ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ КАЗАХСТАН

ПОДВИЖНОЙ СОСТАВ ЖЕЛЕЗНЫХ ДОРОГ
Системы связи, сигнализация и обработки данных
Программное обеспечение для систем управления и защиты на железной дороге

СТ РК МЭК 62279 - 2007

(IEC 62279:2002 Railway applications communications, signaling and processing systems – Software for railway control and protection systems, IDT)

Издание официальное

Комитет по техническому регулированию и метрологии Министерства индустрии и торговли Республики Казахстан (Госстандарт)

Астана

Предисловие

1 РАЗРАБОТАН ТОО «Национальный центр аккредитации»
ВНЕСЕН Комитетом путей сообщения Министерства транспорта и коммуникации республики Казахстан

2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Комитета по техническому регулированию и метрологии Министерства индустрии и торговли Республики Казахстан от 27 сентября 2007 года № 546

3 Настоящий стандарт идентичен международному стандарту МЭК 62279-2002 «Подвижной состав железных дорог. Системы связи, сигнализация и обработки данных. Программное обеспечение для систем управления и защиты на железной дороге» (IEC 62279:2002 Railway applications communications, signaling and processing systems – Software for railway control and protection systems), IDT

4 СРОК ПЕРВОЙ ПРОВЕРКИ **2012 год**
ПЕРИОДИЧНОСТЬ ПРОВЕРКИ **5 лет**

5 ВВЕДЕН ВПЕРВЫЕ

Содержание

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ КАЗАХСТАН

ПОДВИЖНОЙ СОСТАВ ЖЕЛЕЗНЫХ ДОРОГ
Системы связи, сигнализация и обработки данных
Программное обеспечение для систем управления и защиты на железной дороге

Дата введения 2008.07.01.

1 Область применения

Настоящий стандарт устанавливает требования процедуры и технические требования к разработке программируемых электронных систем для использования в программах контроля и защиты, подвижного состава железных дорог. Он предназначен для использования в любой области, подразумевающей последствия выполнения техники безопасности. Они могут варьироваться от очень важных, таких как сигнализация для технической безопасности до второстепенных, таких как информационные системы управления. Данные системы могут реализовываться с использованием специализированных микропроцессоров, логических программируемых контроллеров, многопроцессорных распределительных систем, более крупномасштабных систем центрального процессора или других структур.

Настоящий стандарт применим исключительно к программному обеспечению и взаимодействию между программным обеспечением и системой, частью которой оно является.

Уровни безопасности программного обеспечения выше нуля предназначены для использования в системах, в которых последствия ошибки могут включать сокращение срока службы. Экономические или экологические соображения, однако, могут также служить основанием для использования более высоких уровней безопасности программного обеспечения.

Настоящий стандарт применяется ко всему программному обеспечению, используемому в разработке и реализации систем контроля и защиты железных дорог, включая:

- Прикладное программирование;
- Операционные системы;
- Вспомогательные инструменты;
- Встроенное программное обеспечение.

Прикладное программирование состоит из программирования высокого уровня, программирования низкого уровня и целевого программирования (например, многозвенная логическая схема Логических программируемых контроллеров).

Использование стандарта, имеющегося на рынке программного обеспечения и инструментов также рассматриваются в данном стандарте.

Настоящий стандарт также рассматривает требования систем, конфигурация которых была произведена посредством прикладных данных.

Настоящий стандарт не предназначен для рассмотрения коммерческих вопросов. Они должны рассматриваться как неотъемлемая часть любого соглашения, основанного на договоре. Все пункты этого стандарта требуют тщательного рассмотрения в любой коммерческой ситуации.

Издание официальное

Настоящий стандарт не предназначен для ретроспективного применения. Поэтому он применяется к новым разработкам и применяется в целом к существующим системам, если они подлежат существенным изменениям. Пункт 16 применяется в случае незначительных изменений.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие стандарты:

СТ РК 2.4-2000 ГСИ РК. Поверка средств измерений. Организация и порядок проведения.

СТ РК 2.12-2000 ГСИ РК. Система калибровки Республики Казахстан. Калибровка средств измерений. Организация и порядок проведения.

СТ РК ИСО 9000-2000 Системы менеджмента качества. Основные положения и словарь.

СТ РК ИСО 9001-2001 Системы менеджмента качества. Требования.

СТ РК ИСО/МЭК 17025-2001 Общие требования к компетентности испытательных и калибровочных лабораторий.

СТ РК ИСО МЭК 62280-1-2007 Подвижной состав железных дорог. Системы связи, сигнализации и обработки данных. Часть 1. Обеспечение безопасности связи в закрытых системах передачи.

СТ РК ИСО МЭК 62280-2-2007 Подвижной состав железных дорог. Системы связи, сигнализации и обработки данных. Часть 2. Обеспечение безопасности связи в открытых системах передачи.

3 Термины и определения

В настоящем стандарте применены термины по [1], [2], [3], [4], а также следующие термины с соответствующими определениями:

3.1 Оценка: Процесс анализа для определения того, достигли ли Управление по проектированию и Лицо, подтверждающее подлинность, продукта, который соответствует указанным требованиям и сформировали решение относительного того, подходит ли продукт по назначению.

3.2 Оценщик: Лицо или агент, назначенный для проведения оценки.

3.3 Эксплуатационная готовность: Способность продукта выполнять требуемую функцию при заданных условиях в данный момент, допуская, что требуемые внешние ресурсы предоставлены.

3.4 Коммерчески доступное программное обеспечение (COTS): программное обеспечение, определенное рыночными требованиями, коммерчески доступное, чья пригодность для выполнения поставленной цели демонстрировалась широким спектром коммерческих пользователей.

3.5 Управление по проектированию: Орган, ответственный за формирование проектного решения для выполнения указанных требований и наблюдения за последующим развитием и вводом системы в эксплуатацию в предназначенную среду.

3.6 Постановщик: Одно или несколько лиц, назначенных Управлением по проектированию для анализа и трансформации приведенных требований в приемлемые проектные решения, имеющие требуемую полноту безопасности.

3.7 Элемент: Часть продукта которая была определена как основная единица или структурный элемент. Элемент может быть простым или сложным.

3.8 Ошибка: Отклонение от заданного проекта, что может привести к непредусмотренному поведению системы или сбою.

3.9 Сбой: Отклонение от заданных рабочих показателей системы. Сбой представляет собой последствие неисправности или ошибки в системе.

3.10 Неисправность: Ненормальный режим, который может привести к ошибке или сбою системы. Неисправность может быть случайной или системной.

3.11 Предотвращение неисправностей: Использование проектных методов, которые подразумевают предотвращение неисправностей во время проектирования и разработки системы.

3.12 Устойчивость к ошибкам: Встроенные возможности системы по обеспечению постоянного правильного предоставления услуги как задано при ограниченном количестве неисправностей аппаратного оборудования или программного обеспечения.

3.13 Встроенное программное обеспечение: Заказанный комплект инструкций и связанные данные, которые хранятся таким образом, чтобы быть функционально независимыми от основной памяти, как правило в ПЗУ.

3.14 Комплексное программное обеспечение: Комплексное программное обеспечение - это программное обеспечение, которое может быть использовано для разнообразных инсталляций просто посредством предоставления прикладных данных.

3.15 Внедренец: Один или более лиц, назначенных Управлением по проектированию для трансформации определенных проектов в их физическую форму.

3.16 Продукт: Множество элементов, связанных между собой для формирования системы, подсистемы или единицы оборудования, таким образом, чтобы соответствовать заданным требованиям. В этом стандарте продукт может рассматриваться как состоящий полностью из элементов программного обеспечения или документации.

3.17 Логический программируемый контролер (ЛПК: Система управления на полупроводниковых приборах с памятью, программируемой пользователем для хранения инструкций с целью реализации определенных функций.

3.18 Надежность: Способность единицы выполнять заданную функцию при определенных условиях за определенный период времени.

3.19 Прослеживаемость требований: Задачей прослеживаемости требований является обеспечение того, чтобы требования могли быть показаны как соответственно выполненные.

3.20 Риск: Комбинация частоты или вероятности и последовательность определенных опасных случаев.

3.21 Безопасность: Свобода от неприемлемых уровней риска.

3.22 Управление безопасности: Орган, ответственный за сертификацию системы, связанной с безопасностью, для подтверждения того, что она подходит для работы и соответствует необходимым законодательным и правовым требованиям безопасности.

3.23 Программное обеспечение, связанное с безопасностью: Программное обеспечение, ответственное за безопасность.

3.24 Программное обеспечение: Продукт умственной деятельности, состоящий из программ, процедур, правил и любой связанной документации, относящейся к работе системы.

Примечание - Программное обеспечение независимо средств, используемых для перемещения.

3.25 Жизненный цикл программного обеспечения: Деятельность, происходящая в течение периода времени, который начинается, когда программное обеспечение сконструировано и заканчивается, когда программное обеспечение больше не доступно для использования. Жизненный цикл программного обеспечения, как правило, включает этап реализации, этап развития, этап испытания, этап интеграции, этап установки и этап

обслуживания.

3.26 Ремонтопригодность программного обеспечения: Возможность внесения изменений в систему для исправления неисправностей, улучшения работы других параметров или его адаптация к другой среде.

3.27 Обслуживание программного обеспечения: Действие или набор действий, произведенных с программным обеспечением после его приемки конечным пользователем. Задачей является усовершенствование, увеличение и/или исправлении его функциональности.

3.28 Уровень полноты безопасности программного обеспечения: Классификационный номер, который определяет методы и меры, которые должны быть применены для снижения остаточных неисправностей программного обеспечения до надлежащего уровня.

3.29 Уровень полноты безопасности системы: Номер, который указывает требуемую степень достоверности для того, чтобы система соответствовала указанным требованиям безопасности.

3.30 Прослеживаемость: Уровень создания связи между двумя или более продуктами процесса развития, особенно теми, у которых есть связь предшественника/преемника или главного/второстепенного по отношению друг к другу.

3.31 Подтверждение подлинности: Деятельность демонстрирующая, посредством анализа или испытания, что продукт во всех отношениях, соответствует указанным требованиям.

3.32 Лицо, подтверждающее подлинность: Лицо или агент, назначенный для подтверждения подлинности.

3.33 Проверка подлинности: Деятельность демонстрирующая, посредством анализа или испытания, что результат каждого этапа жизненного цикла соответствует требованиям предыдущего этапа.

3.34 Лицо, проверяющее подлинность: Лицо или агент, назначенный для проверки подлинности.

4 Задачи и согласованность

В каждом из последующих подпунктов приводятся подробности касательно задач и требований пункта.

Для соответствия данному стандарту, необходимо продемонстрировать, что каждое требование было удовлетворено до заданного уровня полноты безопасности таким образом выполняя задачу пункта.

Требование квалифицировано как «до степени, требуемой уровнем полноты безопасности программного обеспечения» указывает на то, что спектр методов и мер может использоваться для удовлетворения требования.

Где применяется пункт 4.3, таблицы подробно приведенные в данном стандарте должны быть использованы для помощи в выборе методов и мер соответствующих уровню полноты безопасности программного обеспечения.

Если метод или мера, ранжированы как высоко рекомендуемые в таблицах, тогда логическое обоснование не использования данного метода должно быть детально приведено и зарегистрировано в Плане гарантии качества программного обеспечения или другом документе, на который дается ссылка в Плане гарантии качества программного обеспечения. В этом нет необходимости при использовании согласованной комбинации методов, приведенных в соответствующей таблице.

В случае предложения к использованию метода или меры, которые не содержаться в таблицах, их эффективность и пригодность в отношении соответствия определенному

требованию и общей задачи пункта должна быть обоснована и зарегистрирована либо в Плане гарантии качества программного обеспечения либо другом документе, на который дается ссылка в Плане гарантии качества программного обеспечения.

Соответствие требованиям определенного пункта и их соответствующие методы, детально приведены в таблицах. Они подлежат оценке посредством проверки документов, требуемых данным стандартом, другими объективными данными, аудитом и освидетельствованием испытаний.

Данный стандарт требует использования пакета методов и их корректного применения. Эти методы обязательны, согласно таблицам, детальная информация представлена в библиографии.

5 Уровни полноты безопасности программного обеспечения

5.1 Задачи

Описать задачу уровней полноты безопасности программного обеспечения для программного обеспечения.

5.2 Требования

5.2.1 Согласно [5], [6] следующее подлежит разработке:

Спецификации требований системы,

Спецификации требовании системы в отношении безопасности,

Описание архитектуры системы,

План безопасности системы,

Что включает:

Функции безопасности;

Конфигурацию или архитектуру системы;

Требования в отношении надежности аппаратного обеспечения;

Требования полноты безопасности.

Уровень полноты безопасности программного обеспечения будет приведен в ходе прохождения общего процесса получения уровня полноты безопасности, заданного в [5].

Требуемый уровень полноты безопасности определяется на основе уровня риска, связанного с использованием программного обеспечения в системе и уровня полноты безопасности системы.

Без дальнейших мер предосторожности, уровень полноты безопасности программного обеспечения должен быть идентичным уровню полноты безопасности системы. Однако, если существуют механизмы предотвращения сбоев модуля программного обеспечения, таким образом приводя систему в небезопасное состояние, уровень полноты безопасности программного обеспечения модуля может быть снижен.

Риски, которые могут быть приняты во внимание – это те, которые связанны со следующими последствиями риска:

Гибель человека или людей;

Травмы или заболеванию людей;

Экологическое загрязнение; и

Потеря или повреждение собственности.

5.2.5 Риски могут быть выражены в количественной форме, однако не представляется возможным привести полноту безопасности программного обеспечения таким же образом. Поэтому полнота безопасности программного обеспечения для данного стандарта будет задана как один из следующих пяти уровней:

Уровень полноты безопасности программного обеспечения	Описание полноты безопасности программного обеспечения
4	Очень высокий
3	Высокий
2	Средний
1	Низкий
0	Не связанный с безопасностью

5.2.6 Уровень полноты безопасности программного обеспечения должен быт задан в спецификациях требований программного обеспечения (пункт 8). Если разные компоненты программного обеспечения имеют разные уровни полноты безопасности программного обеспечения, то он должны быть заданы в спецификациях архитектуры программного обеспечения (пункт 9).

6 Персонал и ответственность

6.1 Задачи

Обеспечить такое состояние дел, чтобы весь персонал ответственный за программное обеспечение был компетентен выполнять данную работу.

6.2 Требования

Поставщик и/или разработчик и потребитель должны реализовать соответствующие части СТ РК ИСО 9001 согласно руководств.

За исключением случаев, когда уровень полноты безопасности программного обеспечения, процесс безопасности должен реализовываться под контролем соответствующей организации по безопасности, которая соответствует подпункту «Организации по безопасности» пункта «Свидетельства безопасного управления» [7].

Весь персонал, вовлеченный во все фазы Жизненного цикла программного обеспечения, включая управленческую деятельность должен быть соответственно обучен, иметь соответствующий опыт и квалификации.

Очень рекомендуется, чтобы обучение, опыт и квалификации всего персонала, вовлеченного во все фазы Жизненного цикла программного обеспечения, включая управленческую деятельность, были обоснованы по отношению к определенному применению за исключением случаев, когда уровень полноты безопасности программного обеспечения равен нулю.

Обоснование, содержащееся в пункте 6.2.4 регистрируется в Плане гарантии качества программного обеспечения и, соответственно, включает свидетельства компетентности в следующих областях:

a) проектирование соответствующее области применения;
b) программное проектирование;
c) проектирование компьютерных систем;
d) проектирование безопасности;
e) правовая и нормативная структура.

Назначается независимый оценщик программного обеспечения.

См. Также 6.2.10 и 14.4.1.

Оценщик получает полномочия на проведение оценки программного обеспечения.

В течение жизненного цикла программного обеспечения, вовлеченные стороны будут независимы до степени, требуемой уровнем полноты безопасности программного обеспечения согласно рисунка 5, что должно интерпретироваться следующим образом.

При всех уровнях полноты безопасности программного обеспечения, оценщик утверждается управлением по безопасности, независимо от поставщика за исключением обстоятельств, определенных в пункте 6.2.10.

Постановщик/внедренец, лицо, проверяющее подлинность и лицо, подтверждающее подлинность могут все принадлежать одной компании, но следующие правила минимальной независимости должны выполняться.

При 0 уровне полноты безопасности программного обеспечения:

Ограничений не существует;

Постановщик/внедренец, лицо, проверяющее подлинность и лицо, подтверждающее подлинность все могут быть одним лицом.

При 1 и 2 уровне полноты безопасности программного обеспечения:

Лицо, проверяющее подлинность и лицо, подтверждающее подлинность могут быть одним лицом, но они не должны быть Постановщиком/внедренцем. Однако, Постановщик/внедренец, лицо, проверяющее подлинность и лицо, подтверждающее подлинность все могут отчитываться через менеджера проекта.

При 3 и 4 уровне полноты безопасности программного обеспечения существует 2 разрешенные схемы.

Лицо, проверяющее подлинность и лицо, подтверждающее подлинность могут быть одним лицом, но они не могут быть постановщиком/внедренцем. Кроме того, лицо, проверяющее подлинность и лицо, подтверждающее подлинность не все отчитываются через менеджера проекта, так как делают постановщик/внедренец, они имеют полномочия предотвращать выпуск продукции.

Постановщик/внедренец, лицо, проверяющее подлинность и лицо, подтверждающее подлинность все должны быть отдельными лицами. Постановщик/внедренец и лицо, проверяющее подлинность могут отчитываться через менеджера проекта, в то время как лицо, подтверждающее подлинность не может этого делать, кроме того, последний полномочен предотвращать выпуск продукта.

6.2.9 Стороны, ответственные за разные пункты приведены ниже:

Спецификации требований программного обеспечения (пункт 8)- постановщик.

Спецификации к требованиям испытаний программного обеспечения (пункт 8)- лицо, подтверждающее подлинность.

Архитектура программного обеспечения (пункт 9) – постановщик.

Проектирование и разработка программного обеспечения (пункт 10) – постановщик.

Проверка подлинности и испытание программного обеспечения (пункт 11) - лицо, проверяющее подлинность

Интеграция программного обеспечения/Аппаратного оборудования (пункт 12) – постановщик.

Подтверждение подлинности программного обеспечения (пункт 13) - лицо, подтверждающее подлинность.

Оценка программного обеспечения (пункт 14) – оценщик.

6.2.10 По распоряжению управления по безопасности, оценщик может быть частью организации поставщика, но в таких случаях, оценщик должен:

Быть уполномочен управлением по безопасности;

Быть полностью независимым от группы проекта;

Напрямую отчитываться управлению по безопасности.

7 Вопросы жизненного цикла и документация

7.1 Задачи

Построить разработку программного обеспечения в определенных фазах и деятельности.

Регистрировать всю информацию, относящуюся к программному обеспечению в течение жизненного цикла программного обеспечения.

7.2 Требования

Выбирается модель жизненного цикла для разработки программного обеспечения. Это должно быть подробно расписано в плане гарантии качества программного обеспечения согласно пункта 15 настоящего стандарта. Например, две модели жизненного цикла показаны на рисунках 3 и 4.

Процедуры гарантии качества ведутся параллельно с деятельностью жизненного цикла и используют одинаковую терминологию.

Вся деятельность, подлежащая выполнению во время фазы определяется до начала таковой. Каждая фаза жизненного цикла программного обеспечения делится на элементарные задачи с четко определенным вводом, выводом и деятельностью по каждой из них.

План гарантии качества программного обеспечения описывает необходимые шаги и отчеты по проверке подлинности.

Все документы строятся таким образом, чтобы позволить постоянное расширение параллельно с процессом развития.

Прослеживаемость документов обеспечивается каждым документом с уникальным ссылочным номером и определенной, документально оформленной связью с другими документами. Каждый термин, акроним или сокращение имеют одно значение в каждом документе. Если по ранее имевшим место причинам это не представляется возможным, разнящиеся значения должны быть перечислены с указанием ссылок.

Кроме того, каждый документ, за исключением документов программного обеспечения COTS (см. пункт 9.4.5) или ранее разработанное программное обеспечение (см. пункт 9.4.6) разрабатывается согласно следующим правилам:

Он должен содержать или реализовывать все применимые условия и требования предыдущего документа, с которым у него есть иерархическая связь;

Он не должен противоречить предыдущему документу;

Каждый термин, акроним или аббревиатура должны иметь одно значение в каждом документе;

Ссылка на каждый предмет или концепцию делается одним названием или описанием в каждом документе.

Выход процесса прослеживаемость является предметом официального управления конфигурацией.

Контекст всех документов регистрируется в форме соответствующей для манипуляции, обработки и хранения.

Документы, перечисленные в Таблице документальной перекрестной ссылки (см. ниже) разрабатываются до степени, требуемой уровнем полноты безопасности программного обеспечения.

В зависимости от размера, сложности и жизненного цикла разрабатываемого программного обеспечения, количество отдельных документов, требуемых к разработке будет разница. Некоторые документы могут быть объединены (при условии, что не будет потерь требуемых деталей в процессе). В случае крупных проектов может быть необходимо разделение перечисленной документации (по иерархичной ступени) на ряд

более легко управляемых меньшего размера документов. Документы, которые были разработаны независимыми группами или лицами не объединяются в единый документ.

7.2.1 Связь между различными документами, определенная в пункте 7, может также быть определена используя таблицу документальной перекрестной ссылки. По каждому документу, перечисленному в колонке «Документы», фаза и пункт, связанные с его созданием могут быть найдены читая таблицу горизонтально и вертикально от ячейки, содержащей символ «■». Фазы, в которых он используется можно найти, читая таблицы вертикально от ячеек, отмеченных символом «♦». Пункт или другая ссылка на определение документа могут быть найдены в колонке «Где определены». Там где пункт предоставлен, следующие пункты также требуют проверки, так как они могут содержать дальнейшую информацию. Следует отметить, что ссылка на план управления конфигурацией программного обеспечения дается в скобках, так как данный пункт просто дает ссылку на СТ РК ИСО 9001.

Таблица документальной перекрестной ссылки

Пункт	8	9	10	11	12	13	14	15	16	Құжат атауы	Где определено
Атауы	SRS	SA	SDD	SVe	S/H	SVal	Ass	Q	Ma		
ФАЗЫ(*) = параллельное										Документы	
ЖҮЙЕГЕ ЕНГІЗУ	♦			♦	♦					Спецификация требований систем	ENB 50129 B.2.3 annex
	♦	♦		♦	♦	♦	♦			Спецификация требований систем по безопасности	ENB 50129 B.2.4 annex
	♦				♦					Описание архитектуры системы	ENB 50129 B.2.1 annex
										План безопасности системы	ENB 50129 IEC 62278
SW ПЛАНИРОВАНИЕ (*)	♦	♦	♦	♦		♦	♦	▮		Sw План гарантии качества	15.4.3
						♦	♦	▮		Sw План управления кофигурацией	(15.4.2)
				▮		♦	♦			Sw План проверки подлинности	11.4.1
				▮		♦	♦			Sw План проверки интеграции	11.4.5
					▮	♦	♦			Sw/Hw План проверки интеграции	12.4.1
						▮	♦			Sw План подтверждения	13.4.3
							♦		▮	Sw План обслуживания	16.4.3
										План подготовки данных	17.4.2.1
										План проверки данных	17.4.2.4
SW ТРЕБОВАНИЯ (*)	▮	♦	♦	♦	♦	♦	♦			Sw Спецификация требований	8.4.1
										Спецификация требований программы	17.4.1.1
	▮			♦	♦	♦	♦			Sw Спецификации требований к испытаниям	8.4.13
				▮						Sw Отчет проверки подлинности	11.4.11
SW ПРОЕКТ		▮	♦	♦	♦	♦	♦			Sw Спецификации архитектуры	9.4.1
			▮	♦	♦	♦	♦			Sw Спецификация проекта	10.4.3
				▮						Sw Отчет проверки подлиности архитектуры	11.4.12
SW ПРОЕКТ МОДУЛЯ			▮	♦	♦	♦	♦			Sw Спецификация проекта модуля	10.4.3
			▮	♦	♦	♦	♦			Sw Спецификация испытания модуля	10.4.14
				▮						Sw Отчет проверки подлинности модуля	11.4.13
КОД			▮	♦	♦	♦	♦			Sw код источника	
				▮		♦	♦			Sw Отчет проверки подлинности кода источника	11.4.14
ИСПЫТАНИЕ МОДУЛЯ			▮	♦						Sw Отчет испытания модуля	10.4.14
SW ИНТЕГРАЦИЯ				▮						Sw Отчет испытания интеграции	11.4.15
										Отчет испытания данных	17.4.2.4
SW/HW ИНТЕГРАЦИЯ					▮					Sw/Hw Отчет испытания интеграции	12.4.8
ПОДТВӨЕ ПОДЛИНОСТИ						▮				SwОтчет подтверждения	13.4.10
							▮			Sw Отчет оценки	14.4.9
ОЦЕНКА (*)									▮	Sw Записи изменений	16.4.9
ОБСЛУЖИВАНИЕ									▮	Sw Записи обслуживания	16.4.8

8 Спецификация требований программного обеспечения

8.1 Задачи

Описать документ, который определяет полный набор требований для программного обеспечения, соответствуя всем требованиям системы до той степени, которая требуется уровнем полноты безопасности программного обеспечения. Он выступает в качестве комплексного документа для каждого программного инженера, таким образом ему не требуется изучать все документы в поисках требований.

Описать Спецификации требования к испытанию программного обеспечения.

8.2 Входящие документы

Спецификации требований системы

Спецификации требований безопасности системы

Описание архитектуры системы

План гарантии качества программного обеспечения

8.3 Выходящие документы

Спецификации требований системы

Спецификации требований к испытанию системы

8.4 Требования

8.4.1 Спецификации требований программного обеспечения выражают требуемые свойства разрабатываемого программного обеспечения, а не процедуры по их разработке. Все эти свойства, которые (за исключением безопасности) определены в [4], включают:

-функциональность (включая возможности и время реагирования);

-надежность и ремонтопригодность;

-безопасность (включая функции безопасности и связанные с ними уровни полноты безопасности программного обеспечения);

-эффективность;

-используемость;

-мобильность.

Уровень полноты безопасности программного обеспечения копируется из определения пункта 5 и регистрируется в спецификациях требований программного обеспечения.

8.4.2 До степени, требуемой уровнем полноты безопасности программного обеспечения, спецификации требований программного обеспечения выражаются и строятся следующим образом:

a) полные, четкие, точные, однозначные, подлежащие проверке, испытанию, обслуживанию и выполнимые;

b) прослеживаемые по отношению ко всем документам, упомянутым в пункте 8.2.

8.4.3 Спецификации требований программного обеспечения включают модели выражения и описания, понятные ответственному персоналу, вовлеченному во весь жизненный цикл системы.

8.4.4 Спецификации требований программного обеспечения определяют и фиксируют документально все интерфейсы с какими-либо другими системами, внутри или за пределами оборудования, находящегося под контролем, включая операторов, независимо от того существует ли прямая связь или только запланирована.

8.4.5 Все соответствующие режимы эксплуатации детально приводятся в спецификациях требований программного обеспечения.

8.4.6 Все соответствующие режимы поведения программируемой электроники, в частности, поведение при сбое детально описываются в спецификациях требований программного обеспечения.

8.4.7 Любые препятствия между аппаратным оборудованием и программным обеспечением определяются и документально регистрируются в спецификациях требований программного обеспечения.

8.4.8 Спецификации требований программного обеспечения приводят степень самопроверки программного обеспечения и заданный уровень проверки аппаратного оборудования программным обеспечением. Самопроверка программного обеспечения включает определение и отчетность, производимыми программным обеспечением в отношении своих сбоев и ошибок.

8.4.9 Спецификации требований программного обеспечения включают требования периодического испытания функций до степени, требуемой спецификациями требований безопасности системы.

8.4.10 Спецификации требований программного обеспечения включают требования позволяющие, всем функциям безопасности подлежать испытаниям во время эксплуатации всей системы до степени, требуемой спецификацией требований безопасности системы.

8.4.11 Когда программное обеспечение должно выполнять функции, особенно связанные с достижением требуемого уровня полноты безопасности системы, такое требование должно быть четко определено в спецификации требований программного обеспечения.

8.4.12 Когда программное обеспечение должно выполнять функции не связанные с безопасностью, это требование должно быть четко определено в спецификации требований программного обеспечения.

8.4.13 Спецификации требовании к испытанию программного обеспечения разрабатываются на основе спецификаций требований программного обеспечения. Данные спецификации относительно испытаний используются для проверки подлинности всех требований, описанных в спецификации требований программного обеспечения, а также как описание испытаний, которые должны быть произведены на завершенном программном обеспечении.

8.4.14 Спецификации требовании к испытаниям программного обеспечения определяют пункты испытания для каждой требуемой функции, наборы тестовых данных, включая:

a) требуемый входной сигнал с их последовательностью и значениями;

b) ожидаемый выходной сигнал с их последовательностью и значениями;

c) критерии приемлемости , включая рабочие и качественные аспекты.

8.4.15 Прослеживаемость требований является важным пунктом в подтверждении подлинности системы, связанной с безопасностью, для демонстрации этого в течение всех фаз жизненного цикла должны быть предоставлены средства.

8.4.16 Любой непрослеживаемый материал должен быть продемонстрирован как не несущий функций безопасности или интеграции системы.

9 Архитектура программного обеспечения

9.1 Задачи

9.1.1 Разработать архитектуру программного обеспечения, которая достигает требований спецификации требований программного обеспечения до степени, требуемой уровнем полноты безопасности программного обеспечения.

9.1.2 Рассмотреть требования, предъявленные к программному обеспечению архитектурой программного обеспечения.

9.1.3 Определить и оценить важность взаимодействия аппаратного

оборудования/программного обеспечения в вопросах безопасности.

9.1.4 Выбрать метод проектирования, если таковой еще не был ранее определен.

9.2 Входящие документы

1) Спецификации требований программного обебспечения;
2) Спецификации требований безопасности системы;
3) Описание архитектуры системы;
4) План гарантии качества программного обеспечения.

9.3 Выходящие документы

Спецификации архитектуры программного обеспечения.

9.4 Требования

9.4.1 Предложенная архитектура программного обеспечения устанавливается поставщиком программного обеспечения и /или разработчиком и подробно приводится в спецификациях архитектуры программного обеспечения.

9.4.2 Спецификации архитектуры программного обеспечения рассматривают возможность достижения спецификаций требовании программного обеспечения при требуемом уровне полноты безопасности программного обеспечения.

9.4.3 Спецификации архитектуры программного обеспечения определяют, оценивают и детально приводят важность взаимодействия аппаратного оборудования /программного обеспечения. Согласно требованиям [5], [6], предварительные исследования о взаимодействии между аппаратным оборудованием и программным обеспечением регистрируются в спецификациях требований безопасности системы.

9.4.4 Спецификации архитектуры программного обеспечения определяют все компоненты программного обеспечения и для этих компонентов определяют следующее:

a) являются ли эти компоненты новыми, существующими или запатентованными;

b) проводилось ли подтверждение подлинности этих компонентов ранее, если да, то каковы были условия подтверждения подлинности;

c) уровень полноты безопасности программного обеспечения компонента.

9.4.5 Программное обеспечение COTS подлежит следующим ограничениям:

a) использование программного обеспечения COTS при 0 уровне полноты безопасности программного обеспечения принимается без дальнейших мер предосторожности;

b) если программное обеспечение COTS должно быть использовано при 1 и 2 уровне полноты безопасности программного обеспечении, оно включается в процесс подтверждения подлинности программного обеспечения;

c) если программное обеспечение COTS должно быть использовано при 3 или 4 уровне полноты безопасности программного обеспечения, то будут предприняты следующие меры предосторожности:

d) Программное обеспечение COTS включается в испытание подтверждения подлинности;

e) Должен проводится анализ возможных сбоев программного обеспечения COTS;

f) Должна быть определена стратегия для определения сбоев программного обеспечения COTS и защиты системы от этих сбоев;

g) Стратегия защиты подлежит испытанию подтверждения подлинности;

h) Журналы регистрации ошибок существуют и оцениваются;

i) насколько возможно используются только простейшие функции программного обеспечения COTS.

9.4.6 Если ранее разработанное программное обеспечение должно использоваться как часть проекта, оно должно быть четко определено и документально зафиксировано. спецификации архитектуры программного обеспечения обосновывают пригодность

программного обеспечения в удовлетворении спецификаций требований программного обеспечения и уровня полноты безопасности программного обеспечения. Воздействие любых изменений программного обеспечения на оставшуюся часть системы должно быть тщательно рассмотрено для того, чтобы определить требуют ли они повторной инспекции и повторной оценки. Должно быть свидетельство того, что спецификации интерфейса с другими модулями, которые проходят повторную проверку и повторную оценку, выполняются.

9.4.7 По возможности существующие проверенные модули программного обеспечения, разработанные согласно данного стандарта должны использоваться в проекте.

9.4.8 Архитектура программного обеспечения должна минимизировать часть безопасности прикладной программы.

9.4.9 Когда программное обеспечение состоит из компонентов различных уровней полноты безопасности программного обеспечения, все компоненты программного обеспечения рассматриваются как принадлежащие высшему уровню полноты безопасности программного обеспечения, только если не имеется свидетельства независимости между компонентами более высокого уровня полноты безопасности программного обеспечения и компонентами более низкого уровня полноты безопасности программного обеспечения. Данное свидетельство регистрируется в спецификациях архитектуры программного обеспечения.

9.4.10 Спецификации архитектуры программного обеспечения определяют стратегию развития программного обеспечения до степени, требуемой уровнем полноты безопасности программного обеспечения. Спецификации архитектуры программного обеспечения выражаются и строятся таким образам, чтобы они были

a) полными, четкими, точными, однозначными, подлежащими проверке, испытанию, обслуживанию и выполнимыми;

b) прослеживаемыми по отношению к спецификациям требований программного обеспечения.

9.4.11 Спецификации архитектуры программного обеспечения обосновывают баланс между стратегиями предотвращения неисправностей и устранением неисправностей.

9.4.12 Спецификации архитектуры программного обеспечения обосновывают методы и меры, выбранные из набора, который удовлетворяет спецификации требовании программного обеспечения при требуемом уровне полноты безопасности программного обеспечения.

10 Проектирование и внедрение программного обеспечения

10.1 Задачи

10.1.1 Произвести проектирование и внедрение программного обеспечения определенного уровня полноты безопасности программного обеспечения из спецификации требовании программного обеспечения и спецификации архитектуры программного обеспечения.

10.1.2 Получить программное обеспечение, которое можно анализировать, испытывать, проверять и обслуживать. Испытание модуля также включено в эту фазу. Так как проверка подлинности и испытания будут критическим элементом в подтверждении подлинности, то особенно стоит рассмотреть проверку подлинности и необходимость в испытании через проектирование и внедрение для обеспечения контроле пригодности результирующей системы и ее программного обеспечения с начального

этапа.

10.1.3 Выбрать пригодный набор инструментов, включая языки и компиляторы, для требуемого уровня полноты безопасности программного обеспечения в течении всего жизненного цикла программного обеспечения, который помогает проверке подлинности, подтверждению подлинности, оценке и обслуживанию.

10.1.4 Произвести интеграцию программного обеспечения.

10.2 Входящие документы

1) Спецификации требовании программного обеспечения;
2) Спецификации архитектуры программного обеспечения;
3) План гарантии качества программного обеспечения.

10.3 Выходящие документы

1) Спецификации проектирования программного обеспечения;
2) Спецификации проекта модуля программного обеспечения;
3) Спецификации модуля испытаний программного обеспечения;
4) Код источника программного обеспечения и подтверждающая документация;
5) Отчет модуля испытания программного обеспечения.

10.4 Требования

10.4.1 Спецификации требований программного обеспечения и спецификации архитектуры программного обеспечения должны быть в наличии, хотя не обязательно в законченной форме до начала процесса проектирования.

10.4.2 Размер и сложность разработанного программного обеспечения удерживается на минимальном уровне.

10.4.3 Спецификации проектирования программного обеспечения описывает проект программного обеспечения на основе разбивки на модули, где каждый модуль имеет спецификации модуля проектирования программного обеспечения и спецификации модуля испытания программного обеспечения.

10.4.4 Спецификация проектирования программного обеспечения включает:

a) компоненты программного обеспечения, которые отслеживаются к архитектуре программного обеспечения и их уровню полноты безопасности;
b) интерфейсы компонентов программного обеспечения со средой;
c) интерфейсы между компонентами программного обеспечения;
d) структуру данных;
e) декомпозицию требований на компоненты;
f) основные алгоритмы и последовательность;
g) диаграммы.

10.4.5 Спецификации проекта модуля программного обеспечения рассматривают (одну спецификацию проекта модуля программного обеспечения на модуль):

a) определение всех низших компонентов программного обеспечения (называемых модулями в настоящих стандартах) отслеженных на верхний уровень;
b) их детальные интерфейсы со средой и другие модули с детальными вводами и выводами;
c) их уровень полноты безопасности;
d) детальные алгоритмы и структуры.

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

10.4.6 Каждый модуль программного обеспечения должен быть считываемым, понятным и контроле пригодным.

10.4.7 Подходящий набор инструментов, включая методы проектирования, языки и компиляторы выбираются для требуемого уровня полноты безопасности программного

обеспечения в течение всего жизненного цикла программного обеспечения.

10.4.8 Где применимо используются инструменты автоматического контроля и инструменты комплексного развития. Это приминает во внимание нужды лица, проверяющего подлинность и лица, подтверждающего подлинность.

10.4.9 До степени требуемой уровнем полноты безопасности программного обеспечения, язык программирования должен иметь транслятор/компилятор, имеющий одно из нижеследующего:

a) «Сертификат подтверждения подлинности» по отношению к признанному национальному/международному стандарту;

b) отчет об оценке, который подробно приводит доводы относительно его целевого использования;

c) процесс, основанный на контроле резервной подписи, который обеспечивает выявление ошибок трансляции.

10.4.10 Выбранный язык должен соответствовать соответствующим требованиям:

a) выбранный язык должен содержать детали, которые способствуют определению ошибок программирования;

b) выбранный язык должен поддерживать детали, которые соответствуют методу проектирования.

10.4.11 Когда пункт 10.4.10 не может быть удовлетворен, тогда обоснование любого альтернативного языка, подробно приводящее его пригодность к использованию по назначению регистрируется в спецификациях архитектуры программного обеспечения или Плане гарантии качества программного обеспечения.

10.4.12 Стандарты кодирования разрабатываются и используются для разработки всего программного обеспечения. На них дается ссылка в плане гарантии качества программного обеспечения (см. пункт 15.4.5).

10.4.13 Стандарты кодирования задают рациональную практику программирования, запрещают небезопасные языковые свойства и описывают процедуры документации кода источника. Каждый модуль программного обеспечения должен содержать информацию в коде источника, которая определена в нижеприведенном (неисчерпывающем) перечне:

- автор;
- история конфигурации;
- краткое описание.

Для данной информации рекомендуется использование стандартной формы. Она должна быть единой для всех модулей.

10.4.14 Испытания модуля программного обеспечения: каждый модуль должен иметь спецификацию испытания модуля программного обеспечения, согласно которого проводятся испытания модуля. Эти испытания показывают, что каждый модуль выполняет свою предназначенную функцию. Спецификация модуля испытания программного обеспечения определяет требуемый уровень тестового покрытия.

Должен быт выпущен отчет об испытании модуля программного обеспечения, включающий следующие детали:

a) справку о результатах испытания и информацию о том, соответствует ли каждый модуль требованиям спецификации модуля проектирования программного обеспечения;

b) справку о тестовом покрытии, которая предоставляется по каждому модулю и показывает, что все инструкции кода источника выполнялись хотя бы раз;

c) он должен быть в проверяемой форме;

d) набор данных испытаний и их результаты регистрируются в машиночитаемой

форме для последующего анализа; испытания должны быть воспроизводимыми и выполняться автоматическими средствами, если осуществимо.

Проверка того, верно ли модуль выполнил спецификацию своего испытания является деятельностью по проверке подлинности, см. пункт 11.

10.4.15 Согласно требуемого уровня полноты безопасности программного обеспечения, выбранный метод проектирования должен содержать свойства способствующие следующему:

a) абстракция, модульность и другие свойства, которые контролируют сложность;

b) четкое и точное выражение следующего:

- функциональность;
- информационный поток между компонентами;
- определение последовательности информации, связанной со временем;
- согласованность;
- структура и свойства данных;

c) человеческое восприятие;

d) проверка и подтверждение подлинности.

10.4.16 Выбранный метод проектирования должен иметь все свойства, способствующие обслуживанию программного обеспечения. Такие свойства включают модульность, укрытие и сокрытие информации.

10.4.17 Интеграция модулей программного обеспечения является процессом прогрессивного объединения индивидуальных и ранее опробованных модулей программного обеспечения в составное целое (или ряд составных подсистем) для того, чтобы интерфейсы модуля и собранное программное обеспечение могли быть должным образом обоснованы до интеграции и испытания системы.

10.4.18 В контексте данного стандарта и до уровня соответствующего заданному уровню полноты безопасности программного обеспечения, прослеживаемость в частности, рассматривает следующее:

a) прослеживаемость требований к проекту или другим объектам, выполняющим его;

b) прослеживаемость объектов проектирования к объектам реализации, которые подтверждают их примерами.

11 Проверка подлинности и испытание программного обеспечения

11.1 Задача

До степени требуемой уровнем полноты безопасности программного обеспечения провести испытание и оценку продуктов заданной фазы для обеспечения корректности и последовательности по отношению к продукции и стандартам, предоставленным как ввод в данную фазу.

11.2 Входящие документы

1) Спецификации требований системы;
2) Спецификации требований безопасности программного обеспечения;
3) Спецификации требований программного обеспечения;
4) Спецификации требований к испытаниям программного обеспечения;
5) Спецификации архитектуры программного обеспечения;
6) Спецификации проектирования программного обеспечения;
7) Спецификации проекта модуля программного обеспечения;
8) Спецификации модуля испытаний программного обеспечения;
9) Код источника программного обеспечения и подтверждающая документация;

10) План гарантии качества программного обеспечения;

11) Отчет испытания модуля программного обеспечения.

11.3 Выходящие документы

1) План проверки подлинности программного обеспечения;

2) Отчет о требованиях проверки подлинности программного обеспечения;

3) Отчет проверки подлинности архитектуры и проекта программного обеспечения;

4) Отчет по модулю проверки подлинности программного обеспечения;

5) Отчет по проверке подлинности кода источника программного обеспечения;

6) План испытания интеграции программного обеспечения;

7) Отчет испытания интеграции программного обеспечения.

11.4 Требования

11.4.1 План проверки подлинности программного обеспечения разрабатывается для того, чтобы деятельность по проверке подлинности могла быть должным образом направлена и определенный проект или другие нужды проверки подлинности могли быть соответственно предоставлены. Во время разработки (и в зависимости от размера системы) план может быть подразделен на ряд более мелких документов и естественным образом добавляться по мере того как потребности проверки подлинности становятся более ясными.

11.4.2 План проверки подлинности программного обеспечения документирует все критерии, методы и инструменты, подлежащие использованию в процессе проверки подлинности по данной фазе.

11.4.3 План проверки подлинности программного обеспечения описывает деятельность к выполнению для гарантии корректности и последовательности по отношению к продуктам и стандартам, предоставленным в качестве ввода в данную фазу.

11.4.4 План проверки подлинности программного обеспечения рассматривает следующее:

a) выбор стратегии и методов проверки подлинности. Во избежание чрезмерной сложности при оценке деятельности по проверке подлинности и испытаниям, предпочтение должно быт отдано выбору набора тестовых данных и методов и т.д., которые сами по себе легко анализируемы;

b) выбор и использование контрольной аппаратуры программного обеспечения;

c) выбор и документация деятельности по проверке подлинности;

d) оценка полученных результатов проверки подлинности;

e) оценка требований надежности;

f) роли и ответственность вовлеченных в процесс испытания;

g) степень тестового покрытия, требуемая к достижению.

11.4.5 План испытания интеграции программного обеспечения документирует следующее:

a) набор тестовых данных и данные испытаний;

b) типы испытаний, подлежащих выполнению;

c) режимы испытаний, инструменты, конфигурация и программы;

d) критерии испытаний, на основе которых будет обосновываться завершение испытания.

11.4.6 Каждая фаза развития требует демонстрации того, что требования к функциональности, надежности, рабочим показателям и безопасности выполняются.

11.4.7 Проверка подлинности проводится независимой стороной до степени требуемой уровнем полноты безопасности программного обеспечения, как определено на Рисунке 5.

11.4.8 Испытания, которые не были в полной мере документированы и были выполнены постановщиком до проверки подлинности не рассматриваются как часть проверки подлинности.

11.4.9 Результаты каждой проверки подлинности сохраняются в форме определенной или упомянутой в Плане проверки подлинности программного обеспечения, таким образом, чтобы данная информация могла быть проверяемой.

11.4.10 После каждой проверки подлинности выпускается отчет о проверке подлинности, констатирующий то, что программное обеспечение прошло проверку подлинности или приводящий причины не прохождения данной проверки. Отчеты о проверке подлинности рассматривают следующее:

a) пункты, которые не соответствуют Спецификации требований программного обеспечения, Спецификации проектирования программного обеспечения или Спецификации модуля проектирования программного обеспечения;

b) пункты, которые не соответствуют Плану гарантии качества программного обеспечения;

c) модули, данные, структуры и алгоритмы плохо адаптированные к проблеме;

d) выявленные ошибок или недостатков;

e) подлинность и конфигурация проверенных пунктов.

11.4.11 Проверка подлинности требований программного обеспечения: после учреждения спецификаций требований программного обеспечения проверка подлинности рассматривает следующее:

a) соответствие спецификации требований программного обеспечения в выполнении требований, заданных в спецификациях требований системы, спецификациях требований безопасности системы и плане гарантии качества программного обеспечения;

b) соответствие спецификации требований испытания программного обеспечения как испытания спецификаций требований программного обеспечения;

c) внутренняя последовательность спецификаций требований программного обеспечения. Результаты регистрируются в отчете проверки подлинности требований программного обеспечения.

11.4.12 Проверка подлинности архитектуры и проекта программного обеспечения: после учреждения спецификаций архитектуры программного обеспечения и спецификаций проекта программного обеспечения, в рамках проверки подлинности требуется рассмотреть следующее:

a) соответствие спецификации архитектуры программного обеспечения и спецификации проекта программного обеспечения при выполнении спецификации требований программного обеспечения;

b) соответствие спецификации проектирования программного обеспечения спецификациям требований программного обеспечения в отношении последовательности и завершенности;

c) соответствие плана испытания интеграции программного обеспечения в качестве набора тестовых данных для спецификации архитектуры программного обеспечения и спецификации проектирования программного обеспечения;

d) внутренняя последовательность спецификаций архитектуры программного обеспечения и проектирования программного обеспечения. Результаты регистрируются в отчете проверки подлинности архитектуры и проектирования программного обеспечения.

11.4.13 Проверка подлинности модуля программного обеспечения: после учреждения каждой спецификации модуля проектирования программного обеспечения, проверка подлинности рассматривает следующее:

a) соответствие спецификации модуля проектирования программного обеспечения

в выполнении спецификации проектирования программного обеспечения;

b) соответствие спецификации модуля испытания программного обеспечения в качестве набора тестовых данных для спецификации модуля проектирования программного обеспечения;

c) декомпозиция спецификации проектирования программного обеспечения на модули программного обеспечения и спецификации модуля проектирования программного обеспечения со ссылкой на следующие пункты:

- Выполнимость требуемых работ;
- Контролепригодность для дальнейшей проверки подлинности;
- Считываемость для группы разработки и проверки подлинности;
- Ремонтопригодность для дальнейшего развития;

d) соответствие отчетов модуля испытаний программного обеспечения как запись испытаний, проведенных согласно спецификации модуля испытания программного обеспечения.

Результаты регистрируются в отчете проверки подлинности модуля программного обеспечения.

11.4.14 Проверка подлинности кода источника программного обеспечения: до степени, требуемой уровнем полноты безопасности программного обеспечения, код источника программного обеспечения проверяется для гарантии соответствия спецификации модуля проектирования программного обеспечения и плана гарантии качества программного обеспечения. Данная деятельность включает проверку для выявления того корректно ли были применены стандарты кодирования.

Результаты регистрируются в отчете проверки подлинности кода источника программного обеспечения.

11.4.15 Отчет испытания интеграции программного обеспечения выпускается следующим образом:

a) Отчет испытания интеграции программного обеспечения выпускается с указанием результата испытания и констатацией того были ли выполнены задачи и критерии плана испытания интеграции программного обеспечения. В случае не выполнения, регистрируются причины;

b) Отчет испытания интеграции программного обеспечения должен быть в проверяемой форме;

c) набор тестовых данных и их результаты регистрируются, желательно в машиночитаемой форме для последующего анализа;

d) испытания должны быть воспроизводимыми и выполняться автоматическими средствами, если возможно;

e) подлинность и конфигурация пунктов проверена.

11.4.16 По вопросу интеграции программного обеспечения/аппаратного обеспечения, см. 12.4.8.

12 Интеграция программного обеспечения/аппаратного оборудования

12.1 Задачи

12.1.1 Продемонстрировать, что программное обеспечение и аппаратное оборудование взаимодействуют корректно для выполнения их функций.

12.1.2 Объединить программное обеспечение и аппаратное оборудование, обеспечивая их совместимость, выполнить спецификацию требований безопасности системы и требований заданного уровня полноты безопасности программного обеспечения.

12.2 Входящие документы

1) Спецификация требований системы;
2) Спецификация требований безопасности системы;
3) Описание архитектуры системы;
4) Спецификации требований программного обеспечения;
5) Спецификации требований испытания программного обеспечения;
6) Спецификации архитектуры программного обеспечения;
7) Спецификации проектирования программного обеспечения;
8) Спецификации модуля проектирования программного обеспечения;
9) Спецификации модуля испытания программного обеспечения;
10) Код источника программного обеспечения и подтверждающая документация;
11) Документация аппаратного оборудования.

12.3 Выходящие документы

1) План испытания интеграции программного обеспечения/аппаратного оборудования

2) Отчет испытания интеграции программного обеспечения/аппаратного оборудования

12.4 Требования

12.4.1 План испытания интеграции программного обеспечения/аппаратного оборудования в отношении уровней полноты безопасности программного обеспечения выше нуля создается на ранней стадии развития жизненного цикла для того, чтобы деятельность по интеграции могла быть соответственно направлена, определенный проект и другие потребности интеграции могли быть надлежаще обеспечены. Во время разработки (и в зависимости от размера системы) план может быть подразделен на ряд меньших документов и естественно добавляться по мере развития проектов аппаратного оборудования и программного обеспечения, а также уточнения детальных потребностей интеграции.

12.4.2 План испытания интеграции программного обеспечения/аппаратного оборудования документирует следующее:

a) наборы тестовых данные и данные испытания;

b) типы испытаний, подлежащих исполнению;

c) режимы испытаний, включая инструменты, описание вспомогательного программного обеспечения и конфигурации;

d) критерии испытаний, на основе которых будет оцениваться завершение испытания.

12.4.3 План испытания интеграции программного обеспечения/аппаратного оборудования разделяет деятельность, которая может быть выполнена разработчиком на его территории и деятельность, требующая доступа на сайт пользователя.

12.4.4 План испытания интеграции программного обеспечения/аппаратного оборудования разделяет между следующей деятельностью:

a) объединение программного обеспечения в целевое аппаратное оборудование; ii) интеграция системы;

12.4.5 Инструменты и оборудование, определенные в плане испытания интеграции программного обеспечения/аппаратного оборудования должны быть доступны в самые ранние практически осуществимые сроки.

12.4.6 Во время интеграции программного обеспечения/аппаратного оборудования любая модификация или изменение интегрированной системы подлежит изучению воздействия, которое определит все модули, на которые было оказано влияние и необходимую деятельность по повторной проверке подлинности.

12.4.7 Наборы тестовых данных и их результаты регистрируются, желательно в машиночитаемой форме для последующего анализа.

12.4.8 Отчет о испытании интеграции программного обеспечения/аппаратного оборудования должен быть выпущен согласно следующим пунктам:

a) отчет об испытании интеграции программного обеспечения/аппаратного оборудования должен привести результаты исследования и констатировать были ли выполнены задачи и критерии плана испытания интеграции программного обеспечения/аппаратного оборудования. Если нет, то это должно быть зарегистрировано;

b) отчет об испытании интеграции программного обеспечения/аппаратного оборудования должен быт выполнен в форме подлежащей проверке;

c) набор тестовых данных и их результаты регистрируются, желательно в машиночитаемой форме для последующего анализа;

d) отчет об испытании интеграции программного обеспечения/аппаратного оборудования также должен содержать свидетельство того, что лицо, проверяющее подлинность удовлетворено соответствием отчета об испытании интеграции программного обеспечения/аппаратного оборудования в качестве записи испытаний, проведенных согласно плана испытания интеграции программного обеспечения /аппаратного оборудования;

e) Отчет об испытании интеграции программного обеспечения/аппаратного оборудования должен документировать подлинность и конфигурацию всех вовлеченных пунктов.

13 Подтверждение подлинности программного обеспечения

13.1 Задача

Провести анализ и испытать соответствие спецификации требовании программного обеспечения с особым акцентом на аспектах функциональности и безопасности согласно уровню полноты безопасности программного обеспечения.

13.2 Входящие документы

1) Спецификация требований программного обеспечения;

2) Вся документация по аппаратному оборудованию и программному обеспечению;

3) Спецификация требований безопасности системы.

13.3 Выходящие документы

1) План подтверждения подлинности программного обеспечения;

2) Отчет о подтверждении подлинности программного обеспечения.

13.4 Требования

13.4.1 Анализ и испытания являются основной деятельностью подтверждения подлинности.

13.4.2 Имитация и моделирование могут быть использованы для дополнения процесса подтверждения подлинности.

13.4.3 План подтверждения подлинности программного обеспечения разрабатывается и детализируется в подходящей документации.

13.4.4 План подтверждения подлинности программного обеспечения разрабатывается, выполняется, а его результаты оцениваются независимой стороной до степени, требуемой уровнем полноты безопасности программного обеспечения.

13.4.5 Если требуется уровнем полноты безопасности программного обеспечения, объем и содержание плана подтверждения подлинности программного обеспечения согласовываются с оценщиком. Данное соглашение также предоставляет справку о

присутствии оценщика во время испытаний.

13.4.6 План подтверждения подлинности программного обеспечения включает краткий обзор обосновывающий выбор стратегии подтверждения подлинности. Обоснование включает рассмотрение, согласно требуемому уровню полноты безопасности программного обеспечения, следующих пунктов:

a) руководства или автоматических методов или и того и другого;

b) статических или динамических методов или и того и другого;

c) аналитических или статистических методов или и того и другого.

13.4.7 План подтверждения подлинности программного обеспечения определяет шаги, необходимые для демонстрации соответствия

- Спецификации требований программного обеспечения;
- Спецификации архитектуры программного обеспечения;
- Спецификации проектирования программного обеспечения;
- Спецификации модуля проектирования программного обеспечения,

в реализации требований безопасности, заданных в спецификациях требований безопасности системы. Лицо, подтверждающее подлинность проверяет завершен ли процесс проверки подлинности.

13.4.8 Измерительное оборудование, использованное для подтверждения подлинности подлежит поверке или калибровке.

13.4.9 Программное обеспечение тестируется посредством моделирования входных сигналов присутствующих при обычной эксплуатации, ожидаемых нарушений и нежелательных условий, требующих действия системы.

13.4.10 Результаты подтверждения подлинности документируются в отчете о подтверждении подлинности программного обеспечения в форме, подлежащей проверке.

13.4.11 После окончания интеграции аппаратного оборудования/программного обеспечения, выпускается отчет подтверждения подлинности программного обеспечения, соответствующий следующим пунктам:

a) он констатирует выполнены ли задачи и критерии плана подтверждения подлинности программного обеспечения. Если нет, то это регистрируется;

b) он также приводит результаты исследования и то выполняет ли все программное обеспечение на своем целевом оборудовании требования, заданные в спецификации требований программного обеспечения;

c) предоставляет оценку тестового покрытия требований спецификации требовании программного обеспечения;

d) Отчет подтверждения подлинности программного обеспечения должен быть выполнен в форме, подлежащей проверке;

e) набор тестовых данных и их результатов регистрируется в машиночитаемой форме для последующего анализа;

f) испытания должны быть воспроизводимыми и выполняться автоматическими средствами, если осуществимо.

13.4.12 Отчет подтверждения подлинности программного обеспечения документирует подлинность и конфигурацию всех следующих пунктов:

a) использованного аппаратного оборудования;

b) использованного оборудования;

c) поверке и калибровке оборудования;

d) использованные имитационные модели;

e) обнаруженные несоответствия;

f) выполненные коррективные меры.

13.4.13 Любые обнаруженные несоответствия, включая любые найденные

ошибки должны быть четко определены в отдельном разделе отчета о подтверждении подлинности программного обеспечения и включены в любую сопроводительную записку доставленного программного обеспечения

13.4.14 Программное обеспечение проверяется согласно спецификации испытания программного обеспечения. Эти испытания должны показать, что все требования спецификации требований программного обеспечения выполнены корректно.

Результаты регистрируются в отчете подтверждения подлинности программного обеспечения.

14 Оценка программного обеспечения

14.1 Задача

Сделать выводы на основе оценки процессов жизненного цикла и полученных продуктов, о том, что программное обеспечение соответствует заданному уровню полноты безопасности программного обеспечения и соответствует предназначенной программе.

14.2 Входящие документы

1) Спецификация требований безопасности системы;

2) Вся документация по аппаратному оборудованию и программному обеспечению.

14.3 Выходящие документы

Отчет об оценке программного обеспечения.

14.4 Требования

14.4.1 Программное обеспечения 0 уровня безопасности:

a) Оценщик должен подтвердить, что это соответствующий уровень безопасности программного обеспечения;

b) также возможно согласовать данный уровень между поставщиком и пользователем во время конкурса.

14.4.2 Программное обеспечение с отчетом об оценке программного обеспечения другого оценщика не должно быт объектом полностью новой оценки. Второй оценщик проверяет программное обеспечение на соответствие должному уровню безопасности пригодности применения на предназначенном целевом компьютере.

14.4.3 Оценщик имеет доступ к процессу проектирования и разработки и всей документации, связанной с проектом.

14.4.4 Оценка программного обеспечения проводится Оценщиком, который независим от группы проектирования.

14.4.5 Оценщик оценивает программное обеспечение системы на соответствие предназначенной цели и корректное реагирование на вопросы безопасности спецификации требований безопасности системы.

14.4.6 До степени, требуемой уровнем безопасности программного обеспечения, оценщик определяет были ли выбраны соответствующие методы и применены в каждой фазе жизненного цикла программного обеспечения.

14.4.7 Если требуется уровнем полноты безопасности программного обеспечения, оценщик согласовывает объем и содержание плана подтверждения подлинности программного обеспечения.

14.4.8 Оценщик может затребовать выполнение дополнительной работы по проверке и подтверждению подлинности по его усмотрению.

14.4.9 Оценщик должен составить отчет по каждому обзору, в которых детализируются результаты оценки.

14.4.10 Если по мнению оценщика программное обеспечение соответствует своему предназначению, то в отчете оценки программного обеспечения должно содержаться заявление относительно уровня безопасности полученного программным обеспечением.

14.4.11 При соответствии программного обеспечения своему предназначению, либо не получении ПО требуемого уровня безопасности, оценщик лишь сообщает о несоответствиях в отчете оценки программного обеспечения и не дает каких-либо технических решений.

15 Гарантия качества программного обеспечения

15.1 Задачи

15.1.1 Определение, мониторинг и контроль всех работ, как технических так и организационных, необходимых для обеспечения требуемого качества программного обеспечения. Это необходимо для того, чтобы обеспечить качественную защиту от систематических неисправностей, а также для возможности создания контрольного анализа с целью эффективного выполнения работ по проверке подлинности.

15.1.2 Предоставление доказательства завершения вышеупомянутых работ.

15.2 Исходные документы

Все документы каждого этапа жизненного цикла

15.3 Выходные документы

1) План гарантии качества программного обеспечения
2) План управления конфигурацией программного обеспечения

Все вышеуказанные планы должны быть представлены в начале проекта и обновляться в течение всего жизненного цикла.

15.4 Требования

15.4.1 Поставщик и/или разработчик должен иметь и применять, систему гарантии качества соответствующую СТР РК ИСО 9001 в целях выполнения требований настоящего стандарта. Особо рекомендуется аккредитация по СТ РК ИСО/МЭК 17025.

15.4.2 Поставщик и/или разработчик и заказчик должны внедрить на предприятии СМК по СТ РК ИСО 9001.

15.4.3 Поставщик и/или разработчик должны составить и документировать, по каждому проекту, план гарантии качества программного обеспечения для выполнения требований 15.4.1 и 15.4.2 настоящего стандарта, которые должны выражаться в значениях измерения где это возможно.

15.4.4 В плане гарантии качества программного обеспечения должен быть пункт, в котором подробно излагается процесс обновления на протяжении всего проекта: частота, ответственность, метод.

15.4.5 Все работы, мероприятия, документы и др., необходимые в соответствии с СТ РК ИСО 9001 и настоящим стандартом должны быть описаны или снабжаться ссылками в плане гарантии качества программного обеспечения и привязаны к конкретной разработке.

За исключением нулевого уровня безопасности программного обеспечения, в плане гарантии качества программного обеспечения должны быть указаны или снабжены ссылками нижеследующие пункты.

Это требуется для обеспечения учета всех аспектов безопасности в программном обеспечении в отношении уровня безопасности программного обеспечения.

Настоящий перечень не является исчерпывающим:

а) определение модели для всего жизненного цикла

- определение каждого этапа, в том числе:
- работы и элементарные задачи;
- критерии вводы и вывода;
- входные и выходные данные каждого этапа;
- основные работы по обеспечению качества на каждом этапе;
- организационная единица, отвечающая за отдельную работу и элементарную задачу;

b) прослеживаемость требований;

c) прослеживаемость структуры документации;

d) документация, связанная с разработкой, проверкой и подтверждением подлинности, эксплуатацией и обслуживанием программного обеспечения;

e) процедуры интеграции системы;

f) применяемые стандарты кодирования;

g) оценка предыдущих подтверждений;

h) определение системы показателей (количественные показатели) как по продукции так и по процессу. Для системы показателей программного продукта необходимо делать ссылку на качественные характеристики и инструкции по проведению оценки в соответствии с серией [4].

15.4.6 Управление конфигурацией должно осуществляться в соответствии с инструкциями СТ РК ИСО 9001.

Каждый программный документ должен находиться под конфигурационным контролем прежде чем будет издана его первая одобренная редакция. Исходный код программного обеспечения должен находиться под конфигурационным контролем до начала документируемого тестирования модуля.

Нельзя вносить недозволенные изменения в какой-либо пункт при контроле управления конфигурацией. Необходимо принять меры предосторожности в целях предотвращения и обнаружения ошибок, возникших в машиночитаемом коде во время хранения, перевода, передачи или дублирования.

Управление конфигурацией не должно быть ограничено лишь строгой разработкой и обслуживанием продукта, оно также должно учитывать среду, применяемую в течении всего жизненного цикла. Такое расширение, необходимое для воспроизводимости разработки и обслуживания, должно включать файлы конфигурации компьютера, компонующие программы, программы сборки, отладочные программы и все другие применимые инструменты.

15.4.7 Адекватность и результаты плана проверки подлинности программного обеспечения должны быть изучены.

15.4.8 Поставщик и/или разработчик должен создать, задокументировать и выполнять процедуры Контроля внешних поставщиков, в том числе:

- методы, обеспечивающие соответствие предоставляемого внешними поставщиками программного обеспечения установленным требованиям. Предварительно разработанное программное обеспечение должно соответствовать требованию уровню безопасности программного обеспечения и надежности. Новое программное обеспечение должно быть разработано и обслуживаться в соответствии с планом гарантии качества программного обеспечения поставщика, либо с конкретным планом гарантии качества программного обеспечения, составленным внешним поставщиком согласно плану гарантии качества ПО поставщика;
- методы, обеспечивающие адекватность и полноту требований предлагаемых внешнему поставщику программного обеспечения.

15.4.9 Поставщик и/или разработчик должен создать, задокументировать и

выполнять процедуры предоставления отчетности по проблемным аспектам и принятия корректирующих мер. Такие процедуры, в рамках системы обеспечения качества, должны внедрять соответствующие части СТ РК ИСО 9001, в частности учитывающие следующие аспекты:

определение документации необходимой для предоставления отчетности по проблемным аспектам и/или принятия корректирующих мер, с целью предоставления отзыва ответственным лицам;

определение анализа информации, собранной в отчетах по проблемам с целью выявления причин;

определение методики, необходимой для соблюдения при предоставлении отчетности, прослеживании и решении проблем, выявленных как на этапе разработки, так и во время обслуживания программного обеспечения;

определение превентивных действий для решения проблем на уровне соответствующем требуемому уровню безопасности программного обеспечения;

определение конкретных организационных обязанностей в отношении разработки и обслуживания программного обеспечения;

определение путей применения средств контроля для обеспечения принятия корректирующих мер и их эффективности;

определение применяемых форм;

определение требований для повторного проведения проверки, проверки и подтверждения подлинности и оценки.

Организация процесса отчетности по проблемным аспектам и корректирующим мерам должна применяться в течении жизненного цикла программного обеспечения, начиная непосредственно после интеграции программного обеспечения и до начала процесса официального утверждения программного обеспечения, охватывая весь этап обслуживания ПО.

16 Обслуживание программного обеспечения

16.1 Задачи

Обеспечение необходимой работы программного обеспечения, с сохранением уровеня безопасности программного обеспечения и надежности при принятии корректирующих мер, расширении функциональных возможностей или адаптации самого программного обеспечения.

16.2 Исходные документы

Все документы

16.3 Выходные документы

1) План обслуживания программного обеспечения
2) Учет изменений ПО
3) Учет обслуживания ПО

16.4 Требования

16.4.1 Обслуживание должно осуществляться в соответствии с инструкциями СТ РК ИСО 9000.

Помимо этого, необходимо выполнять следующие требования по обслуживанию программного обеспечения.

16.4.2 Приспособленность к обслуживанию должна быть заложена в системе программного обеспечения, в частности, за счет соблюдения требований пункта 10 настоящего стандарта [4] также должна использоваться для установления требований и осуществления проверки минимального уровня приспособленности к обслуживанию.

16.4.3 Процедуры по облуживанию программного обеспечения должны быть установлены и указаны в плане обслуживания программного обеспечения. Эти процедуры также включают:

a) контроль отчета ошибок, журналы регистрации ошибок, учет обслуживания, разрешение на внесение изменений и конфигурация программного обеспечения/системы;

b) проверка и подтверждение подлинности, оценка; а также

c) определение уполномоченного органа, который утверждает измененное программное обеспечение.

16.4.4 Работы по обслуживанию должны подвергаться проверке согласно плану обслуживания, через определенные промежутки времени, указанные в плане гарантии качества плану обслуживания.

16.4.5 Обслуживание должно осуществляться на том же уровне компетенции, средств, документации, планирования и организации, как и при первоначальной разработке системы. Это распространяется также на управление конфигурацией, контроль изменений, контроль документооборота 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 безопасности программного обеспечения, подрядные организации, прежде чем приступить к работе по внесению изменений, должны принять решение о том, рассматривать ли работы по обслуживанию как значительные или незначительные, или же имеющиеся методы обслуживания адекватны для системы. Для уровней 0, 1 и 2 безопасности программного обеспечения, поставщику необходимо принять аналогичное решение.

16.4.7 Организация контроля внешнего поставщика, отчетности по проблемным аспектам и корректирующих мер должна осуществляться по аналогичным критериям, указанным в соответствующих абзацах пункта о гарантии качества программного обеспечения.

16.4.8 Для каждого изделия программного обеспечения до его первого выпуска должна быть создана учетная ведомость обслуживания программного обеспечения с последующим ее ведением. Помимо требований СТ РК ИСО 9000 для "Отчетов и учета обслуживания", такая учетная ведомость должна включать:

a) ссылку на все записи учета изменений программного обеспечения для данного изделия программного обеспечения;

b) информацию о последствиях внесения изменений;

c) тестовые данные для компонентов, включая повторное утверждение и данные регрессивного тестирования; а также

d) историю конфигурации программного обеспечения.

16.4.9 Учетная ведомость изменений программного обеспечения должна быть создана для каждой работы по обслуживанию. Такая ведомость должна включать:

a) заявку на внесение изменения или видоизменения;

b) анализ воздействия работ по обслуживанию на систему в целом, в том числе аппаратное оборудование, программное обеспечение, взаимодействие с пользователем, среда и возможное взаимодействие;

c) подробное описание видоизменения или изменения; а также

d) повторное подтверждение подлинности, регрессивное тестирование и повторную оценку видоизменения или изменения насколько это требует уровень безопасности программного обеспечения. Ответственность за повторное подтверждение может варьироваться по проектам, согласно уровню безопасности программного

обеспечения. Помимо этого, влияние видоизменения или изменения на процесс повторного утверждения может быть ограничено различными уровнями систем (только измененные модули, все идентифицированные затронутые модули, полная система). Поэтому, план подтверждения подлинности программного оборудования должен рассматривать обе проблемы согласно уровню безопасности программного обеспечения. Степень независимости повторного утверждения должна быть такой же что и при предыдущем утверждении.

17 Системы сконфигурированные данными прикладной программы

17.1 Задачи

17.1.1 Характерной чертой системы управления и системы защиты на железнодорожном транспорте является необходимость проектирования каждой установки с соблюдением индивидуальных требований для конкретной прикладной программы. Система сконфигурированная данными прикладной программы позволяет использовать комплексное программное обеспечение одобренное как тип, при этом, индивидуальные требования для каждой установки определяются как данные (данные конкретной прикладной программы). Эти данные обычно представлены в виде таблиц или на языке конкретной прикладной программы, которые интерпретируются комплексным программным обеспечением.

17.1.2 Для системы имеющей опасное критическое состояние, высокий уровень ресурсов, требуемый для разработки программного обеспечения чтобы достичь необходимый уровень безопасности системы, делает принятие системы, сконфигурированной данными прикладной программы, весьма привлекательным, поскольку это позволяет использовать комплексное программное обеспечение повторно. Тем не менее, поскольку безопасная работа системы, по всей вероятности, будет зависеть от правильности данных, то процедуры, применяемые для выработки данных, должны быть соответствующего уровня безопасности системы.

17.1.3 Нижеследующие подклассы характеризуют требования настоящего стандарта для первоначальной разработки комплексного программного обеспечения для системы, сконфигурированной данными прикладной программы, а также для последующей разработки каждого комплекта данных по конкретной установке.

17.2 Исходные документы

1) Спецификация требований к программному обеспечению
2) Спецификация архитектуры программного обеспечения

17.3 Выходные документы

1) Спецификация требований прикладной программы
2) План подготовки данных
3) План проверки данных
4) Отчет проверки данных

17.4 Требования

17.4.1 Жизненный цикл подготовки данных

На рисунке 6 показан жизненный цикл для прикладной программы, с использованием аппаратного оборудования и комплексного программного обеспечения, вместе с данными по конкретной прикладной программе. Ниже следуют этапы жизненного цикла.

17.4.1.1 Спецификация требований к прикладной программе

Должны быть определены требования к прикладной программе. Они также должны включать требования, имеющие непосредственное отношение к конкретной

установке (например, схема расположения путей, сигнальная установка, ограничения скорости), а также стандарты, которым прокладная программа должны соответствовать (например, принципы сигнализации, уровни безопасности системы).

17.4.1.2 Проектирование общей установки

Должна быть определена архитектура системы, а также указано количество и тип применяемых базовых компонентов. Такие компоненты, содержащие программное обеспечение, должны быть разработаны в соответствии с настоящим стандартом. Функции, указанные в требованиях, должны быть выделены для компонентов, а также определено физическое нахождение каждого компонента.

17.4.1.3 Подготовка данных

Процесс подготовки данных должен включать выработку конкретной информации (например, управляющие таблицы), выработку исходного кода данных и его компиляцию, проверку и другие поверки, а также проверку данных прикладной программы.

17.4.1.4 Интеграция и приемка

Для некоторых систем данные прикладной программы будут интегрированы с комплексным аппаратным оборудованием и программным обеспечением для прохождения заводского испытания, прежде чем они будут установлены на месте работы. Это, возможно, не понадобиться там, где уверенность может быть получена в достаточной степени другими средствами. Затем оборудование устанавливается на месте работ, и на новом оборудовании выполняются комплексные испытания. В конечном итоге, система должна быть введена в эксплуатацию как полностью действующая система, и процесс окончательной приемки осуществляется на комплектном оборудовании.

17.4.1.5 Утверждение и оценка

Работы по утверждению и оценке направлены на проверку функционирования каждого этапа жизненного цикла.

17.4.2 Процедуры и инструменты подготовки данных

Для каждого нового типа системы, сконфигурированной данными прикладной программы, должны быть разработаны процедуры и инструменты подготовки данных чтобы жизненный цикл подготовки данных, указанный в 17.4.1 мог быть применен для установки новой системы. Разработка таких процедур и инструментов должна выполняться в соответствии с настоящим стандартом параллельно с комплексным программным обеспечением и аппаратным оборудованием для системы. Цель работ по проверке и подтверждению подлинности, а также оценке – обеспечение совместимости инструментов подготовки данных и комплексного программного обеспечения.

17.4.2.1 На этапе проектирования программного обеспечения для системы, сконфигурированной данными прикладной программы, 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 процедуры и инструменты подготовки данных, которые применяются в соответствии с уровнями пригодности и безопасности системы. План указывает требования к независимости между персоналом выполняющим верификацию, утверждение и задания по разработке.

17.4.2.2 План подготовки данных должен выделять уровень безопасности для любых инструментов аппаратного оборудования и программного обеспечения, используемых в течение жизненного цикла подготовки данных. Такой уровень безопасности берется из требуемого уровня безопасности системы и степени до которой выходной эффект каждого инструмента проверяется за счет других ручных и

автоматизированных процедур.

17.4.2.3 Где это возможно, план подготовки данных должен предусматривать обозначения по описанию требований и разработки, которые знакомы инженерам по прикладным программам, например, стандартные планы передачи сигналов и управляющие таблицы. При введении новых обозначений, должна быть предоставлена необходимая документация пользователя, а также обучение по необходимости.

17.4.2.4 Отчеты по проверке и подтверждению подлинности и оценке, необходимые для того, чтобы показать, 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. Эта информация должна содержаться в плане проверке данных, а результаты отмечены в отчете проверки данных.

17.4.2.5 Все данные и соответствующая документация должны соответствовать требованиям по управлению конфигурацией пункта 15 настоящего стандарта. Учетные записи по управлению конфигурацией должны указать версии комплексного программного обеспечения с которой были разработаны данные, а также версии инструментов примененных в процесс подготовки данных.

17.4.3 Разработка программного обеспечения

Разработка комплексного программного обеспечения для использования в системе, сконфигурированной данными прикладной программы, должна соответствовать требованиям пунктов 1 по 16 настоящего стандарта.

17.4.3.1 На этапе спецификации требований к программному обеспечению, должны быть определены те функции, которые используют данные прикладной программы в каждой системе и подсистеме. Уровень безопасности системы выделенный для каждой подсистемы будет определять стандарты для применения к последующей разработке данных для всех установок системы.

17.4.3.2 На этапе проектирования программного обеспечения должны быть определены детальные интерфейсы между комплексным ПО и данными прикладной программы, если это не было определено на предшествующих этапах жизненного цикла, использовать существующий язык конкретного приложения.

17.4.3.3 На этапе проектирования модуля программного обеспечения должно быть обеспечено строгое разделение между кодом и данными программы, то есть должна существовать возможность повторно компилировать и обновлять либо комплексное программное обеспечение, либо данные без необходимости обновлять другое, если не было внесено изменение в определенном интерфейсе между ПО и данными. Аналогично, данные конкретного приложения следует разделять от других данных.

17.4.3.4 На этапе обслуживания программного обеспечения процедуры контроля изменений должны обеспечить возможность внесения изменения в комплексное программное обеспечение только после того, как будет установлено, что либо пересматриваемое ПО совместимо с первоначальными данными, либо данные были пересмотрены по необходимости.

17.4.3.5 В процессе проверки и подтверждения подлинности ПО необходимо тщательно убедиться, что учтены все соответствующие сочетания данных.

17.4.3.6 Комплексное программное обеспечение должно быть разработано так, чтобы оно могло обнаруживать поврежденные данные конфигурации где это возможно.

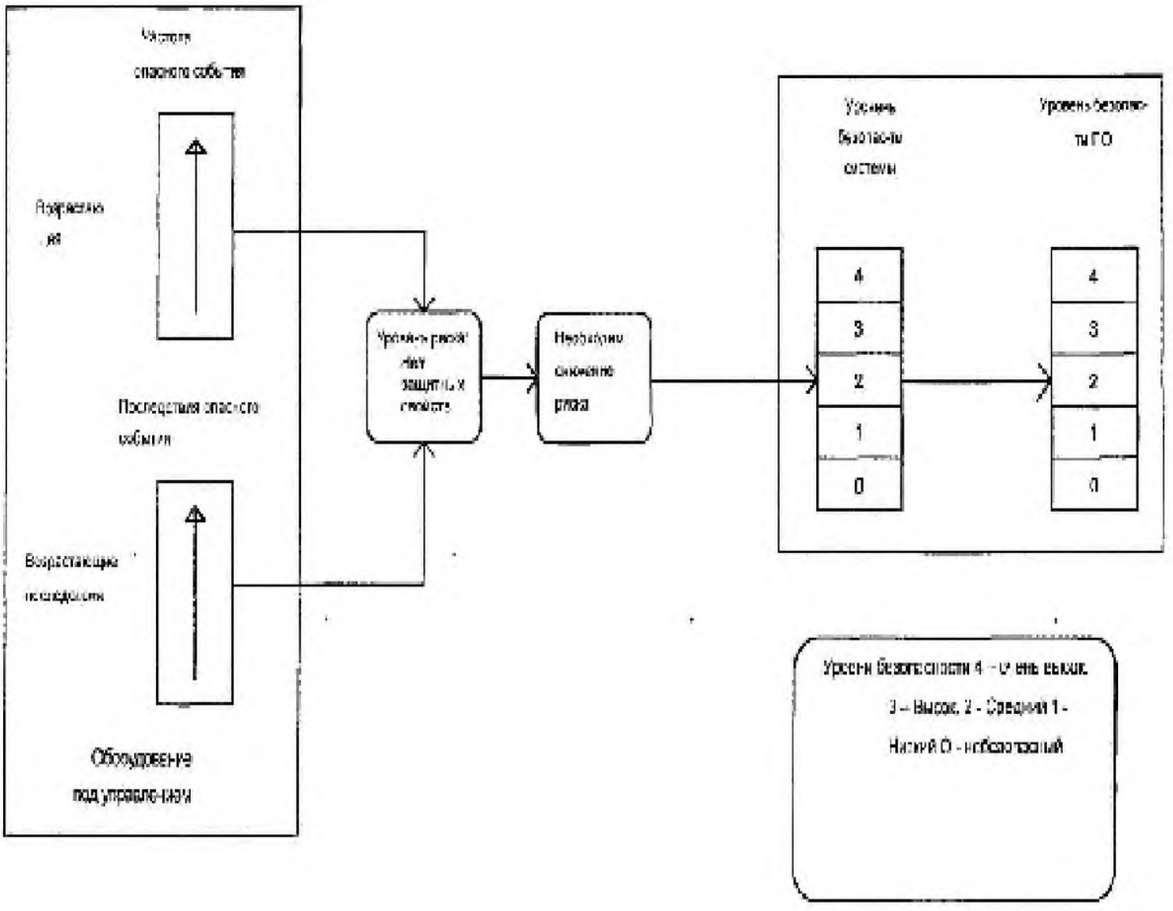

Рисунок 1 – Уровни системы безопасности

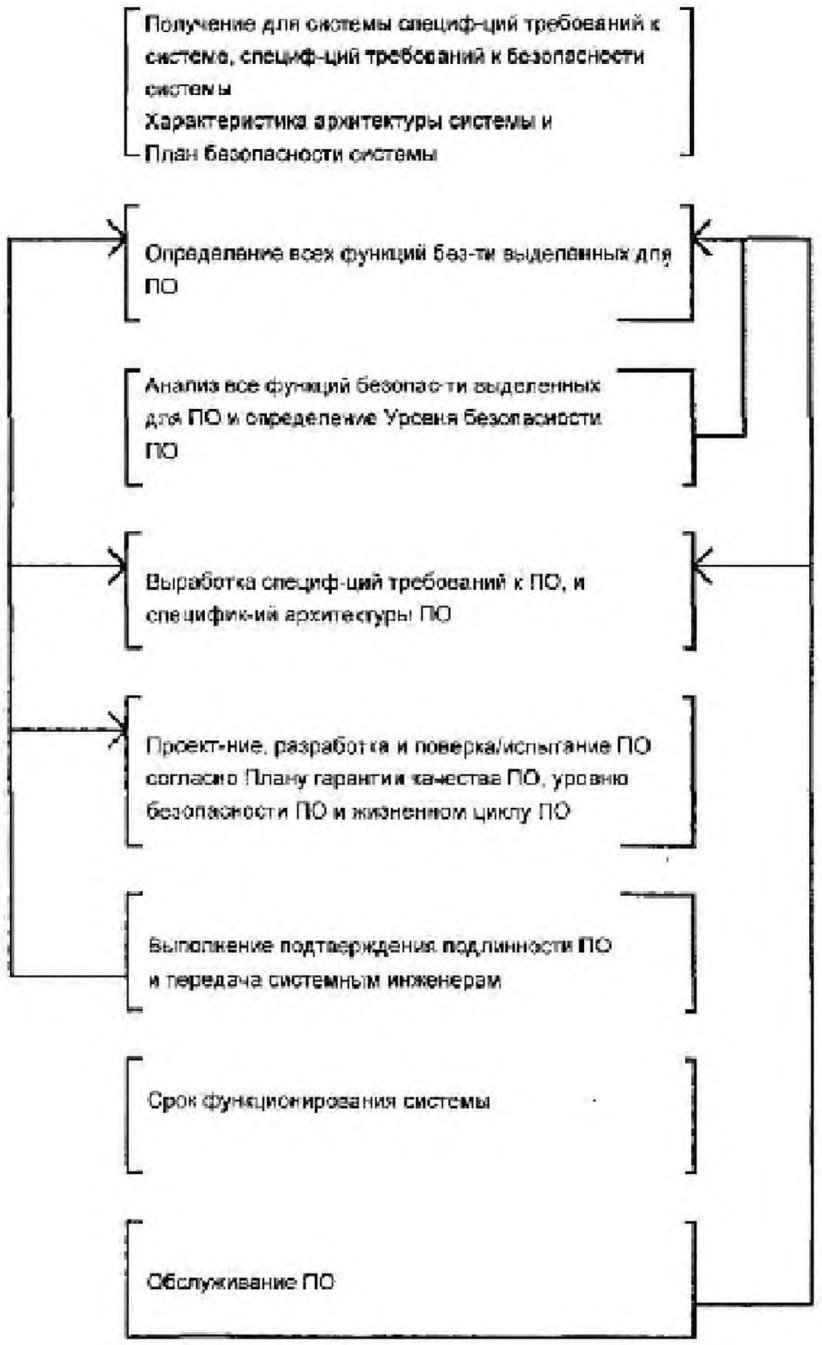

Рисунок 2 – Маршрутная карта безопасности Программного обеспечения

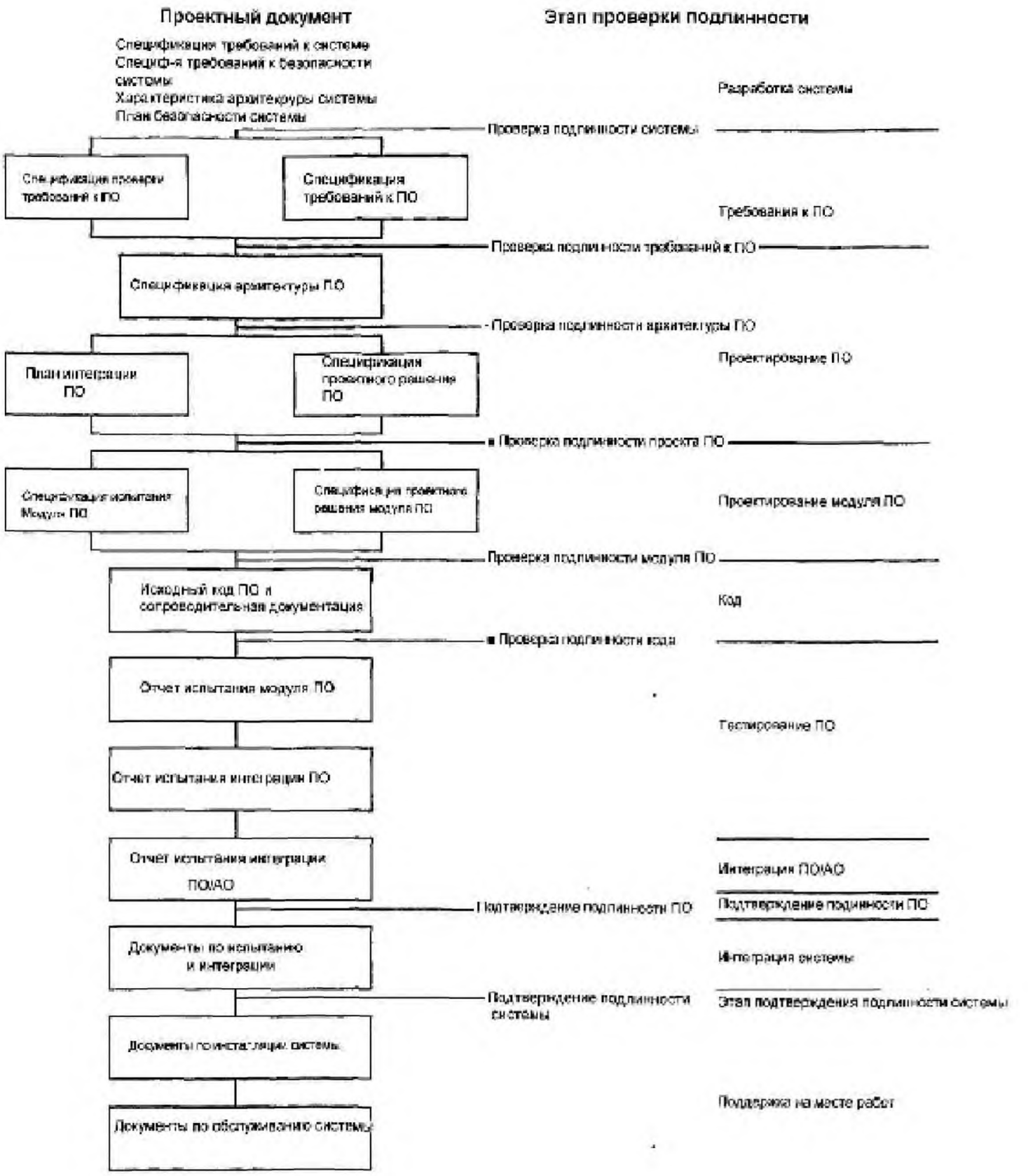

Рисунок 3 Жизненный цикл разработки

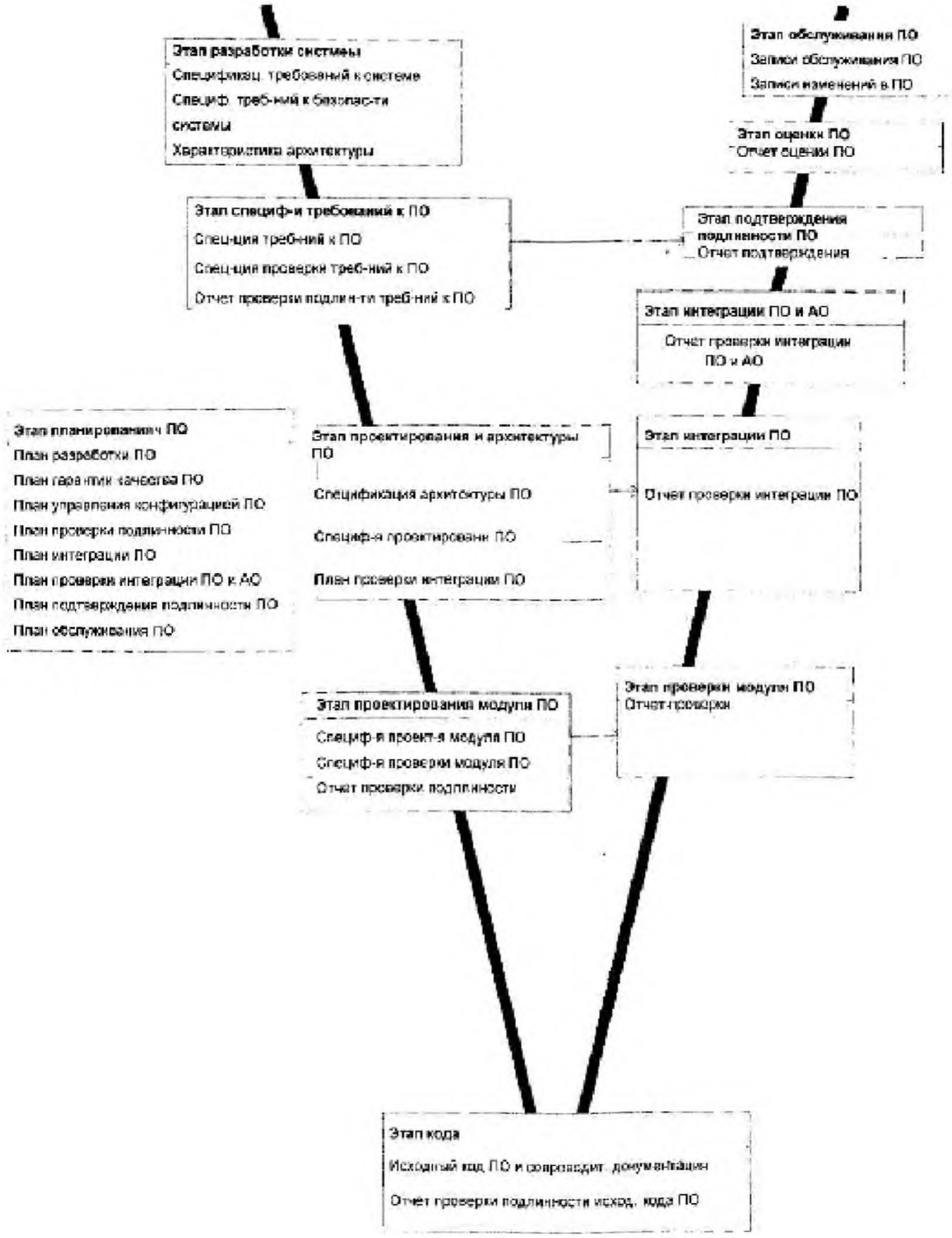

Рисунок 4 – Жизненный цикл разработки 2

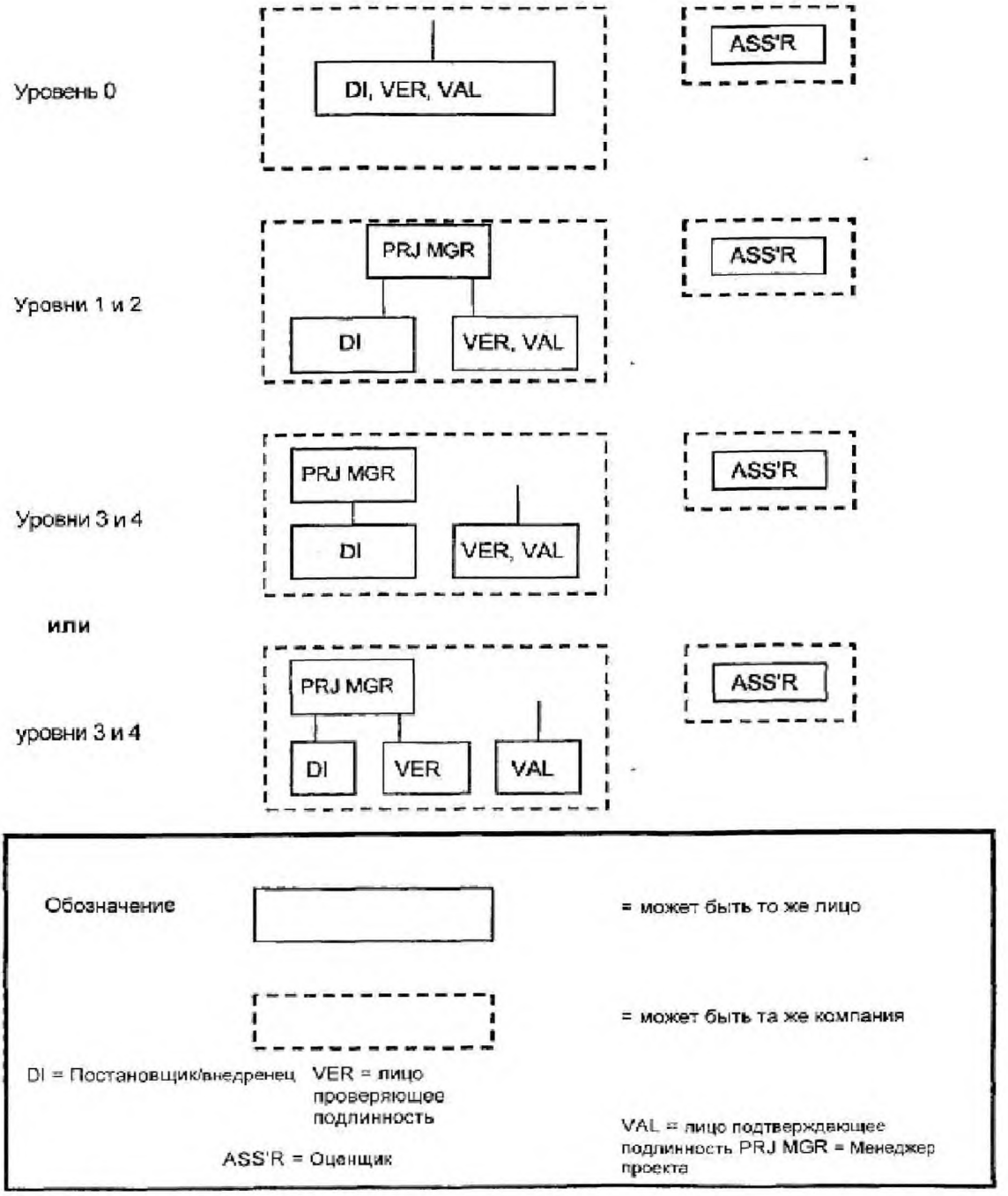

Рисунок 5 – Независимость в отношении уровня безопасности программного обеспечения

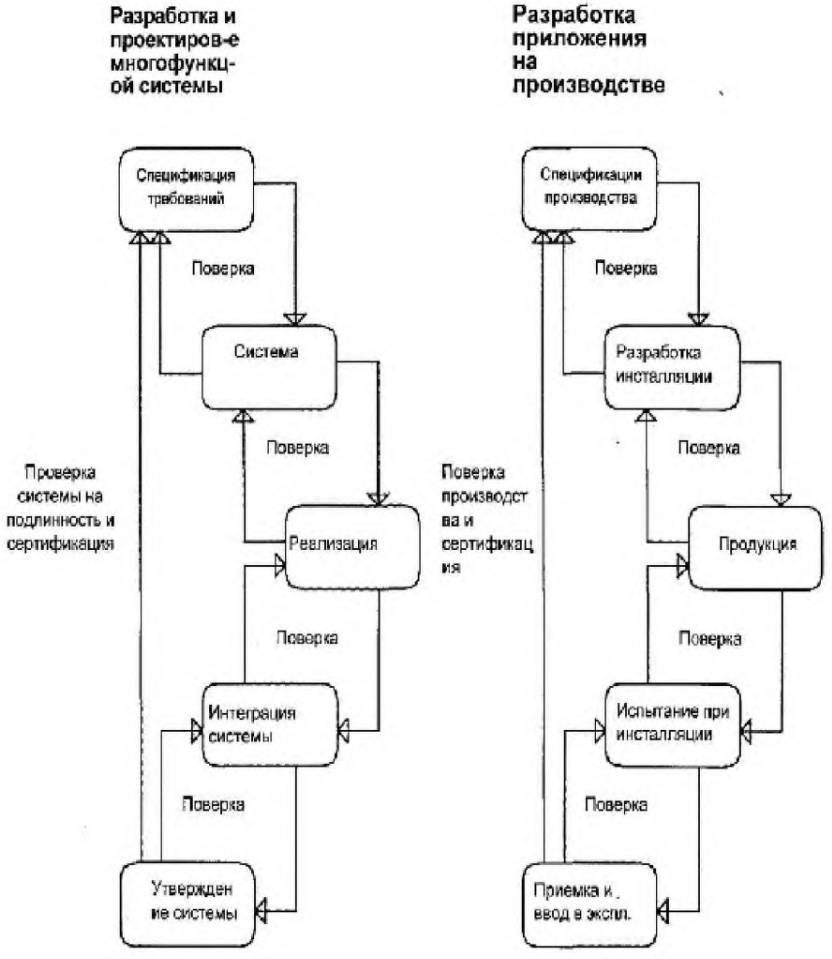

Рисунок 6 – Связь между разработкой комплексной системы и разработкой приложения

Приложение А
(обязательное)

Критерии выбора методов и мер

Каждый из пунктов 7 по 16 настоящего стандарта имеет соответствующую таблицу, которая иллюстрирует средства достижения соответствия. Есть таблицы с более низким уровнем, детальные таблицы (dt)[2], в которых записи раскрываются более подробно. Например, полуформальные методы подробно раскрываются в детальной таблице dt7 (таблица А. 18). Также имеется информативное приложение В на которое имеется ссылка в таблицах к пунктам.

При каждом методе или мере, указанном в таблице существует требование для каждого уровня безопасности программного обеспечения (УБПО) от 1 до 4, а также для уровня 0 отсутствия безопасности. В данной версии документа, требования для уровней 1 и 2 безопасности ПО одинаковые для каждого метода. Аналогично, у каждого метода есть те же самые требования для уровней 3 и 4 безопасности ПО. Такими требованиями могут быть:

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

«СР» данный символ означает, то метод или мера строго рекомендуема для данного уровня безопасности. Если этот метод или мера не используется, то необходимо предоставить подробное обоснование такого не использования в плане гарантии качества программного обеспечения или ином документе, на который делает ссылку план гарантии качества программного обеспечения;

«Р» данный символ означает, что метод или мера рекомендована для данного уровня безопасности. Это более низкий уровень рекомендации чем «СР» и такие методы могут совмещаться и образовывать часть пакета;

- данный символ означает, что метод или мера не имеют рекомендации к применению или неприменению;

«НР» данный символ означает, что метод или мера однозначно не рекомендуется для данного уровня безопасности. Если метод или мера применяется, то необходимо предоставить подробное обоснование такого использования в плане гарантии качества программного обеспечения или ином документе, на который делает ссылку план гарантии качества программного обеспечения.

Сочетания методов или мер должны быть указаны в плане гарантии качества программного обеспечения или ином документе, на который делает ссылку План гарантии качества программного обеспечения, один или более методов или мер должны быть отобраны, если примечание к таблице не содержит иных требований. Такие примечания могут включать ссылку на одобренные методы или одобренные сочетания методов. Если такие методы или сочетания методов применяются, то оценщик принимает их как действительные методы и интересуется лишь правильностью их применения. Если применяется другой комплект методов и он может быть обоснован, то оценщик может признать его приемлемым.

[2] dt = детальная таблица. Первая детальная таблица - А.12; по этой причине она имеет ссылку "dt1".

Таблицы к пунктам

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

ДОКУМЕНТАЦИЯ	Уровень безопасти 0	Уровень безопасти1	Уровень безопасти 2	Уровень безопасти 3	Уровень безопасти 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 по тестированию ПО	**-**	**СР**	**СР**	**СР**	**СР**
7. Отчет по проверке интеграции ПО и аппаратного оборуд	**-**	**СР**	**СР**	**СР**	**СР**
8. Отчет по утверждению ПО	**Р**	**СР**	**СР**	**СР**	**СР**
9. Отчет по оценке ПО	**-**	**СР**	**СР**	**СР**	**СР**
10. Учтенные записи по обслуживанию ПО	**Р**	**СР**	**СР**	**СР**	**СР**
Требование Соответствие СТ РК ИСО 9000 подразумевает выработку документации для всех уровней безопасности. Для уровня 0, разработчик должен выбрать подходящий тип документа.					

Таблица А.2 – Спецификация требований к программному обеспечению (пункт 8)

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Официальные методы, втом числе, напр. CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z и B	B.30	**-**	**Р**	**Р**	**СР**	**СР**
2. Полуоформальные методы	dt7	**Р**	**Р**	**Р**	**СР**	**СР**
3. Структурированная методика, в том числе, напр, JSD, MASCOT, SADT, SDL, SSADM и Yourdon	B.60	**Р**	**СР**	**СР**	**СР**	**СР**
Требований 1. Спецификация требований к ПО всегда требует описания проблемы на естественном языке и любое необходимое математическое обозначение, которое отражает приложение. 2. Таблица отображает дополнительные требования по четкому и ясному определению спецификации. Один или более из этих методов должны быть выбраны для соответствия применяемому уровню безопасности ПО.						

Таблица А.3 – Архитектура программного обеспечения (пункт 9)

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Защитное программирование	B.15	-	**P**	**P**	**CP**	**CP**
2. Выявление неисправностей и диагностика	B.27	-	**P**	**P**	**CP**	**CP**
3. Корректирующие коды	B.20	-	**P**	**P**	**CP**	**CP**
4. Коды обнаружения ошибок	B.20	-	**P**	**P**	**CP**	**CP**
5. Программирование утверждений сбоев	B.25	-	**P**	**P**	**CP**	**CP**
6. Методы подушки безопасности	B.54	-	**P**	**P**	**P**	**P**
7. Разнотипное программирование	B.17	-	**P**	**P**	**CP**	**CP**
8. Блок устранения	B.50	-	**P**	**P**	**P**	**P**
9. Ретроспективное устранение ошибок	B.5	-	**HP**	**HP**	**HP**	**HP**
10. Восстановление без возврата	B.32	-	**HP**	**HP**	**HP**	**HP**
11. Механизмы повторного устранения неисправностей	B.53	-	**P**	**P**	**P**	**P**
12. Запоминание выполненных наборов данных Cases	B.39	-	**P**	**P**	**CP**	**CP**
13. Искусственный интеллект - Исправление неисправностей	B.1	-	**HP**	**HP**	**HP**	**HP**
14. Динамическая реконфигурация ПО	B.18	-	**HP**	**HP**	**HP**	**HP**
15. Анализ последствий ошибок ПО	B.26	-	**P**	**P**	**CP**	**CP**
16. Анализ дерева неисправностей	B.28	**P**	**P**	**P**	**CP**	**CP**

Требования 1. Утвержденные сочетания методов для уровней БПО 3 И 4 должны быть след.: 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. Кроме записей 3, 9, 10, 13 и 14, один или более из этих методов должны быть выбраны для выполнения требований по уровням БПО 1 и 2.
3. Некоторые из этих вопросов могут быть определены на уровне системы.
4. Корректирующие ошибки коды могут быть использованы в соответствии с требованиями СТ РК МЭК 62280-1* и СТ РК МЭК 62280-2.

Таблица А.4- Проектирование и внедрение программного обеспечения (пункт 10)

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Официальные методы, втом числе, напр CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z и B	B.30	-	P	P	CP	CP
2. Полуоформальные методы	dt7	P	CP	CP	CP	CP
3. Структурированная методика, в том числе, напр JSD, MASCOT, SADT, SDL, SSADM и Yourdon	B.60	P	CP	CP	CP	CP
4. Модульный приницп	dt9	CP	O	O	O	O
5. Стандарты проектирования и кодирования	dt1	CP	CP	CP	O	O
6. Анализируемые программы	B.2	CP	CP	CP	CP	CP
7. Язык программирования со строгим контролем типа	B.57	P	CP	CP	CP	CP
8. Структурированное программирование	B.61	P	CP	CP	CP	CP
9. Язык программирования	dt4	P	CP	CP	CP	CP
10. Языковое подмножество	B.38	P	-	-	CP	CP
11. Утвержденный транслятор	B.7	P	CP	CP	CP	CP
12. Зарекомендовавший себя транслятор	B.65	CP	CP	CP	CP	CP
13. Библиотечный комплеккт пригодных/проверенных модулей и компонентов	B.40	P	P	P	P	P
14. Функциональная проверка/проверка методом черного ящика	dt3	CP	CP	CP	M	M
15. Проверка эксплуатационных параметров	dt6	-	CP	CP	CP	CP
16. Проверка интерфейса	B.37	CP	CP	CP	CP	CP
17. Регистрация данных и анализ данных	B.13	CP	CP	CP	O	O
18. Нечёткая логика	B.67	-	-	-	-	-
19. Объектно-ориентированное программирование	B.68	-	P	P	P	P
Требования 1. Соответ. набор методов должен быть выбран согласно уровню БПО. 2. Пр уровне БПО 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 – Проверка подлинности и тестирование (пункт 11)

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Формальное доказательство	B.31	-	**P**	**P**	**CP**	**CP**
2. Вероятностная проверка	B.47	-	**P**	**P**	**CP**	**CP**
3. Статистический анализ	dt8	-	**CP**	**CP**	**CP**	**CP**
4. Динамическйи анализ и проверка	dt2	-	**CP**	**CP**	**CP**	**CP**
5. Система показателей	B.42	-	**P**	**P**	**P**	**P**
6. Матрица обнаруживаемости	B.69	-	**P**	**P**	**CP**	**CP**
7. Анализ последствий ошибки ПО	B26	-	**P**	**P**	**CP**	**CP**
Требования 1. Для Уровней БПО 3 или 4, утвержденное сочетание методов должно быть: a) 1 и 4 или b) 3 и 4 или c) 4, 6 и 7 2. Для Уровней БПО 1 или 2, утвержденные сочетаниия должны быть a) 1 или b) 3 и 4 или c) 4						

Таблица А.6 – Интеграция программного обеспечения и аппаратного оборудования (пункт 12)

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Функциональная проверка/проверка методом черного ящика	dt3	**CP**	**CP**	**CP**	**CP**	**CP**
2. Проверка эксплуатационных параметров	dt6	-	**P**	**P**	**CP**	**CP**
Требования 1. Для уровня БПО 0, метод 1 должен быть утвержденным методом. 2. Для уровней БПО 1, 2, 3 или 4, утвержденным сочетанием методов должны быть 1 и 2.						

Таблица А.7 – Подтверждение подлинности программного обеспечения (пункт 13)

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1 Вероятностная проверка	B.47	-	**P**	**P**	**CP**	**CP**
2. Проверка эксплуатационных	dt6	-	**CP**	**CP**	**O**	**O**
3. Функциональная проверка/проверка методом черного ящика	dt3	**CP**	**CP**	**CP**	**O**	**O**
4. Моделирование	dt5	-	**P**	**P**	**P**	**P**
Требование: Для уровней БПО 1, 2, 3 или 4, утвержденным сочетанием методов должны быть 2 и 3.						

Таблица А.8 – Пункты подлежащие оценке

ПУНКТЫ ПОДЛЕЖАЩИЕ ОЦЕНКЕ	Пункт	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Уровни БПО	5	СР	СР	СР	СР	СР
2. Персонал и функциональные обязанности	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. Проектирование и разработка	10	-	СР	СР	СР	СР
7. Проверка подлинности	11	-	СР	СР	СР	СР
8. Интеграция ПО и АО	12	-	Р	Р	СР	СР
9. Подтверждение подлинности ПО	13	-	СР	СР	СР	СР
10. Гарантия качества	15	-	СР	СР	СР	СР
11. Обслуживание	16	-	СР	СР	СР	СР

Таблица А.9 – Оценка программного обеспечения (пункт 14) методы оценки

МЕТОДЫ ОЦЕНКИ	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Проверочные листы	B.8	СР	СР	СР	СР	СР
2. Статистический анализ ПО	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 по дереву неисправностей	B.28	Р	Р	Р	СР	СР
7. Анализ последствий ошибки ПО	B.26	-	Р	Р	СР	СР
8. Анализ сбоев вызванных общей ошибкой	B.10	-	Р	Р	СР	СР
9. Модели Маркова	B.41	-	Р	Р	Р	Р
10. Блок-схема надёжности	B.51	-	Р	Р	Р	Р
11. Полевое испытание перед вводом в эксплуатацию	-	Р	СР	СР	СР	СР
Требование: Один или более из этих методов должен быть выбран для соответствия используемому уровню БПО.						

Таблица А.10 – Гарантия качества программного обеспечения (пункт 15)

МЕТОД/ МЕРА	Пункт или указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Аккредитация по СТ РК ИСО 9001	15	Р	СР	СР	СР	СР
2. Соответствие СТ РК ИСО 9000	15	О	О	О	О	О
3. Система качества организации	15	О	О	О	О	О
4. Управление конфигурацией ПО	B.56	О	О	О	О	О

Таблица А.11 – Обслуживание программного обеспечения (пункт 16)

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Анализ последствий	B.35	Р	СР	СР	О	О
2 Регистрация данных и анализ данных	B.13	СР	СР	СР	О	О

Детальные таблицы (dt)

Таблица А.12 – Стандарты проектирования и кодирования (dt1) указанные в пункте 10

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Существование 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 ссылок	B.16	-	Р	Р	Р	Р
6. Ограниченное использование рекурсии	B.16	-	Р	Р	СР	СР
7. Отсутствие команд безусловного перехода	B.16	-	СР	СР	СР	СР
Требования 1. Принято, что методы 3, 4 и 5 могут быть присутсвовать как часть утвержденной компиллирующей программы или транслятора. 2. Соответ. набор методов должен быть выбран согласно уровню БПО.						

Таблица А.13 – Динамический анализ и тестирование\ (dt2) указанные в пункте 11 и 14

МЕТОД/ МЕРА	**указат**	Уровень безопас-ти **0**	Уровень безопас-ти **1**	Уровень безопас-ти **2**	Уровень безопас-ти **3**	Уровень безопас-ти **4**
1. Выполнение тестовых данных из анализа граничных значений	B.4	-	**CP**	**CP**	**CP**	**CP**
2. Выполнение тестовых данных из рассуждений с использованием догадок для выявления ошибок	B.21	**P**	**P**	**P**	**P**	**P**
3. Выполнение тестовых данных из подсева ошибок	B.22	-	**P**	**P**	**P**	**P**
4. Моделирование качества функционирования	B.45	-	**P**	**P**	**CP**	**CP**
5. Классы эквивалентности и проверка вводного сегмента	B.19	-	**P**	**P**	**CP**	**CP**
6. Проверка основанная на структуре	B.58	-	**P**	**P**	**CP**	**CP**
Требования 1. Анализ по тестовых данным находится на уровне подсистемы и основан на спецификации и/или спецификации и коде. 2. Соответ. набор методов должен быть выбран согласно уровню БПО.						

Таблица А.14 – Функциональная проверка /Проверка методом черного ящика (dt3) указанные в пунктах 10,12, 13 и 14

МЕТОД/ МЕРА	**указат**	Уровень безопас-ти **0**	Уровень безопас-ти **1**	Уровень безопас-ти **2**	Уровень безопас-ти **3**	Уровень безопас-ти **4**
1. Выполнение тестовых данных из при чинно-следственных диаграмм	B.6	-	-	-	**P**	**P**
2. Создание прототипа/«оживление» изображения	B.49	-	-	-	**P**	**P**
3. Анализ граничных значений	B.4	**P**	**CP**	**CP**	**CP**	**CP**
4. Классы эквивалентности и проверка вводных сегментов	B.19	**P**	**CP**	**CP**	**CP**	**CP**
5. Моделирование процесса	B.48	**P**	**P**	**P**	**P**	**P**
Требования: 1. Полнота симулирования будет зависеть от диапазона уровня БПО, сложности и приложения. 2. Соответ. набор методов должен быть выбран согласно уровню БПО.						

Таблица А.15 – Языки программирования (dt4) указанные в пункте 10

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 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. Множество C илиг 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. Компонующая программа	B.62	**P**	**P**	**P**	-	-
10. Многоступенчатая схема	B.62	**P**	**P**	**P**	**P**	**P**
11. Функциональные блоки	B.62	**P**	**P**	**P**	**P**	**P**
12. Перечень заявлений	B.62	**P**	**P**	**P**	**P**	**P**

Требования:
1. При уровнях БПО 3 и 4, когда применяется языковое подмножество 1, 2, 3 и 4, рекомендация меняется на CP.
2. Для некоторых языков только приложения 7 и 9 могут быть единственными имеющимися. При уровнях БРО 3 и 4 где нет строго рекомендуемого варианта, строго рекомендуется, чтобы было языковое множество чтобы довести рекомендацию до «P», а также должен быть точный набор стандартов кодирования.
3. Чтобы получить информацию по оценке соответствия языка программирования, смотри запись в библиографии «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 - Моделирование (dt5) указанное в Пункте 13

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Диаграммы потока данных	B.12	-	**P**	**P**	**P**	**P**
2. Конечные автоматы	B.29	-	**CP**	**CP**	**CP**	**CP**
3. Формальные методы	B.30	-	**P**	**P**	**CP**	**CP**
4. Моделирование качества функционирования	B.45	-	**P**	**P**	**CP**	**CP**
5. Временные сетки Петри	B.64	-	**CP**	**CP**	**CP**	**CP**
6. Создание прототипа /«оживление» изображения	B.49	-	**P**	**P**	**P**	**P**
7. Структурные диаграммы	B.59	-	**P**	**P**	**CP**	**CP**

Требование: Соответ. набор методов должен быть выбран согласно уровню БПО.

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

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти1	Уровень безопас-ти2	Уровень безопас-ти3	Уровень безопас-ти4
1. Лавинные/нагрузочные испытания	B.3	-	**P**	**P**	**CP**	**CP**
2. Время реакции и факторы ограничения памяти	B.52	-	**CP**	**CP**	**CP**	**CP**
3. Требования к качеству функционирования	B.46	-	**CP**	**CP**	**CP**	**CP**
Требование: Соответ. набор методов должен быть выбран согласно уровню БПО.						

Таблица А.18 – Полуформальные методы (dt7) указанные в пунктах 8 и 10

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Логическая /функциональная блок-схема	-	**P**	**P**	**P**	**CP**	**CP**
2. Диаграммы последовательности	-	**P**	**P**	**P**	**CP**	**CP**
3. Диаграммы потока данных	B.12	**P**	**P**	**P**	**P**	**P**
4. Конечные автоматы /диаграммы перехода из одного состояния в другое состояние	B.29	-	**P**	**P**	**CP**	**CP**
5. Временные сетки Петри	B.64	-	**P**	**P**	**CP**	**CP**
6. Решающие таблицы / таблицы истинности	B.14	**P**	**P**	**P**	**CP**	**CP**
Требование: Соответ. набор методов должен быть выбран согласно уровню БПО.						

Таблица А.19 – Статический анализ (dt8) указанный в пунктах 11 и 14

МЕТОД/ МЕРА	указат	Уровень безопас-ти 0	Уровень безопас-ти 1	Уровень безопас-ти 2	Уровень безопас-ти 3	Уровень безопас-ти 4
1. Анализ граничного значения	B.4	-	**P**	**P**	**CP**	**CP**
2. Проверочные листы	B.8	-	**P**	**P**	**P**	**P**
3. Анализ потока управления	B.9	-	**CP**	**CP**	**CP**	**CP**
4. Анализ потока данных	B.11	-	**CP**	**CP**	**CP**	**CP**
5. Рассуждения с использованием догадок для выявления ошибок	B.21	-	**P**	**P**	**P**	**P**
6. Проверки по Фагану	B.24	-	**P**	**P**	**CP**	**CP**
7. Анализ ложной цепи	B.55	-	-	-	**P**	**P**
8. Символьное выполнение	B.63	-	**P**	**P**	**CP**	**CP**
9. Критический анализ / анализ проектирования	B.66	**CP**	**CP**	**CP**	**CP**	**CP**
Требование: Соответ. набор методов должен быть выбран согласно уровню БПО.						

Таблица А.20 – Модульный принцип (dt9) указанный в пункте 10

МЕТОД/ МЕРА	указат	Уровень безопас-ти0	Уровень безопас-ти1	Уровень безопас-ти2	Уровень безопас-ти3	Уровень безопас-ти4
1. Ограниченный размер модуля	B.43	**CP**	**CP**	**CP**	**CP**	**CP**
2. Сокрытие информации / инкапсуляция	B.36	**P**	**CP**	**CP**	**CP**	**CP**
3. Ограничие числа параметров	B.43	**P**	**P**	**P**	**P**	**P**
4. Пункт одного входа и одного выхода в стандартных подпрограммах и функциях	B.43	**P**	**CP**	**CP**	**CP**	**CP**
5. Полностью определенный интерфейс	B.43	**CP**	**CP**	**CP**	**O**	**O**
Требование: Соответ. набор методов должен быть выбран согласно уровню БПО.						

Приложение Б
(справочное)

Библиография методов

Б.1 Искусственный интеллект – Устранение неисправностей (указанное в пункте 9)

Задачи

Возможность гибкого реагирования на потенциальные опасные факторы за счет введения комплекса (сочетания) методов и моделей процесса, а также анализа безопасности и надежности в режиме онлайн.

Характеристика

Прогнозирование возникновения неисправностей (ходы расчета), устранение неисправностей, обслуживание и надзорные действия могут быть поддержаны системами, основанными на искусственном интеллекте достаточно эффективно в различных каналах системы, поскольку правила могут быть получены непосредственно из спецификаций и проверены относительно них. Определенные общие неисправности, которые косвенным образом вносятся в спецификации уже с учетом правил проектирования и реализации, можно избежать эффективно таким подходом, особенно при применении сочетания моделей и методов функционально или описательно.

Методы выбираются таким образом, чтобы неисправности можно было устранять, а последствия неисправностей минимизировать, для соответствия условиям безопасности и надежности.

Б.2 Анализируемые программы (как указано в пункте 10)

Задачи

Разработка программы, которая позволяет легко осуществлять ее анализ. Реакция программы должна быть контролируемой полностью на основе анализа.

Характеристика

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

- поток управления модулем должен состоять из структурированных конструктивов, то есть последовательностей, шаги цикла и выбор;
- 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 простой;
- сложные расчеты не должны использоваться в качестве основы для разветвления и решений по замкнутой цепи;
- разветвление и решения по замкнутой цепи должны быть просто связаны с входными параметрами модулей;
- границы между различными видами отображения в виде карт должны быть простыми.

Справочный материал: [8],[9].

Б.3 Лавинные/нагрузочные испытания (как указано в таблице dt6)

Задачи

Применение исключительно высокой рабочей нагрузки к объекту испытания с тем,

чтобы показать, что объект испытания легко выдержит нормальную рабочую нагрузку.

Характеристика

Существует ряд различных условий испытания, которые можно применить для лавинных/нагрузочных испытаний. Несколько таких условий даются ниже:

- при работе в режиме опроса, объект испытания получает намного больше входных изменений за единицу времени чем при обычных условиях;
- при работе по запросу, число запросов за единицу времени по объекту испытания увеличивается за пределы обычных условий;
- если размер базы данных играет важную роль, то он увеличивается за пределы обычных условий;
- влияющие средства устанавливаются на максимальную или минимальную скорость соответственно;
- для крайних случаев, все влияющие факторы, насколько это возможно, вносятся в граничные условия одновременно.

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

Б.4 Анализ граничных значений (как указано в таблицах dt2, dt3 и dt8)

Задачи

Устранение ошибок в программном обеспечении, возникающих при ограничениях параметров или границах.

Характеристика

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

- делитель нуля;
- 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 написать набор тестовых данных, чтобы принудительно подвести вывод к его ограниченным значениям. Также следует рассмотреть возможность указать тестовые данные, из-за которого вывод превышает граничные значения спецификации.

Если вывод представляет собой последовательность данных, например, распечатанная таблица, то особое внимание следует уделить первому и последнему элементам и спискам, в которых их нет, 1 и 2 элементам.

Справочный материал: [10].

Б.5 Ретроспективное устранение ошибок (как указано в Пункте 9)

Задачи

Обеспечение правильной функциональной работы в присутствии одной или нескольких неисправностей.

Характеристика

При выявлении неисправности, система переустанавливается на более ранее внутреннее состояние, логичность чего была ранее доказана. Этот метод подразумевает сохранение внутреннего состояния зачастую на так называемых строго определенных

контрольных точках. Это может быть выполнено глобально (для полной базы данных) или с определенным шагом (изменения только между контрольными точками). В таком случае система должна компенсировать изменения, произошедшие в течение этого времени используя протоколирование (контрольный анализ действий), компенсацию (все последствия таких изменений аннулируются) или внешнее (ручное) взаимодействие.

Б.6 Причинно-следственные диаграммы (как указано в пункте 14 и таблице dt3)

Задачи

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 неисправностей и анализ по дереву событий. Начиная с критического события, причинно-следственная диаграмма прослеживается в обратном направлении и вперед. В обратном направлении она эквивалента дереву неисправностей, где критическое событие является конечным событием. В направлении вперед идентифицируются возможные последствия, возникающие из 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 неисправностей. 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: [11].

Б.7 Сертифицированные инструменты и трансляторы (как указано в пункте 10)

Задачи

Инструменты необходимы разработчикам на различных этапах разработки программного обеспечения. Где это возможно, инструменты должны быть сертифицированы с тем, чтобы можно было получить определенную степень уверенности относительно правильности полученных данных.

Характеристика

Сертифицированный инструмент – это инструмент по которому было установлено, что он обладает определенным качеством. Сертификация инструмента обычно выполняется независимым, зачастую национальным, органом по независимо установленным критериям, обычно национальным или международным стандартам. В идеале, инструменты используемые на всех этапах разработки (определение, разработка проекта, кодирование, испытание и подтверждение подлинности) и инструменты используемые в управлении конфигурацией должны проходить сертификацию. На сегодняшний день лишь компилирующие программы (трансляторы) регулярно проходят процедуру сертификации; их устанавливают национальные сертификационные органы, и они проверяют компилирующие программы (трансляторов) по международным стандартам, таким как стандарты для Ada и Pascal.

Справочный материал: [12], [13].

Б.8 Проверочные листы (как указано в пункте 14 и таблице dt8)

Задачи

Обеспечение стимулирующего воздействия на критическую оценку всех аспектов системы, вместо установления определенных требований.

Характеристика

Ряд вопросов, на которые необходимо ответить лицу, выполняющему

контрольную проверку на проверочных листах. Многие вопросы имеют общий характер, и Оценщик должен интерпретировать их как представляющиеся наиболее уместными для конкретной оцениваемой системы.

Чтобы включить широкие вариации в программное обеспечение и аппаратное оборудование, которые проходят процесс подтверждения подлинности, большинство проверочных листов включают вопросы, которые применимы ко многим типам системы. В результате этого, в проверочных листах могут использоваться вопросы, которые не относятся к рассматриваемой системе и которые следует игнорировать. В равной мере может существовать необходимость, для конкретной системы, дополнить стандартный проверочный лист вопросами специально рассчитанными на рассматриваемую систему.

В любом случае, должно быть ясно, что использование проверочных листов особо зависит от компетентности и суждений инженера, выбирающего и использующего проверочные листы. Как результат, решения принятые инженером по выбранным проверочным листам, а также любым дополнительным и ненужным вопросам, должны быть полностью документированы и обоснованы. Задача – обеспечить возможность анализа применения проверочных листов и получения аналогичных результатов если не используются другие критерии.

По-возможности, объект в заполнении проверочного листа должен быть сокращенным. При необходимости предоставления четкого обоснования, следует делать ссылку на дополнительные документы. «Пропуск», «Неуспешное выполнение» и «Неубедительно», либо другие подобные наборы обозначений следует использовать для учета результатов по каждому вопросу. Такая краткость упрощает процесс достижения общего вывода по результатам оценки с использованием проверочных листов.

Справочный материал: [14] ,[15], [16], [17].

Б.9 Анализ потока управления (как указано в таблице dt8)

Задачи

Выявление несовершенных и потенциально неверных программных структур.

Характеристика

Анализ потока управления выявляет сомнительные участки кода, которые не следуют рациональной практике программирования. Программа анализируется чтобы сформировать направленную диаграмму, которую можно анализировать по:

- недоступному коду, например, операции безусловного перехода, которые оставляют блоки кода недосягаемыми;
- узловой код, то есть хорошо структурированный код, чья диаграмма управления сводится последующими редукциями диаграммы к единому узлу. Слабо структурированный код может быть лишь приведен к узловому пункту, состоящему из нескольких узлов.

Справочный материал: [18[, [19].

Б.10 Анализ сбоев, вызванных общей ошибкой (как указано в пункте 14)

Задачи

Выявление потенциальных сбоев в резервных системах или подсистемах, которые сводят на нет преимущества резервирования из-за одновременного появления одинаковых сбоев в частях резервирования.

Характеристика

Компьютерные системы, рассчитанные на обеспечение безопасности установки зачастую используют резервирование в аппаратном оборудовании и мажоритарном голосовании. Данный метод применяется во избежание случайных сбоев компонентов, которые обычно препятствуют верной обработке данных в компьютерной системе.

Однако, некоторые сбои могут быть общими для более чем одного компонента.

Например, если компьютерная система установлена в единственной комнате, изъяны в системе кондиционирования воздуха могут снизить полезность резервирования. То же самое можно сказать и в отношении других внешних факторов воздействия на компьютерную систему, такие как: пожар, потоп, электромагнитные помехи, падения самолета и землетрясения. Компьютерная система может также подвергнуться воздействию в результате аварийных ситуаций связанных с эксплуатацией и обслуживанием. Таким образом, очень важно, чтобы существовали адекватные и должным образом задокументированные процедуры по эксплуатации и обслуживанию. Также важно обеспечить обучение эксплуатации и обслуживанию.

Внутренние факторы воздействия также существенно влияют на отказы, вызванные общей ошибкой (CCF). Они могут происходить в результате ошибок при проектировании в общих или идентичных компонентах и их интерфейсах, а также изнашивания компонентов. При анализе CCF необходимо осуществлять поиск в системе таких потенциальных общих сбоев. Методы анализа CCF представляют собой общий контроль качества, анализ проектирования, проверка подлинности и испытание независимой группы экспертов, а также анализ реальных аварийных ситуаций с учетом опыта схожих систем. Масштаб анализа, тем не менее, выходит за рамки аппаратного оборудования. Даже если в сложных цепочках резервной компьютерной системы используется «различное программное обеспечение», в принципах программного обеспечения 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 использованием сравнительных методов между резервными цепочками, которые должны привести к обнаружению сбоя прежде чем этот сбой окажется общим для всех цепочек. Анализ CCF должен учитывать этот прием.

Справочный материал: [20], [21], [22].

Б.11 Анализ потока данных (как указано в Таблице dt8)

Задачи

Выявление несовершенных и потенциально неверных программных структур.

Характеристика

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

- переменных величин, считываемых прежде чем они написаны. Вероятно, в этом заключается ошибка, и это представляет собой неудачную практику программирования;
- переменных величин, написанных более одно раза без считывания. Это может означать упущенный код;
- переменных величин написанных, но не считываемых. Это может указывать на резервный код.

Существует также дополнительный анализ потока данных, известный как анализ потока информации, где потоки фактических данных (в рамках и между процедурами) сопоставляются с целью проектирования. Обычно это выполняется с использованием компьютеризированного инструмента, когда запланированные потоки данных определяются с использованием структурированного комментария, которое может быть считано инструментом.

Справочный материал: [23], [24].

Б.12 Диаграммы потока данных (как указано в таблицах dt5 и dt7)

Задачи

Описание потока данных через программу в форме диаграммы.

Характеристика

В диаграммах потока данных документально отображается то каким образом ввод данных трансформируется в вывод данных, при этом каждый этап в диаграмме показывает отчетливую трансформацию.

Диаграммы потока данных состоят из трех компонентов:

1. снабженных комментариями стрелок – показывают поток данных в/из центров трансформации, комментарии поясняют какие это данные;
2. снабженных комментариями кружков – показывают центры трансформации, комментарии поясняют трансформацию;
3. операторы (AND, XOR) – эти операторы используются для связывания стрелок.

В диаграммах потока данных описывается, каким образом вводные данные трансформируется в выходные данные. Они не включают, и не должны включать, информацию управления или информацию о задании последовательности. Каждый кружок может рассматриваться как самостоятельный «черный ящик», который, как только его входные данные достушны, трансформирует их в выходные данные.

Одним из принципиальных преимуществ диаграмм потока данных заключается в том, что они показывают трансформацию, не делая каких-либо допущений относительно того, как осуществляются такие трансформации.

Составление диаграмм потока данных осуществляется наилучшим образом путем рассмотрения вводных данных системы и работы по направлению к выходным данным системы. Каждый кружок должен показывать отчетливую трансформацию – его вывод должен, определенным образом, отличаться от его ввода. Не существует правил определения общей структуры диаграммы, и построение диаграммы потока данных является одним из творческих аспектов проектирования системы. Как и любое другое проектирование, это альтернативный процесс попыток усовершенствованный по этапам с получением конечной диаграммы.

Справочный материал: [25].

Б.13 Запись данных и анализ (как указано в пунктах 10 и 16)

Задачи

Запись всех данных, решений и обоснований в проекте по программному обеспечению для более простой проверки и подтверждения подлинности, оценки и обслуживания.

Характеристика

Подробные записи ведутся в течение проекта, как на основе проекта, так и отдельно. Например, инженеру необходимо вести учет, который может включать:

- меры, подробно изложенные по отдельным модулям,
- испытание, проведенное по каждому модулю,
- решения и их обоснования,
- достижение основных этапов проекта,
- проблемы и пути их решения.

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

Б.14 Решающие таблицы (Таблицы истинности) (как указано в пунктах 14 и таблицах dt7)

Задачи

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 перемешными.

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

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

Б.15 Защитное программирование (как указано в пункте 9)

Задачи

Составление программ, которые обнаруживают аномальный поток управления, поток данных или значений данных во время их выполнения и реагируют на них в приемлемым и заданным образом.

Характеристика

Во время программирования могут быть использованы многие методы проверки аномалий управления или данных. Их можно применять систематически через программирование системы для снижения вероятности ошибочной обработки данных.

Можно выделить две накладывающихся друг на друга области защитных методов.

В ошибкоустойчивом программном обеспечении есть свои собственные проектные недостатки. Такие недостатки могут быть вызваны обычной ошибкой проектирования или кодирования, либо ошибочными требованиями. Далее перечислены некоторые защитные методы:

- переменные должны быть проверены на попадание в диапазон;
- где это возможно, значения следует проверять на достоверность;
- параметры к процедурам должны быть проверены на соответствие типу, размеру и диапазону при вводе процедуры.

Эти первые три рекомендации помогают обеспечить разумность чисел манипулируемых программой, как с точки зрения программной функции, так и с точки зрения физического значения переменных.

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

Устойчивое к ошибкам программное обеспечение разрабатывается таким образом, чтобы 'ожидать' сбои в собственной среде или использовать внешние номинальные или предполагаемые условия, а также вести себя установленным образом. Методы включают следующее:

- входные переменные и промежуточные перемешные с физическим значением следует проверять на достоверность;
- следует проверять эффект выходных переменных, предпочтительно путем непосредственного наблюдения за изменениями в состоянии соответствующих систем;
- программное обеспечение должно проверять свою конфигурацию. Это должно предусматривать как существование, так и доступность предполагаемого

аппаратного оборудования, а также комплектность программного обеспечения. Это особенно важно для сохранения целостности после всех процедур по обслуживанию.

Некоторые методы защитного программирования, такие как проверка порядка потока управления, также справляются с внешними сбоями.

Справочный материал: [26], [27], [28].

Б.16 Стандарты проектирования и кодирования (как указано в таблице dt1)

Задачи

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

Характеристика

Правила подлежащие соблюдению согласовываются в начале проекта участниками. 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 случае с языками ориентированными на объект;
- 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, проверки подлинности, оценки и обслуживания. Таким образом, они должны рассматривать имеющиеся инструменты, а именно анализаторы и средства обратного проектирования.

Б.17 Многовариантное программирование (как указано в пункте 9)

Задачи

Выявление и маскировка остаточных ошибок проектирования программного обеспечения при выполнении программы для предотвращения критических сбоев безопасности системы, а также продолжения работы по повышению надежности.

Характеристика

В многовариантном программировании спецификация определенной программы реализованная N-ое число раз различными способами. Те же самые входные значения предоставляются N-ому числу версий, и результаты полученные из N-ых чисел версий сопоставляется. Если результат считается действительным, результат передается на выходные данные компьютера.

N-ое количество версий могут идти параллельно на отдельных компьютерах, а также все версии одновременно могут идти на одном компьютере и результаты подвергаются внутреннему голосованию. Различные стратегии голосования могут использоваться на N-ом количестве версий в зависимости от требований приложения.

Если у системы безопасное состояние, то целесообразно требовать полного согласования (все N согласуются), в противном случае используется устойчивое к отказам выходное значение. Для простых систем аварийного отключения голосование может смещаться в сторону безопасности. В этом случае мерой безопасности будет являться отключение, если какая-либо из двух версий требует отключения. Такой принцип обычно использует две версии (N=2).

Для систем с небезопасным состоянием могут быть применены стратегии мажоритарного голосования. В случаях отсутствия коллективного согласования, могут применяться вероятностные принципы с тем, чтобы максимизировать возможность выбора

корректного значения, например, взяв срединное значение, временную заморозку выходных данных до возврата согласования, и т.д.

Этот метод не искореняет остаточные ошибки проектирования программного обеспечения, но он обеспечивает меру по выявлению и маскировке прежде чем они могут повлиять на безопасность.

Справочный материал: [29], [30].

Б.18 Динамическая реконфигурация (как указано в пункте 9)

Задачи

Сохранение фунциональности системы несмотря на внутреннюю неисправность.

Характеристика

Логическая архитектура системы должна быть такой, чтобы она могла быть спроецирована на ряд имеющихся ресурсов системы. Архитектуре необходимо уметь выявлять сбой в физическом ресурсе и затем повторно проецировать логическую архитектуру обратно на ограниченные ресурсы которые остались функционировать. Хотя концепция традиционно более ограничена восстановлением после вышедших из строя единиц аппаратного оборудования, она также применима к неисправным единицам программного обеспечения при условии достаточного 'динамического резервирования ' чтобы предоставить повторную попытку программному обеспечению, или если существует достаточно резервных данных чтобы отдельный или изолированный сбой был незначительным.

Хотя традиционно и применим к аппаратному оборудованию, этот метод разрабатывается для применения в программнои обеспечении и, таким образом, в общей системе. Он должен учитываться на первом этапе проектирования системы.

Справочный материал: [31], [32].

Б.19 Классы эквивалентности и проверка вводного сегмента (как указано в таблицаxdt2 и dt3)

Задачи

Адекавтная проверка программного обеспечения с использованием минимального набора тестовых данных. Тестовые данные получают за счет выбора сегментов входного интервала необходимого для проверки работы программного обеспечения.

Характеристика

Данная стратегия проверки основывается на эквивалентной связи входных данных, определяющей сегмент входного интервала.

Наборы тестовых данных выбирается 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 ведущих к аналогичному функциональному результату;
- классы эквивалентности могут быть определены по внутренней структуре программы. В этом случае, результаты класса эквивалетности определяются из статического анализа программы, например, набор значений ведущих к аналогичному выполняемому пути (ветви).

Справочный материал: [33].

Б.20 Коды обнаружения и устранения ошибок (как указано в пункте 9)

Задачи

Выявление и устранение ошибок в чувствительной информации.

Характеристика

Для информации n-значных битов вырабатывается кодированный блок k-значных битов, который способствует обнаружению ошибок и их устранению. Вот различные типы кода:

- коды Хэмминга;
- циклические коды;
- полиномиальные коды.

Справочный материал: [34], [35].

Б.21 Расуждения с использованием догадок для выявления ошибок (как указано в таблицах dt2 и dt8)

Задачи

Устранение общих ошибок программирования.

Характеристика

Опыт проведения тестирования и интуиция вместе со знанием и любопытством по поводу системы при тестировании могут добавить некатегоризированные тестовые данные к разработанному набору тестовых данных. Особые значения или сочетания значений могут быть склоны к ошибкам. Некоторые интересные наборы тестовых данных могут быть получены из проверочных листов проверок. Также можно рассмотреть достаточно ли устойчивая система к сбоям. Можно ли нажимать клавиши на лицевой панеле черезчур быстро или слишком часто? Что произойдет если две клавиши нажаты одновременно?

Б.22 Подсев ошибок (как указано в таблице dt2)

Задачи

Убедиться в адекватности набора тестовых данных.

Характеристика

Некоторые известные типы ошибок вносятся в программу, и программа выполняется с контрольными примерами в условиях тестирования. Если хотя бы некоторые из сеянных ошибок обнаруживаются, то набор контрольных примеров является неадекватным. Соотношение обнаруженных сеянных ошибок к общему числу сеянных 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 тестировнию.

$$\frac{\text{Обнаруженные сеянные ошибки}}{\text{Общее число сеянных ошибок}} = \frac{\text{Обнаруженные истинные ошибки}}{\text{Общее число истинных ошибок}}$$

Обнаружение всех сеянных ошибок может указывать, что либо набор тестовых данных адекватен, либо сеянные ошибки обнаружилсиь очень легко. Ограничения метода таковы, что для того, чтобы получить пригодные результаты, типы ошибок, так как и позиции посева, должны отражать статистическое распределение истиных ошибок.

Б.23 Анализ по дереву событий (как указано в Пункте 14)

Задачи

Моделирование, в виде диаграммы, последовательности событий, которая может развиться в систему после исходного события, и таким образом, установление степени серьезности последствий.

Характеристика

На верхушке диаграммы пишутся условия последовательности, относящиеся к разработке после исходного события, которое является целью анализа. Начиная при исходном событии, проводят линию к первому условию в этой последовательности. Там диаграмма разветвляется на ветки 'да' и 'нет', показывая каким образом будущие разработки зависят от условия. Для каждого из этих веток, одна продолжает идти к следующему условию подобным путем. Не все условия, тем не менее, 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 числа условий в этой последовательности.

Справочный материал: [36].

Б.24 Проверки по Фагану (как указано в таблице dt8)

Задачи

Раскрыть ошибки во всех этапах разработки программы.

Характеристика

«Формальная» проверка документов по гарантии качества направленная 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: [37].

Б.25 Програмирование утверждения сбоев (как указано в пункте 9)

Задачи

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

Характеристика

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

Например,

Утверд. <предвар. условие>; действие 1;

Действие x; утверд. <послед. условие>;

Справочный материал: [38], [39], [40].

Б.26 АПОПО (SEEA) – Анализ последствий ошибки программного обеспечения (как указано в пунктах 9, 11 и 14)

Задачи

Идентификация модулей программного обеспечения, их критичность; предложение средств выявления ошибок ПО и повышение устойчивости ПО; оценка объема подтверждения подлинности необходимая по различным компонентам ПО.

Характеристика

Анализ осуществляется в три этапа:

- определение существенно важных модулей программного обеспечения;
- определение глубины анализа (на уровне единой командной магистрали, группы команд, модуля и т.д.) необходимого для каждого модуля программного обеспечения, по его спецификации.

- анализ ошибок программного обеспечения.

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 необходый объем работ по проверке подлинности с учетом критичности каждого модуля.

АПОПО, будучи глубоким анализом выполняемым независимой группой экспертов, представляет собой действенный и доскональный метод.

Справочный материал: [41], [42], [43], [44].

Б.27 Выявление неисправностей и их диагностика (как указано в пункте 9)

Задачи

Выявление неисправностей в системе, которые могут привести к сбою, обеспечивая, таким образом, основу для контрмер в целях минимизации последствий сбоев.

Характеристика

Выявление неисправностей – это процесс проверки системы на наличие ошибочных состояний (которые вызваны, как уже говорилось ранее, неисправностью в системе/подсистеме, которая подлежит проверке). Основной целью выявления неисправностей является блокировка последствий ошибочных результатов. Систему, которая дает либо правильные результаты, либо не дает результатов вообще, называют «самопроверяющей».

Выявление неисправностей основывается на принципах резервирования (главным образом для выявления неисправностей в аппаратном оборудовании) и диверсификации (неисправности ПО). Чтобы принять решение по правильности результатов необходимо что-то наподобие голосования. Специальные применимые методы – это программирование утверждения, программирование N-го числа версии и метод подушки безопасности, а на уровне аппаратного оборудования – это сенсоры, датчики, замкнутая система автоматического регулирования, коды с контролем ошибок и т.п.

Выявление неисправностей 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 затронутыми данными чтобы обеспечить прослеживание сбоя.

Сложные системы состоят из подсистем. Эффективность выявления неисправностей, диагностики и компенсации неисправностей зависит от сложности интеракций между подсистемами, которые влияют на продвижение сигнала неисправностей.

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

Б.28 Анализ дерева неисправностей (как указано в пунктах 9 и 14)

Задачи

Содействие анализу событий, либо комбинации событий, ведущему к опасному фактору или серьезным последствиям.

Характеристика

Начиная с события, которое является непосредственной причиной возникновения опасного фактора или серьезных последствий ('конечное событие'), анализ выполняется по дорожке дерева. Комбинации причин описываются логическими операторами (и, или, и т.д.). Непосредственные причины анализируются аналогично, и далее обратно к базовым событиям где и прекращается анализ.

Метод является графическим, и используется ряд типовых символов для рисования дерева неисправностей. Он главным образом расчитан на анализ систем аппаратного оборудования, но также были попытки применить этот принцип при анализе сбоев в программном обеспечении.

Справочный материал: [45], [46], [47].

Б.29 Конечные автоматы/Диаграммы состояний (как указано в таблицах dt5 и dt7)

Задачи

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, получив ввод I, система может выполнить действие A и перейти к состоянию S2. Характеризуя действия системы для каждого ввода в каждом состоянии, мы может полностью охарактеризовать систему. Полученная в результате этого модель системы называется Конечный автомат. Он 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 измерения это состояние и ввод, а клетки матрицы содержат действие и новое состояние, полученное в результате получения ввода в данном состоянии.

Там где система сложная, или имеет естественную структуру, это можно отразить в многослойном конечном автомате (FSM). Спецификацию или проектное решение выраженное 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 состояния/ввод) и на доступность (возможно ли перейти от одного состония к другому за счет последовательности входных данных). Это важные свойства для критических систем и их можно проверить. Средства для таких проверок легко расписываются. Есть также алгоритмы, которые позволяют обеспечить автоматическое генерирование тестовых данных для проверки ввода в работу конечного автомата или оживления модели конечного автомата.

Справочный материал: [48].

Б.30 Официальные методы (ссылки в статьях 8 и 10, таблице dt5)

Задачи

Разработка программного обеспечения методом, основанном на математике. Это включает официальное проектирование и официальные методы кодирования.

Характеристика

Официальные методы обеспечивают методом разработки вида системы на определенном этапе ее спецификации, проектирования или кодирования разработки. Итоговый вид принимает математическую форму и может быть подвергнут

математическому анализу для обнаружения различных классов несовместимости или неправильности. Кроме того, данная характеристика в некоторых случаях может быть подвергнута анализу машины с точностью, подобной проверке синтаксиса исходной программы составителем или анимации для проявления различных аспектов поведения описываемой системы. Анимация может дать дополнительную уверенность в том, что система соответствует реальным требованиям в той же мере, как и официально определенным требованиям.

Официальный метод обычно предлагает систему обозначений (обычно это какая-то форма используемой дискретной математики), технологию получения характеристики в данном обозначении и различные формы анализа для проверки характеристики на другие свойства точности.

В следующих подсекциях данного библиографического справочника приводится несколько примеров Официальных методов. К описываемым Официальным методам относятся CCS, CSP, HOL, LOTOS, OBJ, временная логика, VDM and Z.

Б.30.1 CS – Расчет коммуникационных систем

Задачи

CCS - метод характеристики и обоснования изменения свойств системы согласующихся, коммуникационных процессов

Характеристика

Подобно CSP, CCS представляет собой математический расчет, имеющий отношение к изменению свойств системы. Проектирование системы моделируется в виде сети независимых процессов, оперирующих последовательно или параллельно. Процессы связываются через порты (подобные каналам CSP), коммуникация происходит, только 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 коммуникационных процессов, чье общее поведение будет соответствовать требованиям системы. Равным образом представляется возможным работать в восходящей манере, комбинируя процессы и получая свойства итоговой системы с использованием правил вывода, имеющих отношение к правилам композиции.

Справочный материал: [49], [50].

Б.30.2 CSP - Взаимодействующие последовательные процессы

Задачи

CSP представляет собой метод спецификации сопутствующих программных систем, т.е. систем взаимодействующих согласованно оперирующих процессов.

Характеристика

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

Система моделируется в виде сети независимых процессов. Каждый процесс описывается в виде всех их 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, стоящая за CSP, была непосредственно включена в архитектуру транспьютера INMOS, язык OCCAM позволяет обеспечить выполнение системы, определенной на CSP в сети транспьютеров.

Источник: [51].

Б.30.3 HOL- Логика высшего порядка

Задачи

Это официальный язык, служащий в качестве основы для спецификации и проверки аппаратного оборудования.

Характеристика

HOL (Логика высшего порядка) относится к особой логической нотации и ее системе машинного обеспечения, которые были разработаны в компьютерной лаборатории Кембриджского Университета. Логическая нотация в основном берется из Элементарной теории типов Чёрча, система машинного обеспечения основывается на системе LCF (Логика вычислительных функций).

Источник: [52].

Б.30.4 LOTOS Задачи

LOTOS представляет собой метод характеристики и обоснования изменений свойств систем согласующихся, коммуникационных процессов.

Характеристика

LOTOS (язык для спецификации временного расположения) основывается на CCS с дополнительными характеристиками соответствующих алгебр CSP и CIRCAL (расчет схемы). Он превосходит слабость 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: [53].

Б.30.5 OBJ

Задачи

Обеспечить точной системной спецификацией при помощи отзывов пользователей и проверки системы до обеспечения выполнения.

Характеристика

OBJ – это язык алгебраической спецификации. Пользователи определяют требования в отношении алгебраического уравнения. Поведенческие или конструктивные аспекты системы определяются в отношении операций, действующих по абстрактным типам 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 формальным технологиям доказательства, как и другим формальным подходам. Более того, поскольку конструктивные аспекты спецификации OBJ являются автоматически выполнимыми, они непосредственно должны выполнять проверку 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 нижележащими технологиями формальной спецификации.

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

Справочный материал: [54], [55], [56], [57].

Б.30.6 Временная логика

Задачи

Непосредственное выражение безопасности и операционных требований, а также официальная демонстрация того, что эти свойства сохраняются в последующих этапах при разработке.

Характеристика

Стандартная логика предиката первого порядка не содержит временной концепции. Временная логика протяженна до логики первого порядка через добавление модальных операторов (например, «в последующем» и «в результате»). Данные операторы могут быть использованы для квалификации тезисов по системе. К примеру, свойства целости 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 (поведение). Что составляет «состояние» - зависит от выбранного уровня описания. Это может относиться ко всей системе, компоненту системы или компьютерной программе. Количественно выраженные интервалы времени и ограничения не управляются непосредственно во временной логике. Необходимо использовать абсолютные интервалы времени путем создания состояний дополнительного времени как часть определения состояния.

Справочный материал: [58], [59], [60].

Б.30.7 VDM – Венский метод разработки

Задачи

Систематическая спецификация и выполнение последовательных программ.

Характеристика

VDM – это метод спецификации, основанный на математике, и метод для обработки реализаций, путем, позволяющим доказательство верности в отношении спецификации.

Метод спецификации основан на использовании модели в плане того, что состояние системы моделируется в отношении теоретико-множественных структур, на которых определяются инварианты (предикаты), и операции в данном состоянии моделируются определением их пре - и послеусловий в отношении состояния системы. Операции могут быть испробованы для сохранения инвариантов системы.

Обеспечение выполнения спецификации выполняется посредством конкретизации состояния системы в отношении структур, данных в итоговом языке и путем обработки операций в отношении программы в итоговом языке. Этапы конкретизации и обработки дают рост обязательствам доказательства, что устанавливает их точность. Выполняются данные обязательства или нет – это выбор проектировщика.

VDM в основном используется на этапе спецификаци, но также может быть использован на этапах проектирования реализации, ведущих к исходному коду. Он применяется только в отношении последовательных программ к последовательным процессам в сопутствующих системах.

Справочный материал: [61], [62], [63], [64].

Б.30.8 Z и B

Задачи

Z – это нотация языка спецификации для последовательных систем и проектировочный метод, позволяющий разработчику продолжать работу со спецификации Z до выполняемых алгоритмов, путем, позволяющим доказательство из точности в отношении спецификации.

Z в основном используется на этапе спецификации, но метод был разработан для

перехода из спецификации в проектирование и реализацию. Наиболее всего он подходит к разработке последовательных информационно-ориентированных систем.

В является ассоциированным методом.

Характеристика

Подобно VDM, метод спецификации основан на использовании такой модели, на которой моделируется состояние системы в отношении теоретико-множественных структур, на которых определяются инварианты (предикаты), и операции в таком состоянии моделируются определением их пре- и послеусловий в отношении состояния системы. Операции могут быть испробованы для сохранения инвариантов системы, таким образом демонстрируя их состоятельность. Формальная часть спецификации делится на схемы, позволяющие структуризацию спецификаций посредством обработки.

В основном спецификация Z представляет собой композицию официального Z и неформального объяснительного текста в 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 нотацию, чем завершенный метод. Однако был разработан ассоциированный метод (называемый B), который может быть использован в соединении с Z. Метод B основан на принципе пошаговой обработки.

Справочный материал: [65], [66], [67].

Б.31 Нормальная проба (как указано в статье 11)

Задачи

С использованием теоретических и математических моделей и правил представляется возможным испытывать точность программы без ее выполнения.

Характеристика

В разных частях программы дается ряд утверждений, которые используются в качестве пре - и послеусловий к различным путям в программе. Доказательство состоит из демонстрации того, что программа переносит пре-условия в послеусловия согласно набору логических правил и прерывания программы.

В данной библиографии описывается несколько формальных методов, к примеру, CCS, CSP, HOL, LOTOS, OBJ, Временная логика, VDM и Z. Их характеристики даны в статье B.30.

Справочный материал: [68].

Б.32 Восстановление без возврата (как указано в статье 9)

Задачи

Обеспечение правильности функционального действия при наличии одного или более дефектов.

Характеристика

При обнаружении дефекта текущее состояние системы манипулируется для получения состояния, которое позднее станет постоянным. Данная концепция особенно приемлема 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 подвергнута среде и только часть состояния системы подвержена влиянию среды.

Б.33 Плавное снижение эффективности

Задачи

Управление доступных и наиболее критических функций системы, несмотря на дефекты, вызванные опуском менее критических функций.

Характеристика

Данный метод расставляет приоритеты среди различных функций, выполняемых системой. Дизайн обеспечивает следующее: если для выполнения всех функций системы недостаточно ресурсов, выполняются функции высшего приоритета. К примеру, функции регистрации ошибок и событий могут быть менее приоритетными, чем функции контроля системы. Контроль системы будет продолжаться, если аппаратное оборудование, связанное с регистрацией ошибок, не сработает. Далее, если не срабатывает аппаратное оборудование управления системой, но при этом работает аппаратное оборудование регистрации ошибок, то аппаратное оборудование регистрации ошибок предпримет контрольные функции. Это преимущественно применяется по отношению к аппаратному оборудованию, но может применяться и к системе в целом, что должно приниматься в расчет с самого верхнего этапа проектирования.

Справочный материал: [69], [70], [71].

Б.34 Исследование рисков и эксплуатационных возможностей (HAZOP)

Задачи

Посредством серии систематических проверок компонентных секций компьютерной системы и ее работы, определить виды повреждений, ведущих к потенциально опасным ситуациям в контролируемой системе.

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

Предполагается, что опасные происшествия в контролируемых системах были определены в отдельном Анализе Рисков, где были классифицированы по степеням важности.

Анализ, затрагивающий все этапы жизненного цикла проекта, от спецификации через проектирование и до обслуживания и модификации и намерение идентифицировать виды неисправностей на каждом этапе, которые могут привести к потенциальным рискам, и, как следствие, их устранение.

Характеристика

Проводится анализ группы инженеров (затрагивающий компьютерные дисциплины, КИПиА, электрику, технологию, безопасность и эксплуатацию), возглавляемой специалистом, обученным методикам анализа рисков, посредством проведения запланированных совещаний.

Важно, чтобы был запланированный график совещаний на все время проекта; каждый из них должен быть запланирован, по крайней мере, на полдня и для эффективности их должно быть не более четырех раз в неделю, а также необходимо обеспечение соответствующей документацией.

До начала анализа должны быть составлены согласованные проверочные таблицы для систематической проверки, а также для 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 включают детальную проверку соответствующих частиц самих компьютерных систем.

В каждой части или на каждом этапе анализа целью является идентификация видов неисправностей, ведущих к потенциальным рискам в контролируемой системе и степени их действия. Большая часть составляющих частей компьютерной системы делятся далее и при необходимости подвергаются отдельному анализу.

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

Важно вести подробные записи, поскольку они формируют значительную часть досье рисков/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: [72], [73], [74], [75].

Б.35 Анализ последствий (как указано в статье 16)

Задачи

Идентификация эффекта, произведенного изменением или усовершенствованием системы программного обеспечения на другие модули системы программного обеспечения, а также на другие системы.

Характеристика

До изменения или усовершенствования программного обеспечения необходимо провести анализ для идентификации влияния модификации или усовершенствования на программное обеспечение, а также идентификации вовлеченных программных систем и модулей.

После проведения анализа необходимо решение, имеющее отношение к перепроверке программных обеспечений. Это зависит от количества вовлеченных модулей, критичности влиявших модулей и характера изменений. Возможные решения:

a) Перепроверке подвергается только измененный модуль;

b) Перепроверке подвергаются все идентифицированные вовлеченные модули;

c) Перепроверке подвергаются целая система.

Б.36 Утаивание информации/Инкапсуляция (как указано в таблице dt9)

Задачи

Повышение надежности и обслуживания программного обеспечения.

Характеристика

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

Утаивание информации – это общепринятый подход для минимизации сложностей. Структуры основных данных «спрятаны» и могут манипулироваться посредством определенного набора процедур доступа. Это позволяет внутренним структурам изменяться или дополнять дальнейшими процедурами без влияния на функциональное поведение оставшегося программного обеспечения. К примеру, 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 оставшегося программного обеспечения с использованием этих процедур.

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

Справочный материал: [76], [77].

Б.37 Проверка интерфейсов (как указано в статье 10)

Задачи

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

Характеристика

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

- всех переменных интерфейса в их крайних положениях;
- всех переменных интерфейса индивидуально 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 маленьких интерфейсов;
- определенных условий проверки релевантных к каждому переходу каждой подпрограммы.

Эти проверки представляют собой особенную важность, если их интерфейсы не содержат операторов контроля, которые обнаруживают неправильные параметральные значения, особенно они приобрели особую важность после производства новых конфигураций предсуществовавших подпрограмм.

Б.38 Языковое подмножество (как указано в статье 10 и таблице dt4)

Задачи

Уменьшение возможностей введения программных ошибок и увеличение возможностей обнаружения каких-либо оставшихся ошибок.

Характеристика

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

Б.39 Запоминание решенных проблем (как указано в статье 9)

Задачи

Усиление программного обеспечения для его отказоустойчивости в случае его выполнения нелицензированного маршрута.

Характеристика

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

Результатом индивидуальных путей «от решения к решению» (DDpaths) или же результатом индивидуальных доступов к массиву данных, записям/томам или и к тому и другому, может быть учет выполнения.

Возможны различные методы хранения путей запусков. Можно использовать методы кодирования Нэша для обозначения последовательности выполнения одним большим единственным числом или последовательностью чисел. Во время нормальной работы значение пути выполнения должно проверяться по отношение 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 одним целым может

и не представиться возможным. В этом случае возможно применение метода на модульном уровне.

Справочный материал: [78].

Б.40 Библиотека надежных/проверенных модулей и компонентов (как указано в статье 10)

Задачи

Предупреждение необходимости в перепроверке или значительном перепроектировании модулей программных обеспечений и компонентных схем аппаратного оборудования для каждой новой программы. А также благоприятствовать схемам, которые не были формально или неукоснительно утверждены, но имеющие значительную операционную историю.

Характеристика

Хорошо спроектированные и структурированные PES состоят из большого числа компонентов аппаратного оборудования и программного обеспечения и четко выраженных модулей, которые взаимодействуют друг с другом в четко определенном виде.

Различные PES, спроектированные для различия приложений, содержат ряд одинаковых или подобных модулей или компонентов. Составление каталога таких общеприменимых модулей позволяет ресурсам, необходимым для утверждения проектов, быть использованными более чем одним приложением.

Более того, использование таких модулей в многократных приложениях обеспечивает эмпирическое доказательство успешного операционного применения. Такие эмпирические доказательства увеличивают степень доверия пользователей по отношению к модулям.

Б.41 Модели Маркова (как указано в статье 14)

Задачи

Определение надежности, безопасности и доступности системы.

Характеристика

Компонуется график системы. График представляет собой состояние системы в отношении состояния ее отказа (Неисправные состояния представляются точками пересечения графика). Области между точками пересечений, представляющие неисправные состояния или случаи их исправления, загружены соответствующей частотой неисправностей или частотой исправлений. Обратите внимание, что случаи, состояния и частота неисправности детализируются таким образом, что можно получить точную характеристику системы, например, выявленные или невыявленные дефекты, описание более крупных неисправностей и т.д.

Метод Маркова приемлем для моделирования резервных систем, в которых уровень резерва варьируется со временем благодаря неисправности и исправлению компонента. Другие классические методы, как, например, FMEA и FTA, не могут быть сразу адаптированы к моделированию эффектов неисправностей за весь жизненный цикл системы, поскольку не существует простых комбинаторных формул для исчисления соответствующих возможностей.

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

Справочный материал: [79], [80], [81], [82].

Б.42 Система показателей (как указано в статьях 11 и 14)

Задачи

Предсказание атрибутов программ по свойствам программного обеспечения, но не разработки или истории проверки обеспечения.

Характеристика

Данные модели оценивают некоторые структурные свойства программного обеспечения и относят их к предпочтительным атрибутам, таким как надежность или сложность. Для оценки большинства мер необходимы программные инструменты. Ниже приводятся некоторые из рекомендуемых систем показателей:

- Графическая теоретическая сложность: данная мера может применяться на ранней стадии жизненного цикла для доступа к взаимовлиянию, и основан на сложности программного контрольного графика, представленного его цикломатическим числом;
- Количество методов для активации определенного модуля (доступность): чем доступнее будет модуль, тем скорее он будет налажен;
- Теория программного обеспечения: данная мера вычисляет длину программы путем подсчета количества операторов и операндов. Обеспечивает меру сложности и оценивает ресурсы разработки;
- Количество входов и выходов на модуль: минимизация количества входов/выходов является ключевым свойством структурного проектирования и программных методов.

Справочный материал: [83], [84], [85].

Б.43 Модульный подход (как указано в таблице dt9)

Задачи

Декомпозиция программных систем на меньшие доступные части для ограничения сложности системы.

Характеристика

Модульный подход или разбиение на модули содержит несколько правил проектирования, кодирования и обслуживания этапов проектирования программного обеспечения. Данные правила варьируются согласно методу проектирования, примененного во время проекта. 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 крепкой;
- Сборки подпрограмм должны компоноваться при условии наличия нескольких уровней модулей;
- Размеры субпрограмм должны быть ограничены до каких-то определенных значений, обычно от 2 до 4 ситового размера;
- 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 и их использование должно обосновываться в каждой инстанции;
- Все модульный интерфейсы должны быть полностью документированы;
- Каждый модуль должен утаивать что-либо из своей среды;
- Интерфейс любого модуля должен содержать минимальное количество параметров, необходимых для функций модулей,
- Необходимо определить подходящее ограничение числа параметров – обычно 5.

Б.44 Моделирование методом Монте-Карло

Задачи

Моделирование эффекта реального мира в программном обеспечении 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 стохастического феномена реального мира.
- Детерминированные проблемы, которые математически транслируются в эквивалентную вероятностную проблему.

Моделирование методом Монте-Карло вводит потоки случайных чисел для имитации шума при аналитическом сигнале или для добавления случайных систематических ошибок или толерантность. Моделирование методом Монте-Карло используется для производства больших образов, от которых получают статистические результаты.

При использовании моделирования 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 точными и возможными, чем решение проблемы в начальном виде.

Б.45 Моделирование функционирования (как указано в таблицах dt2 и dt5)

Задачи

Обеспечение такой работоспособности системы, которая бы соответствовала определенным требованиям.

Характеристика

Спецификация требований включает требования производительности и реагирования к определенным функциям, возможно объединенных с давлением при использовании всех источников системы. Проект предлагаемой системы сравнивается с определенными требованиями путем:

- Определения модели процессов системы и их взаимодействий,
- Определения использования ресурсов каждым процессом, например, время процессора, пропускная способность канала связи, устройства памяти и т.д.,
- Определения распределения требований размещенных на системе под средними и наихудшими условиями,
- Вычисления производительности средних и наихудших случаев и время реагирования для индивидуальных системных функций.

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

До детального моделирования может применяться простая проверка «ресурсного бюджета», которая суммирует требования ресурсов всех процессов. Если такие требования превышают спроектированную работоспособность системы, проект считается недопустимым.

Даже если проект проходит через эту проверку, моделирование работы может показать, что чрезмерные задержки и время реагирования происходят вследствие ресурсной перегрузки. Во избежание такой ситуации инженеры часто проектируют такие системы, в

которых возможно использование некоторых долей (например, 50 %) общих ресурсов, так что уменьшается возможность ресурсной перегрузки.

Б.46 Требования к функционированию (как указано в таблице dt6)

Задачи

Обеспечение удовлетворительных требований к функционированию программной системы.

Характеристика

Анализ выполняется по спецификациям требований системы и программного обеспечения для идентификации всех общих и особых, определенно выраженных и предполагаемых требований к функционированию.

Каждое требование к функционированию проверяется поочередно для определения:

- Получаемого критерия успеха,
- Возможности получения меры против успеха критерия,
- Потенциальной точности таких измерений,
- Этапов проекта, на которых возможна оценка измерений
- Этапов проекта, на которых возможно проведение измерений.

Целесообразность каждого функционального требования затем анализируется для получения перечня функциональных требований, критерия успеха и потенциальных измерений. К основным целям относится следующее:

a) Каждое функциональное требование связано, по крайней мере, с одним измерением;

b) По возможности, выбираются точные и эффективные измерения, которые могут быть в процессе разработки с самого начала;

c) Определяются неотъемлемые и добровольные функциональные требования и критерий успеха;

d) При возможности, извлекается выгода из возможности использования одного измерения для более чем одного требования.

Б.47 Вероятностная проверка (как указано в статьях 11 и 13)

Задачи

Получить количественное выражение свойств надежности исследуемого программного обеспечения. Данное число может относиться к соответствующим степеням достоверности и важности, а также

a) Вероятности неисправности на спрос,

b) Вероятности неисправности во время определенного периода времени

c) Вероятности ограничения распространения ошибки.

Из этих чисел возможно извлечение других параметров, таких как:

- вероятность безотказного исполнения,
- вероятность неразрушения,
- доступность,
- среднее время безотказной работы или частота неисправностей
- вероятность безопасного исполнения.

Характеристика

Вероятностные вопросы основываются на вероятностном испытании или производственном опыте. Обычно количество испытаний наблюдаемых случаев весьма велико.

Для того чтобы упростить проверку, обычно применяются автоматические пособия. Они имеют отношение к деталям обеспечения проверочных данных и контролю результатов теста. Крупные проверки применяются с крупными ведущими компьютерами с соответствующей периферией моделирующего процесса. Проверочные данные

выбираются согласно систематической, а также случайной точки зрения. Первое касается общего контроля проверки, например, гарантии профиля проверочных данных. Случайный отбор касается индивидуальных случаев проверки.

Оборудование для индивидуальных проверок, контрольных примеров, проверочных наблюдений определяются деталями сплошных проверок, описанных выше. Другие важные условия определяются через необходимую математическую подготовку, которую необходимо выполнять для того, чтобы допустить оценку проверки ввиду намеченной цели проверки.

Вероятностные числа в отношении поведения любого объекта теста также могут быть выведены из производственного опыта. При условии соблюдения аналогичных условий применяется такая же математика, как и в случае с оценкой результатов проверки.

Б.48 Имитация работы (как указано в таблице dt3)

Задачи

Проверка функционирования программной системы вместе с интерфейсом на удаленные ЭВМ, без ее допущения на модификацию реального окружения каким-либо образом.

Характеристика

Создание в целях проверки системы, воспроизводящей поведение системы, контролируемой системой в рамках проверки.

Имитация может быть представлена только программным обеспечением или комбинацией программного обеспечения и аппаратного оборудования. Она должна:

- обеспечивать все входные устройства проверяемой системы таким образом, чтобы они существовали и при монтаже системы,
- реагировать на выходные устройства из системы посредством, исправно представляющим контролируемую часть,
- иметь обеспечение вводных устройств оператора для предоставления каких-либо нарушений, с которыми проверяемой системе предстоит справиться.

Когда программное обеспечение подвергается проверке, имитация может являться имитацией объектного аппаратного оборудования с входными и выходными устройствами.

Справочный материал: [86], [87].

Б.49 Создание прототипа/анимация (как указано в таблицах dt3 и dt5)

Задачи

Проверка возможности реализации системы по отношению к данным ограничивающим условиям. Связь интерпретации описателя системы с заказчиком для определения расположения.

Характеристика

Выбирается подкомплект функций системы и ограничивающих условий и требований к функционированию. Компонуется прототип с использованием высококачественных инструментов. На данном этапе рассматриваются ограничивающие условия, такие, как объектный компьютер, язык выполнения, размер программы, сопровождаемость, надежность и доступность. Оценивается прототип по отношению к критериям пользователя и возможна модификация требований системы в свете данной оценки.

Справочный материал: [88], [89].

Б.50 Блок восстановления (как указано в статье 9)

Задачи

Увеличение вероятности программы, выполняющей свою заданную функцию.

Характеристика

Записываются несколько секций различных программ, чаще самостоятельно, каждая из которых планируется к выполнению одинаковых функции. Окончательная программа конструируется из данных секций. Первая секция, называемая первичной, выполняется в первую очередь. За ней следуют приемные испытания вычисляемого результата. Если тест проходит, принимается результат и проходит на последующие части системы. Если тест не проходит, все побочные эффекты первой секции переустанавливаются и запускается вторая секция, называемая первой альтернативой. За ней следует приемное испытание и обрабатывается как и в первом случае. При желании возможно обеспечение второй, третьей или большего числа альтернатив.

Справочный материал: [90].

Б.51 Блок-схема надежности (как указано в статье 14)

Задачи

Моделирование в схематичной форме набора событий, которые должны произойти, и условий, обязательных к соблюдению для успешного выполнения системы или задачи.

Характеристика

Цель анализа представлена в виде успешного выполнения 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 выполняется логика соединения. Если он доходит до вертекса, 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: [91], [92].

Б.52 Время реакции и ограничение памяти (как указано в таблице dt6)

Задачи

Обеспечения соответствия системы своим временным и запоминающим требованиям.

Характеристика

Спецификация требований для системы памяти программного обеспечения 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 путями, например, в сравнении с существующей системой или в прототипе и аналоге критичных по времени систем.

Б.53 Повторный запуск механизмов по устранению неисправностей (как указано в статье 9)

Задачи

Попытка функционального устранения выявленного несправного состояния при помощи повторного запуска определенных механизмов.

Характеристика

В случае выявления неисправности или состояния ошибки предпринимаются

попытки исправления ситуации путем перезапуска того же самого шифра. Восстановление путем повторного запуска может быть таким же полным, как и процедура перезапуска или повторного включения или небольшое задание перепланирования и повторного включения для выполнения задачи, после «сторожевого» действия задачи или блокировки времени программы.

Технологии повторного запуска широко используются в случаях с неисправностями связи или при исправлении ошибок и условия повторного запуска могут быть отмечены из коммуникационного протокола ошибок (контрольная сумма и пр.) или из блокировки времени реагирования на подтверждение прохождения.

Справочный материал: [93].

Б.54 Подушка безопасности (как указано в статье 9)

Задачи

Защита от остаточной спецификации и погрешностей выполнения в программном обеспечении, которое отрицательно влияют на безопасность.

Характеристика

Подушка безопасности – это внешний монитор, выполняемый на отдельном компьютере к другой спецификации. Такая подушка безопасности имеет отношение только к обеспечению выполнения основным компьютером безопасных, необязательно правильных действий. Подушка безопасности постоянно регулирует главный компьютер. Подушка безопасности предотвращает попадание системы в опасное состояние. Кроме того, если она обнаруживает, что основной компьютер входит в потенциально опасное состояние, система возвращается назад в безопасное состояние при помощи подушки безопасности или основного компьютера.

Справочный материал: [94].

Б.55 Анализ ложных цепей (как указано в таблице dt8)

Задачи

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 аппаратного оборудования, программного обеспечения, действий оператора или комбинаций этих компонентов. Ложные пути не являются результатом сбоя аппаратного оборудования, но представляют собой скрытые условия непреднамеренно спроектированные в системе или зашифрованные в программы, которые могут привести к неисправному функционированию в определенных условиях.

К категориям ложных путей относятся:

- Ложные пути, которые могут вызвать электрический ток, энергию или логическое звено для потока вдоль непредвиденного пути или в непреднамеренном направлении;
- Ложный расчет времени, в котором события происходят в неожиданном или противоречивом последствии;
- Ложные указатели, вызывающие однозначный или ложный показ действующих условий системы и таким образом способные привести к нежелательным действиям оператора;
- Ложные ярлыки, которые неправильно или неточно отмечают функции системы, например, ввод, контроль, дисплей, интерфейс системы и т.д. и таким образом способные привести оператора к заблуждению при применении неверных стимулов к системе.

Анализ ложных цепей основывается на распознавании основных топологических образцов со структурой аппаратного или программного оборудования (например, определяются шесть основных образцов для программного обеспечения). Анализ

проводится с применением пособия - проверочного листа с вопросами по использованию и отношениями между основными топологическими компонентами.

Справочный материал: [95], [96].

Б.56 Управление конфигурацией программного обеспечения (как указано в статье 15)

Задачи

Задачей управления конфигурации программного обеспечения является обеспечение последовательности групп результатов разработки при изменении результатов. Управление конфигурации, в целом, применяются по отношению и к аппаратному оборудованию и к программному обеспечению.

Характеристика

Управление конфигурацией программного обеспечения – это метод, используемый при разработке. В сущности, он требует регистрации производства каждой версии каждого "значительного" результата и каждого отношения между различными версиями различных результатов. Регистрация результатов позволяет разработчику определить эффект на другие результаты изменения по отношению к одному результату (особенно по одному из его компонентов). В частности, системы или подсистемы могут надежно быть переконструированы из последовательных наборов версий компонентов.

Справочный материал: [97], [98], [99], [100].

Б.57 Языки программирования со строгим контролем типов (как указано в статье 10)

Задачи

Уменьшение возможных неисправностей путем использования языка, предусматривающего высокий уровень проверки компилирующей программой.

Характеристика

Такие языки обычно позволяют определение типов данных, определенных пользователем, через типы данных основного языка (такого, каks INTEGER, REAL). Эти типы могут быть далее использованы точно таким же образом, как и для основных типов, но задаются строгие проверки для гарантии 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 аспекты хорошей практики проектирования программных обеспечений, такие, как легко анализируемые контрольные структуры (например IF .. THEN .. ELSE, DO .. WHILE, etc.), которые ведут к хорошо структурированным программам.

К типичным примерам языков со строгим контролем типов относятся паскаль, Ада и Модула 2.

Справочный материал: [101], [102].

Б.58 Структурно-основанные испытания (как указано в таблице dt2)

Задачи

Применение тестов, выполняющих определенные подмножества структуры программы.

Характеристика

На основе анализа программы выбирается набор вводных данных, таким образом, что выполняется большая часть элементов выбранной программы. Выполненные элементы программы могут варьироваться в зависимости от уровня необходимой строгости.

Утверждения: это наименее строгий тест, поскольку возможно выполнение всех

кодовых утверждений без выполнения обеих ветвей условного утверждения.

Ветви: необходимо проверять обе стороны каждой ветви. Это может быть непрактично для некоторых типов защитного кода.

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

LCSAJ: последовательность линейного кода и переход – это линейная последовательность кодовых утверждений, включая условные переходы, прекращенные переходом. Многие потенциальные внутренние контуры будут критичными из-за ограничений на вводные данные, наложенные выполнением кода.

Поток информации: выбирается выполняемые ветви на основе использования данных, например, ветвь, где подобные переменные могут сами писать или же быть написанными.

Граф вызова: программа составляется из подпрограмм, которые могут быть активизированы из других подпрограмм. Граф вызова представляет собой дерево активизации подпрограммы в программе. Тесты спроектированы для покрывания всех активизаций дерева.

Полный путь: запускает все возможные пути через код. Полное испытание обычно не представляется возможным вследствие огромного числа потенциальных путей.

Справочный материал: [103].

Б.59 Структурные диаграммы (как указано в таблице dt5)

Задачи

Демонстрация структуры программы посредством диаграммы.

Характеристика

Структурные диаграммы представляют собой нотацию, образовывающую дополнение к диаграммам потока данных. Они описывают программирующую систему и иерархию путей, а также отображает это графически, в виде дерева. Они документируют, как диаграмма потока данных может быть выполнена как иерархия программных единиц.

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

a) Прямоугольник, отмеченный именем единицы;

b) стрелка, соединяющая эти прямоугольники;

c) круглая стрелка, отмеченная именем данных, прошедших в и из компонентов, структурной таблицы. Обычно круглая стрелка изображается параллельно стрелке, соединяющей прямоугольники в таблице.

Из любой нетривиальной диаграммы потока данных можно вывести ряд различных структурных таблиц.

Структурные таблицы, выведенные из диаграмм потока данных представляют структуру системы первого уровня, где каждый прямоугольник структурной таблицы представляет кружок диаграммы потока данных. Само собой разумеется, что более глубокие уровни могут быть описаны с использованием такой же технологии.

Справочный материал: [104], [105], [106].

Б.60 Структурная методология (как указано в статьях 8 и 10)

Задачи

Основной целью структурных методологий является преобразование качества разработки программного обеспечения путем фокусирования внимания на ранних этапах жизненного цикла. Методы, нацеленные на достижение этого, посредством точных и интуитивно-понятных процедур и нотаций (при помощи компьютеров) для определения

существования требований и вводных свойств в логическом порядке и структурированной манере.

Характеристика

Существует ряд структурных методологий. Некоторые из них, такие, как SSADM, LBMS спроектированы для традиционной обработке данных и функций обработки транзакций, тогда как другие (MASCOT, JSD, Йордон в реальном времени) больше сориентированы на приложения контроля обработки и приложения в реальном времени (которые обычно являются более определяющими безопасность).

Структурные методы по существу являются "продуманными инструментами" для систематического восприятия и разбиения проблемы или системы. Их основные свойства - это:

- логический порядок идеи, разбивающей большую проблему на управляемые стадии;
- идентификация всей системы, включая среду и необходимую систему;
- разбивка данных и функций в необходимой системе;
- проверочные таблицы, например, перечни типов объектов, которые необходимо определить;
- низкие интеллектуальные издержки – простые, интуитивные, прагматичные.

Поддерживающие нотации больше направлены на точность в определении логического объекта проблемы и системы (например, способы обработки и потоки данных), но обрабатывающие функции, выполняемые этими, направлены на выражение с использованием неформальных нотаций. Однако некоторые методы выполняют частичное использование (математически) формальных нотаций (например, JSD использует регулярное выражение: Йордон, SOM и SDL применяют машины определенных состояний). Такая четкость не только уменьшает объем недопонимания, она обеспечивает объем для автоматической обработки.

Другим бенефитом структурированной нотации является ее визуальность, которая позволяет интуитивную проверку спецификации или проекта пользователем, против его эффективных, но неупомянутых знаний.

Данная библиография детально описывает некоторые из этих структурных методологий; например, выражение контролируемых требований, разработка системы Джексона, MASCOT, Йордон реального времени и структурный анализ и метода проектирования (SADT).

Справочный материал: [107], [108], [109].

Б.60.1 Выражение контролируемых требований (CORE)

Задачи

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 помогает процессу коммуникации – CORE больше спроектирован для выражения требований, чем для спецификаций. Структурируется подход, и представление проходит через различные уровни обработки. Процесс CORE поддерживает более широкий вид на проблему, внося знание среды, в которой будет использоваться система и различающиеся точки наблюдения различных типов пользователя. CORE включает руководство и тактику для распознающих отправлений из «великого проекта». Отправления могут быть откорректированы или точно идентифицированы и задокументированы. Таким образом, спецификации могут быть неполными, но определяются нерешенные проблемы и участки высокого риска, которые должны адресоваться в последующем проекте.

Б.60.2 JSD – разработка системы Джексона

Метод разработки, покрывающий разработку систем программного обеспечения от требований через код, со специальным акцентом на системы реального времени.

Характеристика

JSD представляет собой поэтапный процесс разработки, в котором разработчик моделирует процессы реального времени, на основе которых должны основываться функции системы, а также определяет необходимые функции и вводит их в модель и трансформирует итоговую спецификацию в одну, являющуюся выполнимой в объектной среде. Таким образом, процесс охватывает традиционные этапы определения, проектирования, и выполнения, в то же время представляет отличный вид от традиционных методов из тех, которые относятся к нисходящему программированию.

Более того, он акцентируется на раннем этапе определения сущность-объектов в реальном мире, которые представляют собой задачу строящейся системы, а также на их моделировании их и на тех результатах, которые могут произойти. Как только этот анализ «реального мира» был проведен, и модель создана, анализируются необходимые функции системы для определения того, как они могут соответствовать по отношению к модели реального мира. Получающаяся модель системы пополняется структурным описанием всех процессов в модели и затем вся модель трансформируется в программы, которые будут оперировать в итоговых средах программного обеспечения и аппаратного оборудования.

Справочный материал: [110], [111].

Б.60.3 MASCOT Задачи

Проектирование и ввод в работу систем реального времени.

Характеристика

MASCOT (Модулярный подход к компоновке, эксплуатации и испытанию программных обеспечений) – это метод разработки, поддерживаемый программирующей системы. Это систематический метод выражения структуры реального времени методом, независимым от объектного аппаратного оборудования или языка выполнения. Он задает упорядоченный подход для проектирования, который выдает высоко-модулированную структуру, обеспечивая тесное соотношение между функциональными компонентами в проектировании и компоновочными компонентами, возникающими в интеграции системы. Система проектируется через сеть параллельных процессов, связанных через каналы. Каналы могут быть банком фиксированных данных или очередями (магистралью данных). Контроль доступа к каналам определяется независимо от процессов. Он определяется через механизмы доступа, которые тоже усиливают правила оптимизации по процессам. Последние версии MASCOT были спроектированы с вводом Ada.

MASCOT поддерживает стратегию доступа, основанную на тесте и верификации единичных модулей и больших коллекций функционально соотнесенных модулей. Ввод MASCOT нацелен на компоновку, основанной на ядре MASCOT – набор планирующих базисных единиц, которые придают особое значение выводу и поддержке механизмов доступа.

Справочный материал: [112].

Б.60.4 Йордон в режиме реального времени

Задачи

Метод разработки полного программного обеспечения, состоящий из спецификации и методов проектирования, ориентированных на разработку систем реального времени.

Характеристика

Схема разработки этого метода предполагает трехэтапное 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, представляют собой необходимое поведение. Третья фаза включает фактическое поведение системы в аппаратном оборудовании и программном обеспечении. Три фазы приблизительно соответствуют обычным этапам определения, проектирования и ввода, но выделяют тот факт, что на каждом этапе разработчик вовлечен в деятельность моделирования.

Сущностная модель представляется в двух частях: модель, относящаяся к окружению и содержащая определение границы между системой и ее средой и описание внешних событий, на которые должны отвечать система, и поведенческая модель, которая содержит схемы, определяющие трансформацию, проводимую системой, в ответ на события и определение данных, которые должна содержать система для реагирования.

Модель ввода также делится на подмодели: в таком случае охватывая раскладку индивидуальных процессов к процессорам и декомпозицию процессов в модули.

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

Кроме того, метод содержит способы для стимулирования проекта предлагаемой системы, на бумаге или механически, из составленных моделей.

Справочный материал: [113].

Б.60.5 SADT – структурный анализ и метод проектирования

Задачи

Моделирование и идентификация в схематичном виде и использованием информационных стрелок, процессов принятия решений и заданий управлений, связанных со сложной системой.

Характеристика

В SADT главную роль играет концепция диаграммы фактора деятельности. Диаграмма ФД состоит из транзакций, 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 во 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 пригодны для манипуляции 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 для выполнения любой деятельности своей задачи.
- Вывод: может обозначать что угодно, производимое действием, обозначенное стрелкой из прямоугольника действия с правой стороны.

Когда действия строго соотносятся друг с другом многими факторными отношениями, возможно лучше рассматривать эти действия как неделимую группу, которая содержится в одном прямоугольнике действий, которое не отдает себя во временное пользование для дальнейшего детализирования своего содержания. Руководствующий принцип для группирования действий в один прямоугольник действий – это то, что итоговые прямоугольники, соединены парами только несколькими факторами.

Модельная иерархия диаграмм Ф/Д следуется до того момента, когда последующее детализирование прямоугольников действий становится бессмысленным. Этот этап достигается, когда действия внутри прямоугольников является неразделимыми или когда последующее детализирование прямоугольников действий выпадает за объемы системного анализа.

Справочный материал: [114], [115], [116], [117].

Б 61 Структурированное программирование (как указано в статье 10)

Задачи

Проектирование и ввод программы таким образом, чтобы сделать практичным анализ программы. Анализ должен обнаружить все значительные поведенческие программы.

Характеристика

Программа должна содержать минимум структурной сложности. Необходимо избегать сложных разветвлений. Петлевые ограничения и разветвления должны попросту быть (по возможности) соотнесены с вводными параметрами. Программа должна быть соответствующим образом поделена на маленькие модули, и взаимодействие этих модулей должно быть ясным. Свойства программирующего языка, способствующие вышеуказанному подходу, должны быть использованы в предпочтении к другим свойствам, которые (по утверждению) более эффективны, за исключением того, что такая эффективность принимает абсолютную приоритетность (например, некоторые системы определяющие безопасность).

Справочный материал: [118], [119], [120].

Б.62 Подходящие программирующие языки (как указано в таблице dt4)

Задачи

Максимальная поддержка требований международного стандарта, в частности, защитного программирования, строгого контроля типов, структурного программирования и возможных операторов контроля. Выбранный программирующий язык должен вести к легко проверяемому коду с минимумом усилий и облегчать разработку, проверку и обслуживание программы.

Характеристика

Язык должен полностью и недвусмысленно быть определяем. Язык должен быть скорее ориентирован на пользователя и проблему, чем быть машинно-ориентированным. Широко используемые языки или их подгруппы предпочтительны для языков со специальным предназначением.

Кроме того, помимо вышеназванных свойств, язык должен обеспечивать:

- Блочную структуру,
- Время трансляции,
- Тип оперативного времени и проверку границы массива,
- Проверку параметров.

Язык должен обуславливать

- Использование маленьких и управляемых модулей,
- Ограничение доступа к данным в определенных модулях,
- Определение переменных подмножества значений,
- Другие типы конструктивов ограничивающих ошибок.

Необходимо, чтобы язык поддерживался подходящим транслятором, соответствующими библиотеками предсуществующих модулей, отладчиком и средствами для контроля и разработки версии.

К свойствам, которые делают верификацию трудной, и которые, следовательно должны избегаться, относятся:

- операции безусловного перехода, исключающие переходы к подпрограмме;
- рекурсия;
- ссылки, динамическая память или любой вид динамических переменных или объекты;
- прерывание оперирования на уровне исходной программы
- множественные вводы или выходы из цепей, блоки или подпрограммы;
- начальная загрузка или декларация подразумевай величины;
- вариантная запись или эквивалентность;
- процедурные параметры.

Языки низкого уровня, в частности, входной язык ассемблера, представляют проблемы вследствие их машинно-ориентированного характера.

Б.63 Символическое выполнение (как указано в таблице dt8)

Задачи

Демонстрация согласованности между исходной программой и спецификацией.

Характеристика

Программа выполняется с замещением левой части правой частью во всех заданиях. Условные ветви и звенья транслируются в булевские выражения. Окончательный результат – это символическое выражение для каждой переменной программы. Это может быть проверено в противовес ожидаемого выражения.

Справочный материал: [121], [122].

Б.64 Временные сети Петри (как указано в таблицах dt5 и dt7)

Задачи

Моделирование относительных аспектов поведения системы, оценка и возможное улучшение требований к безопасности и оперированию посредством анализа и перепроектирования.

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 трансформаций. Позиции могут быть «помеченными» или «непомеченными». Трансформация включается, когда все позиции ввода помечены. После запуска она разрешена (но не обязана) к «пуску». Если она запускается, обозначения ввода снимаются и вместо этого помечается каждая позиция вывода от трансформации.

Потенциальные опасности в модели представляются как особые состояния (обозначения). Расширенные сети Петри позволяют моделирование временных свойств системы. Несмотря на то, что классические сети Петри концентрируются на аспектах контрольных потоков, предлагалось включить поток данных в модель некоторые расширения.

Справочный материал: [123], [124], [125], [126].

Б.65 Транслятор, испытанный в пользовании (как указано в статье 10)

Задачи

Избегать сложностей, возникающих вследствие неисправностей, которые могут возникать во время разработки, проверки и обслуживания пакета программ.

Характеристика

Используется такой транслятор, чье исправное выполнение функций доказывалось во многих проектах. Транслятор без опыта работы или имеющие серьезные ошибки запрещены.

Если транслятор продемонстрировал какой-то недостаток, относительные

конструктивы языка фиксируются и избегаются во время проекта безопасности.

Другая версия такого метода работы – это ограничение пользования языком только до его распространенных свойств.

Данная рекомендация основана на опыте многих проектов. Демонстрировалось, что незрелые трансляторы представляют собой серьезное препятствие разработке любой программы. Они делают разработку программных обеспечений связанных с безопасностью, неосуществимым.

Также известно, что в настоящее время не существует метода доказательства правильности для всех частей транслятора.

Б.66 Обзор критического анализа/проекта (как указано в таблице dt8)

Задачи

Быстрое и по возможности экономичное обнаружение ошибок в некоторых программах процесса разработки.

Характеристика

МЭК опубликовала международный стандарт 61160, который дает рекомендации к выполнению процедур по обзору проекта в качестве стимулирующей программы и для среды процесса. Он включает руководство к планированию и проведению проверок проекта, а также особые детали, касающиеся содействия специалистов, вкладывающих свою надежность, обслуживание, поддержку в техническом обслуживании и доступность. Этот стандарт также включает подразделы о вкладе других специалистов, работающих с сопутствующими предметами, такими, как качество, среда, безопасность (программы и пользователя), человеческие факторы и законные вопросы.

Кроме того, помимо общего понимания «программы» указывается включение других аспектов, например, аппаратное оборудование и программное обеспечение, перечни каталогов, спецификации, описывающие наименование, транспортные упаковки, монтаж и сборка, инструкции и руководство по эксплуатации, этикетки и предупреждения, обслуживание, перечни запасных частей, гарантии, проспекты и рекламная литература.

Критический анализ программы состоит из группы по проведению критического анализа, выбирающая небольшой набор случаев документальных проверок, представительские комплекты вводов и соответствующих предполагаемых выводов программы. Далее данные проверки мануально отслеживаются через логику программы.

Справочный материал: [127], [128].

Б.67 Нечеткая логика (как указано в статье 10)

Задачи

Нечеткая логика – это математическая дисциплина, основанная на теории размытого комплекта, допускающая степени достоверности и лжи. Это обобщение двухуровневой логики, которая предоставляет модель для обоснования. Она задействует объединение человеческой логики в автоматические системы. Путем признания сложности определения точных границ, нечеткая логика уменьшает 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 консеквенты связаны с размытыми комплектами.

ПРИМЕР

Если скорость высокая, расстояние_до_остановки среднее, ТО ускоритель около нуля и торможение нерезкое.

В данном примере, "скорость" это языковая переменная, характеризуется нечетким набором "быстроты", который может быть интерпретирован как "скорость выше, чем

приблизительно 70 км/ч". Если скорость меньше 60 км/ч, членское значение "быстроты" равно 0. Если скорость выше 80 км/ч, членское значение "быстроты" равно 1. Если скорость между 60 и 80 ккм/ч, членское значение "быстроты" варьируется между 0 и 1.

Логика, принимающая решение, основана на математических классах операторов: треугольные нормы и треугольные «со-нормы».

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 нашла большое количество приложений в сферах варьирующихся от финансов и до проектирования землетрясений. В частности, возник нечеткий контроль для контролирования высоко нелинейных систем, или систем, чьи математические свойства неизвестны или слишком сложны для аналитической обработки или систем, в которых имеющиеся измерения имеют низкое качество.

К важным приложениям 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: [129], [130], [131], [132], [133], [134].

Б.68 Объектно-ориентированное программирование (как указано в статье 10)

Задачи

Разрешить быстрое создание прототипа, облегчить многократное применение существующих компонентов программного обеспечения, достичь сокрытия информации, уменьшить вероятность ошибок во время целого жизненного цикла, уменьшить необходимые усилия в течение этапа обслуживания для разбивки сложных проблем на более легкие управляемые небольшие проблемы, уменьшить зависимости между компонентами программного обеспечения, создать более легкие расширяемые приложения.

Характеристика

Объектно-ориентированное программирование – это фундаментально новый метод мышления о программном обеспечении, основанный на абстрактностях, которые существуют в реальном мире, а не основаны на машинных абстракциях. Объектно-ориентированное программирование организует программное обеспечение как коллекцию объектов, которые объединяют и структуру, и поведение данных. Это находится в контрасте с условно приятным программированием, в котором структуры и поведение данных соединены свободным образом.

Объект: объект состоит из участка приватных данных и набора операций, так называемых методов – на данном объекте. Методы могут быть публичными и приватными. Ни для одного другого программного компонента не предусматривается прямое чтение или изменение приватных данных объекта. Любой другой программный компонент должен использовать публичные методы для этого объекта для чтения или записи данных в участке приватных данных объекта.

Класс объекта: определением класса объекта (часто в форме определения типа) вы запускаете конкретизацию многочисленных объектов такого же класса, т.е все конкретизации имеют участок приватных данных и методы, определенные в классе объекта.

(Множитель) наследование свойств: класс объекта может наследовать участок

приватных данных и методы одного (или более) суперклассов (объектных классов, находящихся выше по классу в иерархии) предусматривая то, что можно добавлять некоторые приватные данные, добавлять некоторые методы или изменять выполнение наследованных методов. Используя наследие можно построить деревья класса множественных объектов.

Полиморфизм: одна и та же операция может вести себя по-разному для разных классов объектов, например, правильная операция для конечных этапов пишет символы на этот терминал, и правильная операция на файловый объект пишет символы на данный файл.

Справочный материал: [135], [136].

Б.69 Прослеживаемость (как указано в статье 11)

Задачи

Целью прослеживаемости является обеспечение того, что все требования могут быть надлежащим образом продемонстрированы для их же удовлетворения, а также того, чтобы не было ни одного непрослеживаемого материала.

Характеристика

Прослеживаемость требований должна быть важным учетом в обосновании системы, а также необходимо обеспечение средства для обеспечения демонстрации на всех этапах жизненного цикла.

Прослеживаемость считается применимой к функциональным и нефункциональным требованиям и особенно должна быть направлена на:

a) Прослеживаемость требований к проекту или другим объектам, которые выполняют эти требования,

b) Прослеживаемость проектных объектов к объектам выполнения, которые их подвергают обработке,

c) Прослеживаемость требований и проектных объектов к операционным и обслуживающим объектам, необходимых быть примененными в безопасном и надлежащем использовании системы,

d) Прослеживаемость требований, объектов проекта, выполнения, операции и обслуживания к верификации проверочным планам и спецификациям, которые определят их приемлемость,

e) Прослеживаемость верификационных и проверочных планов и спецификаций к проверочным или другим отчетам, которые записывают результаты их приложений.

В случаях, когда требования, проект или другие объекты подвергаются обработке как ряд отдельных документов, прослеживаемость должна обслуживаться в пределах документальных структур и в иерархической манере.

Вывод процесса прослеживаемость должен представлять собой субъект формального управления конфигурации.

Приложение
(справочное)

Библиография

[1] МЭК 60050-191 Международный электротехнический словарь. Глава 191: Надежность и качество услуг.

[2] IEEE STD 610.12, стандарт IEEE глоссарий терминов технологии программирования.

[3] ИСО/МЭК 2382 (все части, информационные технологии). *Словарь.*

[4] ИСО/МЭК 9126 (все части). Техника программного обеспечения. Качество продукции.

[5] МЭК 62278-2002 Железные дороги. Технические условия и демонстрация надежности, эксплуатационной готовности, ремонтопригодности и безопасности.

[6] DIN V ENV 50129-1999 Железные дороги. Электронные сигнализационные системы безопасности.

[7] Проверка подлинности. Практические проблемы Б. Уэбб и Д. Маннеринг, SARSS 87, ноя. 1987, Алтриншем, Англия, Элсевьер Эшлайд Сайенс, ISBN 1-85166-167-0, 1987

[8] Опыт в проектировании и подлинности программного обеспечения для системы защиты реактора. С. Болонья, Э. Агостино *et al.,* семинар IFAC, SAFECOMP 1979, Штутгарт, 16-18 мая 1979, Пергамон Пресс 1979

[9] Искусство испытания программного обеспечения. Г. Майерс, Уили энд Санз, Нью-Йорк, США, 1979.

[10] Метод причинно-следственной диаграммы как основа для количественного анализа происшествий. Д. Нилсен, Riso-M-1374, 1971.

[11] Набор для подтверждения подлинности по Pascal. Ю Кей дистрибьютор: Гарантия качетсва по BSI, post box 375, Милтон Кейнез, MK14 6LL.

[12] Набор для подтверждения по Ada. Ю Кей дистрибьютор: Национальный расчетный центр (NCC), Оксфорд Роуд, Манчестер, Англия.

[13] Программируемые электронные системы в приложениях связанных с безопасностью. Health and Safety Executive, Издательство Ее Величества, Лондон 1987.

[14] Рекомендации по оценке безопасности и надежности высоко интегрированных промышленных компьютерных систем. EWICS TC7, WP 489, 6 окт. 1987.

[15] Искусство испытания программного обеспечения. Г. Майерс, Уили энд Санз, Нью-Йорк, США, 1979.

[16] Программное обеспечение для компьютеров в системах безопасности на атомных электростанциях. IEC 60880, 1986.

[17] Обзор сбоев вызванных общей ошибкой. И. Уотсон, UKAEA, Центр надежности системы, Уигшо Лейн, WA3 4NE, Англия, NCSR R 27, июль 1981.

[18] Сбои общего характера в системах резервирования. И.Уотсон и Г. Эдвардс Ядерная технология Том. 46, De. 1979.

[19] Программируемые электронные системы в приложениях связанных с безопасностью. Health and Safety Executive, Издательство Ее Величества.

[20] RXVP80 - Системы проверки и подтверждения подлинности для FORTRAN. Руководство для пользователя. Дженерал ресерч коропрейшн, Санта Барбара, Калифорния, США.

[21] Поток информации и поток данных программ повторения. Дж. Бергеретти и Б. Каррe, ACMTrans. on Prog. Lang. and Syst., 1985.

[22] Проектирование программного обеспечения. И. Сомервиль, Эдисон-Уэсли 1982.

ISBN 0-201-13795-X.

[24] Зависимость критических компьютерных систем - Часть 1. Е. Редмил (ed) Элсевьер Эшлайд Сайенс 1988.

[25] Зависимость критических компьютерных систем - Часть 2. Е. Редмил (ed) Элсевьер Эшлайд Сайенс 1988.

[26] Аспекты проектирования программного обеспечения в рамках концепций программирования в режиме реального времени. Е.Шойтц, Физические связи компьютера 41 (1986) Северная Голландия, Амстердам.

[27] Гарантоспособные вычисления: От концепции к диверсификации проектных решений. А. Авизинис, Дж. Лапри, Proc. IEEE, том. 74, 5, май 1986.

[28] Теоретическая основа анализа многоверсионного программного обеспечения подвергающегося совпадающим сбоям. Д. Экхартанд Л. Ли, IEEE Trans. SE-11, No. 12, 1985

[29] Критические вопросы при проектировании реконфигурируемого управляющего компьютера. Х. Шмид, Дж. Лам, Р. Наро и К. Вейр, FTCS 14 июнь 1984, IEEE 1984.

[30] Определение процессов для процессоров: Подход устойчивый к ошибкам. июнь 1984, Г. Кар и К. Николау, НИЦ Уотсон, Йорктаун

[31] Искусство испытания программного обеспечения. Г. Майерс, Уили энд Санз, Нью-Йорк, США, 1979.

[32] Технология устраняющих ошибки кодов. Е. Берлекамп, Proc. of the IEEE, 68, 5, 1980.

[33] Краткий курс по устраняющим ошибки кодам. Н. Слоан, Спрингер-Верлаг, Вена, 1975.

[34] Дерева событий и их обработка на персональных компьютерах. Н. Лимниус и Дж. Жанет, Техника обеспечения надежности, том 18, No. 3 1987.

[35] Проверки проектирования и кодирования для сокращения количества ошибок в разработке программы. М. Фаган, Журнал систем IBM, No. 3, 1976.

[36] Дисциплина программирования. Е.Дижкстра, Прентис-Холл, 1976.

[37] Наука программирования. Д. Гриес, Спрингер-Верлаг, 1981.

[38] Разработка программного обеспечения – Строгий подход. С. Джоунс, Прентис-Холл, 1980.

[39] Постоянные железнодорожные устройства и подвижной состав. Обработка данных. Функциональная надежность программного обеспечения – Адаптированные методы анализа безопасности ПО, французский стандарт NF F 71-013, дек. 1990.

[40] "Режимы сбоя программного обеспечения и анализ последствий" Транзакция на надежности IEE - Том. R28 No. 3 авг. 79 – сс. 247/249

Дж. Фрагола и Дж. Спамм

[41] "Анализ последствий ошибки программного обеспечения, средство качественного проектирования "

[42] IEEE Симпозиум « Компьютер с надежностью программного обеспечения»

1973-сс. 90/93

[43] Методика по технике обеспечения надежности системы: Обсуждение последних достижений. Дж. Фуселанд, Дж. Аренд, Ядерная безопасность 20(5), 1979.

[44] Руководство по дереву неисправностей. В. Весли, *et al.*, NUREG-0942, Отдел управления по безопасности систем, Положение по ядерным реакторам, США. Надзорная ядерная комисиия, Вашингтон, округ Колумбия 20555, 1981.

[45] Технология надежности. А. Грин и А. Борн, Вили - Интерсаейнс, 1972.

[46] Введение в теорию машин с конечным числом состояний. А. Джилл, МакГроу-Хилл, 1962.

[47] Расчет коммуникационных систем. Р. Милнер, Отчет № ECS-LFCS-86-7, Лаборатория для основ компьютерных наук, Департамент компьютерных наук, Университет Эдинбурга, Великобритания.

[48] Спецификация сложных систем. Б. Коэн, В.Т. Харвуд, М.И. Джексон. Эдисон-Уэсли, 1986.

[49] Взаимодействующие последовательные процессы. CAR. Хоар, Прэнтис-Холл, 1985

[50] HOL: Машинно-ориентированная формулировка логики высшего порядка. М. Гордон, Технический отчет Кембриджского университета, № 68.

[51] Системы обработки информации – Внутренняя связь открытых систем - LOTOS – Официальный метод описания, основанный на Временном Расположении Наблюдаемого Поведения. ISO/DIS 8807, 20 июля, 1987 г.

[52] Введение в OBJ; Язык для написания и проверки спецификаций. Дж.А. Гогуэн и Дж. Тардо, Спецификация надежных программных обеспечений, ИИЭЭ Пресс 1979, переиздано в Технологии программных спецификаций, Н. Гехани, А. Макгетрик (eds), Эдисон-Уэсли, 1985.

[53] Алгебраические спецификации для производства практических программных обеспечений. К. Раттрэй, Коган Пресс, 1987.

[54] Алгебраический подход к стандартизации и сертификации графических программных обеспечений. Р. Гнатц, Форум по компьютерной графике 2(2/3), 1983.

[55] Руководство по DTI STARTS. 1987, NCC, Оксфорд Роуд, Манчестер, Великобритания.

[56] Временная логика программ. Ф. Крогер EATCS Монографии по компьютерным наукам, Том 8, Спрингер Верлаг, 1987.

[57] Проектирование для безопасности с использованием Временной Логики. Дж. Горски. SAFECOMP 86, Сарлат Франция, Пергамон Пресс, октябрь 1986.

[58] Виды логики для компьютерного программирования. Д. Габэй. Эллис Хорвуд.

[59] Разработка программных обеспечений – Точный подход. К.Б. Джоунз. Прэнтис-Холл, 1980.

[60] Формальная спецификация и разработка программных обеспечений. Д. Бьорнер и К.Б. Джоунз. Прэнтис-Холл, 1982.

[61] Систематическая разработка программных обеспечений с использованием VDM. К.Б. Джонс. Прэнтис-Холл, 1986.

[62] Спецификация сложных систем. Б. Коэн, В.Т. Харвуд и М.И. Джексон. Эддисон-Уэсли, 1986.

[63] Нотация Z – Ссылочное руководство. Дж. М. Спиви, Прэнтис Холл, 1988. Анализ проблем спецификаций. Редактировано И. Хэйз, Прэнтис Холл, 1987.

[64] Спецификация файлового хранилища UNIX. К. Морган и Б. Суфрин. Транзакции ИИЭЭ по проектированию программных обеспечений, SE-10, 2 марта 1984.

[65] Книга B – Определение значений программ. Дж. Р. Абриал, Объединенная пресса Кембриджа, 1996.

[66] Можно ли сделать доказательство правильности программ практичным. Дж. Даль, Исследовательский отчет, ISBN 82-90230-26-5 No. 33, Осло, май.

[67] Программные обеспечения для Шаттлов. К.Т. Шеридан, Автоматическая обработка данных, Том 24, июль 1978.

[68] Развитие отказоустойчивых вычислений. Том 1. Гарантоспособные вычисления и отказоустойчивые системы, редактировано А. Авизьенис, Х. Копец и Дж. К. Лэпри, Спрингер-Верлаг, ISBN 3-211-81941-X, 1987.

[69] Отказоустойчивость, принципы и практики. Том 3 из [2] вышеуказанного издания, Т. Андерсон и П.А. ЛИ, Спрингер-Верлаг, ISBN 3-211-82077-9, 1987.

[70] HAZOP и HAZAN. Т.А. Клетц, 1986, 2-е издание, Институт инженеров-химиков, 165-171 Рэйлуэй Террас, Рагби, CV1 3HQ, Великобритания.

[71] Критерии надежности и рисков для программирующихся электронных систем в химической лаборатории. Е. Джонсон, Заседание 3-го симпозиума по безопасности «Безопасность и надежность PES», PES, Б.К Дэниелз (ed), 28-30 май 1986, Гернсей Чаннел Айлэндс, Прикладные науки Элсевир, 1986.

[72] Техника обеспечения надежности и оценка рисков. И. Дж. Хентли и Х. Кумамото Прэнтис Холл, 1981.

[73] Надежность систем и анализ рисков. И.Дж. Френкель, Мартинус Нийхог 1984.

[74] Программирование: Планирование изменений, Д.А. Ламб, Прэнтис Холл, 1988.

[75] О проектировании и разработке программы для обработки семейства деталей, Д.Л. Парнас, Протоколы ИИЭЭ SE-2, март 1976.

[76] Отказоустойчивые программные приложения – Некоторые принципы и анализ проблемы. В. Эренебергер, заседания SARSS 1987, Элтрингем, Манчестер, Великобритания, Прикладные науки Эльзевира, 1987.

[77] Теория стохастических процессов. Р.Е Кокс и Х.Д. Миллер, Метуэн и Ко. Лимитед, Лондон, Великобритания 1968.

[78] Конечные цепи Маркова. Дж. Г. Кемени и Дж. Л. Снэлл, Д. Ван Нострандд Компани Инк., Принстон, 1959.

[79] Справочник по надежности. Б.А. Козлов и Л.А. Ушаков, Холт Райнхарт и Уинстон Инк., Нью-Йорк, 1970.

[80] Теория и практика проектирования надежных систем. Д.П. Сивьёрк и Р.С. Сварц, Диджитал Прэсс 1982.

[81] Критерии сложности, Т. Дж. МакКейб, Протоколы ИИЭЭ по разработке программных обеспечений, Том. SE-2, No. 4, декабрь 1976.

[82] Модели измерения для оценки качества программных обеспечений. С.Н. Моханти, Исследования Ассоциации по вычислительной технике по обработке данных, том 11, No. 3, сентябрь 1979.

[83] Компоненты теории программного обеспечения. М.Х. Хадстед, Эльзевир, Северная Голландия, Нью-Йорк, 1977.

[84] Имитатор программного обеспечения – Пособие для проведения пусконаладочных работ завода. С. Наннз, EWICS TC7.

[85] Имитация физических неисправностей. Ф. Моррилон, EWICS Номер документа WP460.

[86] Протоколы рабочей конференции по созданию прототипа. Намур, окт. 1983, Буддэ *и другие,* Спрингер-верлаг, 1984.

[87 Использование языка выполняемой спецификации для информационной системы. С. Урбан *и другие,* Протоколы ИИЭЭ по разработке программных обеспечений, том SE-11 No. 7, июль 1985.

[88] Системная структура для отказоустойчивости программы. Б. Рэндалл, Протоколы ИИЭЭ по разработке программных обеспечений, том SE-1, No. 2, 1975.

[89] Методология разработки надежности программы: Разделение состояния развития. Дж.Б. Фасслэнд, Дж.С. Арэнд, Ядерная безопасность 20(5), 1979.

[90] Справочник по дереву ошибок. В.Е. Весли и др., NUREG-0942, Разделение офиса системы безопасности при регулировании ядерного реактора, Американская комиссия по ядерному регулированию, округ Колумбия 20555, 1981.

[91] Теория и практика проектирования надежных систем. Д.П. Севьёрек и Р.С. Шварц, Диджитал Пресс.

[92] Использование всех технологий для улучшения безопасности программных

обеспечений. Протоколы IFAC SAFECOMP 88, Сарлат, Франция, октябрь 1986, Пергамон Пресс 1986.

[93] Ложный анализ и ложный анализ программных обеспечений. С.Г. Годой и Г. Дж. Энгельс, Дж. Эйркрафт, том 15, No. 8, 1978.

[94] Анализ ложных цепей. Дж. П. Ранкин, Ядерная безопасность, том 14, No. 5, 1973

[95] Практика управления конфигурацией для систем, оборудования, боевой техники и компьютерных программ. MIL-STD-483.

[96] Управление конфигурацией программного обеспечения. Дж.К. Бакл, Макмиллан Пресс, 1982.

[97] Управление конфигурацией программного обеспечения. В.А. Бабич, Эдисон-Уэсли, 1986.

[98] Требования к управлению конфигурацией для оборонного оборудования. Стандарт Министерства Обороны Великобритании 05-57 Издание 1, 1980.

[99] Справочное руководство для языка программирования Ада, ANSI/MIL-STD-815A 1983.

[100] В поисках эффективного разнообразия: изучение шести языков отказоустойчивых программных обеспечений для воздушного управления, А. Авизьенис, М.Р. Лью, В. Шутц, 18-й международный симпозиум по отказоустойчивому вычислению, Токио, Япония 27-30 июня 1988, ИИЭЭ Прессы компьютерного общества, ISBN 0-8186-0867-6.

[101] Надежность испытательной стратегии анализа путей. В. Хауден, Протоколы ИИЭЭ Разработка программных обеспечений, том SE 3, 1976.

[102] Разработка программных обеспечений. И. Соммервиль, Эдисон-Уэсли 1982. ISBN 0-201-13795-X.

[103] Структурное проектирование. Л.Л. Константин и И. Ердон, Энглвуд Клиффс, Нью-Джерси, Прэнтис-Холл, 1979.

[104] Надежное программное обеспечение через композитный дизайн. Г. Дж. Майерс, Нью-Йорк, Ван Нострандт, 1975.

[105] Структурная разработка для систем реального времени (3 тома) П.Т. Йордон Пресс, 1985.

[106] Анализ важных систем. Ст.М. Макменамин, Ф Палмер, Йордон Инк., 1984. Нью-Йорк.

[107] Использование структурных методов в разработке больших программно-реализованных радиоэлектронных систем. Д.Дж. Хэйтли, Заседания DASC, Балтимор, 1984.

[108] Обзор JSD. Дж. Р. Камерон, Протоколы ИИЭЭ SE-12, по. 2, февраль1986.

[109] Разработка систем. М. Джексон, Прэнтис-Холл, 1983.

[110] Руководство для пользователя MASCOT 3. Форум пользователей MASCOT, RSRE, Мальверн, Англия, 1987.

[111] Структурная разработка систем реального времени (3 тома) П. Т. Уорд и С. Дж. Меллор. Йордон Пресс, 1985.

[112] Структурный анализ для определения требований, Д. Т. Росс, К.Е. Шоман младший, протоколы ИИЭЭ по разработке программных обеспечений, том SE-3, 1977, 6-15.

[113] Структурный анализ (СА): Язык коммуникационных идей, Д.Т. Росс, протоколы ИИЭЭ по разработке программных обеспечений, том. SE-3,1, 16-34.

[114] Приложения и Расширения SADT, Д.Т. Росс, компьютер, апрель 1985, 25-34.

[115] Структурный анализ и метод проектирования – Приложение по безопасным системам , В. Хайнц, Оценка риска и программное обеспечение программированного обучения, модуль В1, глава 11. 1989, Дельфтский университет технологий, научная группа по безопасности, почтовый ящик 5050, 2600 Великобритания, Дельфт,

Нидерланды.

[116] Алгоритм программирования. И.В. Дижкстра, Энглвуд Клиффс, Нью-Джерси, Прэнтис Холл, 1976.

[117] Оценка класса программных средств. М. А. Хеннел и др., 7-я международная конференция по разработке программных обеспечений, март 1984, Орландо.

[118] Программное средство для нисходящего программирования. Д. К. Инс, Программное обеспечение – практика и опыт, том 13, No. 8, август 1983.

[119] Формальная проверка программы с использованием символического выполнения. Р.Б. Данненберг и Г.В. Эрнст, Протоколы ИИЭЭ по разработке программных обеспечений, том SE-8, No 1, 1982.

[120] Символическое выполнение и проверка программных обеспечений. Дж. К. Кинг, замечания. ACM, том 19, No. 7, 1976.

[121] Теория и приложения сети. В. Брауэр (издатель), лекционные записи в компьютерной науке, том 84, Спрингер 1980.

[122] Теория сети Петри и моделирование систем. Дж. Л. Питерсон, Прэнтис-Холл, 1981.

[123] Анализ безопасности с использованием сетей Петри. Н. Левесон и Дж. Стользи, заседание FTCS 15, Энн. Эрбор, Мичиган, июнь 1985, ИИЭЭ 1985.

[124] Средство для спецификации требований и анализа программных обеспечений реального времени на основе временных сетей Петри. С. Болонья, Ф. Писакейн, К. Геззи, Д. Мандриоли, заседание SAFECOMP 88, 9-11 ноября 1988. Фульда Фед, Германия 1988.

[125] Искусство проверки программного обеспечения. Г. Майерс, Уили и сыновья, Нью-Йорк, 1979.

[126] Международный стандарт МЭК 61160: Формальная проверка проекта (1992).

[127] Нечеткие множества: Задех. Информация и контроль, 1965, том 8, стр. 338-353

[128] Нечеткая логика в системах контролирования: контроллер нечеткой логики, часть I и II. Хуэн Чьен Ли. ИИЭЭ Транзакции по системам, человек и кибернетика, том. 20, No. 2, март-апрель 1990.

[129] Промышленные приложения нечеткого контроля: М. Сугено Эд, Амстердам, Северная Голландия, 1985.

[130] Системы управления автоматических поездов предикативным нечетким контролем: Ясунобу, Миямото, в промышленных приложениях нечеткого контроля, М. Сугено Эд, Амстердам, Северная Голландия, 1985.

[131] Автоматический метод управления для регулирующих стержней на заводах по производству легководного ядерного кипящего реактора: Киношита, Фукусаки, Сато, Мияке, заседания. Совещания специалистов по внутриреакторному оснащению и оценке активной зоны ядерного реактора. Кадараш, Франция, 1988.

[132] Использование систем, основанных на правилах, для технологического контроля. Бернард. ИИЭЭ Contr. Syst. Mag., том 8, No. 5, стр. 3-13, 1988.

[133] Составление программно-ориентированных программных обеспечений, Бертранд Мейер, Англия: Прэнтис-Холл Интернэшнл, 1988.

[134] Классификация как парадигма для вычислений, Питер Вагнер, Технический отчет CS-86-11, Университет Браун, май 1986.

[135] Изучая язык, Питер Вегнер, Байт (издание МакГрой-Хилл), март 1989.

[136] Все OOPSLA (системы объектно-ориентированного программирования, языки и приложения) и заседания конференции ECOOP (Европейская конференция по объектно-ориентированному программированию).

УДК 629.4.018 **МКС 45.060.10**

Ключевые слова: Комплексное программное обеспечение, спецификация, управление по проектированию, устойчивость к ошибкам, встроенное программное обеспечение, логический программируемый контролер, программное обеспечение, предотвращение неисправностей, подтверждение подлинности, прослеживаемость, надежность.

Басуға _____ж. қол қойылды Пішімі 60х84 1/16
Қағазы офсеттік. Қаріп түрі «KZ Times New Roman»,
«Times New Roman»
Шартты баспа табағы 1,86. Таралымы ____ дана. Тапсырыс ____

«Қазақстан стандарттау және сертификаттау институты»
республикалық мемлекеттік кәсіпорны
010000, Астана қаласы,
Есіл өзенінің сол жақ жағалауы, Орынбор көшесі, 11 үй,
«Эталон орталығы» ғимараты
Тел.: 8 (7172) 240074