка приведена в разделе 9.)

Цель

Обеспечить возможность гибкого реагирования на потенциальные опасности за счет введения комплекса методов и моделей процесса, а также анализа безопасности и надежности в режиме реального времени.

Характеристика

Прогнозирование возникновения сбоев (рассчитанные тенденции), устранение сбоев, обслуживание и надзорные действия могут быть поддержаны системами с искусственным интеллектом достаточно эффективно в различных каналах системы, поскольку правила могут быть выведены непосредственно из спецификаций и проверены относительно них. При таком подходе определенных общих ошибок, которые потенциально могут быть внесены в спецификации уже с учетом правил проектирования и реализации, можно эффективно избежать, особенно при функциональном или описательном применении сочетания моделей и методов.

Методы выбираются таким образом, чтобы сбои можно было устранять, а последствия сбоев – минимизировать, обеспечивая соответствие условиям безопасности и надежности.

В.2 Поддающиеся анализу программы

(Ссылка приведена в разделе 10.)

Цель

Разработка легкоанализируемых программ. Работа программы должна быть полностью доступна проверке на основе анализа.

Характеристика

Задача – разработать легкоанализируемые программы с применением методов статического анализа. Чтобы достичь этого, необходимо следовать правилам структурированного программирования, например:

– алгоритм управления модулем должен состоять из структурированных конструктивных элементов, то есть последовательностей, циклов и выбора;

– модули должны быть небольшими;

– число возможных путей через модуль небольшое;

– индивидуальные части программы должны быть составлены таким образом, чтобы они были как можно больше разъединены;

– связь между входными и выходными параметрами должна быть как можно проще;

– сложные расчеты не должны использоваться в качестве основы для выбора ветви и цикла;

– выбор ветви и цикла должен быть просто связан со входными параметрами модулей;

– границы между различными типами преобразования должны быть простыми.

Справочный материал

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

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 Лавинное/нагрузочное тестирование

(Ссылка приведена в таблице dt6.)

Цель

Применение исключительно высокой рабочей нагрузки на объект тестирования с целью проверки того, выдержит ли объект тестирования нормальную рабочую нагрузку.

Характеристика

Существует ряд различных условий тестирования, которые можно применить для лавинного/нагрузочного тестирования. Некоторые из таких условий приведены ниже:

– при работе в режиме последовательного опроса объект тестирования подвергается намного большим исходным изменениям за единицу времени, чем при обычных условиях;

– при работе по запросу число запросов за единицу времени к объекту тестирования увеличивается, выходя за пределы обычных условий;

– если размер базы данных играет важную роль, он увеличивается, выходя за пределы обычных условий;

– влияющие устройства устанавливаются на максимальную или минимальную скорость соответственно;

– для крайних случаев все влияющие факторы, насколько это возможно, вносятся в граничные условия одновременно.

При данных условиях тестирования может быть оценено временное поведение объекта тестирования. Можно наблюдать за влиянием изменений нагрузки. Можно проверить правильный объем внутренних буферов динамических переменных, стековых запоминающих устройств и т. д.

B.4 Анализ граничных значений

(Ссылки приведены в таблицах dt2, dt3 и dt8.)

Цель

Устранение ошибок в ПО, возникающих при ограничениях параметров или границ.

Характеристика

Входной интервал программы разделен на ряд классов ввода. Тестирование должно охватывать границы и крайние точки классов. Благодаря тестированию можно проверить, совпадают ли границы в исходном интервале спецификации с границами в программе. Использование нулевого значения как в прямом, так и в косвенном переводе, зачастую ведет к ошибке и требует особого внимания:

– делитель нуль;

– пустые символы в коде ASCII;

– пустая стековая память или элемент списка;

– нулевая матрица;

– нулевой элемент таблицы.

Обычно границы для ввода имеют прямое соответствие границам диапазона вывода. Следует обеспечить набор тестовых данных, чтобы принудительно подвести вывод к ограниченным значениям. Также следует рассмотреть возможность обеспечения тестовых данных, из-за которых вывод превышает граничные значения спецификации.

Если вывод представляет собой последовательность данных, например распечатанную таблицу, то особое внимание следует уделить первому и последнему элементам и спискам, в которых нет элементов, 1 элемент и 2 элемента.

Справочный материал

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

B.5 Восстановление предыдущего состояния файла

(Ссылка приведена в разделе 9.)

Цель

Восстановление правильной функциональной работы после одного или нескольких сбоев.

Характеристика

При появлении сбоя система переустанавливается на более раннее внутреннее состояние, стабильность которого была проверена ранее. Этот метод подразумевает частое сохранение внутреннего состояния на так называемых строго определенных контрольных точках. Это может быть выполнено глобально (для полной базы данных) или с определенным шагом (изменения только между кон-

трольными точками). Затем система должна компенсировать изменения, произошедшие в течение этого времени, используя протоколирование (контрольный анализ действий), компенсацию (все последствия таких изменений аннулируются) или внешнее (ручное) взаимодействие.

B.6 Причинно-следственные диаграммы

(Ссылки приведены в разделе 14 и таблице dt3.)

Цель

Моделирование в форме диаграммы последовательности событий, которые могут развиться в систему как следствие сочетания базовых событий.

Характеристика

Это можно рассматривать как сочетание анализа по дереву сбоев и анализа по дереву событий. Начиная с критического события, причинно-следственная диаграмма прослеживается в обратном направлении и в направлении вперед. В обратном направлении она эквивалента дереву сбоев, где критическое событие является конечным событием. В направлении вперед идентифицируются возможные последствия, возникающие из-за события. Диаграмма может содержать символы вершины, описывающие условия распространения различных ветвей от вершины. Также могут быть включены задержки во времени. Эти условия также могут быть описаны с помощью дерева сбоев. Линии распространения могут быть совмещены с логическими символами, чтобы сделать диаграмму более компактной. Определяется набор стандартных символов для использования в причинно-следственных диаграммах. Диаграммы могут быть применены для расчета вероятности возникновения определенных критических последствий.

Справочный материал

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

B.7 Сертифицированные средства и трансляторы

(Ссылка приведена в разделе 10.)

Цель

Средства необходимы разработчикам на различных этапах разработки ПО. По возможности средства должны быть сертифицированы, чтобы в определенной степени удостовериться в правильности полученных данных.

Характеристика

Сертифицированное средство – это средство, в отношении которого было установлено, что оно обладает определенным качеством. Сертификация средства обычно выполняется независимым, часто национальным, органом по независимо установленным критериям, обычно национальным или международным стандартам. В идеале средства, используемые на всех этапах разработки (определение, разработка проекта, кодирование, тестирование и валидация), и средства, используемые в управлении конфигурацией, должны проходить сертификацию. В настоящее время лишь компилирующие программы (трансляторы) регулярно проходят процедуру сертификации; их устанавливают национальные органы по сертификации, и они проверяют компилирующие программы (трансляторы) по международным стандартам, таким как стандарты для Ada и Pascal.

Справочный материал

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.

B.8 Проверочные списки

(Ссылки приведены в разделе 14 и таблице dt8.)

Цель

Стимулирование критической оценки всех аспектов системы вместо установления определенных требований.

Характеристика

В проверочных списках есть ряд вопросов, на которые необходимо ответить лицу, выполняющему контрольную проверку. Многие вопросы имеют общий характер, и эксперт должен интерпретировать их как представляющиеся наиболее уместными для конкретной оцениваемой системы.

Для охвата широкого спектра вариаций ПО и аппаратных средств, которые проходят процесс валидации, большинство проверочных списков содержат вопросы, которые применимы ко многим типам системы. В результате этого в проверочных списках могут присутствовать вопросы, которые не относятся к рассматриваемой системе и которые следует игнорировать. В равной мере может существовать необходимость для конкретной системы в дополнении стандартного проверочного списка вопросами, специально предназначенными для рассматриваемой системы.

В любом случае должно быть ясно, что использование проверочных списков значительно зависит от компетентности и суждений инженера, выбирающего и применяющего проверочные списки. В результате решения, принятые инженером по выбранным проверочным спискам, а также любым дополнительным и детализированным вопросам, должны быть полностью документально оформлены и обоснованы. Цель – обеспечить возможность анализа применения проверочных списков и получения аналогичных результатов, если не используются другие критерии.

По возможности при заполнении проверочного списка следует соблюдать краткость. При необходимости предоставления четкого обоснования следует приводить ссылку на дополнительные документы. «Пропуск», «Неуспешное выполнение» и «Неокончательно» либо другие подобные наборы обозначений следует использовать для записи результатов по каждому вопросу. Такая краткость упрощает процесс получения общего вывода по результатам оценки с использованием проверочных списков.

Справочный материал

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office, London 1987.

Guidelines for the Assessment of the Safety and Reliability of High Integrity Industrial Computer Systems. EWICS TC7, WP 489, 6th October 1987.

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

Software for computers in the safety systems of nuclear power stations. IEC 60880, 1986.

B.9 Анализ потока управления

(Ссылка приведена в таблице dt8.)

Цель

Выявление неверных и потенциально некорректных программных структур.

Характеристика

При анализе потока управления выявляются сомнительные участки кода, в которых не соблюдается рациональная практика программирования. Программа анализируется для формирования направленной диаграммы, которую можно анализировать по:

– недоступному коду, например операции безусловного перехода, которые оставляют блоки кода недосягаемыми;

– узловой код, т. е. хорошо структурированный код, диаграмма управления которого сводится последующими понижениями диаграммы к единому узлу. Слабо структурированный код может быть лишь приведен к узловому пункту, состоящему из нескольких узлов.

Справочный материал

RXVP80 – The Verification and Validation System for FORTRAN. Users Manual. General Research Corporation, Santa Barbara, California, USA.

Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACM Trans. on Prog. Lang. and Syst., 1985.

B.10 Анализ отказов по общим причинам

(Ссылка приведена в разделе 14.)

Цель

Идентификация потенциальных отказов в резервных системах или подсистемах, которые сводят на нет преимущества резервирования из-за одновременного появления одинаковых отказов в резервных элементах.

Характеристика

Компьютерные системы, предназначенные для обеспечения безопасности установки, часто используют резервирование аппаратного и мажоритарного голосования. Данный метод применяется для предупреждения случайных отказов компонентов, которые обычно препятствуют верной обработке данных в компьютерной системе.

Однако некоторые отказы могут быть общими для более чем одного компонента. Например, если компьютерная система установлена в комнате, проблемы в системе кондиционирования воздуха могут снизить полезность резервирования. То же можно отметить и в отношении других внешних факторов воздействия на компьютерную систему, таких как пожар, потоп, электромагнитные помехи, падение самолета и землетрясение. Компьютерная система может также подвергнуться воздействию в результате аварийных ситуаций, связанных с эксплуатацией и обслуживанием. Таким образом, крайне важно, чтобы существовали адекватные и документально оформленные надлежащим образом процедуры по эксплуатации и обслуживанию. Также важно обеспечить обучение эксплуатации и обслуживанию.

Внутренние факторы воздействия также существенно влияют на общие причины отказов (CCF). Они могут происходить в результате ошибок при проектировании в аналогичных или идентичных компонентах и их интерфейсах, а также в результате изнашивания компонентов. При анализе CCF необходимо осуществлять поиск таких потенциальных общих отказов в системе. Методы анализа CCF представляют собой общий контроль качества, анализ проектирования, верификацию и тестирование независимой группой экспертов, а также анализ реальных аварийных ситуаций с учетом опыта эксплуатации подобных систем. Однако анализ распространяется не только на аппаратные средства. Даже если в сложных цепочках резервной компьютерной системы используется «различное ПО», в подходах ПО может существовать некая унифицированность, которая могла бы инициировать CCF (например, ошибки в общей спецификации).

Когда CCF не возникает на одной и той же линии, могут быть приняты меры предосторожности с применением сравнительных методов между резервными цепочками, которые должны привести к обнаружению отказа, прежде чем этот отказ станет общим для всех цепочек. При анализе CCF этот прием должен учитываться.

Справочный материал

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, Dec. 1979.

Programmable Electronic Systems in Safety Related Applications. Health and Safety Executive, Her Majesty's Stationary Office.

B.11 Анализ потока данных

(Ссылка приведена в таблице dt8.)

Цель

Выявление неверных и потенциально некорректных программных структур.

Характеристика

Анализ потока данных объединяет информацию, полученную в результате анализа потока управления, с информацией, по которой переменные величины считываются или записываются в различных блоках кода. При анализе можно проверить наличие:

– переменных величин, считываемых прежде, чем они записаны. Вероятно, в этом заключается ошибка, и это является примером неудачной практики программирования;

– переменных величин, записанных более одно раза без считывания. Это может означать упущенный код;

– переменных величин, записанных, но не считываемых. Это может указывать на резервный код.

Существует также дополнительный анализ потока данных, известный как анализ потока информации, где потоки реальных данных (в рамках и между процедурами) сопоставляются с целью проектирования. Обычно это выполняется с помощью компьютеризированного средства, когда запланированные потоки данных определяются с использованием структурированного комментария, который может быть считан средством.

Справочный материал

RXVP80 – The Verification and Validation System for FORTRAN. Users Manual. General Research Corporation, Santa Barbara, California, USA.

Information Flow and Data Flow of While Programs. J. F. Bergeretti and B. A. Carre, ACMTrans. on Prog. Lang. and Syst., 1985.

B.12 Диаграммы потока данных

(Ссылки приведены в таблицах dt5 и dt7.)

Цель

Описание потока данных через программу в форме диаграммы.

Характеристика

С помощью диаграмм потока данных документируется, каким образом входные данные преобразуются в выходные данные, при этом каждый этап в диаграмме представляет отдельное преобразование.

Диаграммы потока данных состоят из трех компонентов:

1) стрелки с комментариями – иллюстрируют поток данных в центры/из центров преобразования, в комментариях поясняется, какие это данные;

2) окружности с комментариями – иллюстрируют центры преобразования, в комментариях поясняется преобразование;

3) операторы (AND, XOR) – эти операторы используются для связывания стрелок с комментариями.

С помощью диаграмм потока данных описывается, каким образом входные данные преобразуются в выходные данные. Они не включают и не должны включать управляющую информацию или информацию о задании последовательности. Каждая окружность может рассматриваться как самостоятельный «черный ящик», который, как только становятся доступны входные данные, преобразует их в выходные данные.

Одно из принципиальных преимуществ диаграмм потока данных заключается в том, что они иллюстрируют преобразования без каких-либо допущений относительно того, как осуществляются эти преобразования.

Наилучшим подходом при составлении диаграмм потока данных является рассмотрение входных данных системы и работа по направлению к выходным данным системы. Каждая окружность должна отчетливо иллюстрировать преобразование – выходные данные должны определенным образом отличаться от входных данных. Не существует общих правил определения общей структуры диаграммы, и построение диаграммы потока данных является одним из творческих аспектов проектирования системы. Как и любое другое проектирование, это итеративный процесс, усовершенствованный по этапам для получения итоговой диаграммы.

Справочный материал

Software Engineering. I. Sommerville, Addison-Wesley 1982. ISBN 0-201-13795-X.

B.13 Записи данных и анализ

(Ссылки приведены в разделах 10 и 16.)

Цель

Запись всех данных, решений и обоснований в проекте ПО для более простой верификации, валидации, оценки и обслуживания.

Характеристика

Подробные записи ведутся в ходе проекта, как на основе проекта, так и отдельно. Например, инженеру необходимо вести записи, которые могут включать:

– меры, подробно изложенные по отдельным модулям;

– тестирование, проведенное с каждым модулем;
– решения и их обоснования;
– завершение основных этапов проекта;
– проблемы и пути их решения.

В ходе проекта и по его завершении эти записи могут анализироваться для получения различной информации. В частности, запись данных является очень важной для обслуживания компьютерных систем, поскольку обоснование по некоторым решениям, принятым на этапе проекта разработки, не всегда известно инженерам, осуществляющим обслуживание систем.

B.14 Таблицы истинности

(Ссылки приведены в разделе 14 и таблице dt7.)

Цель

Обеспечение четкой и понятной спецификации и анализа сложных логических комбинаций и связей.

Характеристика

При таких методах используются две пространственные таблицы для краткого описания логических связей между булевыми переменными.

Краткость и табличный характер обоих методов делают их уместными как средство анализа сложных логических комбинаций, выраженных в коде.

Оба метода потенциально могут быть выполнимыми при условии использования их в качестве спецификаций.

B.15 Защитное программирование

(Ссылка приведена в разделе 9.)

Цель

Написание программ, предназначенных для обнаружения ненормального потока управления, потока данных или значений данных во время их выполнения и реагирования на них соответствующим предусмотренным образом.

Характеристика

В процессе программирования могут применяться разные методы проверки ошибок управления или некорректных данных. Их можно систематически применять посредством программирование системы для снижения вероятности ошибочной обработки данных.

Можно выделить две накладывающихся друг на друга области защитных методов.

В устойчивом к ошибкам ПО есть свои проектные недостатки. Такие недостатки могут быть вызваны обычной ошибкой проектирования или кодирования либо ошибочными требованиями. Ниже перечислены некоторые защитные методы:

– переменные должны быть проверены на соответствие диапазону;
– где это возможно, значения следует проверять на достоверность;
– параметры процедур должны быть проверены на соответствие типу, величине и диапазону на входе процедуры.

Эти первые три рекомендации помогают обеспечить обоснованность численных критериев, используемых программой, как с точки зрения функции программы, так и с точки зрения физического значения переменных.

Параметры «только считывание» и параметры «считывание/запись» следует разделять и проверять доступ к ним. Функции должны обрабатывать все параметры как «только считывание». Литеральные константы не должны давать доступ для записи. Это помогает выявлять случайное наложение записи или ошибочное использование переменных.

Устойчивое к ошибкам ПО разрабатывается таким образом, чтобы «ожидать» сбои в собственной среде или использовать внешние номинальные или предполагаемые условия и реагировать заранее установленным образом. Методы включают следующие указания:

– входные переменные и промежуточные переменные с физическим значением следует проверять на достоверность;
– следует проверять влияние выходных переменных, предпочтительно путем непосредственного наблюдения за изменениями в состоянии соответствующих систем;

– ПО должно проверять свою конфигурацию. Это должно предусматривать как существование, так и доступность предполагаемых аппаратных средств, а также комплектность ПО. Это особенно важно для сохранения целостности после всех процедур по обслуживанию.

Некоторые методы защитного программирования, такие как проверка порядка потока управления, также устраняют последствия внешних сбоев.

Справочный материал

Dependability of Critical Computer Systems – Part 1. E. F. Redmill (ed.) Elsevier Applied Science 1988.

Dependability of Critical Computer systems – Part 2. E. F. Redmill (ed.) Elsevier Applied Science 1988.

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

B.16 Стандарты проектирования и кодирования

(Ссылка приведена в таблице dt1.)

Цель

Обеспечение единой схемы проектных документов и вырабатываемого кода, обеспечение выполнения анонимного программирования и стандартного метода проектирования.

Характеристика

Правила, подлежащие соблюдению, согласовываются участниками в начале проекта. Такие правила должны состоять из следующих пунктов:

– метод разработки и соответствующие стандарты кодирования, подлежащие соблюдению, например JSP, MASCOT, Petri-Nets и т. д.;

– подробности модуляризации, например формы интерфейса, размеры модуля;

– использование инкапсуляции, наследования и полиморфизма в случаях с объектно-ориентированными языками;

– использование или избежание определенных языковых конструкций, например GOTO, EQUIVALENCE, динамические объекты, динамические данные, рекурсия, ссылки, выходной канал и т. п.;

– что, где и как комментировать.

Эти правила составляются, чтобы обеспечить простоту разработки, верификации, оценки и обслуживания. Таким образом, при этом должны применяться имеющиеся средства, в частности анализаторы и средства обратного проектирования.

B.17 Многовариантное программирование

(Ссылка приведена в разделе 9.)

Цель

Выявление и маскировка остаточных сбоев разработки ПО при выполнении программы для предотвращения критических сбоев безопасности системы, а также продолжения работы по повышению надежности.

Характеристика

При многовариантном программировании заданная спецификация программы реализуется N раз различными способами. Одинаковые входные значения заданы для N версий реализации, и результаты, полученные при реализации каждой версии, сравниваются между собой. Если результат считается допустимым, то они выводятся из компьютера.

N версий могут реализовываться параллельно на разных компьютерах или на одном компьютере, и результаты подвергаются мажоритарной выборке. Для разных версий могут быть использованы различные способы выборки в зависимости от требований к приложению.

Для систем с безопасным состоянием целесообразно требовать полного согласования (все N согласований), в противном случае необходимо использовать устойчивое к отказам выходное значение. Для простых отключающих систем акцент может быть сделан на безопасности. В этом случае мерой безопасности будет являться отключение, если какая-либо из двух версий требует отключения. При таком принципе обычно используются две версии (N = 2).

Для систем с небезопасным состоянием могут быть применены способы мажоритарной выборки. В случаях отсутствия общего согласования могут применяться вероятностные принципы, чтобы макси-

мизировать возможность выбора корректного значения, например, взяв среднее значение, временную задержку выходных данных до возврата согласования и т. д.

Этот метод не устраняет полностью остаточные сбои проектирования ПО, но обеспечивает меру по выявлению и маскировке прежде, чем они могут повлиять на безопасность.

Справочный материал

Dependable Computing: From Concepts to Design Diversity. A. Avizienis, J. C. Laprie, Proc. IEEE, Vol. 74, 5 May 1986.

A Theoretical Basis for the Analysis of Multi-version Software subject to Co-incident Failures. D. E. Eckhardt and L. D. Lee, IEEE Trans. SE-11, No. 12, 1985.

B.18 Динамическая реконфигурация

(Ссылка приведена в разделе 9.)

Цель

Сохранение функциональности системы, несмотря на внутренний сбой.

Характеристика

Логическая архитектура системы должна быть такой, чтобы она соответствовала совокупности доступных ресурсов системы. Необходимо, чтобы архитектура предусматривала возможности выявления сбоя физического ресурса и затем повторно сопоставлялась с совокупностью оставшихся ограниченных ресурсов, которые продолжают функционировать. Хотя данная концепция в основном традиционно ограничивается восстановлением функционирования после выхода из строя тех или иных коспонентов аппаратных средств, она также применима к неисправным единицам ПО при условии достаточного «динамического резервирования», чтобы предоставить повторную попытку ПО или, если существует достаточно резервных данных, чтобы отдельный или изолированный сбой был незначительным.

Хотя этот метод традиционно применим к аппаратным средствам, он разрабатывается для применения в ПО и, таким образом, во всей системе. Он должен учитываться на первом этапе проектирования системы.

Справочный материал

Critical Issues in the Design of a Reconfigureable 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. June 1984, G. Kar and C. N. Nikolaou, Watson Research Centre, Yorktown.

B.19 Классы эквивалентности и тестирование сегмента входа

(Ссылки приведены в таблицах dt2 и dt3.)

Цель

Тестирование ПО в достаточной мере с использованием минимального набора тестовых данных. Тестовые данные получают за счет выбора сегмента области входа, необходимого для тестирования работы ПО.

Характеристика

Данный метод тестирования основывается на эквивалентной связи входных данных, определяющей сегмент области входа.

Наборы тестовых данных выбираются с целью охвата всех подмножеств данного сегмента. По крайней мере один набор тестовых данных берется от каждого класса эквивалентности.

Существует две основные возможности для входного деления на сегменты:

– классы эквивалентности могут быть определены по спецификации. Интерпретация спецификации может быть либо ориентирована на ввод (например, выбранные значения обрабатываются аналогичным образом), либо ориентирована на вывод (например, набор значений, приводящих к аналогичному функциональному результату);

– классы эквивалентности могут быть определены по внутренней структуре программы. В этом случае результаты классификации эквивалентности определяются из статического анализа программы (например, набор значений, приводящих к аналогичному выполнению задачи).

Справочный материал

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

B.20 Коды обнаружения и устранения ошибок

(Ссылка приведена в разделе 9.)

Цель

Выявление и устранение ошибок в важной информации.

Характеристика

Для информации в n бит разрабатывается блок кода в k бит, предназначенный для обнаружения и устранения ошибок. Ниже указаны различные типы кода:

– коды Хэмминга;
– циклические коды;
– полиномиальные коды.

Справочный материал

The Technology of Error Correcting Codes. E. R. Berlekamp, Proc. of the IEEE, 68, 5, 1980.
A Short Course on Error Correcting Codes. N. J. A. Sloane, Springer-Verlag, Wien, 1975.

B.21 «Провоцирование» ошибок

(Ссылки приведены в таблицах dt2 и dt8.)

Цель

Устранение общих ошибок программирования.

Характеристика

Опыт проведения тестирования и интуиция вместе со знанием и интересом к тестируемой системе могут добавить несколько некатегоризированных тестовых данных к уже разработанному набору тестовых данных. Особые значения или комбинации значений могут вызвать ошибки. Некоторые интересные наборы тестовых данных могут быть получены из проверочных списков. Также можно рассмотреть, достаточно ли устойчива система к сбоям. Можно ли нажимать на кнопки на лицевой панели слишком быстро или слишком часто? Что произойдет, если нажать на две кнопки одновременно?

B.22 Подсев ошибок

(Ссылка приведена в таблице dt2.)

Цель

Убедиться в адекватности набора тестовых данных.

Характеристика

Некоторые известные типы ошибок вносятся в программу, и программа выполняется по тестовым данным в условиях тестирования. Если хотя бы некоторые из подсеянных ошибок обнаруживаются, то тестовые данные являются недостаточными. Соотношение обнаруженных подсеянных ошибок к общему числу подсеянных ошибок является оценкой соотношения обнаруженных истинных ошибок к общему числу ошибок. Это дает возможность рассчитать число оставшихся ошибок и, таким образом, оставшийся объем работ по тестированию.

$$\frac{\text{Обнаруженные подсеянные ошибки}}{\text{Общее число подсеянных ошибок}} = \frac{\text{Обнаруженные истинные ошибки}}{\text{Общее число истинных ошибок}}.$$

Обнаружение всех подсеянных ошибок может указывать на то, что либо тестовые данные достаточны, либо подсеянные ошибки обнаружились очень легко. Ограничения метода таковы, что для того, чтобы получить пригодные результаты, типы ошибок, так же как и позиции подсева, должны отражать статистическое распределение истинных ошибок.

B.23 Анализ дерева событий

(Ссылка приведена в разделе 14.)

Цель

Моделирование в виде диаграммы последовательности событий, которая может развиться в систему после исходного события, и установление того, насколько серьезными могут оказаться последствия.

Характеристика

Наверху диаграммы записывают условия последовательности, относящиеся к разработке после исходного события, которое является целью анализа. Начиная с исходного события, проводят линию к первому условию в этой последовательности. Затем диаграмма разветвляется на ветки «да» и «нет», показывая, каким образом будущие разработки зависят от условия. Для каждой из этих веток одна продолжает идти к следующему условию подобным образом. Тем не менее не все условия соответствуют всем веткам. Одна продолжает идти к концу последовательности, и каждая ветка дерева, построенная таким образом, представляет собой возможное последствие. Дерево событий можно использовать для расчета вероятности различных последствий, основанных на вероятности и числе условий в этой последовательности.

Справочный материал

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

B.24 Проверки по Фагану

(Ссылка приведена в таблице dt8.)

Цель

Выявление ошибок на всех этапах разработки программы.

Характеристика

«Формальная» проверка документов по обеспечению качества направлена на обнаружение ошибок и упущений. Процесс проверки состоит из пяти фаз: планирование, подготовка, проверка, исправление и проверка исполнения. Каждая из этих фаз имеет собственную отдельную задачу. Должна быть проверена полная разработка системы (спецификация, проектирование, кодирование и тестирование).

Справочный материал

Design and Code Inspections to Reduce Errors in Program Development. M. E. Fagan, IBM Systems Journal, No. 3, 1976.

B.25 Программирование с регистрацией отказов

(Ссылка приведена в разделе 9.)

Цель

Выявление остаточных ошибок проектирования ПО во время выполнения программы в целях предотвращения критических отказов безопасности системы и продолжения работы по повышению надежности.

Характеристика

Утверждение метода программирования следует после проверки предварительного условия (прежде чем будет выполнена последовательность предписаний, первичные условия проверяются на предмет их правильности) и последующего условия (результаты проверяются после выполнения последовательности предписаний). Если предварительное условие или последующее условие не выполнено, обработка процесса останавливается с ошибкой.

Например,

```
утверждение <предварительное условие>;
    действие 1;
        :
        :
```

действие x;
утверждение <последующее условие>

Справочный материал

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.

B.26 SEEA – анализ последствий ошибки программного обеспечения

(Ссылки приведены в разделах 9, 11 и 14.)

Цель

Идентификация модулей ПО, их критичности; предложение средств выявления ошибок ПО и повышение устойчивости ПО; оценка объема работ по валидации, необходимых для различных компонентов ПО.

Характеристика

Анализ осуществляется в три этапа:
– определение существенных модулей ПО;
– определение глубины анализа (на уровне единой командной магистрали, группы команд, модуля и т. д.), необходимого для каждого модуля ПО, по его спецификации;
– анализ ошибок ПО.
Результат этого этапа – таблица, в которой указывается следующая информация:
– наименование модуля;
– рассматриваемая ошибка;
– последствия ошибки на уровне модуля;
– последствия на уровне системы;
– нарушенный критерий безопасности;
– критичность ошибки;
– предложенные средства выявления ошибок;
– нарушенный критерий, если было использовано средство выявления ошибок;
– остаточная критичность, если было использовано средство выявления ошибок.
– синтез.
Синтез определяет оставшиеся небезопасные сценарии и необходимый объем работ по валидации с учетом критичности каждого модуля.

SEEA, являясь глубоким анализом, выполняемым независимой группой экспертов, представляет собой действенный и доскональный метод.

Справочный материал

Railway fixed equipment and rolling stock. Data processing. Software dependability – Adapted methods for software safety analysis, French standard NF F 71-013, December 1990.

IRSE International Technical Committee Report No. 1 «SAFETY SYSTEM VALIDATION with regard to cross acceptance of signalling systems by the railways».

P. THIREAU «Méthodologie d'Analyse des Effets des Erreurs Logiciel (AEEL) appliquée à l'étude d'un logiciel de haute sécurité» 5th International Conference on Reliability and Maintainability. Biarritz – France – 1986.

D. J. REIFER «Software failures modes and effects analysis» Transaction on reliability IEE – Vol. R28 No. 3 août 79 – Pages 247/249.

J. R. FRAGOLA et J. F. SPAMM «The software error effects analysis, a qualitative design tool» IEEE Symposium Computer on Software Reliability 1973 – Pages 90/93.

B.27 Выявление сбоев и их диагностика

(Ссылка приведена в разделе 9.)

Цель

Выявление сбоев в системе, которые могут привести к отказу, обеспечение основы для контрмер в целях минимизации последствий отказов.

Характеристика

Выявление сбоев – это процесс проверки системы на наличие ошибочных состояний (которые вызваны, как уже было указано ранее, сбоем в системе/подсистеме, подлежащей проверке). Основной целью выявления сбоев является предотвращение последствий ошибочных результатов. Систему, которая либо дает правильные результаты, либо не дает результатов вообще, называют «самопроверяющей».

Выявление сбоев основывается на принципах резервирования (главным образом, для выявления сбоев в аппаратных средствах) и диверсификации (сбои ПО). Чтобы принять решение по правильности результатов, необходимо что-то наподобие голосования. Специальные применимые методы – это программирование с регистрацией, программирование N-го числа версий и методов (технические приемы) мониторинга безопасности, а на уровне аппаратных средств – это сенсоры, датчики, замкнутая система автоматического регулирования, коды с контролем ошибок и т. п.

Выявление сбоев может быть обеспечено за счет проверок в интервале значений или в интервале времени на различных уровнях, особенно на физическом (температура, напряжение и т. п.), логическом (коды обнаружения ошибок), функциональном (утверждения) или внешнем уровне (проверки достоверности). Результаты этих проверок могут храниться и связываться с некорректными данными, чтобы обеспечить прослеживание отказа.

Сложные системы состоят из подсистем. Эффективность выявления сбоев, диагностики и исправления сбоев зависит от сложности взаимодействия между подсистемами, которые оказывают влияние на продвижение сбоев.

Диагностика сбоев изолирует наименьшие подсистемы, которые можно идентифицировать. Небольшие подсистемы позволяют обеспечить более детальную диагностику сбоев (идентификацию ошибочных состояний).

B.28 Анализ дерева сбоев

(Ссылки приведены в разделах 9 и 14.)

Цель

Содействие анализу событий либо комбинации событий, ведущих к опасности или серьезным последствиям.

Характеристика

Анализ должен выполняться по дереву, начиная с события, которое является непосредственной причиной возникновения опасности или серьезных последствий («конечное событие»). Комбинации причин описываются логическими операторами (и, или и т. д.). Непосредственные причины анализируются аналогично. На этом и прекращается анализ.

Метод является графическим, и используется ряд типовых символов для рисования дерева сбоев. Он, главным образом, предназначен для анализа систем аппаратных средств, но также были попытки применить этот принцип при анализе отказов в ПО.

Справочный материал

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

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, D. C. 20555, 1981.

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

B.29 Конечные автоматы/диаграммы состояний

(Ссылки приведены в таблицах dt5 и dt7.)

Цель

Определение или реализация управляющей структуры системы.

Характеристика

Многие системы могут быть охарактеризованы с точки зрения их состояний, входных данных и действий. Так, находясь в состоянии S1 и получив ввод I, система может выполнить действие A и перейти к состоянию S2. Характеризуя действия системы для каждого ввода в каждом состоянии, можно полностью охарактеризовать систему. Полученная в результате этого модель системы называется

конечным автоматом. Она зачастую рисуется в виде так называемой диаграммы состояний, на которой показано, как система переходит от одного состояния к другому, или в виде матрицы, в которой главные величины – это состояние и ввод, а клетки матрицы содержат действие и новое состояние в результате получения ввода в данном состоянии.

Если система завершенная или имеет естественную структуру, это можно отразить в многослойном конечном автомате (FSM). Спецификацию или проектное решение в виде конечного автомата можно проверить на полноту (у системы должно быть действие и новое состояние для каждого ввода в каждом состоянии), на непротиворечивость (только одно изменение состояния определяется для каждой пары состояние/ввод) и на доступность (возможно ли перейти от одного состояния к другому за счет последовательности входных данных). Это важные свойства для критических систем, и их можно проверить. Средства для таких проверок легко расписываются. Существуют также алгоритмы, которые позволяют обеспечить автоматическое генерирование тестовых данных для проверки ввода в работу конечного автомата или оживления модели конечного автомата.

Справочный материал

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

B.30 Формальные методы

(Ссылки приведены в разделах 8 и 10, таблице dt5.)

Цель

Разработка ПО математическими методами; включает формальное проектирование и формальные методы кодирования.

Характеристика

Формальные методы обеспечивают описание разработки системы на определенном этапе ее спецификации, проектирования или кодирования. Итоговое описание принимает математическую форму и может быть подвергнуто математическому анализу для обнаружения различных классов несовместимости или некорректности. Кроме того, данное описание в некоторых случаях может быть подвергнуто машинному анализу с точностью, подобной точности при проверке синтаксиса исходной программы составителем или анимации для проявления различных аспектов поведения описываемой системы. Анимация может обеспечить дополнительную уверенность в том, что система соответствует реальным требованиям в той же мере, что и формально установленным требованиям.

Формальный метод обычно предполагает систему обозначений (обычно используется какая-либо форма дискретной математики), технологию получения описания в данной системе обозначений и различные формы анализа для проверки описания на различные свойства точности.

В следующих подразделах приводится несколько примеров формальных методов. К описываемым формальным методам относятся CCS, CSP, HOL, LOTOS, OBJ, временная логика, VDM и Z.

B.30.1 CCS – расчет взаимодействующих систем

Цель

CCS – средство описания и обоснования поведения параллельных систем взаимодействующих процессов.

Характеристика

Подобно CSP, CCS представляет собой математический расчет, имеющий отношение к изменению свойств системы. Проектирование системы моделируется в виде сети независимых процессов, протекающих последовательно или параллельно. Процессы связываются через порты (подобные каналам CSP), коммуникация происходит только тогда, когда готовы оба процесса. Возможно моделирование недетерминизма. Начиная с абстрактной высокоуровневой характеристики целой системы (известной как трассировка), представляется возможным проводить пошаговую обработку системы в композицию взаимодействующих процессов, общее поведение которой будет соответствовать требованиям системы. Равным образом представляется возможным работать, применяя принцип «снизу вверх», комбинируя процессы и получая свойства итоговой системы с использованием правил вывода, имеющих отношение к правилам композиции.

Справочный материал

A Calculus of Communicating Systems. R. Milner, Report number ECS-LFCS-86-7, Laboratory for Foundations of Computer Science, Département informatique, Université d'Edimbourg, Royaume-Uni.

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

B.30.2 CSP – взаимодействующие последовательные процессы

Цель

CSP представляет собой метод составления спецификации взаимодействующих программных систем, т. е. систем взаимодействующих согласованных рабочих процессов.

Характеристика

CSP обеспечивает язык для спецификации систем процессов и доказательство для подтверждения того, что выполнение процессов удовлетворяет их спецификациям (описываемых как последовательность событий, разрешенных к трассированию).

Система моделируется в виде сети независимых процессов. Каждый процесс описывается в виде всех возможных типов поведения. Система моделируется путем последовательной или параллельной композиции процессов. Процессы могут взаимодействовать (синхронно или обмениваться данными) через каналы; взаимодействие происходит только тогда, когда готовы оба процесса. Можно смоделировать относительный расчет времени событий.

Теория, основанная на CSP, была непосредственно включена в архитектуру транспьютера INMOS, а язык OCCAM позволяет обеспечить применение системы, определенной на CSP, в сети транспьютеров.

Источник:

Communicating Sequential Processes. CAR. Hoare, Prentice-Hall, 1985.

B.30.3 HOL – логика высшего порядка

Цель

Это официальный язык, служащий в качестве основы для спецификации и верификации аппаратных средств.

Характеристика

HOL (логика высшего порядка) относится к особой логической форме и ее системе машинного обеспечения, которые были разработаны в компьютерной лаборатории Кембриджского университета. Логическая форма в основном заимствуется из элементарной теории типов Черча, система машинного обеспечения основывается на системе LCF (логика вычислительных функций).

Источник:

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

B.30.4 LOTOS

Цель

LOTOS представляет собой метод описания и обоснования параллельных систем взаимодействующих процессов.

Характеристика

LOTOS (язык для спецификации временного упорядочивания) основывается на CCS с дополнительными характеристиками соответствующих алгебр CSP и CIRCAL (расчет схемы). Он не имеет недостатков CCS в управлении структурами данных и выражении величин путем комбинации со вторым компонентом, основанным на языке абстрактного типа данных ACT ONE. Однако компонент определения процесса LOTOS может быть использован с другими формулами для характеристики абстрактного типа данных.

Справочный материал

Information Processing Systems – Open Systems Inter-connection – LOTOS – A Formal Description Technique based on the Temporal Ordering of Observational Behaviour.

ISO/DIS 8807, July 20, 1987.

B.30.5 OBJ

Цель

Обеспечение точной спецификации системы с помощью обратной связи с пользователем и валидации системы до ее реализации.

Характеристика

OBJ – это язык алгебраической спецификации. Пользователи устанавливают требования в отношении алгебраического уравнения. Поведенческие или конструктивные аспекты системы определяются в отношении операций, действующих на абстрактные типы данных (ADT). ADT подобны пакету ADA, в котором поведение оператора является видимым, в то время как детали реализации являются скрытыми.

Спецификация OBJ и последующая пошаговая реализация подвержена таким же методам формального доказательства, как и другие формальные подходы. Более того, поскольку конструктивные аспекты спецификации OBJ исполняются машиной, они непосредственно должны выполнять валидацию системы, исходя из самой спецификации. Выполнение, по сути, является оценкой функции путем замены формул (переписывание), которая продолжается до получения определенного значения. Такая выполняемость позволяет конечным пользователям рассматриваемых систем получить «вид» последующей системы на стадии спецификации системы без необходимости ознакомления с базовыми технологиями формальной спецификации.

Как и в случаях с другими технологиями ADT, OBJ применяется только по отношению к последовательным системам или к последовательным аспектам сопутствующих систем. OBJ широко использовался для спецификации как небольших, так и крупномасштабных промышленных приложений.

Справочный материал

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. McGettrick (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.

DTI STARTS Guide. 1987, NCC, Oxford Road, Manchester, UK.

B.30.6 Временная логика

Цель

Непосредственное выражение требований безопасности и эксплуатационных требований, а также формальная демонстрация того, что эти свойства сохраняются на последующих этапах при разработке.

Характеристика

Стандартная логика предиката первого порядка не содержит временной концепции. Временная логика расширяет логику предиката первого порядка путем добавления модальных операторов (например, «в последующем» и «в результате»). Данные операторы могут быть использованы для уточнения тезисов о системе. Например, свойства безопасности могут быть востребованы для поддержания «в последующем», в то время как другие желаемые состояния системы могут понадобиться для достижения «в результате» какого-то другого начального состояния. Временные формулы интерпретируются в последовательности состояний (поведения). Что составляет «состояние», зависит от выбранного уровня описания. Это может относиться ко всей системе, элементу системы или компьютерной программе. Количественно выраженные интервалы времени и ограничения не управляются непосредственно во временной логике. Абсолютное время должно быть обработано путем создания состояний дополнительного времени как части определения состояния.

Справочный материал

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.

Logics for Computer Programming. D. Gabay. Ellis Horwood.

B.30.7 VDM – венский метод разработки

Цель

Систематические спецификации и выполнение последовательных программ.

Характеристика

VDM – это математически обоснованный метод спецификации и метод для обработки реализаций путем, позволяющим доказать корректность спецификации.

Метод спецификации основан на моделировании состояния системы в отношении теоретико-множественных структур, на которых определяются инварианты (предикаты), и операции в данном состоянии моделируются определением их предварительных и последующих условий в отношении состояния системы. Операции могут быть испробованы для сохранения инвариантов системы.

Реализация спецификации осуществляется посредством воспроизведения состояния системы исходя из структуры данных на целевом языке и путем обработки операций исходя из программы на целевом языке. Этапы воспроизведения и обработки приводят к доказательствам обязательств, устанавливающих их правильность. Выполнять данные обязательства или нет – на усмотрение проектировщика.

VDM в основном применяется на этапе спецификации, но также может быть применен на этапах проектирования и реализации, ведущих к исходному коду. Он применяется только в отношении последовательных программ или последовательных процессов в согласованных системах.

Справочный материал

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.

Systematic Software Development using VDM. C. B. Jones. Prentice-Hall, 1986.

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

B.30.8 Z и B

Цель

Z – это форма языка спецификации для последовательных систем и метод проектирования, позволяющий разработчику работу из спецификации Z путем, позволяющим доказать их точность по отношению к спецификации.

Z в основном используется на этапе формирования технических требований, ТЗ, спецификации, но метод был разработан для перехода от спецификации к проектированию и реализации. Больше всего он подходит к разработке последовательных информационно ориентированных систем.

B является ассоциативным методом.

Характеристика

Подобно VDM метод спецификации основан на использовании моделирования состояния системы в зависимости от теоретико-множественных структур, по которым определяются инварианты (предикаты), и работа в таком состоянии моделируется посредством определения их предварительных и последующих условий в зависимости от состояния системы. Операции могут быть испробованы для сохранения инвариантов системы, таким образом демонстрируется их состоятельность. Формальная часть спецификации делится на схемы, позволяющие структуризацию спецификаций посредством обработки.

В основном спецификация Z представляет собой композицию формального Z и неформального поясняющего текста на естественном языке. (Формальный текст сам по себе может быть слишком сжатым для легкого чтения, и часто его цель необходимо объяснять, в то время как естественный неформальный язык может легко стать неясным и неточным.)

В отличие от VDM Z представляет собой скорее форму, чем завершенный метод. Однако был разработан ассоциативный метод (называемый B), который может быть использован вместе с Z. Метод B основан на принципе пошаговой обработки.

Справочный материал

The Z Notation – A Reference Manual. J. M. Spivey, Prentice Hall, 1988.

Specification Case Studies. Edited by I. Hayes, Prentice-Hall, 1987.

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

The B Book – Assigning Programs to Meanings. J. R. Abrial, Cambridge United Press, 1996.

B.31 Нормальное доказательство

(Ссылка приведена в разделе 11.)

Цель

Доказательство корректности программы без ее выполнения с использованием теоретических и математических моделей и правил.

Характеристика

В разных частях программы установлено множество условий, которые служат в качестве предварительных и последующих условий к разным путям в программе. Доказательством будет демонстрация того, что программа переносит предварительные условия в последующие условия согласно набору логических правил и что программа прерывается.

В настоящем приложении описывается несколько формальных методов, например CCS, CSP, HOL, LOTOS, OBJ, временная логика, VDM и Z. Их характеристики приведены в B.30.

Справочный материал

Can Program Proving be made Practical. J. Dahl, Research Report, ISBN 82-90230-26-5 No. 33, Oslo, May.

B.32 Восстановление без возврата

(Ссылка приведена в разделе 9.)

Цель

Обеспечение правильности функционирования при наличии одного или более сбоев.

Характеристика

При обнаружении сбоя текущее состояние системы изменяется до состояния, которое позднее станет основным. Данная концепция является особенно подходящей для систем, работающих в реальном времени с маленькой базой данных и быстрым темпом изменений внутреннего состояния. Предполагается, что по крайней мере часть состояния системы может быть привязана к среде и только часть состояния системы подвержена влиянию среды.

B.33 Постепенный вывод из работы

Цель

Обеспечение выполнения наиболее критичных доступных функций системы, несмотря на отказы, вызванные исключением менее критичных функций.

Характеристика

С помощью данного метода происходит распределение приоритета среди различных функций, выполняемых системой. Проект предусматривает, что если для выполнения всех функций системы недостаточно ресурсов, выполняются функции высшего приоритета. Например, функции регистрации ошибок и событий могут быть менее приоритетными, чем функции контроля системы. Контроль системы будет продолжаться, если аппаратные средства, связанные с регистрацией ошибок, не сработают. Далее, если не срабатывают аппаратные средства управления системой, но при этом работают аппаратные средства регистрации ошибок, то аппаратные средства регистрации ошибок перенимают контрольные функции. Это преимущественно применяется по отношению к аппаратным средствам, но может применяться и к системе в целом, что должно учитываться с самого верхнего этапа проектирования.

Справочный материал

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, ISBN 3-211-81941-X, 1987.

Fault Tolerance, Principle and Practices. Vol. 3 of [2] above, T. Anderson and P. A. Lee, Springer-Verlag, ISBN 3-211-82077-9, 1987.

B.34 Исследование опасности и работоспособности (HAZOP)

Цель

Определение видов повреждений, ведущих к потенциально опасным ситуациям в контролируемой системе, посредством проведения серии систематических проверок компонентных секций компьютерной системы и ее работы.

К типичным опасным событиям в контролируемых системах относятся пожар, взрыв, выброс токсических материалов (химических или ядерных) или серьезные экономические потери.

Предполагается, что опасные события в контролируемых системах были определены при отдельном анализе рисков, где они были классифицированы по степени важности.

Анализ, охватывающий все этапы жизненного цикла проекта от спецификации через проектирование и до обслуживания и модификации, предназначен для идентификации на каждом этапе событий/видов отказов, которые могут привести к потенциальным опасностям, и, следовательно, их устранения.

Характеристика

Анализ проводится группой инженеров (отвечающих за вычислительные системы, контрольно-измерительные приборы, электрическое оборудование, технологический процесс, безопасность и эксплуатацию), возглавляемой специалистом, обученным методикам анализа рисков, посредством проведения запланированных совещаний.

Важно, чтобы был запланирован график совещаний на все время проекта; каждое из них должно быть запланировано по крайней мере на полдня, и для эффективности они должны проводиться не более четырех раз в неделю; также необходимо обеспечение соответствующей документацией.

До начала анализа должны быть составлены согласованные проверочные списки для систематической проверки, а также в отношении каждой секции системы должны быть заданы наводящие вопросы, такие как «Что будет, если это произойдет?», «Как это может произойти?», «Когда это может произойти?», «Имеет ли это большое значение?». После получения положительных ответов задаются следующие вопросы: «Что можно сделать в этом случае?», «Когда это должно произойти?», «Существует ли альтернатива?» и т. д.

В первой части анализа предполагается проверка установки компьютера в целом. Вторая и последующие части предполагают детальную проверку соответствующих частиц самих компьютерных систем.

В каждой части или на каждом этапе анализа целью является идентификация видов отказов, ведущих к потенциальным опасностям в контролируемой системе, и степени их воздействия. Большая часть составляющих частей компьютерной системы подразделяется далее и, при необходимости, подвергается отдельному анализу.

В любое необходимое время в ходе систематического анализа можно проводить обзорные совещания.

Важно вести подробные записи, поскольку они формируют значительную часть досье опасностей/безопасности системы.

После проведения ряда совещаний предполагается проведение обзорного совещания для обеспечения выполнения всех необходимых действий и включения модификаций, предложенных во время ознакомительных совещаний, в общий анализ и т. д.

Справочный материал

HAZOP and HAZAN. T. A. Kletz, 1986, 2nd Edition, Institution of Chemical Engineers, 165-171 Railway Terrace, Rugby, CV1 3HQ, UK.

Reliability and Hazard Criteria for Programmable Electronic Systems in the 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.

Reliability Engineering and Risk Assessment. E. J. Henlty and H. Kumamoto, Prentice-Hall, 1981.

Systems Reliability and Risk Analysis. E. G. Frenkel, Martinus Nijhogg 1984.

B.35 Анализ последствий

(Ссылка приведена в разделе 16.)

Цель

Идентификация последствий, вызванных изменением или усовершенствованием системы ПО на другие модули системы ПО, а также на другие системы.

Характеристика

До модификации или усовершенствования ПО необходимо провести анализ для идентификации влияния модификации или усовершенствования на ПО, а также идентификации вовлеченных программных систем и модулей.

После проведения анализа необходимо принять решение в отношении перепроверки ПО. Это зависит от количества вовлеченных модулей, критичности влиявших модулей и характера изменений. Возможные решения:

i) перепроверке подвергается только измененный модуль;

ii) перепроверке подвергаются все идентифицированные вовлеченные модули;

iii) перепроверке подвергается целая система.

B.36 Скрытие информации/инкапсуляция

(Ссылка приведена в таблице dt9.)

Цель

Повышение надежности и удобства обслуживания ПО.

Характеристика

Информация, глобально доступная всем составляющим ПО, может быть случайно или по ошибке изменена любым из этих составляющих. Любое изменение в структурах данных может потребовать детальной проверки кода и масштабных модификаций.

Скрытие информации – это общепринятый подход для минимизации проблем. Структуры основных данных «скрыты» и могут изменяться посредством определенного набора процедур доступа. Это позволяет внутренним структурам изменять или дополнять дальнейшими процедурами без влияния на функциональное поведение остального ПО. Например, у каталога имени могут быть следующие процедуры доступа: вставить, удалить и найти. Процедуры доступа и структуры внутренних данных могут быть переписаны (например, использовать другой искомый метод или сохранить имена на жестком диске) без влияния на логическое поведение остального ПО с использованием этих процедур.

Такая концепция типа абстрактных данных напрямую поддерживается рядом языков программирования, но можно применять основной принцип независимо от типа используемого языка программирования.

Справочный материал

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

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

B.37 Тестирование интерфейса

(Ссылка приведена в разделе 10.)

Цель

Демонстрация того, что интерфейсы подпрограмм не содержат каких-либо ошибок или таких ошибок, которые могли бы привести к отказу в особом программном приложении, или выявление всех релевантных ошибок.

Характеристика

Применяется несколько уровней детальной или полной проверки. Самые важные уровни – это проверка:

– всех переменных интерфейса в их крайних положениях;

– всех переменных интерфейса по отдельности в их крайних значениях с другими переменными интерфейса при нормальных значениях;

– всех значений домена каждой переменной интерфейса с другими переменными интерфейса при нормальных значениях;

– всех значений всех переменных в комбинации. Это осуществимо только для небольших интерфейсов;

– определенных условий проверки релевантных к каждому переходу каждой подпрограммы.

Эти проверки особенно важны, если их интерфейсы не содержат операторов контроля, которые обнаруживают неправильные параметрические значения. Они приобретают особую важность после появления новых конфигураций ранее существовавших подпрограмм.

B.38 Подмножество языка

(Ссылки приведены в разделе 10 и таблице dt4.)

Цель

Снижение вероятности возникновения программных сбоев и повышение вероятности обнаружения любых остающихся сбоев.

Характеристика

Проверяется язык для идентификации структурных программных компонентов, которые либо склонны к ошибкам, либо сложны для анализа, например применение методов статистического анализа. Затем определяется подмножество языка, исключающее такие структурные компоненты.

B.39 Запоминание решенных проблем

(Ссылка приведена в разделе 9.)

Цель

Повышение отказоустойчивости ПО в случае его нелицензионного выполнения.

Характеристика

Во время лицензирования ведется запись всех соответствующих деталей выполнения каждой программы. В процессе нормальной работы каждая программа сравнивается с набором лицензионного запуска. Если результаты отличаются, принимаются меры безопасности.

Учет выполнения может быть результатом прохождения индивидуальных путей «от решения к решению» (DDpaths), или индивидуальных доступов к массивам данных, записям/томам, или и того и другого.

Возможны различные методы хранения путей запусков. Можно использовать методы кодирования Нэша для обозначения последовательности выполнения одним большим числом или последовательностью чисел. В процессе нормальной работы значение пути выполнения должно проверяться по отношению к сохраненным случаям до проведения каких-либо операций по выводу.

Поскольку возможные комбинации путей «от решения к решению» во время одной программы достаточно велики, работа с программами как с единым целым может не представляться возможной. В этом случае допускается применение метода на модульном уровне.

Справочный материал

Fail-safe Software – Some Principles and a Case Study. W. Ehrenberger, Proc. SARSS 1987, Altringham, Manchester, UK, Elsevier Applied Science, 1987.

B.40 Библиотека надежных/проверенных модулей и компонентов

(Ссылка приведена в разделе 10.)

Цель

Исключение необходимости повторной проверки и повторной разработки программных модулей и аппаратных элементов для каждого нового приложения, а также содействие разработкам, которые не были формально или строго подвергнуты валидации, но имеют значительную эксплуатационную историю.

Характеристика

Хорошо разработанные и структурированные PES состоят из большого числа элементов аппаратных средств и ПО и четко выраженных модулей, которые взаимодействуют друг с другом четко определенным образом.

Различные PES, разработанные для различных приложений, будут содержать ряд одинаковых или подобных модулей или элементов. Составление каталога таких общеприменимых модулей позволяет использовать ресурсы, необходимые для валидации разработок, более чем в одном приложении.

Более того, использование таких модулей в разных приложениях обеспечивает эмпирическое доказательство успешного эксплуатационного применения. Такие эмпирические доказательства повышают степень доверия пользователей к модулям.

B.41 Модели Маркова

(Ссылка приведена в разделе 14.)

Цель

Оценивание надежности, безопасности и доступности системы.

Характеристика

Строится график системы. График представляет собой зависимость состояния системы от состояния отказа (состояния отказа отображаются точками на графике). Области между точками, представляющими состояния отказа или случаи их исправления, отображаются в соответствии с частотой отказов или исправлений. Необходимо обратить внимание на то, что событие, состояние или частота отказа конкретизируются таким образом, что можно получить точную характеристику системы, например обнаруженные или необнаруженные отказы, описание большего отказа и т. д.

Метод Маркова применим при моделировании резервных систем, в которых уровень резерва изменяется со временем из-за отказа и исправления элемента. Другие классические методы, например FMEA и FTA, не могут быть сразу адаптированы к моделированию эффектов отказов за весь жизненный цикл системы, поскольку не существует простых комбинаторных формул для исчисления соответствующих возможностей.

В самых простых случаях формулы, описывающие возможности системы, легкодоступны в литературе или могут быть вычислены вручную. Для более сложных случаев существуют методы симплификации (абсорбирующее состояние). Для особо сложных случаев результаты могут быть вычислены компьютерной имитацией графика.

Справочный материал

The Theory of Stochastic Processes. R. E. Cox and H. D. Miller, Methuen and Co. Ltd, London, UK, 1968.
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. Ushakov, 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.

B.42 Система показателей

(Ссылки приведены в разделах 11 и 14.)

Цель

Получение характеристик ПО непосредственно по его свойствам, а не в процессе его разработки или по истории его тестирования.

Характеристика

С помощью данных моделей оценивают некоторые структурные свойства ПО и относят их к предпочтительным характеристикам, таким как надежность или сложность. Для оценки большинства мер необходимы программные инструменты. Ниже приведены некоторые из рекомендуемых систем показателей:

– мера сложности, основанная на теории графов: данная мера может применяться на ранней стадии жизненного цикла для оценки альтернатив, которая основана на сложности графа управления программы, представленного его цикломатическим числом;

– количество методов для активации определенного модуля (доступность): чем доступнее будет модуль, тем скорее он будет налажен;

– теория ПО: данная мера вычисляет длину программы путем подсчета количества операторов и операндов. Это позволяет обеспечить меру сложности и оценить ресурсы разработки;

– количество входов и выходов на модуль: минимизация количества входов/выходов является ключевым свойством структурного проектирования и программных методов.

Справочный материал

A Complexity Measure. T. J. McCabe, IEEE Trans. on Software Engineering, Vol. SE-2, No. 4, Dec. 1976.
Models and Measurements for Quality Assessments of Software. S. N. Mohanty, ACM Computing Surveys, Vol. 11, No. 3, Sep. 1979.
Elements of Software Science. M. H. Halstead, Elsevier, North Holland, New York, 1977.

B.43 Модульный подход

(Ссылка приведена в таблице dt9.)

Цель

Разложение программных систем на более мелкие простые части для уменьшения сложности системы.

Характеристика

Модульный подход или разбиение на модули предполагает несколько правил разработки, кодирования и обслуживания этапов проектирования ПО. Данные правила варьируются в зависимости от метода, примененного при разработке. Большинство методов содержат следующие правила:

– модуль должен иметь одно четко определенное задание или функцию для выполнения;

– соединения между модулями должны быть ограниченными и четко определенными, согласованность в одном модуле должна быть строгой;

– сборки подпрограмм должны компоноваться при условии наличия нескольких уровней модулей;

– размеры подпрограмм должны быть ограничены до определенных установленных значений, обычно от 2 до 4 размеров экрана;

– у подпрограмм должно быть по одному входу и выходу;

– модули должны связываться с другими модулями через их интерфейсы. При применении глобальных или общих переменных они должны быть четко структурированы, доступ должен контролироваться, их использование должно обосновываться в каждой инстанции;

– все модульные интерфейсы должны быть полностью документально оформлены;

– каждый модуль должен скрывать что-либо из своей среды;

– интерфейс любого модуля должен содержать минимальное количество параметров, необходимых для функционирования модулей;

– должно быть установлено подходящее ограничение количества параметров – обычно 5.

B.44 Моделирование методом Монте-Карло

Цель

Моделирование эффекта реального окружения в ПО с использованием случайных чисел.

Характеристика

Моделирование методом Монте-Карло используется для решения проблем. Такие проблемы делятся на два класса:

– вероятностные проблемы, когда случайные числа используются для создания стохастической модели реального окружения;

– детерминированные проблемы, которые математически транслируются в эквивалентные вероятностные проблемы.

Моделирование методом Монте-Карло подразумевает ввод потоков случайных чисел для имитации шума при аналитическом сигнале или для добавления случайных систематических ошибок или устойчивости. Моделирование методом Монте-Карло применяется при производстве больших образцов, с помощью которых получают статистические результаты.

При моделировании методом Монте-Карло необходимо уделять внимание обеспечению приемлемости значений ошибок, толерантности и шума.

Общим принципом моделирования методом Монте-Карло является скорее переутверждение и переформулировка проблемы таким образом, чтобы полученные результаты были как можно более точными, чем решение проблемы в первоначальном виде.

B.45 Моделирование функционирования

(Ссылки приведены в таблицах dt2 и dt5.)

Цель

Обеспечение такой работоспособности системы, которая бы соответствовала установленным требованиям.

Характеристика

Спецификация требований содержит требования производительности и реагирования на определенные функции, возможно, в сочетании с ограничениями по использованию всех ресурсов системы. Разработка конкретной системы сравнивается с определенными требованиями путем:

– определения модели процессов системы и их взаимодействий;

– определения использования ресурсов каждым процессом, например время счета, пропускная способность канала связи, устройства памяти и т. д.;

– определения распределения требований, предъявляемых к системе при средних и наихудших условиях;

– расчета производительности и времени реагирования в средних и наихудших случаях для индивидуальных системных функций.

Для простых систем возможно аналитическое решение, в то время как для более сложных систем потребуется какая-либо форма моделирования для получения точных результатов.

До детального моделирования может применяться простая проверка «ресурсного бюджета», при которой суммируют требования ресурсов всех процессов. Если такие требования превышают расчетные возможности системы, то проект считается неосуществимым.

Даже если проект проходит эту проверку, моделирование работы может показать, что чрезмерные задержки и время реагирования происходят вследствие ресурсной перегрузки. Во избежание такой ситуации инженеры часто проектируют системы, в которых возможно использование некоторых долей (например, 50 %) общих ресурсов, вследствие чего уменьшается возможность ресурсной перегрузки.

B.46 Требования к функционированию

(Ссылка приведена в таблице dt6.)

Цель

Обеспечение удовлетворения требований к функционированию ПО системы.

Характеристика

Анализ выполняется в отношении спецификаций требований системы и ПО для идентификации всех общих и особых, конкретно выраженных и предполагаемых требований к функционированию.

Каждое требование к функционированию проверяется поочередно для определения:

– применяемых критериев успешной работы;

– того, могут ли быть обеспечены меры для применения критериев успешной работы;

– потенциальной точности таких измерений;

– этапов проекта, на которых возможно оценивание измерений;

– этапов проекта, на которых возможно выполнение измерений.

Целесообразность каждого функционального требования затем анализируется для получения перечня функциональных требований, критериев успеха и потенциальных измерений. Основными считаются следующие цели:

i) каждое функциональное требование связано по крайней мере с одним измерением;

ii) по возможности выбираются точные и эффективные измерения, которые могут быть выполнены в процессе разработки с самого начала;

iii) определяются неотъемлемые и добровольные функциональные требования и критерии успеха;

iv) следует стремиться извлекать выгоду за счет возможности выполнения одного измерения для более чем одного требования.

B.47 Вероятностное тестирование

(Ссылки приведены в разделах 11 и 13.)

Цель

Получение количественного выражения свойств надежности ПО. Данное число может относиться к соответствующим степеням достоверности и важности, а также:

i) вероятности отказа на каждый запрос;

ii) вероятности отказа за определенный период времени;

iii) вероятности локализации ошибки.

Возможно извлечение других параметров, таких как:
– вероятность безотказного исполнения;
– вероятность безотказной работы;
– доступность;
– среднее время наработки на отказ или интенсивность отказов;
– вероятность безопасного исполнения.

Характеристика

Вероятностные расчеты основываются на вероятностном тестировании или опыте эксплуатации. Обычно количество наблюдаемых случаев весьма велико.

Для того чтобы упростить тестирование, обычно применяются автоматические вспомогательные средства. Они имеют отношение к деталям обеспечения тестовых данных и контролю результатов теста. Сложные тесты проводят на сложном ведущем компьютере с соответствующим периферийным моделирующим процессом. Тестовые данные выбираются систематически, а также случайно. Первое касается общего контроля теста, например обеспечения профиля тестовых данных. Случайный отбор касается индивидуальных тестовых данных.

Оборудование для индивидуального тестирования, проведение тестирования, тестовых наблюдений определяется подробными целями сплошных тестов, описанных выше. Другие важные условия определяются через необходимую математическую подготовку, которую необходимо выполнять для того, чтобы оценить результаты тестирования согласно поставленной задаче тестирования.

Вероятностные числа, характеризующие поведение любого объекта тестирования, также могут быть получены из опыта эксплуатации. Если соблюдены аналогичные условия, применяется такая же математика, как и в случае с оценкой результатов проверки.

B.48 Имитация работы

(Ссылка приведена в таблице dt3.)

Цель

Функциональное тестирование ПО системы с ее внешними интерфейсами, при этом без какого-либо изменения реального окружения.

Характеристика

В целях тестирования необходимо создать систему, воспроизводящую поведение тестируемой системы.

Моделью может быть только ПО или ПО и аппаратные средства. Она должна:
– обеспечивать тестируемую систему всеми входными данными, которые будут существовать и при установке системы;
– реагировать на выходные данные системы способом, соответствующим объекту управления;
– иметь обеспечение для операторов ввода для предоставления каких-либо нарушений, с которыми тестируемой системе предстоит справиться.

При тестировании ПО могут имитироваться целевые аппаратные средства со своими входными и выходными данными.

Справочный материал

A Software Simulator – An Aid to Plant Commissioning. S. Nunns, EWICS TC7.
Physical Fault Simulation. F. Morillon, EWICS Document number WP460.

B.49 Создание прототипа/анимация

(Ссылки приведены в таблицах dt3 и dt5.)

Цель

Проверка возможности реализации системы вопреки ограничивающим условиям. Сообщение клиенту интерпретации технических требований, спецификаций на систему для выявления неверного понимания (ошибочного толкования).

Характеристика

Выбирается подкомплект функций системы и ограничивающих условий и требований к функционированию. Компонуется прототип с использованием высококачественных средств. На данном этапе рассматриваются ограничивающие условия, такие как целевой компьютер, реализация языка, размер программы, удобство обслуживания, надежность и доступность. Оценивается прототип по отношению к критериям пользователя, и в свете данной оценки возможна модификация требований системы.

Справочный материал

Proc. Working Conference on Prototyping. Namur Oct 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.

B.50 Блок восстановления

(Ссылка приведена в разделе 9.)

Цель

Повышение вероятности выполнения программой своей функции.

Характеристика

Записывается несколько различных частей программы, чаще независимо, каждая из которых предназначена для выполнения одной и той же заданной функции. Готовая программа конструируется из этих частей. Первая часть, называемая первичной, выполняется в первую очередь. За ней следуют приемочное испытание вычисляемого результата. Если результаты теста положительные, то результат принимается и передается последующим частям системы. Если результаты теста отрицательные, то все побочные эффекты первой части устраняются и запускается вторая часть, называемая первой альтернативой. За ней следует приемочное испытание, и результаты обрабатывается, как и в первом случае. При желании можно предусмотреть две, три или большее число альтернатив.

Справочный материал

System Structure for Software Fault Tolerance. B. Randall, IEEE Trans. Software Engineering, Vol. SE-1, No. 2, 1975.

B.51 Блок-схема надежности

(Ссылка приведена в разделе 14.)

Цель

Моделирование в виде схем набора событий, которые должны произойти, и условий, обязательных к соблюдению для успешной работы системы или выполнения задачи.

Характеристика

Целью анализа является успешное прохождение пути, состоящего из блоков, линий и логических соединений. Успешный путь начинается с одной стороны диаграммы и продолжается через блоки и соединения до другой стороны диаграммы. Блок представляет собой условие или событие, и путь может проходить через него, если условие является подлинным или произошло событие. Если путь приходит к соединению, он продолжается, если соблюдать логическое соединение. Если он доходит до вершины, он может продолжаться вдоль всех исходящих линий. Если существует по крайней мере один благоприятный путь через диаграмму, считается, что задача анализа выполняется верно.

Справочный материал

System Reliability Engineering Methodology: A Division of the State of the Art. J. B. Fusseland 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, U. S. Nuclear Regulatory Commission, Washington, D. C. 20555, 1981.

B.52 Ограничения времени реакции и памяти

(Ссылка приведена в таблице dt6.)

Цель

Обеспечение соответствия системы требованиям к памяти и временным требованиям.

Характеристика

Спецификация требований для системы и ПО включает память и требования к реагированию по специфическим функциям, возможно, объединенных с ограничениями по использованию общих ресурсов системы. Выполняется анализ, который определяет требования по распределению в рамках средних и наихудших условий. Данный анализ требует оценки использования ресурсов и затраченного времени для каждой функции системы. Данная оценка может быть получена несколькими способами, например в сравнении с существующей системой или в прототипе и аналоге критичных по времени систем.

B.53 Механизмы восстановления после сбоев путем повторного запуска

(Ссылка приведена в разделе 9.)

Цель

Попытка функционального устранения выявленного условия сбоя при помощи механизмов повторного запуска.

Характеристика

В случае выявления условия сбоя или ошибки предпринимаются попытки исправления ситуации путем перезапуска того же кода. Восстановление путем повторного запуска может быть таким же полным, как и процедура перезапуска и повторного включения, или решаться в виде ограниченной задачи перепланирования и повторного включения после перерыва в работе программы или после работы в ждущем режиме.

Методы повторного запуска широко применяются в случаях сбоев связи или при исправлении ошибок. Условия повторного запуска могут быть определены, исходя из ошибки коммуникационного протокола (контрольная сумма и др.) или из блокировки времени реагирования на подтверждение прохождения.

Справочный материал

The Theory and Practise of Reliable System Design. D. P. Siewiorek and R. S. Schwarz, Digital Press.

B.54 Метод мониторинга безопасности

(Ссылка приведена в разделе 9.)

Цель

Защита от ошибок в конечной спецификации и реализации ПО, которые отрицательно влияют на безопасность.

Характеристика

Методы мониторинга безопасности – это внешний мониторинг, выполняемый на отдельном компьютере применительно к другой спецификации. Такой мониторинг безопасности имеет отношение только к обеспечению выполнения основным компьютером безопасных, но необязательно правильных действий. Мониторинг безопасности постоянно осуществляет мониторинг главного компьютера. Метод мониторинга безопасности предотвращает переход системы в опасное состояние. Кроме того, если он обнаруживает, что основной компьютер переходит в потенциально опасное состояние, система возвращается в безопасное состояние при помощи внешнего контролирующего блока или основного компьютера.

Справочный материал

Using AI Techniques to Improve Software Safety. Proc. IFAC SAFECOMP 88, Sarlat, France, Oct. 1986, Pergamon Press 1986.

B.55 Анализ ложной цепи

(Ссылка приведена в таблице dt8.)

Цель

Обнаружение непредвиденных путей или логического потока в системе, который в определенных условиях инициирует нежелательную функцию или препятствует выполнению ожидаемой функции.

Характеристика

Путь ложной цепи может состоять из аппаратных средств, ПО, действий оператора или комплекса этих элементов. Ложные пути не являются результатом отказа аппаратных средств, но представляют собой скрытые условия, непреднамеренно спроектированные в системе или зашифрованные в программах, которые могут привести к нарушению функционирования при определенных условиях.

К ложным цепям относятся:

– скрытые пути, которые могут привести к возникновению электрического тока, энергии или логической последовательности по непредвиденному пути или в непреднамеренном направлении;

– ложный расчет времени, в котором события происходят в неожиданной или противоречивой последовательности;

– ложные индикаторы, приводящие к неоднозначному или неверному отображению режимов работы системы и, таким образом, способные привести к нежелательным действиям оператора;

– ложные ярлыки, которые неправильно или неточно обозначают функции системы, например ввод, контроль, дисплей, интерфейс системы и т. д., и, таким образом, способные ввести оператора в заблуждение при выполнении неверных действий с системой.

Анализ ложных цепей основывается на распознавании основных топологических образцов со структурой аппаратных средств или ПО (например, определяются шесть основных образцов для ПО). Анализ проводится с помощью перечня вопросов по применению и взаимосвязи между основными топологическими элементами.

Справочный материал

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.

B.56 Управление конфигурацией ПО

(Ссылка приведена в разделе 15.)

Цель

Задачей управления конфигурации ПО является обеспечение последовательности групп результатов разработки при изменении результатов. Управление конфигурацией в основном применяется как к аппаратным средствам, так и к ПО.

Характеристика

Управление конфигурацией ПО – это метод, применяемый при разработке. В сущности, он требует регистрации разработки каждой версии каждого «существенного» комплектующего узла и каждой связи между различными версиями различных комплектующих узлов. Регистрация результатов позволяет разработчику определить воздействие изменения одного комплектующего узла на другие комплектующие узлы (особенно на их элементы). В частности, системы или подсистемы могут быть надежно перестроены на основе согласованного множества версий элементов.

Справочный материал

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 Standard 05-57 Issue 1, 1980.

B.57 Языки программирования со строгим контролем типов

(Ссылка приведена в разделе 10.)

Цель

Снижение вероятности возникновения сбоев путем использования языка, предусматривающего высокий уровень проверки компилирующей программой.

Характеристика

Такие языки обычно позволяют определять типы данных, установленные пользователем, через основные типы данных языка (таких, как INTEGER, REAL). Эти типы могут быть далее использованы точно таким же образом, как и основные типы, но должны задаваться строгие проверки для обеспе-

чения использования верного типа. Данные проверки задаются в течение всей программы, даже если она выстраивается из отдельно скомпилированных единиц. Данные проверки также гарантируют соответствие количества и типа аргументов процедуры, даже когда на них ссылаются из отдельно скомпилированных модулей.

Языки со строгим контролем типов также поддерживают другие составляющие хорошей практики разработки ПО, такие как легкоанализируемые управляющие структуры (например, IF .. THEN .. ELSE, DO .. WHILE и т. д.), с помощью которых получают хорошо структурированные программы.

К типичным примерам языков со строгим контролем типов относятся Pascal, Ada и Modula 2.

Справочный материал

Reference Manual for the Ada Programming Language, ANSI/MIL-STD-815A 1983.

In search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software, A. Avizienis, M. R. Lyu, W. Schutz, The Eighteenth International Symposium on Fault Tolerant Computing, Tokyo, Japan 27–30 June 1988, IEEE Computer Society Press, ISBN 0-8186-0867-6.

B.58 Структурное тестирование

(Ссылка приведена в таблице dt2.)

Цель

Применение тестирования с использованием определенных подмножеств структуры программы.

Характеристика

На основе анализа программы набор входных данных выбирается таким образом, чтобы использовалась большая часть элементов выбранной программы. Использованные элементы программы могут отличаться в зависимости от уровня необходимой строгости.

Утверждения: это самый простой тест, т. к. все тестировщики могут выполнить его без перехода по обеим ветвям условного утверждения.

Ветви: каждую ветвь необходимо проверять по обоим направлениям. Это может быть непрактично для некоторых типов защитного кода.

Объединенные условия: каждое условие в объединенной условной ветви (т. е. связанные с использованием AND/OR).

LCSAJ: последовательность линейной программы и переход – это любая линейная последовательность кодовых утверждений, включая условные переходы, законченные переходом. Многие подветви кода могут быть неосуществимы из-за ограничений, налагаемых на входные данные, возникающих вследствие выполнения предшествующего кода.

Поток информации: выбираются выполняемые ветви на основе использования данных, например ветвь, где одна и та же переменная может быть записана или может записывать сама.

Граф вызова: программа составляется из подпрограмм, которые могут быть активизированы из других подпрограмм. Граф вызова представляет собой дерево активизации подпрограммы в программе. Тесты разрабатываются таким образом, чтобы покрыть все активизации подпрограммы в дереве.

Полный путь: запускает все возможные пути через код. Полное тестирование обычно не представляется возможным вследствие очень большого числа потенциальных путей.

Справочный материал

Reliability of the Path Analysis Testing Strategy. W. Howden, IEEE Trans. Software Engineering, Vol. SE 3, 1976.

B.59 Структурные диаграммы

(Ссылка приведена в таблице dt5.)

Цель

Представление структуры программы в виде диаграммы.

Характеристика

Структурные диаграммы представляют собой условные обозначения, дополняющие диаграмму потока данных. Они описывают систему программирования и иерархию частей, отображая это графически в виде дерева. Они документируют, как элементы диаграммы потока данных могут быть внедрены в виде иерархии программных единиц.

Структурная схема показывает отношение между программными единицами без включения какой-либо информации о порядке активации этих единиц. Они изображаются при помощи использования следующих трех символов:

i) прямоугольника, отмеченного наименованием единицы;

ii) стрелки, соединяющей эти прямоугольники;

iii) стрелки в круге, отмеченной наименованием данных, входящих и исходящих из элементов в структурной схеме. Обычно стрелка в круге наносится параллельно стрелке, соединяющей прямоугольники в схеме.

Из любой нетривиальной диаграммы потока данных можно получить ряд различных структурных схем.

Структурные схемы, полученные из диаграмм потока данных, представляют структуру системы первого уровня, где каждый прямоугольник структурной схемы представляет круг диаграммы потока данных. Само собой разумеется, что более глубокие уровни могут быть описаны с применением того же метода.

Справочный материал

Software Engineering. I. Sommerville, Addison-Wesley 1982. ISBN 0-201-13795-X.
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.

B.60 Структурная методология

(Ссылки приведены в разделах 8 и 10.)

Цель

Основной целью структурных методологий является улучшение качества разработки ПО путем фокусирования внимания на ранних этапах жизненного цикла. Цель методов – достичь этого через точные и интуитивно понятные процедуры и условные обозначения (с помощью компьютера) для установления того, что требования и особенности выполнения логически упорядочены и структурированы.

Характеристика

Существует ряд структурных методологий. Некоторые из них, такие как SSADM, LBMS, разработаны для традиционной обработки данных и функций обработки транзакций, тогда как другие (MASCOT, JSD, Йордон в режиме реального времени) больше сориентированы на приложения управления процессом в режиме реального времени (которые являются наиболее критичными с точки зрения безопасности).

Структурные методы по существу являются «инструментами мышления» для систематического восприятия и разбиения проблемы или системы. Их основные свойства – это:

– логический порядок мышления для разделения крупной проблемы на поддающиеся управлению стадии;

– идентификация всей системы, включая среду и необходимую систему;

– разделение данных и функций в необходимой системе;

– проверочные таблицы, например перечни типов объектов, которые необходимо определить;

– низкие интеллектуальные издержки – простые, интуитивные, прагматичные.

Поддержка условных обозначений больше направлена на точность в определении логического объекта проблемы и системы (например, способы обработки и потоки данных), но обрабатывающие функции, выполняемые ими, направлены на выражение с использованием неформальных условных обозначений. Однако некоторые методы частично используют (математически) формальные условные обозначения (например, JSD использует регулярное выражение; Йордон, SOM и SDL используют FSM). Такая четкость не только снижает вероятность неправильного толкования, она предоставляет возможности для автоматической обработки.

Другим преимуществом структурированного условного обозначения является его визуальность, которая дает возможность пользователю осуществить интуитивную проверку спецификаций или проектов, противоречащих его опыту и знаниям.

В настоящем приложении подробно описаны некоторые из этих структурных методологий, например выражение управляемых требований, разработка системы Джексона, MASCOT, Йордон в режиме реального времени и метод структурного анализа и проектирования (SADT).

Справочный материал

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

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

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

B.60.1 Выражение управляемых требований (CORE)

Цель

Обеспечение идентификации и выражения всех требований.

Характеристика

Данный подход направлен на восполнение пробелов между клиентом/конечным пользователем и аналитиком. Он не является математически строгим, но способствует облегчению процесса коммуникации – CORE спроектирован больше для выражения требований, чем для спецификаций. Подход структурируется, и представление проходит через различные уровни обработки. Процесс CORE способствует формированию более широкого подхода к проблеме, добавляя знание среды, в которой будет использоваться система, и различающиеся точки зрения различных типов пользователей. CORE включает руководящие указания и тактику для распознавания отправных точек «большого проекта». Отправные точки могут быть откорректированы или точно идентифицированы и документально оформлены. Таким образом, спецификации могут быть неполными, но определяются нерешенные проблемы и участки высокого риска, которые должны быть рассмотрены при последующей разработке.

B.60.2 JSD – разработка системы Джексона

Метод разработки, охватывающий разработку систем ПО от установления требований до кодирования с упором на системы, работающие в режиме реального времени.

Характеристика

JSD представляет собой поэтапный процесс разработки, в котором разработчик моделирует процессы реального времени, на основе которых должны базироваться функции системы, а также определяет необходимые функции и вводит их в модель и изменяет итоговую спецификацию в такую, которая является выполнимой в целевой среде. Таким образом, процесс охватывает традиционные этапы определения, проектирования и выполнения, но его отличие от традиционных методов заключается в том, что он не является нисходящим.

Более того, он акцентирует внимание на раннем этапе идентификации особенностей существования объектов действительности, которые следует учитывать при построении системы, а также на их моделировании и на том, что может с ними произойти. Как только этот анализ «реального мира» проведен и создана модель, анализируются необходимые функции системы для определения того, как они могут соответствовать модели реального мира. Получающаяся модель системы дополняется структурным описанием всех процессов в модели, и затем вся модель преобразуется в программы, которые будут оперировать в итоговых средах ПО и аппаратных средств.

Справочный материал

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

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

B.60.3 MASCOT

Цель

Разработка и внедрение систем, работающих в режиме реального времени.

Характеристика

MASCOT (модульный подход к компоновке, эксплуатации и тестированию ПО) – это метод разработки, поддерживаемый программируемой системой. Это систематический метод выражения структуры реального времени методом, независимым от объектных аппаратных средств или языка выполнения. Он устанавливает упорядоченный подход к разработке, который выдает высокомодулированную структуру, обеспечивая тесную связь между функциональными элементами в разработке и структурными элементами, возникающими при интеграции системы. Система разрабатывается через сеть параллельных процессов, связанных через каналы. Каналы могут представлять собой банки фиксированных данных или обеспечивать поочередность обработки данных (магистраль данных). Контроль доступа к каналам определяется независимо от процессов. Он определяется через механизмы дос-

тупа, которые также усиливают правила оптимизации по процессам. Последние версии MASCOT были разработаны с учетом внедрения Ada.

MASCOT поддерживает принцип приемки, основанный на тестировании и верификации единичных модулей и бóльших комбинаций функционально связанных модулей. Внедрение MASCOT должно быть основано на ядре MASCOT – наборе запланированных базисных элементов, которые придают особое значение внедрению и поддержке механизмов доступа.

Справочный материал

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

B.60.4 Йордон в режиме реального времени

Цель

Метод разработки полного ПО, состоящий из спецификации и методов разработки, ориентированных на системы, работающие в режиме реального времени.

Характеристика

Схема разработки этого метода предполагает трехфазное развитие разрабатываемой системы. Первая фаза включает компоновку базовой модели – такой модели, которая описывает поведение, необходимое для системы. Вторая фаза включает компоновку модели ввода, описывающей структуры и механизмы, которые при вводе реагируют требуемым образом. Третья фаза включает фактическое поведение системы в аппаратных средствах и ПО. Три фазы приблизительно соответствуют обычным этапам определения, разработки и внедрения, но при этом выделяется тот факт, что на каждом этапе разработчик вовлечен в деятельность по моделированию.

Необходимая модель представляется в двух частях: модель среды, содержащая определение границы между системой и ее средой и описание внешних событий, на которые должны отвечать система, и поведенческая модель, которая содержит схемы, определяющие преобразование, проводимое системой, в ответ на события и определение данных, которые должна содержать система для реагирования.

Модель внедрения также делится на подмодели: в таком случае охватываются распределение индивидуальных процессов по процессорам и разложение процессов на модули.

Для того чтобы собрать эти модели, метод комбинирует ряд хорошо известных методов: диаграммы потока данных, структурный английский, диаграммы состояний и сети Петри.

Кроме того, метод содержит способы моделирования разработки предлагаемой системы на бумаге или механически из составленных моделей.

Справочный материал

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

B.60.5 SADT – метод структурного анализа и проектирования

Цель

Моделирование и идентификация в схематичном виде с использованием информационных потоков, процессов принятия решений и задач управлений, связанных со сложной системой.

Характеристика

В SADT главную роль играет понятие диаграммы коэффициента использования (A/F). Диаграмма A/F состоит из действий, сгруппированных в так называемые «прямоугольники действий». Каждый прямоугольник действий имеет уникальное имя и связан с другими «прямоугольниками действий» через факторные отношения (изображенные в виде стрелок), которым также присвоены уникальные имена. Каждый «прямоугольник действий» может быть иерархично декомпозирован на второстепенные «прямоугольники действий» и связи. Существует четыре вида факторов: ввод, контроль, механизмы и вывод.

– Ввод: обозначен стрелкой, входящей в «прямоугольник действий» с левой стороны. Ввод может представлять материальные или нематериальные объекты и пригоден для манипуляции посредством одного или более действий в «прямоугольнике действий».

– Контроль: обычно представляет собой инструкции, процедуры, критерии выбора и т. д. Контролирует руководство по выполнению действия, обозначается стрелками, входящими в верхнюю часть «прямоугольника действий».

– Механизм: это ресурс, такой как персонал, организационные единицы или оборудование, необходимый при любой деятельности для выполнения задачи.

– Вывод: может означать что угодно, производимое действием, обозначенное стрелкой из «прямоугольника действия» с правой стороны.

Когда действия строго соотносятся друг с другом многими управляющими связями, вероятно, лучше рассматривать эти действия как неделимую группу, которая содержится в одном «прямоугольнике действий», содержание которого не подлежит дальнейшей детализации. Руководствующий принцип для группирования действий в один «прямоугольник действий» – это то, что итоговые прямоугольники соединены попарно только несколькими факторами.

Модельная иерархия диаграмм A/F ведется до того момента, пока дальнейшее детализирование «прямоугольников действий» не становится бессмысленным. Этот этап достигается, когда действия внутри прямоугольников являются неразделимыми или когда дальнейшее детализирование «прямоугольников действий» выходит за рамки системного анализа.

Справочный материал

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

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, April 1985, 25-34.

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

B.61 Структурное программирование

(Ссылка приведена в разделе 10.)

Цель

Разработка и внедрение программы таким образом, чтобы сделать анализ программы практичным. При анализе должны быть обнаружены все существенные реакции программы.

Характеристика

Программа должна иметь минимальную структурную сложность. Необходимо избегать сложных разветвлений. Петлевые ограничения и разветвления должны (по возможности) быть просто связаны с вводными параметрами. Программа должна быть соответствующим образом поделена на небольшие модули, и взаимодействие этих модулей должно быть ясным. Свойства языка программирования, способствующие применению вышеуказанного подхода, должны быть использованы предпочтительно по сравнению с другими свойствами, которые (по утверждению) более эффективны, за исключением того, что такая эффективность имеет абсолютную приоритетность (например, некоторые системы, критичные для безопасности).

Справочный материал

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

Assessing a Class of Software Tools. M. A. Hennell *et al.*, 7th International conference on Software Engineering, March 1984, Orlando.

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

B.62 Приемлемые языки программирования

(Ссылка приведена в таблице dt4.)

Цель

Максимальная поддержка требований настоящего стандарта, в частности защитного программирования, строгого контроля типов, структурного программирования и возможных операторов контроля. Выбранный язык программирования должен приводить к легко проверяемому коду с минимальными усилиями и облегчать разработку, верификацию и обслуживание программы.

Характеристика

Язык должен быть определен полностью и недвусмысленно. Язык должен быть скорее ориентирован на пользователя и проблему, чем быть машинно ориентированным. Широкоиспользуемые языки или их подмножества более предпочтительны по сравнению с языками со специальным предназначением.

Кроме того, помимо вышеназванных свойств, язык должен обеспечивать:

– блочную структуру;

– проверку времени трансляции;

– проверку типа оперативного времени и границы массива;

– проверку параметров.

Язык должен обусловливать:

– использование небольших и управляемых модулей;

– ограничение доступа к данным в определенных модулях;

– определение переменных подмножества значений;

– другие типы компонентов, ограничивающих ошибки.

Необходимо, чтобы язык поддерживался подходящим транслятором, соответствующими библиотеками существующих модулей, отладчиком и средствами для контроля и разработки версии.

К свойствам, которые затрудняют верификацию и которых, следовательно, необходимо избегать, относятся:

– операции безусловного перехода, исключающие переходы к подпрограмме;

– рекурсия;

– ссылки, динамическая память или любой вид динамических переменных или объектов;

– прерывание оперирования на уровне исходной программы;

– множественные входы или выходы из циклов, блоков или подпрограмм;

– срытое задание или описание переменной;

– вариантная запись или эквивалентность;

– процедурные параметры.

Языки низкого уровня, в частности ассемблерные языки, представляют проблемы вследствие их машинно ориентированного характера.

B.63 Символическое выполнение

(Ссылка приведена в таблице dt8.)

Цель

Обеспечение согласованности между исходной программой и спецификацией.

Характеристика

Программа выполняется с замещением левой части правой во всех заданиях. Условные ветви и звенья транслируются в булевские выражения.

Окончательный результат – это символическое выражение для каждой переменной программы. Это может быть проверено относительно ожидаемого выражения.

Справочный материал

Formal Program Verification using Symbolic Execution. R. B. Dannenberg and G. W. Ernst, IEEE Trans. Software Engineering, Vol. SE-8, No 1, 1982.

Symbolic Execution and Software Testing. J. C. King, Comm. ACM, Vol. 19, No. 7, 1976.

B.64 Временные сети Петри

(Ссылки приведены в таблицах dt5 и dt7.)

Цель

Моделирование соответствующих разновидностей поведения системы, оценка и возможное улучшение требований безопасности и эксплуатационных требований посредством анализа и переработки.

Характеристика

Сети Петри принадлежат к классу графических теоретических моделей, которые соответствуют представлению информации и контрольного потока в системах, демонстрирующих совпадение и асинхронное поведение.

Сеть Петри – это сеть позиций и преобразований. Позиции могут быть «помеченными» или «непомеченными». Преобразование запускается, когда все позиции ввода помечены. После запуска оно разрешено к «пуску» (но пуск не является обязательным). Если оно запускается, обозначения ввода снимаются, и вместо этого помечается каждая позиция вывода от преобразования.

Потенциальные опасности в модели представляются как особые состояния (обозначения). Расширенные сети Петри позволяют моделировать временные свойства системы. Несмотря на то, что классические сети Петри концентрируются на аспектах потоков управления, некоторые расширения были предложены для включения в модель потока данных.

Справочный материал

Net Theory and Applications. W. Brauer (ed), Lecture Notes in Computer Science, Vol. 84, Springer 1980.

Petri Net Theory and Modelling of Systems. J. L. Peterson, Prentice-Hall, 1981.

Safety Analysis using Petri Nets. N. Leveson and J. Stolzy, Proc. FTCS 15, Ann Arbor, Michigan, June 1985, IEEE 1985.

A Tool for Requirements Specification and Analysis of Real Time Software Based on Timed Petri Nets. S. Bologna, F. Pisacane, C. Ghezzi, D. Mandrioli, Proc. SAFECOMP 88, 9-11 Nov. 1988. Fulda Fed. Rep. of Germany 1988.

B.65 Транслятор, испытанный практическим применением

(Ссылка приведена в разделе 10.)

Цель

Недопущение сложностей, возникающих вследствие отказов, которые могут возникать во время разработки, верификации и обслуживания пакета программ.

Характеристика

Используется такой транслятор, исправность выполнения функций которого была продемонстрирована во многих проектах. Трансляторы, не применявшиеся на практике или имеющие любые серьезные известные ошибки, запрещены.

Если было продемонстрировано, что транслятор имеет какой-либо недостаток, относительные конструктивы языка фиксируются и избегаются в ходе проекта безопасности.

Другая версия такого способа работы – это ограничение пользования языком только до его распространенных свойств.

Данная рекомендация основана на опыте многих проектов. Было продемонстрировано, что незрелые трансляторы представляют собой серьезное препятствие в разработке любой программы. Они делают разработку ПО, связанного с безопасностью, неосуществимым.

Также известно, что в настоящее время не существует метода доказательства правильности для всех частей компилятора.

B.66 Прогоны/обзоры разработки

(Ссылка приведена в таблице dt8.)

Цель

По возможности быстрое и экономичное обнаружение ошибок в некоторых программах в процессе разработки.

Характеристика

IEC опубликовала IEC 61160, который содержит рекомендации по выполнению процедур по обзору разработки в качестве стимулирующей программы и для среды процесса. Он включает руководство по планированию и проведению проверок проекта, а также особые детали, касающиеся содействия специалистов, вносящих вклад в надежность, обслуживание, поддержку в обслуживании и доступность. Указанный стандарт также включает подразделы о вкладе других специалистов, работающих с сопутствующими объектами, такими как качество, среда, безопасность (программы и пользователя), человеческие факторы и правовые вопросы.

Кроме того, помимо общего понимания «программы», указывается включение других аспектов, например аппаратные средства и ПО, перечни каталогов, спецификации, описывающие наименование, транспортные упаковки, установку и сборку, инструкции и руководство по эксплуатации, этикетки и предупреждения, обслуживание, перечни запасных частей, гарантии, проспекты и рекламная литература.

При анализе программы группа по проведению анализа выбирает небольшой набор документально оформленных тестовых данных, представительские выборки входных данных и соответствующих предполагаемых выходных данных программы. Затем данные проверки отслеживаются вручную посредством алгоритма программы.

Справочный материал

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

International Standard IEC 61160: *Formal design review* (1992).

B.67 Нечеткая логика

(Ссылка приведена в разделе 10.)

Цель

Нечеткая логика – это математическая дисциплина, основанная на теории нечетких множеств, допускающая степени истинности и ложности. Это обобщение двухуровневой логики, которая предоставляет модель для обоснования. Она подразумевает включение человеческой логики в автоматические системы. Путем признания сложности определения точных границ нечеткая логика снижает необходимую четкость и таким образом приводит к высокоуровневым и простым, легко контролируемым решениям.

Характеристика

Основная часть решения нечеткой логики представляет собой комплект языковых правил (правила ЕСЛИ – ТОГДА), где предпосылки и последствия связаны с нечеткими множествами.

Пример – ЕСЛИ скорость высокая и расстояние_до_остановки среднее, ТОГДА ускорение около нуля и торможение нерезкое.

В данном примере «скорость» – это языковая переменная, характеризующаяся нечетким набором «быстроты», который может быть интерпретирован как «скорость выше примерно 70 км/ч». Если скорость ниже 60 км/ч, значение «быстроты» равно 0. Если скорость выше 80 км/ч, значение «быстроты» равно 1. Если скорость между 60 и 80 км/ч, значение «быстроты» варьируется между 0 и 1.

Логика, на основе которой принимается решение, основана на математических классах операторов: треугольные нормы и треугольные сонормы.

Системы нечеткой логики, основанные на правилах, отличаются от других экспертных систем во многом: (1) небольшое количество правил; (2) использование только прямого построения цепочки; (3) несвязывание интерференций (все правила выполняются параллельно в одном цикле); (4) статистически определенные правила; (5) скорость выполнения вследствие простоты решений; (6) детерминизм.

В течение последних нескольких лет нечеткая логика получила широкое распространение в различных областях: от финансов и до моделирования землетрясений. В частности, возник нечеткий контроль для контролирования высоконелинейных систем, систем, математические свойства которых неизвестны или слишком сложны для аналитической обработки, или систем, в которых доступные измерения имеют низкое качество.

К важным областям применения контроля нечеткой логики относится контроль авиационных полетов и электрических систем и контроль ядерного реактора. Позднее нечеткий контроль был успешно применен в системах управления автоматических поездов.

Справочный материал

Fuzzy Sets: Zadeh. Information and Control, 1965, vol. 8, p. 338–353.

Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I and II. Chuen Chien Lee. IEEE Transactions on systems, man, and cybernetics, vol. 20, No. 2, March/April 1990.

Industrial Applications of Fuzzy Control: M. Sugeno Ed., Amsterdam North Holland, 1985.

Automatic Train Operation System by Predictive Fuzzy Control: Yasunobu, Miyamoto, in Industrial Applications of Fuzzy Control, M. Sugeno Ed., Amsterdam North Holland, 1985.

An automatic operation method for control rods in BWR plants: Kinoshita, Fukusaki, Satoh, Miyake, Proc. Specialists Meeting on In-Core Instrumentation and Reactor Core Assessment. Cadarache, France 1988.

Use of rule-based system for process control. Bernard. IEEE Contr. Syst. Mag., Vol. 8, No. 5, p. 3-13, 1988.

B.68 Объектно ориентированное программирование

(Ссылка приведена в разделе 10.)

Цель

Обеспечение возможности быстрого создания прототипа, облегчение многократного применения существующих элементов ПО, достижение сокрытия информации, снижение вероятности возникновения ошибок в течение всего жизненного цикла, уменьшение необходимых усилий на этапе обслуживания, разделение сложных проблем на более легко управляемые небольшие проблемы, снижение зависимости между элементами ПО, создание более легко расширяемых приложений.

Характеристика

Объектно ориентированное программирование – это фундаментально новый метод мышления в отношении ПО, основанный на абстрактных конструкциях, которые существуют в реальном мире, а не на машинных абстракциях. Объектно ориентированное программирование организует ПО как коллекцию объектов, которые объединяют и структуру, и поведение данных. Это отличается от обычно применяемого программирования, в котором структуры и поведение данных свободно соединены.

Объект: объект состоит из участка приватных данных и набора операций – так называемых методов – на данном объекте. Методы могут быть общедоступными и приватными. Ни для какого другого элемента ПО не предусматривается прямое чтение или изменение приватных данных объекта. Любой другой элемент ПО должен применять общедоступные методы в отношении этого объекта для чтения или записи данных на участке приватных данных объекта.

Класс объекта: путем установления класса объекта (часто в форме определения типа) запускают конкретизацию многочисленных объектов такого же класса, т. е. все конкретизации имеют участок приватных данных и методы, определенные в классе объекта.

(Множественное) наследование свойств: класс объекта может наследовать участок приватных данных и методы одного (или более) суперклассов (объектных классов, находящихся выше по классу в иерархии) с учетом того, что можно добавлять некоторые приватные данные, добавлять некоторые методы или изменять выполнение наследованных методов. Применяя наследование, можно построить деревья класса множественных объектов.

Полиморфизм: одна и та же операция может вести себя по-разному для разных классов объектов, например операция записи для объекта терминала записывает символы на этот терминал, а операция записи для файлового объекта записывает на данный файл.

Справочный материал

Object Oriented Software Construction, Bertrand Meyer, England: Prentice-Hall International, 1988.

Classification as a paradigm for computing, Peter Wegner, Technical Report CS-86-11, Brown University, May 1986.

Learning language, Peter Wegner, Byte (McGraw-Hill publication), March 1989.

All OOPSLA (Object-Oriented Programming Systems, Languages and Applications) and ECOOP (European Conference on Object-Oriented Programming) conference proceedings.

B.69 Прослеживаемость

(Ссылка приведена в разделе 11.)

Цель

Целью прослеживаемости является обеспечение того, что все требования могут быть надлежащим образом выполнены, а также того, чтобы не было непрослеживаемых данных.

Характеристика

Прослеживаемости требований следует уделять особое значение при валидации системы, а также необходимо предоставить средства для обеспечения демонстрации этого на всех стадиях жизненного цикла.

Прослеживаемость считается применимой к функциональным и нефункциональным требованиям, и должны, в частности, рассматриваться:

a) прослеживаемость требований к разработке или другим объектам, которые выполняют эти требования;

b) прослеживаемость объектов разработки до объектов внедрения, которые их реализуют;

c) прослеживаемость требований и объектов разработки до эксплуатационных объектов и объектов обслуживания, которые должны применяться для безопасности и надлежащего использования системы;

d) прослеживаемость объектов требований, разработки, внедрения, эксплуатации и обслуживания до верификации и тестовых планов и спецификаций, которые определят их приемлемость;

e) прослеживаемость верификационных и тестовых планов и спецификаций до тестовых или других отчетов, в которых фиксируются результаты их применения.

В случаях, когда объекты требований, разработки или другие объекты подвергаются обработке как ряд отдельных документов, прослеживаемость должна поддерживаться в пределах документальных структур и в иерархическом порядке.

Вывод процесса прослеживаемости должен подвергаться формальному управлению конфигурацией.

Приложение Д.А
(справочное)

Сведения о соответствии государственных стандартов ссылочным международным стандартам

Таблица Д.А.1 – Сведения о соответствии государственного стандарта ссылочному международному стандарту

Обозначение и наименование международного стандарта	Степень соответствия	Обозначение и наименование государственного стандарта
ISO 9000:2000 Системы менеджмента качества. Основные положения и словарь	IDT	СТБ ИСО 9000-2006 Системы менеджмента качества. Основные положения и словарь

Таблица Д.А.2 – Сведения о соответствии государственного стандарта ссылочному международному стандарту другого года издания

Обозначение и наименование международного стандарта	Обозначение и наименование международного стандарта другого года издания	Степень соответствия	Обозначение и наименование государственного стандарта
ISO 9001:1994 Система качества. Модель для обеспечения качества при проектировании, разработке, производстве, монтаже и обслуживании	ISO 9001:2008 Системы менеджмента качества. Требования	IDT	СТБ ISO 9001-2009 Системы менеджмента качества. Требования

Ответственный за выпуск *В. Л. Гуревич*

Сдано в набор 10.05.2011. Подписано в печать 02.06.2011. Формат бумаги 60×84/8. Бумага офсетная.
Гарнитура Arial. Печать ризографическая. Усл. печ. л. 9,41 Уч.-изд. л. 5,97 Тираж экз. Заказ

Издатель и полиграфическое исполнение:
Научно-производственное республиканское унитарное предприятие
«Белорусский государственный институт стандартизации и сертификации» (БелГИСС).
ЛИ № 02330/0552843 от 08.04.2009.
ул. Мележа, 3, комн. 406, 220113, Минск.