ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р МЭК 60880— 2010

АТОМНЫЕ ЭЛЕКТРОСТАНЦИИ

Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории А

IEC 60880:2006
Nuclear power plants —
Instrumentation and control systems important for safety —
Software aspects for computer-based systems performing category A functions
(IDT)

Издание официальное

Москва
Стандартинформ
2011

Предисловие

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

Сведения о стандарте

1 ПОДГОТОВЛЕН на основе аутентичного перевода на русский язык стандарта, указанного в пункте 4, который выполнен Открытым акционерным обществом «Всероссийский научно-исследовательский институт атомных электростанций» (ОАО «ВНИИАЭС») и Автономной некоммерческой организацией «Измерительно-информационные технологии» (АНО «Изинтех»)

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 322 «Атомная техника»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 30 ноября 2010 г. № 739- ст

4 Настоящий стандарт индентичен международному стандарту МЭК 60880:2006 «Атомные электростанции. Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории А» (IEC 60880:2006 «Nuclear power plants — Instrumentation and control systems important for safety — Software aspects for computer-based systems performing category A functions»).

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

5 ВВЕДЕН ВПЕРВЫЕ

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

© Стандартинформ, 2011

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

Введение

a) Техническое обоснование, основные проблемы и организация настоящего стандарта

Проектирование систем контроля и управления, основанных на программном обеспечении и используемых для целей ядерной безопасности, является сложной задачей из-за требований безопасности, которые должны быть выполнены. Программное обеспечение безопасности, используемое на атомных электростанциях (АЭС), которое часто требуется только в аварийных ситуациях, должно пройти полную валидацию и аттестацию перед его применением в эксплуатации. Для достижения необходимой высокой надежности систем контроля и управления специальные меры должны приниматься в течение всего жизненного цикла программного обеспечения, начиная с формулирования базовых требований на различных фазах проектирования и кончая процедурами верификации и валидации для эксплуатации и обслуживания. Целью настоящего стандарта является освещение соответствующих аспектов безопасности и обеспечение требований для достижения высокого качества программного обеспечения.

Настоящий стандарт разработан на основе использования и обобщения опыта применения его первого издания, которое было выпущено в 1986 г. с целью интерпретации принципов безопасности, применявшихся ранее к аппаратным системам с жесткими связями, применительно к цифровым системам — многопроцессорным распределенным системам, а также к системам более высокого уровня с центральным процессором, включенных в системы безопасности атомных электростанций.

Данное издание широко использовалось в атомной промышленности для определения требований и как руководство в отношении программного обеспечения систем контроля и управления, важных для безопасности атомных электростанций.

Настоящий стандарт предназначен для использования разработчиками систем, подразделениями АСУТП АЭС, экспертами и лицензирующими организациями.

b) Место настоящего стандарта в структуре серии стандартов подкомитета 45А

Непосредственные ссылки на МЭК 60880 имеются в МЭК 61513, который рассматривает вопросы системных аспектов и высокоинтегрированного компьютерного контроля и управления, используемого в системах безопасности атомных станций.

МЭК 60880 является документом второго уровня по классификации ПК 45А и касается аспектов программного обеспечения систем контроля и управления, выполняющих функции категории А.

Требования к программному обеспечению, связанному с выполнением функций категорий В и С, описаны в МЭК 62138.

МЭК 60880 и МЭК 62138 в целом охватывают аспекты программного обеспечения компьютерных систем, используемых на атомных станциях для выполнения функций, важных для безопасности.

Настоящее второе издание МЭК 60880 должно использоваться совместно с МЭК 60987 и МЭК 61226, а также с серией стандартов ПК 45А по техническому обеспечению и стандартами по классификации.

Более полная информация о структуре серии стандартов ПК 45А приведена в пункте d) настоящего введения.

c) Рекомендации и ограничения по использованию настоящего стандарта

Важно отметить, что настоящий стандарт не устанавливает дополнительных требований к системам безопасности.

Специальные требования и рекомендации касаются следующих аспектов:

1) общий подход к разработке программного обеспечения, обеспечивающего его высокую надежность, включая надежность взаимосвязей между программным и техническим обеспечениями;

2) общий подход к верификации программного обеспечения и валидации программных аспектов компьютерных систем;

3) метод управления модификацией и конфигурацией программного обеспечения;

4) требования к использованию инструментальных программ;

5) методы оценки соответствия ранее разработанного программного обеспечения.

Общепризнанным является тот факт, что технология программного обеспечения продолжает развиваться быстрыми темпами, поэтому невозможно включить в настоящий стандарт все ссылки на современные методы и технологии проектирования.

Для обеспечения пригодности настоящего стандарта и в будущем основное внимание сосредоточено на принципиальных вопросах, а не на конкретных технологиях программного обеспечения.

При разработке новых технологий будет возможность оценки пригодности этих технологий, применяя принципы безопасности, содержащиеся в настоящем стандарте.

d) Описание структуры серии стандартов ПК 45А и их взаимосвязи с другими документами МЭК (МАГАТЭ и ИСО)

Документом высшего уровня серии стандартов ПК 45А является МЭК 61513. Этот стандарт касается требований к системам контроля и управления, важных для безопасности атомных станций (АС), и лежит в основе серии стандартов ПК 45А.

В МЭК 61513 имеются непосредственные ссылки на другие стандарты ПК 45А по общим вопросам, связанным с категоризацией функций и классификацией систем, оценкой соответствия, разделением систем, защитой от отказов по общей причине, аспектами программного и технического обеспечений компьютерных систем и проектированием пунктов управления. Стандарты, на которые имеются непосредственные ссылки, следует использовать на втором уровне совместно с МЭК 61513 в качестве согласованной подборки документов.

К третьему уровню серии стандартов ПК 45А, на которые в МЭК 61513 нет непосредственных ссылок, относятся стандарты, связанные с конкретным оборудованием, техническими методами или конкретной деятельностью. Обычно документы, в которых по общим вопросам имеются ссылки на документы второго уровня, могут использоваться самостоятельно.

Четвертому уровню, продолжающему серию стандартов ПК 45А, соответствуют технические отчеты, не являющиеся нормативными документами.

Для МЭК 61513 принята форма представления, аналогичная форме представления базовой публикации по безопасности МЭК 61508, с его структурой общего жизненного цикла безопасности и структурой жизненного цикла системы; в нем приведена интерпретация общих требований МЭК 61508-1, МЭК 61508-2 и МЭК 61508-4 для применения в ядерной области. Согласованность с этим стандартом будет способствовать соответствию требованиям МЭК 61508, интерпретированным для ядерной области. В этой структуре МЭК 60880 и МЭК 62138 соответствуют МЭК 61508-3, применительно к ядерной области.

В МЭК 61513 приведены ссылки на стандарты ИСО, а также на документ МАГАТЭ 50-С-QA по вопросам, связанным с обеспечением качества.

В серии стандартов ПК 45А последовательно реализуются и детализируются принципы и базовые аспекты безопасности, предусмотренные правилами МАГАТЭ по безопасности атомных электростанций, а также серией документов МАГАТЭ по безопасности, в частности требованиями NS-R-1 «Безопасность атомных электростанций: Проектирование» и руководством по безопасности NS-G-1.3 «Системы контроля и управления, важные для безопасности атомных электростанций». Термины и определения, применяемые в стандартах серии ПК 45А, согласованы с терминами и определениями, применяемыми в МАГАТЭ.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

АТОМНЫЕ ЭЛЕКТРОСТАНЦИИ

Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории А

Nuclear power plants. Instrumentation and control systems important for safety. Software aspects for computer-based systems performing category A functions

Дата введения — 2012 — 01— 01

1 Область применения

Настоящий стандарт устанавливает требования к компьютерным системам контроля и управления атомных электростанций, выполняющим функции категории А, определенные в МЭК 61226.

В соответствии с определением, приведенным в МЭК 61513, системы контроля и управления класса безопасности 1 предназначены, главным образом, для поддержания функций категории А, однако они могут также поддерживать функции более низких категорий. Тем не менее, требования к системе всегда определяются выполняемыми функциями наивысших категорий.

Для программного обеспечения систем контроля и управления, выполняющих на АЭС функции только категорий В и С, в соответствии с определениями МЭК 61226 применяют требования и рекомендации МЭК 62138.

Цель требований настоящего стандарта состоит в разработке программного обеспечения высокой степени надежности. Требования настоящего стандарта относятся к каждому этапу разработки программного обеспечения и документации, включая спецификацию требований, проектирование, разработку, верификацию, валидацию и эксплуатацию.

В основу этих требований при разработке положены следующие принципы:

- наилучшая установившаяся практика;
- методы проектирования сверху вниз;
- модульность;
- верификация на каждом этапе;
- четкая документация;
- легко проверяемая документация;
- валидационные тестирования.

Дополнительные указания и информация в обеспечении требований основной части настоящего стандарта приведены в приложениях А - I.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие международные стандарты:

МЭК 60671 Периодические тестирования и контроль системы защиты ядерных реакторов (IEC 60671, Periodic tests and monitoring of the protection system of nuclear reactors)

МЭК 61069-2:1993 Измерение и управление промышленным процессом. Определение свойств системы с целью ее оценки. Часть 2: Методология оценки (IEC 61069-2: 1993, Industrial-process measurement and control — Evaluation of system properties for the purpose of system assessment — Part 2: Assessment methodology)

Издание официальное

МЭК 61226 Атомные станции. Системы контроля и управления, важные для безопасности. Классификация функций контроля и управления (IEC 61226, Nuclear power plants — Instrumentation and control systems important for safety — Classification of instrumentation and control functions)

МЭК 61508-4 Функциональная безопасность электрических/ электронных/ программируемых электронных систем, связанных с безопасностью. Часть 4: Определения и сокращения (IEC 61508-4, Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 4: Definitions and abbreviations)

МЭК 61513 Атомные станции. Системы контроля и управления, важные для безопасности. Общие требования к системам (IEC 61513, Nuclear power plants — Instrumentation and control systems important to safety — General requirements for systems)

ИСО/МЭК 9126 Технология программирования. Качество программного продукта (ISO/IEC 9126, Software engineering — Product quality)

Руководство МАГАТЭ NS-G-1.2 Оценка и верификация безопасности для атомных электростанций (IAEA guide NS-G-1.2, Safety assessment and verification for nuclear power plant)

Руководство МАГАТЭ NS-G-1.3 Системы контроля и управления, важные для безопасности на атомных электростанциях (IAEA guide NS-G-1.3, Instrumentation and control systems important to safety in nuclear power plants)

Для датированных ссылок применяют указанный вариант. Для недатированных ссылок применяют последнее издание документа (включая все изменения и поправки).

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1 **анимация** (animation): Процесс, посредством которого указанное в спецификации поведение демонстрируется с реальными значениями, полученными из задающих поведение выражений и некоторых входных величин.

3.2 **прикладная функция** (application function): Функция системы контроля и управления по выполнению задачи, связанной с контролируемым процессом, а не с функционированием самой системы.

[МЭК 61513, пункт 3.1]

3.3 **проблемно-ориентированный язык** (application oriented language): Компьютерный язык, специально разработанный для определенного типа применений и используемый лицами, являющимися специалистами в данном типе применения.

[МЭК 62138, пункт 3.3]

Примечание 1 — Группы оборудования обычно характеризуются проблемно-ориентированными языками, обеспечивающими удобное приспособление оборудования к специфичным требованиям.

Примечание 2 — Проблемно-ориентированные языки могут использоваться для обеспечения функциональных требований к системе контроля и управления и/или для установления или разработки прикладного программного обеспечения. Они могут базироваться на текстах, графике или на том и другом.

Примечание 3 — Например, языки диаграмм блоков функций, языки, определенные МЭК 61131-3.

3.4 **прикладное программное обеспечение** (application software): Часть программного обеспечения системы контроля и управления, которая обеспечивает выполнение прикладных функций.

[МЭК 61513, пункт 3.2]

3.5 **автоматизированная генерация кода** (automated code generation): Функция автоматизированных инструментов, позволяющая преобразовывать проблемно-ориентированный язык в форму, пригодную для компиляции или выполнения.

3.6 **канал** (channel): Совокупность взаимосвязанных компонентов внутри системы, имеющая один выход. Канал теряет свою идентичность тогда, когда сигналы на единственном выходе сочетаются с сигналами от других каналов, например, от канала контроля или канала активизации защиты.

[Глоссарий МАГАТЭ NS-G-1.3]

3.7 **уплотнение кода (программы)** (code compaction): Целесообразное уменьшение размеров памяти, необходимой для программы, путем исключения избыточных или посторонних команд.

3.8 **отказ по общей причине (ООП)** [common cause failure (CCF)]: Отказ двух или более конструкций, систем или компонентов вследствие единичного конкретного события или единичной конкретной причины.

[Глоссарий МАГАТЭ NS-G-1.3]

3.9 **компьютер** (computer): Программируемое функциональное устройство, которое состоит из одного или нескольких процессоров и периферийного оборудования, управляется хранящимися внутри программами и способно выполнять основные вычисления, включая многочисленные арифметические или логические операции без вмешательства в этот процесс человека.

[ИСО 2382-1]

Примечание — Компьютер может быть автономным или состоять из нескольких взаимосвязанных устройств.

3.10 **компьютерная программа** (computer program): Набор упорядоченных команд и данных, которые описывают операции в форме, приемлемой для их выполнения компьютером.

3.11 **компьютеризированная система** (computer-based system): Система контроля и управления, функции которой в большей своей части зависят от использования микропроцессоров, программируемого электронного оборудования или компьютеров либо полностью определяются таким использованием.

[МЭК 61513, пункт 3.10].

Примечание — Эквивалентны следующему определению: цифровые системы, системы с программным обеспечением, программируемые системы.

3.12 **данные** (data): Представление информации или команд в виде, пригодном для передачи, интерпретации или обработки с помощью компьютера.

[IEEE 610, модифицировано]

Примечание — Данные, необходимые для определения параметров и реализации прикладных и служебных функций в системе, называются «прикладными данными».

3.13 **глубокоэшелонированная защита** (defence in depth): Применение более одной защитной меры для достижения определенной цели безопасности так, чтобы цель была достигнута даже при отказе одной из защитных мер.

[Глоссарий МАГАТЭ по безопасности]

3.14 **разнообразие** (diversity): Наличие двух или более путей или средств достижения установленной цели. Разнообразие специально создается как защита от отказа по общей причине. Оно может быть достигнуто наличием систем, которые физически отличаются одна от другой, или с помощью функционального разнообразия, если аналогичные системы достигают установленной цели различными путями.

3.15 **динамический анализ** (dynamic analysis): Процесс оценки системы или компоненты, основанный на их поведении в процессе работы. В противоположность статическому анализу.

[IEEE 610]

3.16 **отказ** (failure): Отклонение реального функционирования от запланированного.

[МЭК 61513, пункт 3.21, изменено]

3.17 **дефект** (fault): Неисправность или ошибка в компоненте технического обеспечения, программного обеспечения или системы.

[МЭК 61513, пункт 3.22]

3.18 **устойчивость к дефектам и ошибкам** (fault tolerance): Встроенные возможности системы обеспечивать непрерывную и правильную работу при наличии ограниченного числа дефектов технического или программного обеспечения.

3.19 **функциональное разнообразие** (functional diversity): Применение разнообразия на функциональном уровне (например, активация останова при достижении предельных значений как давления, так и температуры).

3.20 **универсальный язык** (general-purpose language): Компьютерный язык, предназначенный для всех видов применения.

[МЭК 62138, пункт 3.17]

Примечание 1 — Программное обеспечение операционной системы групп оборудования обычно реализуется с использованием универсальных языков.

Примечание 2 — Примеры: Ада, Си, Паскаль.

3.21 **ошибка человека** (human error): Действие человека, приводящее к непреднамеренному результату.

3.22 **инициализировать** (initialize): Установить счетчики, переключатели, адреса или содержимое устройств памяти на нулевое значение или другие начальные значения в начале или в заданной точке выполнения компьютерной программы.

3.23 **комплексные тестирования** (integration tests): Тестирования, проводимые во время процесса интеграции технического и программного обеспечения до валидации компьютерной системы с целью проверки совместимости программного обеспечения и технического обеспечения компьютера.

3.24 **библиотека** (library): Набор связанных элементов программного обеспечения (ПО), сгруппированных вместе, но индивидуально отбираемых для включения в окончательный продукт ПО.

3.25 **N-версионное программное обеспечение** (N-version software): Набор различных программ, называемых версиями, разработанных под общие требования и общие приемо-сдаточные тестирования. Версии выполняются одновременно и независимо, обычно на резервированных аппаратных средствах. Используются идентичные входы в тестовых системах или соответствующие входы резервированных систем. В случае противоречия между выходами различных версий используется заранее определенная стратегия, такая, например, как голосование.

3.26 **операционное системное программное обеспечение** (operation system software): Программное обеспечение, выполняемое на целевом процессоре во время работы, такое, например, как драйверы и сервисы входа/выхода, управление прерываниями, планировщик, драйверы связи, библиотеки прикладных программ, диагностирование во время работы, управление резервированием и смягченной деградацией.

3.27 **исходное постулированное событие (ИПС)** (postulated initiating event — PIE): Событие, приводящее к ожидаемым происшествиям или аварийным ситуациям и, как следствие, к явлениям отказов.

[МЭК 61513, пункт 3.41]

3.28 **ранее разработанное программное обеспечение (РПО)** (predeveloped software — PDS): Часть программного обеспечения, которая уже существует, доступна как коммерческий или запатентованный продукт и предлагается к использованию.

[МЭК 62138, пункт 3.24, модифицировано]

3.29 **резервирование** (redundancy): Использование альтернативных (одинаковых или неодинаковых) конструкций, систем или компонентов таким образом, чтобы все они могли выполнять требующуюся функцию независимо от эксплуатационного состояния или выхода из строя любого из них.

[Глоссарий МАГАТЭ NS-G-1.3]

3.30 **ролевое управление доступом** (role-based access control): Управление доступом на основе правил, определяющих разрешение доступа пользователей к объекту (функции, данные) не на индивидуальном основании, а на основании принадлежности к группам с идентичными задачами.

3.31 **функция безопасности** (safety function): Специфичная цель, которая должна быть достигнута для обеспечения безопасности.

[Глоссарий МАГАТЭ NS-R-1]

3.32 **система безопасности** (safety system): Система, важная для безопасности, обеспечивающая безопасный останов реактора или отвод остаточного тепла в активной зоне либо ограничивающая последствия ожидаемых при эксплуатации событий и проектных аварий.

[Глоссарий МАГАТЭ NS-R-1]

3.33 **траектория сигнала** (signal trajectory): Динамика изменения всех состояний оборудования, внутренних состояний, входных сигналов и действий оператора, определяющих выходы системы.

3.34 **программное обеспечение (ПО)** (software): Программы (т.е. набор упорядоченных команд), данные, правила и любая связанная с этим документация, имеющая отношение к работе компьютеризированной системы контроля и управления.

[МЭК 62138, пункт 3.27]

3.35 **разработка ПО** (software development): Стадия жизненного цикла ПО, которая приводит к созданию ПО системы контроля и управления или программного продукта. Она охватывает деятельность, начиная от спецификации требований и до валидации и установки на объекте.

[МЭК 62138, пункт 3.30]

3.36 **модификация ПО** (software modification): Изменение в уже согласованном документе (или документах), ведущее к изменению рабочей программы.

Примечание — Модификации ПО могут происходить в процессе первоначальной разработки ПО (например, устранение ошибок, обнаруженных на поздних этапах разработки) либо когда ПО уже находится в эксплуатации.

3.37 **жизненный цикл безопасности ПО** (software safety lifecycle): Необходимая деятельность при разработке и использовании программного обеспечения СКУ, важной для безопасности, осуществляемая в течение всего периода времени, начиная с разработки спецификации требований к программному обеспечению и заканчивая выведением программного обеспечения из эксплуатации.

[МЭК 62138, пункт 3.31]

3.38 **версия ПО** (software version): Экземпляр программного продукта, полученный путем модификации или корректировки предыдущего программного продукта.

[IEEE 610, модифицировано]

3.39 **спецификация** (specification): Документ, определяющий в полной, точной, проверяемой форме требования, дизайн, поведение или другие свойства системы либо компонента, и, зачастую, процедуры для определения, выполняются ли эти требования.

[IEEE 610]

Примечание — Существуют различные типы спецификаций, например, спецификация требований к ПО или спецификация проекта.

3.40 **статический анализ** (static analysis): Процесс оценки системы или ее компонентов, основанный на ее форме, структуре, содержании или документации.

3.41 **системное программное обеспечение** (system software): Часть ПО системы контроля и управления, созданная для конкретного компьютера или семейства оборудования с целью облегчения разработки, эксплуатации и модификации этих объектов и связанных с ними программ.

[МЭК 62138, пункт 3.33]

3.42 **валидация системы** (system validation): Подтверждение путем проверки и предоставления других свидетельств того, что система в целом соответствует спецификации требований (функциональность, время отклика, устойчивость к дефектам и ошибкам, запас прочности).

3.43 **верификация** (verification): Подтверждение экспертизой и предоставлением иного объективного свидетельства того, что результаты функционирования соответствуют целям и требованиям, определенным для такого функционирования.

[МЭК 62138, пункт 3.35]

4 Сокращения

АПСП — автоматизированное проектирование и создание программ;
ООП — отказ по общей причине (см. 3.8);
РПО — ранее разработанное программное обеспечение (см. 3.28);
ИПС — исходные постулированные события (см. 3.27);
ОК — обеспечение качества;
В&В — верификация и валидация;
СКУ — система контроля и управления.

5 Общие требования к проектам программного обеспечения

5.1 Общая информация

Процесс создания систем контроля и управления, применяемых на атомных станциях, установлен в МЭК 61513, в котором концепция жизненного цикла безопасности системы вводится в качестве средства, с помощью которого можно управлять процессом. Кроме того, принятие этой концепции должно привести к получению данных, необходимых для обоснования работы систем безопасности. Описанный в МЭК 61513 жизненный цикл безопасности системы включает в себя и устанавливает (но не навязывает) требования к мероприятиям проекта, которые должны осуществляться при создании систем (см. рисунок 1).

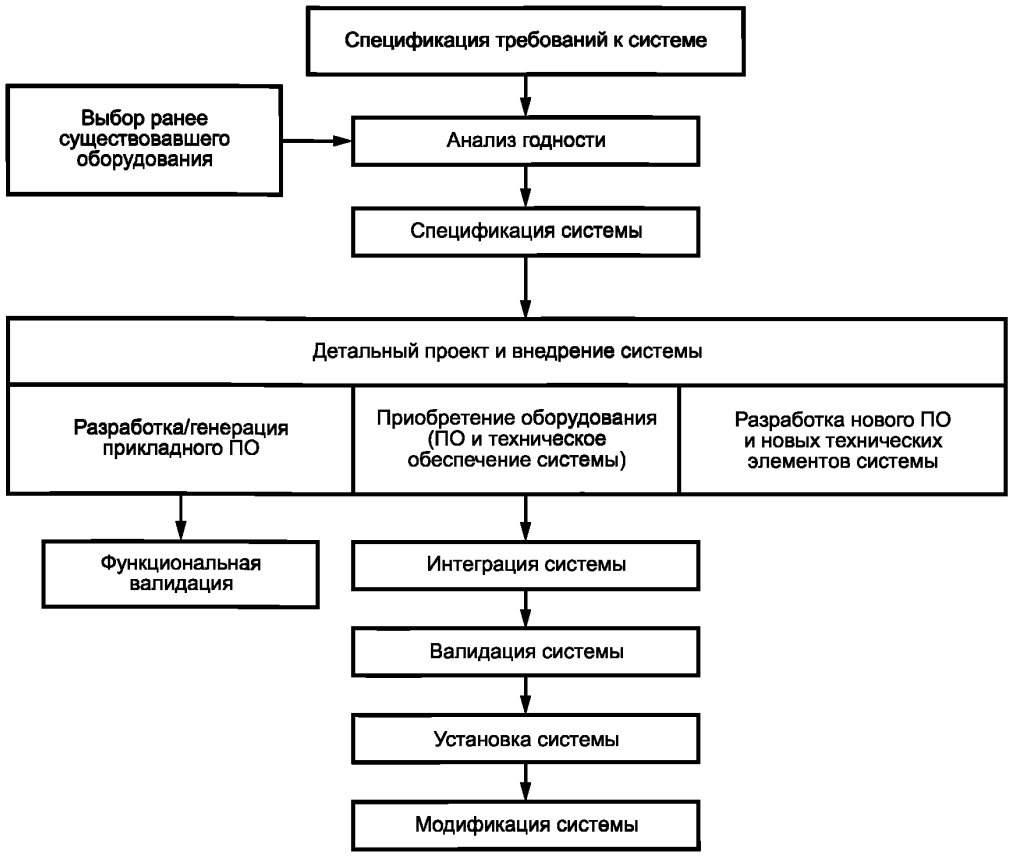


Рисунок 1 — Деятельность по жизненному циклу безопасности системы (как определено в МЭК 61513)

Жизненный цикл безопасности системы для компьютерных (т.е. цифровых) систем расширяется и включает в себя концепцию жизненного цикла безопасности ПО (см. рисунок 2). В соответствии с этой концепцией техническое и программное обеспечения разрабатываются параллельно, исходя из общей спецификации, а затем объединяются на этапах жизненного цикла, соответствующих интеграции и установке.

Следующие процессы обеспечивают поэтапный процесс разработки при создании ПО:

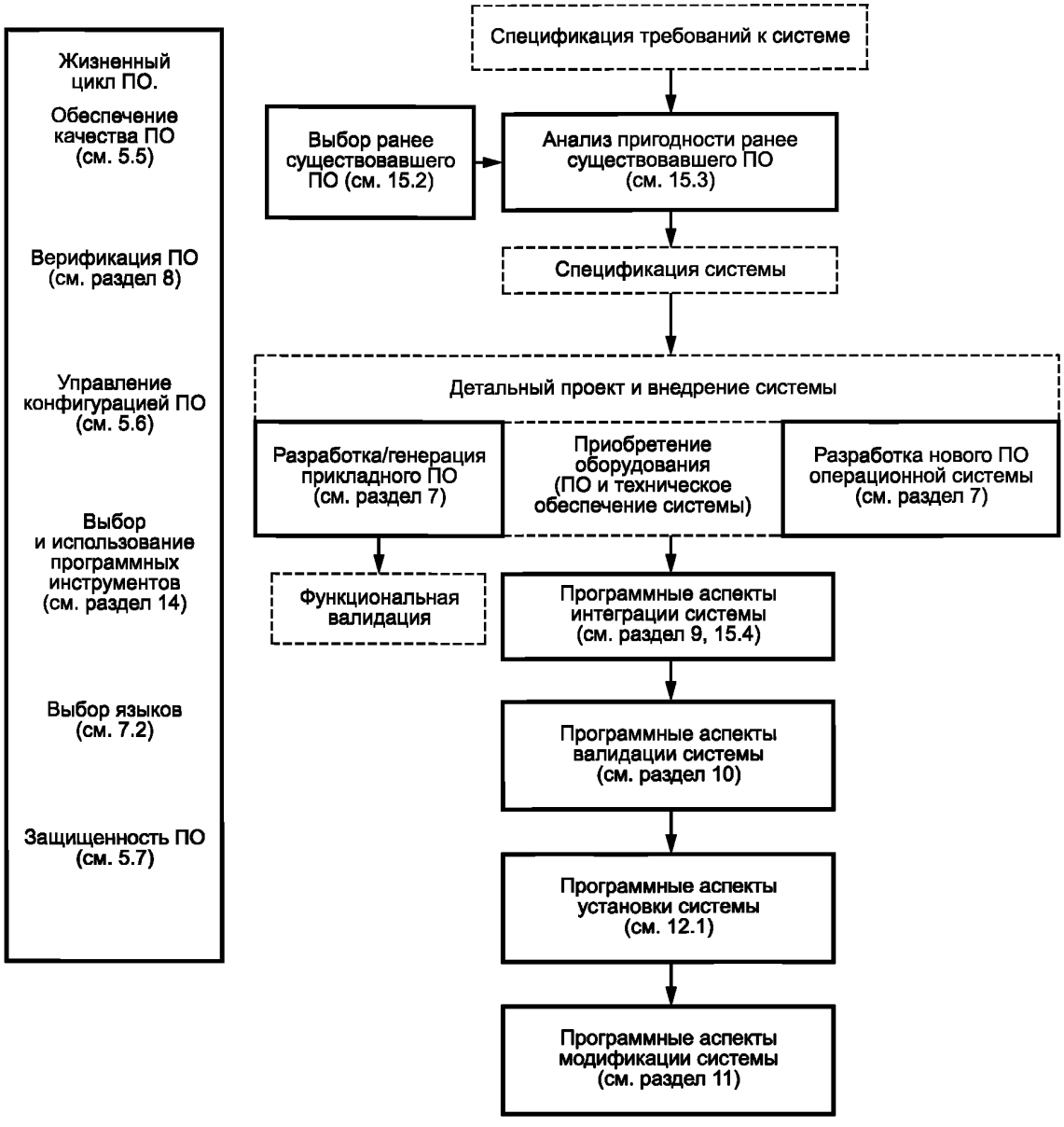
- управление проектом ПО (см. 5.4);
- обеспечение качества и контроль качества ПО (см. 5.5);
- управление конфигурацией ПО (см. 5.6);
- защищенность ПО (см. 5.7);
- верификация ПО (см. раздел 8).

Существует также деятельность, включающая в себя выбор языков (см. 7.2, приложение D), выбор программных инструментов, помогающих разработке (см. раздел 14), предупреждение ООП (см. раздел 13) и изготовление документации (см. 7.4, приложение F).

Итоговая деятельность, связанная с ПО, в жизненном цикле безопасности системы и вспомогательные процессы показаны на рисунке 2 (блоки, выделенные полужирным шрифтом и со ссылками на соответствующие подразделы в скобках).

Разработку ПО следует осуществлять на основании традиционной модели «V», поскольку этот подход отражен и распространен в других стандартах, в частности в документе МАГАТЭ NS-G-1.3, при этом допускается необходимая корректировка, поскольку очевидно, что некоторые этапы разработки могут осуществляться автоматически, а процесс разработки может быть итерационным.

В подразделах 5.2 и 5.3 приведены различные типы ПО, различные подходы к разработке, рассматриваемые в настоящем стандарте.



Примечание — Блоки, заключенные в тонкие пунктирные линии, представляют деятельность в отношении системы, не освещаемую в настоящем стандарте.

Рисунок 2 — Деятельность по жизненному циклу безопасности системы, связанная с ПО

5.2 Типы ПО

Компоненты ПО системы часто определяют как принадлежащие к ПО операционной системы (обмен информацией, управление входами/выходами, стандартные функции, самопроверка и т.п.) либо к прикладному ПО (логика блокировок, контуры регулирования, формат отображения, аварийная логика и т.п.).

Прикладное ПО обычно использует средства, предоставляемые ПО операционной системы, уменьшая таким образом необходимость дублирования программ внутри модулей, что снижает общий объем ПО.

Прикладное ПО обычно является специфичным для данного проекта.

ПО операционной системы может использоваться в различных проектах.

Многие проекты систем широко применяют конфигурационные данные. Конфигурационные данные могут относиться к ПО операционной системы либо к прикладному ПО. Конфигурационные данные, связанные с прикладным ПО, состоят, главным образом, из технических данных станции, вытекающих из проекта

станции, и часто подготавливаются проектантами станции, которые не обязательно должны обладать навыками в области программирования.

Конфигурационные данные могут быть разделены на:

- элементы данных, которые не должны изменяться в режиме «онлайн» операторами станции и к которым предъявляются такие же требования, как и к остальной части ПО;

- параметры, т.е. элементы данных, которые могут быть изменены операторами во время работы станции (например, уровни аварийной сигнализации, уставки, данные по калибровке измерительной аппаратуры) и для которых необходимы специфичные требования.

Многие современные платформы оборудования для контроля и управления обеспечены обширным набором инструментальных программ, позволяющим инженерам системотехникам проектировать и реализовывать рабочие программы.

Например, типичная СКУ, разработанная с использованием компонентов комплекса оборудования, включает в себя:

- компоненты ранее разработанного программного обеспечения, такие как ядро операционного системного программного обеспечения и библиотека прикладных функций. Обычно эти компоненты разработаны с применением универсальных языков;

- данные конфигурации, необходимые для адаптации ядра операционного системного программного обеспечения к средствам ввода-вывода и сервисам, требующимся в данном применении;

- прикладное программное обеспечение, разработанное с использованием проблемно-ориентированных языков.

5.3 Подход к разработке программного обеспечения

Обычно программное обеспечение вносит существенный вклад в функции, выполняемые СКУ. Оно может также поддерживать дополнительные функции, предусмотренные в проекте системы (например, инициализацию и контроль за техническим обеспечением, связь и синхронизацию между подсистемами). Таким образом, в большинстве случаев жизненный цикл программного обеспечения тесно связан с жизненным циклом безопасности системы. В частности, спецификация требований к программному обеспечению является частью спецификации требований к системе и частью проекта системы либо непосредственно вытекает из них.

Несмотря на то, что верификация новых компонентов программного обеспечения определенно является частью жизненного цикла безопасности программного обеспечения, часто не существует разделения и четкой границы между интеграцией программного обеспечения с системой. Поэтому в настоящем стандарте интеграция программного обеспечения рассматривается как часть интеграции системы. Валидация программного обеспечения также не является деятельностью, связанной только с программным обеспечением: в настоящем стандарте она рассматривается как часть интеграции и/или валидации системы.

В настоящем стандарте предполагается, что жизненный цикл программного обеспечения, первоначально предназначенный для разработки программного обеспечения с помощью универсальных языков, распространяется также на проблемно-ориентированные языки и конфигурацию ранее разработанного программного обеспечения.

Тем не менее, в нем признаются следующие различия в процессе разработки за счет введения специализированных процедур для каждого вида программного обеспечения на уровне реализации:

- реализация с использованием универсальных языков;

- реализация с использованием проблемно-ориентированных языков вместе с генераторами кода;

- отбор, использование и конфигурация ранее разработанного программного обеспечения.

Поскольку блоки «Разработка/генерация прикладного ПО» и «Разработка нового ПО операционной системы» на рисунке 2 представляют большую и существенную часть жизненного цикла безопасности программного обеспечения, на рисунке 3 приведена ее «расшифровка», в которой более детально представлена деятельность между определением спецификации требований к программному обеспечению и его валидацией, с четким обозначением трех различных путей реализации. На рисунке 3 в скобках указаны соответствующие разделы и пункты настоящего стандарта.

Дополнительные требования к программному обеспечению приведены в приложении В.

Принципы, отраженные в требованиях настоящего стандарта, касаются качества конечной программы, и они применимы независимо от того, разработана ли программа с использованием универсальных языков, проблемно-ориентированных языков с автоматической генерацией рабочей программы, а также независимо от конфигурации.

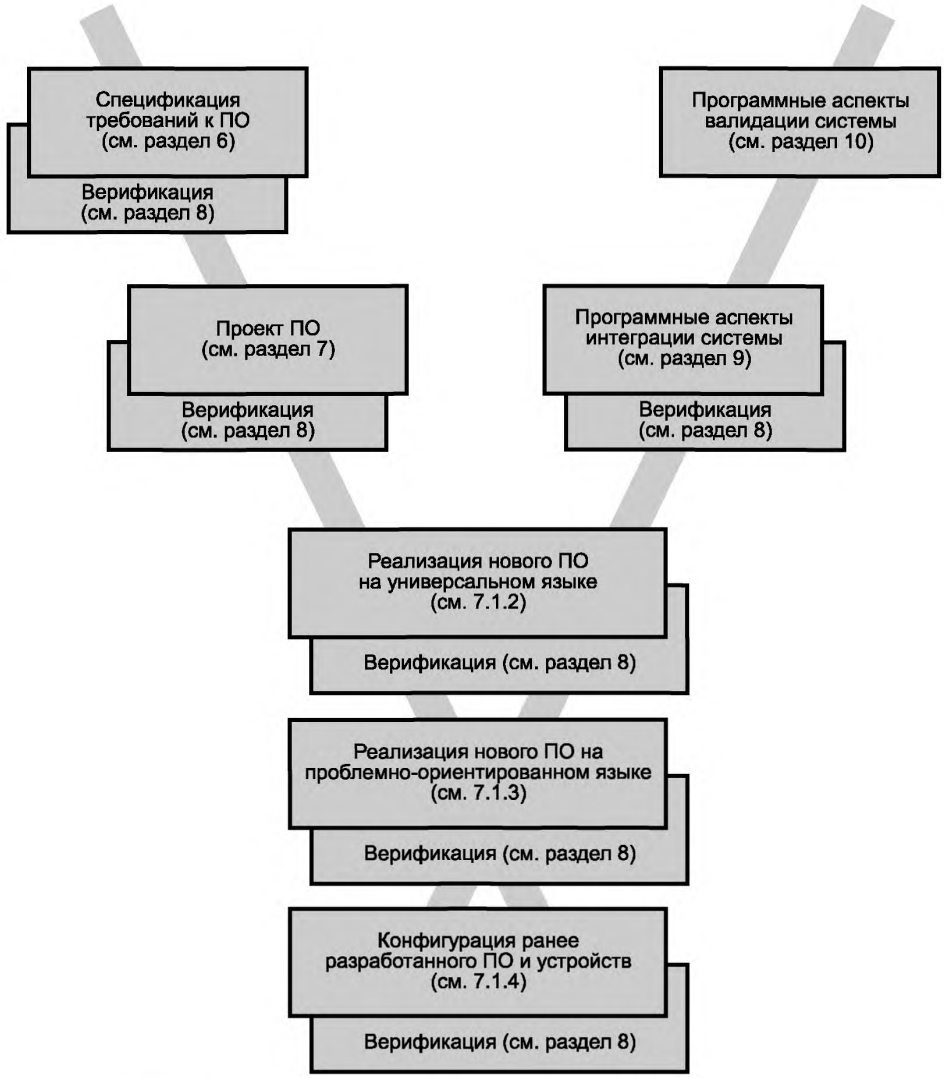


Рисунок 3 — Деятельность по разработке в рамках жизненного цикла безопасности программного обеспечения по МЭК 60880

5.4 Управление проектированием программного обеспечения

5.4.1 Любое проектирование программного обеспечения должно быть разбито на несколько этапов.

Каждый этап проектирования является до некоторой степени самостоятельным, но будет влиять на другие этапы и, в свою очередь, зависеть от них. Эти этапы отличаются друг от друга видом выполняемой деятельности.

Этапы и связанная с ними деятельность при проектировании программного обеспечения составляют процесс, называемый в настоящем стандарте «разработкой программного обеспечения». Подразумевается, что этот процесс может быть итеративным при условии выполнения требований последнего абзаца введения к разделу 6 МЭК 61513.

Деятельность и ее этапы при реализации проекта программного обеспечения определяются следующими общими факторами:

5.4.2 Осуществляемая на этапах процесса разработки деятельность должна быть определена в соответствии с выбранным для проекта подходом к разработке программного обеспечения (см. 5.2 и 5.3).

5.4.3 Деятельность по разработке программного обеспечения должна соотноситься с полным жизненным циклом безопасности программного обеспечения.

5.4.4 Каждый этап разработки программного обеспечения, указанный в 5.4.1, должен быть разделен на четко определенные виды деятельности.

5.4.5 Этапы разработки программного обеспечения должны быть официально приняты и ни один из них не должен быть пропущен.

5.4.6 Если деятельность по разработке программного обеспечения автоматизируется с использованием инструментальных программ, эта автоматизированная деятельность должна быть документально оформлена, включая документацию по входным и выходным данным, относящимся к конкретному этапу.

5.4.7 Входные и выходные данные на каждом этапе должны быть определены и документально оформлены.

5.4.8 Все выходные данные каждого этапа должны систематически проверяться [см. приложение В, пункт В.1, перечисление с) и пункт В.4, перечисление g)].

5.4.9 Каждый этап должен включать в себя создание соответствующих документов (см. приложение F).

5.4.10 Каждый этап должен завершаться систематическим анализом, включающим в себя проверку соответствующих документов.

5.4.11 Перечень документации, требующейся в течение жизненного цикла безопасности программного обеспечения, должен быть установлен во время разработки программного обеспечения. Пример типового перечня приведен в приложении F.

5.5 План обеспечения качества программного обеспечения

5.5.1 На ранней стадии жизненного цикла безопасности программного обеспечения должен существовать или быть установлен план обеспечения качества.

Специальные планы обеспечения качества могут быть приняты для отдельных этапов изделия или отдельных компонентов программного обеспечения в соответствии с национальными стандартами или стандартами организаций при условии соблюдения принципов, определенных в настоящем стандарте.

5.5.2 Любые отклонения от требований настоящего стандарта или его нормативных приложений должны быть установлены и обоснованы.

5.5.3 При осуществлении действий, отличных от изложенных в нормативных приложениях, они должны быть оформлены документально и быть доступны для проверки в соответствии с требованиями основных разделов настоящего стандарта.

5.5.4 В частности, должно быть рассмотрено влияние данных действий на СКУ и на программное обеспечение.

5.5.5 Все необходимые технические процедуры, проводимые при выполнении каждого этапа жизненного цикла безопасности программного обеспечения, должны быть указаны в плане обеспечения качества.

5.5.6 В плане обеспечения качества должно содержаться требование о том, чтобы реализация этапов была поручена компетентным лицам с предоставлением им необходимых ресурсов.

5.5.7 В плане обеспечения качества должно содержаться требование о том, чтобы модификации уже утвержденных документов осуществлялись на основе четкого определения конкретных изменений, их анализа и последующего утверждения уполномоченными лицами.

5.5.8 В плане обеспечения качества должно содержаться требование о том, чтобы используемые методы, языки, инструментальные программы, правила и стандарты были определены и оформлены документально, а также были известны соответствующим лицам и ими освоены.

5.5.9 В плане обеспечения качества должно содержаться требование о том, чтобы при использовании нескольких методов, языков, инструментальных программ, правил и/или стандартов было ясно, какой именно из них используется для каждого вида деятельности.

5.5.10 В плане обеспечения качества должно содержаться требование, чтобы специфичные для проекта термины, выражения, сокращения и условные обозначения были четко определены.

5.5.11 В плане обеспечения качества должно содержаться требование о том, чтобы любые возникшие в связи с качеством проблемы были отслежены и разрешены.

5.5.12 В плане обеспечения качества должно содержаться требование о том, чтобы результаты его реализации были документально оформлены.

5.5.13 Каждый шаг верификации или анализа должен заканчиваться отчетом о проведенном анализе, сделанных выводах и принятых согласованных решениях. Отчет должен быть включен в документацию.

5.5.14 Любые отклонения от плана обеспечения качества должны быть оформлены документально и обоснованы.

5.6 Управление конфигурацией

В 6.2.1.2 МЭК 61513 (план управления конфигурацией системы) приведены требования управления конфигурацией на уровне СКУ.

В данном подпункте приведены дополнительные требования, специфичные для программного обеспечения или имеющие для него особое значение.

5.6.1 Управление конфигурацией программного обеспечения должно осуществляться в соответствии с положениями плана управления конфигурацией или плана обеспечения качества.

5.6.2 Эти положения плана должны быть согласованы с соответствующими положениями на уровне управления конфигурацией системы.

5.6.3 Документально оформленные процедуры управления конфигурацией программного обеспечения должны быть установлены на ранней стадии проекта программного обеспечения при соблюдении следующих требований:

5.6.4 Каждая создаваемая версия любого программного продукта должна иметь уникальную идентификацию.

5.6.5 Должна быть возможность идентификации соответствующих версий всей документации, связанной с каждым программным продуктом.

5.6.6 Программное обеспечение, находящееся в разработке, должно быть отделено от программного обеспечения, достигшего разрешенного или верифицированного статуса.

5.6.7 Должна быть возможность идентификации версий всех программных продуктов, которые совместно образуют полную версию конечного программного продукта.

5.6.8 Должна быть возможность верификации полноты программных продуктов.

5.6.9 Должна быть возможность идентификации версии программного обеспечения целевой системы.

5.6.10 Должна быть возможность ретроспективной идентификации всех программных продуктов, на которые влияет реализация модификаций.

5.6.11 Доступ ко всем программным продуктам, введенным под управление конфигурацией, должен быть объектом соответствующего управления для обеспечения того, чтобы программное обеспечение не модифицировалось не уполномоченными на это лицами и сохранялась защищенность программного обеспечения.

5.6.12 Должна быть возможность идентификации всех трансляторов и всех версий инструментальных программ, используемых для получения каждой выполняемой программы (см. 14.3.3).

5.7 Защищенность программного обеспечения

Цель защищенности программного обеспечения и данных состоит в том, чтобы неуполномоченные лица и не предназначенные для этой цели системы не могли их считывать или изменять, и, в то же время, был обеспечен необходимый доступ для уполномоченных лиц и предназначенных для этой цели систем.

В 5.4.2 (общий план защищенности) и 6.2.2 (план защищенности системы) МЭК 61513 представлены требования по защищенности на уровне архитектуры К&У и конкретной СКУ.

Хотя использование программного обеспечения действительно представляет определенную потенциальную угрозу защищенности, основные контрмеры обычно осуществляются на уровне системы, например меры физической защиты, подсоединение блокирующих устройств. Основные требования к защищенности, налагаемые на программное обеспечение, могут дополнять защитные меры на системном уровне и способны свести к минимуму уязвимость.

В настоящем подразделе представлены требования к защищенности, специфичные или особенно важные для программного обеспечения.

5.7.1 Анализ защищенности

5.7.1.1 Должен быть проведен анализ потенциальных угроз защищенности в отношении программного обеспечения. В анализе должны быть учтены соответствующие этапы жизненного цикла безопасности системы и программного обеспечения. В нем также должны быть определены требования к защите и доступности данных и программного обеспечения, основанные на требованиях настоящего пункта.

5.7.1.2 В плане обеспечения качества программного обеспечения или системы либо в плане защищенности программного обеспечения или системы должен быть учтен анализ защищенности программного обеспечения.

5.7.1.3 Если анализ показывает, что контрмеры на системном уровне недостаточны, то в нем должны быть определены требования к контрмерам при проектировании программного обеспечения.

5.7.2 Проект защищенности

5.7.2.1 Требования к контрмерам при проектировании конкретного программного обеспечения, полученные при анализе защищенности, должны быть включены в требования к проектированию программного обеспечения.

5.7.2.2 Любое новое программное обеспечение должно проектироваться так, чтобы минимизировать уязвимость системы.

5.7.2.3 Любому ранее разработанному программному обеспечению нужно придать такие конфигурацию и параметры, чтобы минимизировать уязвимость системы, например, путем минимизации функций до необходимого предела или с помощью существующих функций защищенности программного обеспечения.

5.7.2.4 Должна быть исключена возможность изменения хранящихся программ оператором.

5.7.2.5 Если для выполнения функций К&У оператору необходим доступ к изменению данных, то устройства человеко-машинного интерфейса должны ограничивать этот доступ необходимыми пределами.

5.7.2.6 Там, где необходимо противостоять возможным угрозам защищенности, должны быть включены в проект, конфигурацию и/или процедуру присвоения параметров программного обеспечения эффективные защитные меры, касающиеся:

- управления выборочного доступа пользователя к функциям программного обеспечения;
- связи данных с системами, имеющими меньшую важность для безопасности;
- прослеживаемости модификаций программного обеспечения или параметров.

5.7.2.7 В проектной документации должны быть определены и описаны функции, критические для защищенности, и элементы защищенности, примененные в программном обеспечении.

5.7.2.8 Во время верификации программного обеспечения должна быть подтверждена защищенность функций.

5.7.2.9 Во время валидации СКУ с помощью подходящих тестов должна быть продемонстрирована эффективность функций защищенности.

5.7.3 Доступ пользователя

5.7.3.1 Там, где необходимо, программное обеспечение должно поддерживать технические меры по эффективной процедуре аутентификации, прежде чем пользователю будет разрешен доступ.

5.7.3.2 Там, где доступ пользователя является элементом, критическим для защищенности, в программном обеспечении следует применять процедуру аутентификации, осуществляемую техническими средствами на основе получения комбинации информации о знании (например, пароль), личной собственности (например, ключ, карта с встроенным микропроцессором) и/или персональных характеристиках (например, отпечаток пальца), а не полагаться исключительно на пароль.

5.7.3.3 Программному обеспечению и основанному на ролевом имени управлению доступом должны быть сообщены такие конфигурация и параметры, которые ограничивают разрешенный доступ пользователя к функциям и данным необходимыми пределами.

5.7.3.4 Права доступа пользователя должны быть ограничены до определенной степени с учетом возможностей и последствий потенциальных угроз защищенности.

Одним из возможных путей правильной реализации этого требования является криптографический метод (шифрование данных).

5.7.3.5 Удаленные доступы из любых мест, находящихся вне технической среды станции (например, из административных зданий или из мест, находящихся вне станции), способных повлиять на функции программного обеспечения или данные, не допускаются.

5.7.4 Защищенность во время разработки

5.7.4.1 Жизненный цикл безопасности разработки программного обеспечения должен учитывать потенциальные угрозы защищенности во время деятельности по разработке и обслуживанию.

5.7.4.2 Должны быть предусмотрены меры против скрытых функций в прикладном программном обеспечении или системном программном обеспечении (например, верификация кода), т.к. они могут поддерживать потенциальный несанкционированный доступ.

5.7.4.3 Если меры против скрытых функций не могут быть реализованы в отношении ранее разработанного программного обеспечения, применение такого программного обеспечения может быть обосновано с учетом потенциальной угрозы обеспечения безопасности, важности безопасности функций контроля и управления (I&C), характеристик системы и программного обеспечения.

5.7.4.4 Должна быть подтверждена выявляемость во время деятельности по верификации потенциальных средств преднамеренной модификации программного обеспечения, способных приводить к ошибочному режиму работы с задержкой по времени или изменению параметров режима работы.

6 Требования к программному обеспечению

6.1 Спецификация требований к программному обеспечению

6.1.1 Требования к программному обеспечению должны извлекаться из требований к системам безопасности, такие требования к программному обеспечению являются частью спецификации компьютерной системы.

6.1.2 Требования к программному обеспечению должны описывать то, что́ должно делать программное обеспечение, а не то, ка́к оно должно это делать.

6.1.3 В требованиях к программному обеспечению должны указываться:

- прикладные функции, заложенные в программное обеспечение;
- различные типы поведения программного обеспечения и соответствующие условия перехода;
- интерфейсы и взаимодействия программного обеспечения со средой [например, с операторами, остальными элементами СКУ, другими системами (если они существуют), с которыми оно взаимодействует или разделяет ресурсы], включая роли, типы, форматы, диапазоны и ограничения на вводы и выводы данных;
- параметры программного обеспечения, которые могут быть модифицированы вручную во время операции (если существуют), их роли, типы, форматы, диапазоны и ограничения, а также проверки, которые должно осуществлять программное обеспечение, если эти параметры изменяются;
- требования к параметрам программного обеспечения, в частности требования ко времени отклика;
- то, что программное обеспечение не должно делать или чего должно избегать (если это применимо);
- требования к среде программного обеспечения или допущения, принятые относительно этой среды;
- требования к пакетам стандартных программ (при их наличии).

6.1.4 Вследствие важности данного этапа разработки программного обеспечения процесс установления требований должен быть строгим.

6.1.5 Спецификация требований к программному обеспечению должна быть такой, чтобы она могла продемонстрировать согласованность с требованиями МЭК 61513.

Детализация требований, дополняющих спецификацию требований к программному обеспечению, приведена в приложении А.

6.1.6 Должны быть описаны ограничения, связанные со взаимодействием между программным и техническим обеспечением (см. А.2.1 приложения А).

6.1.7 В спецификации требований программного обеспечения должна быть ссылка на спецификацию требований технического обеспечения для каждого влияющего элемента проекта технического обеспечения.

6.1.8 Специальные условия работы, такие как ввод станции в эксплуатацию, перегрузка топлива, должны быть описаны вплоть до уровня программного обеспечения подверженной влиянию функции.

6.2 Самоконтроль

6.2.1 Программное обеспечение компьютерных систем должно контролировать техническое обеспечение во время работы в определенные промежутки времени, а также свое собственное поведение (см. А.2.2 приложения А).

Программное обеспечение рассматривается в качестве фактора первостепенной важности для достижения высокой надежности всей системы.

6.2.2 Те части памяти, которые содержат коды или постоянные данные, должны контролироваться для обнаружения непредусмотренных изменений.

6.2.3 Самоконтроль программного обеспечения должен быть способнен в практически выполнимых пределах обнаруживать:

- случайные отказы компонентов технического обеспечения;
- ошибочное поведение программного обеспечения (например, отклонения от установленной работы программы или установленных условий эксплуатации либо искажение данных);
- ошибочную передачу данных между различными обрабатывающими устройствами.

6.2.4 Если во время эксплуатации станции обнаружен отказ, то программное обеспечение должно вовремя и соответствующим образом на него отреагировать. Это реагирование должно реализовываться в соответствии с требуемыми в спецификации реакциями системы, а также в соответствии с правилами, установленными МЭК 61513 для проекта системы.

С целью исключения ложных срабатываний может потребоваться соответствующий анализ.

6.2.5 Самоконтроль не должен негативно влиять на выполнение системой непредусмотренных функций.

6.2.6 Следует предусмотреть возможность автоматического сбора полезной диагностической информации, получаемой в результате самоконтроля программного обеспечения.

6.3 Периодические тестирования

6.3.1 Для компьютерных систем безопасности следует применять основные принципы МЭК 60671 для тех компонентов, которые должным образом не охватываются самоконтролем.

6.3.2 Программное обеспечение должно быть спроектировано так, чтобы соответствовало следующим требованиям периодических тестирований, проводимых в пределах установленных максимальных интервалов (например, в периоды останова):

1) каждая функция безопасности должна охватываться периодическими тестированиями;

2) любой отказ при выполнении функции безопасности должен быть обнаружен.

6.3.3 Следует предусмотреть возможность автоматического сбора полезной диагностической информации, получаемой в результате периодических тестирований программного обеспечения.

6.3.4 Рекомендуется, чтобы качество программного обеспечения вспомогательных устройств, предназначенных для тестирований, соответствовало качеству оборудования, используемого для валидации, как указано в 10.2.

Нет необходимости в том, чтобы программное обеспечение вспомогательных устройств, предназначенных для тестирований систем класса 1, соответствовало всем требованиям настоящего стандарта.

6.4 Документация

6.4.1 Основная цель документации по спецификации требований к программному обеспечению состоит в формировании основы для разработки программного обеспечения. Однако не следует пренебрегать аспектами лицензирования, поскольку данная документация может быть представлена регулирующим органам. Поэтому она может содержать аспекты, не являющиеся важными для разработки программного обеспечения, но являющиеся основой для лицензирования.

Такими важными для лицензирования аспектами могут быть:

- рассмотрение рисков;
- рекомендации для функций или конструкционных элементов безопасности;
- другие элементы, обеспечивающие основу для специфичных требований;
- специальные требования регулирующих органов к структуре программного обеспечения, анализу кода, B&B и т.п.

6.4.2 Спецификация требований к программному обеспечению должна быть представлена в стандартизованном формате, который не должен влиять на понятность документа (см. пункт А.2.3 приложения А).

6.4.3 Спецификация требований к программному обеспечению должна быть однозначной, тестируемой или верифицируемой, а также достижимой.

Для улучшения согласованности и полноты аспектов спецификации требований к программному обеспечению может применяться формализованный язык или проблемно-ориентированный язык.

Для этой цели могут использоваться автоматизированные программные инструменты.

6.4.4 Спецификация требований к программному обеспечению должна быть представлена ведущим участникам процесса проектирования.

7 Проектирование и реализация

7.1 Принципы проектирования и реализации

7.1.1 Общие сведения

7.1.1.1 Проект программного обеспечения должен включать самоконтроль (см. А.2.2 приложения А).

7.1.1.2 При обнаружении отказа должны быть предприняты надлежащие действия в соответствии с 6.2.

7.1.1.3 Структура программного обеспечения должна основываться на модульном принципе.

7.1.1.4 Структура программного обеспечения должна быть простой и понятной как в целом, так и в деталях.

7.1.1.5 Следует избегать изощренных приемов, рекурсивной структуры и сжатия кода.

7.1.1.6 Исходная программа должна быть понятной для квалифицированных специалистов, не участвующих в процессе разработки.

7.1.1.7 Исходная программа должна соответствовать документально оформленным правилам, предназначенным для улучшения ясности модифицируемости и тестируемости.

7.1.1.8 Следует обосновывать любые несоответствия правилам проектирования программного обеспечения.

7.1.1.9 Должна быть представлена полная и четко написанная документация к программному обеспечению.

7.1.1.10 Линии связи должны быть спроектированы в соответствии с требованиями к передаче данных, приведенными в 5.3.1.3 МЭК 61513.

7.1.1.11 Линии связи, используемые в одном резервном устройстве последовательных элементов, должны быть детерминистскими.

7.1.1.12 Из этих положений вытекают следующие рекомендации:

1) меры по реализации требований безопасности программного обеспечения, включая самоконтроль, следует выбирать в начале проектирования (см. раздел В.3 приложения В);

2) подход к проектированию программного обеспечения «сверху — вниз» предпочтительнее подхода «снизу — вверх» (см. раздел В.1 приложения В);

3) в начале проектирования каждого программного обеспечения следует устанавливать концептуальную модель его архитектуры (см. раздел В.2 приложения В);

4) написание программы следует осуществлять таким образом, чтобы это обеспечивало простоту проведения верификации (см. разделы В.4 и В.5 приложения В);

5) там, где используется стандартное программное обеспечение от производителя или поставщика, в дополнение к перечислению с) раздела В.2 приложения В применяются требования раздела 15;

6) использование проблемно-ориентированных языков предпочтительнее использования машинно-ориентированных языков (см. примечание раздела В.5 приложения В).

7.1.2 Реализация нового программного обеспечения на универсальных языках

В настоящем пункте рассматривается ситуация, когда часть или все функции безопасности категории А обеспечиваются разработкой компонентов программного обеспечения, использующих универсальные языки.

Универсальные языки являются обычно языками высокого уровня, такими как Ада, Си, Паскаль, либо языками ассемблера, предназначенными для используемой аппаратной платформы. Языки могут применяться для реализации функции любого рода при условии соблюдения соответствующих правил проектирования и кодирования.

7.1.2.1 На этапе проектирования программного обеспечения должны быть определены его компоненты, которые нужно разработать с помощью универсальных языков.

7.1.2.2 Для этих компонентов в процессе разработки следует определить этапы детального проектирования и кодирования.

7.1.2.3 Деятельность на этапе детального проектирования состоит в уточнении выходных данных этапа проектирования с тем, чтобы можно было на систематической основе проводить кодирование на выбранном языке (например, путем определения необходимых алгоритмов, структур данных, функциональных интерфейсов, ограничений и т.п.).

7.1.2.4 Уровень детализации информации на этапе детального проектирования зависит от используемого универсального языка. Если используется язык Ассемблера, то проект должен обеспечить подробно структурированные алгоритмы и представление данных.

7.1.2.5 На этапе кодирования детальный проект должен быть транслирован в исходный код в соответствии с заранее определенными правилами программирования, основанными на требованиях приложения В.

7.1.3 Реализация нового программного обеспечения на проблемно-ориентированных языках

В настоящем пункте рассматривается ситуация, когда часть функции безопасности категории А или все такие функции обеспечиваются путем разработки новых компонентов программного обеспечения с использованием проблемно-ориентированных языков.

Проблемно-ориентированные языки опираются на методы формализации (такие как логические диаграммы или диаграммы функциональных блоков и т.п.), которые могут применяться для выражения всей или части спецификации требований к программному обеспечению. Эти части могут использоваться в качестве входной информации для генерации исполняемого кода с помощью автоматизированных средств.

7.1.3.1 Рекомендуется, чтобы методы формализации обладали следующими свойствами: невысокая сложность, ясность и стандартность расположения и представления, модульность, наличие соответствующих комментариев, отсутствие небезопасных элементов. Эти свойства, как правило, упрощают понимание, верификацию, тестирование и последующую модификацию.

7.1.3.2 В проблемно-ориентированных языках следует использовать формат, понятный для специалистов, ответственных за анализ спецификации программного обеспечения, например, для технологов и специалистов по контролю и управлению, имеющих дело с системами, для которых установлены функции контроля и управления.

7.1.3.3 Рекомендуется, чтобы проблемно-ориентированные языки поддерживали простую структуру программного обеспечения, например, линейные программы.

7.1.3.4 Рекомендуется, чтобы проблемно-ориентированные языки позволяли разработчикам учитывать спецификацию проекта архитектуры СКУ, например, давали возможность назначать функции компонентам системы и поддерживали способность проекта сохранять устойчивость к дефектам элементов технического обеспечения.

7.1.4 Конфигурация ранее разработанного программного обеспечения

В данном пункте рассматривается ситуация, когда функции безопасности категории А обеспечиваются программным обеспечением с использованием данных конфигурации, т.е. данных, специфичных для конкретного применения.

Ранее разработанное программное обеспечение может быть связано с комплексом оборудования либо оно может быть отдельной программой, которая затем интегрируется с выбранной платформой технического обеспечения.

7.1.4.1 При использовании ранее разработанного программного обеспечения должна быть проведена оценка его возможностей (см. 15.3) для обеспечения пригодности для предназначенной роли.

Требования к анализу пригодности приведены в 15.3.1.2. При существовании ограничения на использование программного обеспечения его пригодность может быть достигнута с помощью программного обеспечения, являющегося внешним по отношению к ранее разработанному программному обеспечению.

Для конфигурирования программного обеспечения предпочтителен подход, основанный на инструментальных программах, который сужает область возможных ошибок человека.

7.1.4.2 Все ограничения, связанные с данными, должны быть оформлены документально с указанием, например, допустимых форматов данных, диапазонов, правил вычислений.

7.1.4.3 Данные конфигурации должны быть оформлены документально.

7.1.4.4 Надлежащим образом должны быть обоснованы значения величин, используемых в сочетании с исходными проектными данными.

7.1.4.5 Прослеживаемость: рекомендуется, чтобы была возможность определить, когда и кем были проведены изменения данных конфигурации.

7.1.4.6 Удобство сопровождения: процесс разработки должен обеспечивать с помощью структурированного подхода и использования комментариев к данным и/или сопроводительной документацией понятность и сопровождаемость структуры данных в течение назначенного срока службы системы, к которой эта структура принадлежит.

Для дальнейшего руководства по использованию инструментов прикладных данных см. 14.3.5.

7.2 Язык и связанные с ним трансляторы и инструментальные средства

7.2.1 Общие требования

При проектировании и реализации программного обеспечения систем класса 1 нельзя требовать использования специальных языков, однако следующие положения могут рассматриваться в качестве общих основных правил для языков, используемых в этих целях:

7.2.1.1 Используемые языки должны соответствовать строгим (или строго очерченным) правилам семантики и синтаксиса.

7.2.1.2 Синтаксис языка должен быть полностью и четко определен и оформлен документально.

7.2.1.3 В необходимых случаях использование языка должно быть ограничено «безопасным» сокращенным вариантом, например, примитивами, которые пригодны для определения необходимых функций.

7.2.1.4 Должны использоваться языки с тщательно проверенным транслятором.

7.2.1.5 Если применяется транслятор, не прошедший тщательной проверки, то дополнительная верификация должна предоставить свидетельство того, что результаты трансляции будут верными.

7.2.1.6 Следует иметь в распоряжении инструментальные программы для автоматизированной проверки.

Рекомендуется использование автоматизированных инструментальных программ. При этом применяются требования раздела 14.

7.2.2 Универсальные языки

Универсальные языки для систем класса 1 и их трансляторы не должны препятствовать использованию следующих схем, ограничивающих возможные ошибки:

- проверка типа переменных модуля трансляции, проверка параметров подпрограммы;
- проверка границ массива рабочего модуля.

Руководство по выбору языка, транслятора иредактора связи. приведено в приложении D.

7.2.3 Проблемно-ориентированные языки и соответствующая автоматизированная генерация кода

7.2.3.1 Проблемно-ориентированные языки следует преобразовывать в универсальный язык с помощью автоматизированных инструментальных программ (например, генератора кода) до трансляции программы в выполняемую форму.

7.2.3.2 Должна быть проведена оценка соответствия генерируемого кода требованиям настоящего стандарта к проектированию и кодированию программного обеспечения, а несоответствия должны быть обоснованы.

7.2.3.3 Структура генерируемой программы должна быть определена в общем виде, например, должно быть определено расположение описаний по отношению к кодовым операторам.

7.2.3.4 Генерируемый код не должен изменяться в результате непосредственных действий с кодом.

7.2.3.5 При модификации спецификации входных данных код должен быть генерирован повторно, например, в результате деятельности по B&B.

Дополнительные рекомендации по автоматизированной генерации кода приведены в перечислении f) раздела B.5 приложения B.

7.3 Подробные рекомендации

7.3.1 Общие положения

В приложении B приведены рекомендации, в которых детализированы аспекты, указанные в 7.1.

К двум основным этапам разработки программного обеспечения применимы рубрики индивидуальных рекомендаций приложения B, как показано в таблице 1.

Таблица 1 — Процедурные и программные аспекты проектирования и реализации

Этап разработки программного обеспечения	Процедурные аспекты	Приложение B	Программные аспекты	Приложение B
Проектирование	Модифицируемость	B.1a	Структуры управления	
	Подход сверху вниз	B.1b	и доступа	B.2a
	Верификация промежуточных программ при		Модули	B.2b
	проектировании	B.1c	Операционное системное программное обеспечение	B.2c
	Управление модификацией в процессе разработки		Время выполнения	B.2d
			Прерывания	B.2e
		B.1d	Арифметические выражения	B.2f
			Проверки достоверности	B.3a
			Безопасный вывод данных	B.3b
			Ветвления и циклы	B.4a
			Подпрограммы	B.4b
			Иерархические структуры	B.4c
			Структуры данных	B.4e
			Проблемно-ориентированные языки	B.5e

Окончание таблицы 1

Этап разработки программного обеспечения	Процедурные аспекты	Приложение В	Программные аспекты	Приложение В
Реализация	Верификация промежуточных программ при проектировании	B.1c	Модули	B.2b
			Время выполнения	B.2d
			Прерывания	B.2e
	Управление модификацией в процессе разработки	B.1d	Арифметические выражения	B.2f
			Проверки достоверности	B.3a
			Безопасный вывод данных	B.3b
	Тестирования устройства и интеграционные тестирования	B.4g	Содержание памяти	B.3c
			Проверка на наличие ошибок	B.3d
			Ветвления и циклы	B.4a
	Правила кодирования	B.5d	Подпрограммы	B.4b
			Иерархические структуры	B.4c
			Адресация и массивы	B.4d
			Структуры данных	B.4e
			Динамические изменения	B.4f
			Последовательность и упорядоченность	B.5a
			Комментарии	B.5b
			Ассемблер	B.5c
			Автоматизированная генерация кода	B.5f

7.3.2 Применение требований и рекомендаций

7.3.2.1 В процессе разработки программного обеспечения должны быть выполнены требования и рекомендации, приведенные в приложении В, а их невыполнение должно быть обосновано и оформлено документально.

7.3.2.2 Обоснование следует выполнять в начале процесса проектирования.

7.4 Документация

7.4.1 Во время разработки программного обеспечения этап проектирования должен заканчиваться составлением спецификации проекта программного обеспечения.

Этот документ служит основой для официального рассмотрения проекта и последующей реализации программы.

7.4.2 Детализация должна быть достаточной для того, чтобы реализация программы могла быть осуществлена без дальнейшего разъяснения проекта.

7.4.3 Документ должен быть структурирован в соответствии с уровнями процесса проектирования программного обеспечения.

Спецификация проекта программного обеспечения может быть представлена в виде одного документа или полного набора отдельных документов.

7.4.4 В этом случае каждый документ должен иметь определенную связь с другими документами и быть четко ограниченным по теме.

7.4.5 Форматы документов следует выбирать в соответствии с конкретной целью, включая:

- словесное описание;
- арифметические выражения;
- графическое представление.

7.4.6 В документах следует приводить необходимые диаграммы и чертежи. Как правило, предпочтительней выбирать графическое представление.

7.4.7 При необходимости документацию следует согласовывать с национальными стандартами.

8 Верификация программного обеспечения

8.1 Процедура верификации программного обеспечения

Деятельность по верификации, предпринимаемая как часть разработки программного обеспечения, обычно лежит на ответственности поставщика и осуществляется персоналом, не зависимым от лиц, создающих программное обеспечение; лучший путь — это привлечение верификационной группы.

Допускается проведение третьей стороной дополнительной верификации как части оценки третьей стороной программного обеспечения и процесса его разработки с тем, чтобы обеспечить уверенность в соответствии программного обеспечения намеченным показателям качества. Существует много путей обеспечения независимой верификации ресурсами и ее реализации; выбор пути зависит от предпочтений национальных регулирующих органов.

8.1.1 Верификационная группа должна состоять из лиц, не принимавших участия в создании программы и обладающих необходимыми компетенцией и знаниями.

Четко определяют требуемый уровень независимости:

8.1.2 Руководство верификационной группы должно быть отдельным и не зависимым от руководства разрабатывающей группы.

8.1.3 Общение между верификационной группой и разрабатывающей группой — будь то уточнения или отчет о дефектах — должно осуществляться формализованным образом в письменном виде с уровнем детализации, допускающим проверку.

8.1.4 При взаимодействии между двумя сторонами следует стремиться к сохранению независимости суждений верификационной группы.

8.1.5 Верификационная группа должна быть обеспечена соответствующими ресурсами и средствами. Ей должно быть предоставлено время, необходимое для осуществления деятельности по верификации.

8.1.6 Верификационная группа должна иметь четко определенные ответственность и обязательства.

8.1.7 Верификационная группа должна иметь необходимые основания для формулирования своих выводов.

8.1.8 Выходные данные каждого этапа разработки программного обеспечения (см. рисунок 3) должны быть верифицированы.

8.1.9 Деятельность по верификации программного обеспечения должна подтверждать соответствие спецификации требований к программному обеспечению требованиям, предъявляемым к программному обеспечению в спецификации требований к системе.

8.1.10 Деятельность по верификации программного обеспечения должна подтверждать соответствие спецификации проекта программного обеспечения требованиям к программному обеспечению.

8.1.11 Деятельность по верификации программного обеспечения должна подтверждать соответствие кода и спецификации проекта программного обеспечения, полученной на этапе проектирования. Специальные требования приведены в 8.2.3.2 для случая использования инструментальной программы АПСП с такими элементами, как автоматизированная генерация кода.

8.1.12 Любую производственную деятельность следует начинать на основе верифицированных входных данных или документов.

8.1.13 Верификацию результатов этапа как части разработки программного обеспечения рекомендуется проводить до начала следующего этапа, и она должна быть завершена до завершения (т.е. до верификации) следующего этапа.

Возможные подготовительные работы для последующего этапа могут быть выполнены до верификации и утверждения предыдущего этапа.

8.1.14 Если исходная информация или документы, необходимые для выполнения определенных действий, были изменены, то эти действия, а также последующие действия должны быть проведены повторно в необходимом объеме, учитывая потенциальное влияние произведенных изменений.

8.1.15 Верификация всего программного обеспечения должна быть завершена до введения системы в эксплуатацию.

8.2 Действия по верификации программного обеспечения

При верификации необходимо выполнить следующие действия.

8.2.1 План верификации

8.2.1.1 До начала выполнения действий по верификации должен быть составлен план верификации программного обеспечения.

8.2.1.2 В плане должны быть документально оформлены все критерии, методы и инструменты, используемые в процессе верификации.

8.2.1.3 В плане должны быть описаны действия, выполняемые для оценки каждого объекта программного обеспечения, каждой инструментальной программы, используемой в процессе разработки программного обеспечения, и каждого этапа с тем, чтобы выяснить соответствие этих объектов спецификации требованиям к программному обеспечению.

8.2.1.4 Степень детализации должна быть такой, чтобы верификационная группа могла реализовать план верификации и вынести объективное суждение о соответствии программного обеспечения предъявляемым к нему требованиям.

8.2.1.5 План верификации должен быть составлен верификационной группой и включать в себя:

1) выбор стратегии верификации, систематической или выборочной, либо комбинирующей то и другое, с выбором контрольного тестирования в соответствии с требуемыми функциями, особенностями структуры программы либо с учетом того и другого (см. приложение Е);

2) выбор и использование инструментальных программ для верификации программного обеспечения;

3) проведение верификации;

4) документальное оформление действий по верификации;

5) оценку результатов верификации, полученных непосредственно от испытательного оборудования, а также из проведенных тестирований оценку выполнения требований по безопасности.

8.2.1.6 Проводимые тестирования должны обеспечить обширную проверку программного обеспечения. Среди критериев, которые необходимо включить в план, важнейшими следует считать критерии тестового покрытия.

8.2.1.7 В плане верификации должны быть определены любые объективные свидетельства, требующиеся для подтверждения обширности тестирований. С этой целью критерии тестового охвата, выбранные в соответствии с проектом (см. приложение Е), должны быть обоснованы и документально оформлены.

8.2.1.8 Должны быть предусмотрены адекватные меры по разрешению всех проблем безопасности, возникших во время действий по верификации, проводимых поставщиком в процессе разработки программного обеспечения либо третьей стороной в процессе оценки.

8.2.1.9 Все проблемы безопасности должны быть решены с помощью соответствующих корректировок или смягчающих мероприятий.

8.2.2 Верификация проекта

8.2.2.1 Верификация проекта должна касаться:

1) достаточности спецификации проекта программного обеспечения в отношении требований, касающихся их взаимного соответствия и полноты, вплоть до модульного уровня включительно;

2) разбиения проекта на функциональные модули и способа такого разбиения с учетом:

- возможности технической реализации проекта,
- контролепригодности для последующей верификации,
- понятности для разрабатывающей и верификационной групп,
- модифицируемости для возможности последующей модификации;

3) правильной реализации требований безопасности.

8.2.2.2 Результат верификации проекта должен быть оформлен документально.

8.2.2.3 В документацию должны быть включены выводы и четко обозначены следующие проблемы, требующие действий:

- пункты, по которым нет соответствия требованиям к программному обеспечению;
- пункты, по которым нет соответствия стандартам на проектирование;
- модули, данные, структуры и алгоритмы, плохо адаптированные к задаче.

8.2.3 Верификация реализации

Независимо от того, как была разработана рабочая программа, методы тестирований, используемые для верификации результатов этапа реализации, следует отбирать в соответствии с подразделом Е.4.2 приложения Е.

8.2.3.1 Верификация реализации на универсальных языках

8.2.3.1.1 Верификация кода

8.2.3.1.1.1 Верификация реализации должна включать в себя действия, основанные на анализе и тестированиях исходного кода. Анализ исходного кода может проводиться с использованием таких методов верификации, как проверка кода, возможно, с помощью автоматизированных инструментальных программ.

8.2.3.1.1.2 Верификацию кода следует начинать с анализа исходного кода модуля, за которым следуют тестирования модуля.

8.2.3.1.1.3 Верификация модуля должна подтвердить, что каждый модуль выполняет предназначенную ему функцию и не выполняет непредназначенных функций.

8.2.3.1.1.4 Интеграционные тестирования модуля должны проводиться с целью демонстрации на раннем этапе разработки того, что все модули правильно взаимодействуют и выполняют предназначенные им функции. При использовании инструментария АПСП он также должен соответствовать требованиям раздела 14.

8.2.3.1.1.5 Результаты верификации кода должны быть оформлены документально.

8.2.3.1.2 Спецификация тестирований программного обеспечения

Спецификация тестирований программного обеспечения является одним из принципиальных документов, с которыми должен быть согласован план верификации.

8.2.3.1.2.1 Спецификация тестирований программного обеспечения должна основываться на спецификации проекта программного обеспечения и детальной проверке требований к программному обеспечению.

8.2.3.1.2.2 В спецификации должна быть проведена детальная информация о тестированиях, которые нужно провести в отношении каждого компонента программного обеспечения (модулей и их составляющих).

8.2.3.1.2.3 Спецификация тестирований программного обеспечения должна включать в себя:

1) среду, в которой проводятся тестирования;

2) процедуры тестирований;

3) критерии приемки, т.е. детальное определение критериев, которые должны быть соблюдены, для того чтобы принять модули и основные компоненты на уровнях подсистемы и системы;

4) процедуры обнаружения дефектов;

5) перечень документов, которые должны быть оформлены.

8.2.3.1.3 Отчет о тестированиях программного обеспечения

8.2.3.1.3.1 В отчете о тестированиях программного обеспечения должны быть представлены результаты верификации, описанные в спецификации тестирований программного обеспечения и устанавливающие, работает ли программное обеспечение в соответствии со спецификацией проекта программного обеспечения.

8.2.3.1.3.2 В отчете должны быть отмечены расхождения между проектом и реализацией, обнаруженные в процессе тестирований.

8.2.3.1.3.3 Отчет о тестированиях программного обеспечения должен включать в себя следующие пункты как на уровне модуля, так и на уровне основного проекта:

1) конфигурация технических средств, используемых для тестирований, определение и обоснование использования любых технических приспособлений и программного обеспечения;

2) используемые носители информации и требования к доступу испытуемого конечного кода;

3) входные значения, связанные с тестированиями;

4) ожидаемые и получаемые выходные значения;

5) дополнительные данные, касающиеся синхронности, последовательности событий и т.п.;

6) соответствие критериям приемки, указанным в спецификации тестирований;

7) регистрация возникающих дефектов с описанием характеристик каждого дефекта.

8.2.3.2 Верификация реализации на проблемно-ориентированных языках

Использование проблемно-ориентированных языков обычно рассматривается в целях улучшения качества (т.е. для снижения уровня дефектов проекта и реализации, вносимых в технологическом процессе) и эксплуатационной надежности программного обеспечения.

8.2.3.2.1 Рекомендуется, чтобы прикладное программное обеспечение, которое автоматически генерируется из спецификации, использующей проблемно-ориентированный язык, имело систематизированную структуру с целью поддержания эффективной верификации.

8.2.3.2.2 Программное обеспечение, написанное на проблемно-ориентированных языках, должно верифицироваться на правильность и согласованность с помощью, например, визуальной проверки или с использованием автоматических инструментальных программ, которые позволяют моделировать работу программного обеспечения в режиме отладки.

8.2.3.2.3 Процесс верификации должен подтверждать, что:

- все элементы проекта правильно реализованы;

- функциональность программного обеспечения согласуется с целями, определенными в спецификации требований к программному обеспечению;

- проект программного обеспечения соответствует требованиям стандартов, указанных в плане обеспечения качества.

8.2.3.2.4 Соответствующий и обоснованный выбор методик, таких как анимация, тестирование, обзор, сквозной контроль, формализованный анализ и подтверждение, должен быть применен для улучшения понимания спецификаций и верификации их функциональной корректности и согласованности.

8.2.3.2.5 Инструментальные программы, используемые для верификации или валидации, должны быть аттестованы, как это требуется в разделе 14.

8.2.3.3 Верификация конфигурации ранее разработанного программного обеспечения

В настоящем подразделе рассматривается ситуация, когда функция безопасности категории А обеспечивается ранее разработанным программным обеспечением, конфигурированным с помощью данных конфигурации, т.е. данных, специфичных для конкретного применения. Программное обеспечение может быть связанным с семейством оборудования или быть самостоятельной программой, которая затем интегрируется с выбранными базовыми техническими средствами.

Необходимое условие применения требований данного подраздела состоит в том, что ранее разработанное программное обеспечение уже аттестовано для данного применения (см. раздел 15).

8.2.3.3.1 Верификация данных должна проводиться с использованием комбинации проверки/анализа и/или тестирований. Соответствующие средства тестирований влияния данных конфигурации должны быть проанализированы и оформлены документально (при рассмотрении роли моделирования, эмуляции, испытательных стендов и прототипов).

8.2.3.3.2 Структура документации, одобренная для любой конфигурации ранее разработанного программного обеспечения, должна позволять, при необходимости, создание промежуточных документов для того, чтобы процесс проектирования был полностью отражен в документации, например, чтобы установить, как требование ко времени отклика учитывается при конфигурировании частоты сканирования и частоты передачи буфера сообщений.

8.2.3.3.3 Процесс верификации должен подтвердить, что данные конфигурации согласуются с проектной документацией и со всеми установленными ограничениями и правилами.

Верификация может осуществляться путем визуальной проверки либо с применением инструментальных программ, либо путем комбинирования обеих методик.

8.2.3.3.4 Если данные получают с использованием инструментальных программ и если этап верификации данных предполагается пропустить с непосредственным переходом к интеграции системы, то должно быть представлено обоснование достаточной уверенности в корректности инструментальных программ.

Дальнейшее руководство по верификации данных, получаемых с использованием прикладных инструментальных программ, приведено в 14.3.5.

9 Программные аспекты интеграции системы

Процесс интеграции системы состоит в объединении верифицированных модулей технического обеспечения и программного обеспечения в подсистемы (компьютерные узлы) и, в конечном счете, в завершенную систему.

Этот процесс состоит из четырех частей:

a) сборка и взаимное соединение модулей технического обеспечения в соответствии с проектной документацией;

b) построение законченного программного обеспечения из программных модулей;

c) загрузка законченного программного обеспечения технических средств, для которых оно разрабатывалось;

d) верификация того, что:

- программное обеспечение соответствует спецификации проекта;

- требования к интерфейсу между техническим и программным обеспечениями удовлетворены;

- программное обеспечение работает в конкретном техническом обеспечении.

К программным аспектам интеграции системы относятся действия, по перечислениям b), c) и d).

В настоящем разделе приведены дополнительные требования к плану интеграции системы, а также требования, касающиеся программных аспектов интеграции системы, в дополнение к 6.1.4 МЭК 61513.

9.1 Программные аспекты плана интеграции системы

9.1.1 Данный план должен быть подготовлен и документально оформлен на этапе проектирования и верифицирован по классу 1 требований к системе.

9.1.2 План должен быть подготовлен на достаточно ранней стадии процесса разработки для того, чтобы требования по интеграции системы могли быть включены в проект системы и ее техническое и программное обеспечения.

9.1.3 В плане должны быть установлены стандарты и процедуры, которым нужно следовать при интеграции системы.

9.1.4 В данный план должны быть включены те положения плана обеспечения качества системы, которые относятся к интеграции системы.

9.1.5 В плане интеграции системы должны быть учтены ограничения в рамках любых модулей технического и/или программного обеспечения, указанные в проекте системы технического обеспечения и программного обеспечения. В план должны быть также включены требования к процедурам и методам управления, охватывающие:

- управление конфигурацией системы (см. 5.6);
- интеграцию системы;
- верификацию интегрированной системы;
- устранение дефектов.

9.1.6 В плане интеграции системы должны быть определены оба аспекта управления конфигурацией (идентификация и контроль) в соответствии с требованиями 6.2.1.2 МЭК 61513.

9.1.7 В процессе верификации отдельных модулей технического и программного обеспечения некоторые аспекты проекта этих модулей могут верифицироваться на уровне подсистем (компьютерных узлов) или на уровне завершенной системы (если это более целесообразно). Когда верификация с помощью тестирований на этих уровнях невозможна, все требования к проекту отдельного модуля должны верифицироваться другими средствами.

9.1.8 Все взаимосвязи между верификацией отдельных модулей и верификацией интегрированной системы должны быть оформлены документально в плане интеграции системы.

9.2 Интеграция системы

Конкретные процедуры интеграции системы зависят от характера проекта системы (см. перечисления а) — с), раздел 9).

9.2.1 Такие процедуры должны быть установлены в плане интеграции системы и должны включать в себя следующие действия:

- получение модулей, соответствующих плану управления конфигурацией (см. 6.2.1.2 МЭК 61513);
- интегрирование модулей технического обеспечения в систему (например, расположение модуля, адрес ячейки памяти, выбор, разводка соединений);
- объединение программных модулей и загрузку законченного программного обеспечения в соответствующее техническое обеспечение;
- предварительные функциональные тестирования функций интегрированной системы (см. приведенные ниже требования);
- документальное оформление процесса интеграции и конфигурации системы, которые будут подвергаться тестированиям;
- надлежаще оформленное предоставление системы для тестирований.

9.2.2 Если устранение дефектов требует модификации какого-либо верифицированного программного обеспечения или какого-либо проектного документа, то этот дефект должен быть отражен в отчете в соответствии с процедурой, установленной в 9.4.

Любые дефекты, обнаруженные во время интеграции системы, которые являются исключительно ошибками самого процесса интеграции и не влияют на проектную документацию, могут быть устранены без оформления отчета о дефектах.

9.3 Верификация интегрированной системы

Верификация системы определяет, должным ли образом интегрированы в систему модули и подсистемы технического и программного обеспечения, совместимы ли и работают ли так, как им предназначено.

9.3.1 Система тестирования должна быть настолько законченной, насколько это практически целесообразно для данных тестирований.

9.3.2 Контрольные тестирования, выбранные для верификации системы, должны проверять взаимосвязь модулей, а также основную работу самих модулей.

9.3.3 В плане интеграции системы должно быть показано, что моделирование любой части системы является существенным и эквивалентным реальной части.

9.3.4 В плане интеграции системы должны быть определены тестирования для каждого требования к интерфейсу каждого компьютерного узла.

9.3.5 Тестирования интегрированной системы должны анализироваться, а результаты тестирований оцениваться верификационной группой, обладающей хорошим знанием спецификации системы.

9.3.6 Оборудование, используемое для верификации системы, должно быть должным образом откалибровано.

9.3.7 Для инструментальных программ, используемых при верификации, должны быть установлены меры по обеспечению качества, соответствующие важности этих инструментальных программ верификации.

9.3.8 Верификация интегрированной системы должна подтвердить, что все компоненты системы обладают должными эксплуатационными характеристиками (например, устройства обработки и устройства связи).

9.4 Процедуры устранения дефектов

9.4.1 Процедуры, связанные с отчетом о дефектах, обнаруженных во время верификации интеграции системы, и с их устранением, должны быть установлены до начала верификации интегрированной системы.

9.4.2 Эти процедуры должны применяться ко всем дефектам, обнаруженным во время верификации системы, а также к дефектам, обнаруженным во время интеграционных функциональных тестирований и требующим модификации верифицированной системы или проектной документации на систему.

9.4.3 Данные процедуры должны обеспечивать проведение любой необходимой повторной верификации проекта системы, модулей технического или программного обеспечения в соответствии с планом управления конфигурацией системы.

9.4.4 Данные процедуры должны обеспечивать проведение любой необходимой модификации проекта системы, технического или программного обеспечения в соответствии с процедурой модификации в соответствии с разделом 11 и планом управления конфигурацией системы.

9.4.5 Должна быть проведена оценка каждого дефекта, указанного в отчете, с целью определения возможных систематических упущений, а также возможность выявления обнаруженных дефектов на более раннем этапе верификации.

9.4.6 Если установлено, что это так (т.е. дефекты должны быть обнаружены на более раннем этапе), то должно быть проведено исследование более раннего этапа верификации для определения возможных систематических упущений существующей верификации.

9.4.7 Если оценка дефектов показала, что существуют упущения в верификации, ставшие причиной того факта, что дефекты модулей программного обеспечения остались необнаруженными, то эти упущения должны быть определены, исправлены или обоснованы.

9.5 Программные аспекты отчета о верификации интегрированной системы

9.5.1 Результаты верификации интегрированной системы должны быть документально оформлены в отчете (см. приложение F).

9.5.2 В данном отчете должны быть указаны используемое техническое и программное обеспечение, используемое испытательное оборудование, параметры его калибровки и параметры установки программного и технического обеспечения, моделирование компонентов системы или интерфейса, а также любые обнаруженные по результатам тестирований расхождения вместе с корректирующими действиями в соответствии с 9.4.

9.5.3 Результаты тестирований должны сохраняться в форме, доступной для проверки лицами, непосредственно не задействованными в плане верификации.

9.5.4 Разрешение вопросов, связанных с устранением всех отмеченных в отчете дефектов, и результаты последующей оценки должны быть в достаточных деталях документально оформлены так, чтобы лица, непосредственно не задействованные в разработке системы и плане верификации, могли осуществлять их проверку.

10 Программные аспекты валидации системы

a) Для валидации системы и ее программного обеспечения по классу 1 требований к интегрированной системе должны быть проведены тестирования.

b) Валидация должна включать в себя тестирования, проводимые с системой в ее конечной конфигурации, включая конечную версию программного обеспечения.

10.1 Программные аспекты плана валидации системы

10.1.1 Валидация системы должна проводиться в соответствии с планом валидации системы.

10.1.2 В плане должны быть определены статические и динамические контрольные тестирования.

10.1.3 План валидации компьютерной системы должен разрабатываться, а результаты валидации должны оцениваться лицами, не участвовавшими в проектировании и реализации.

10.2 Валидация системы

10.2.1 Система должна проверяться с помощью статического и динамического моделирования входных сигналов, существующих при нормальной эксплуатации, при ожидаемых эксплуатационных событиях и при аварийной ситуации, требующей реакции испытываемой компьютерной системы.

10.2.2 Каждая связанная с реактором функция категории А системы должна быть подтверждена с помощью репрезентативных тестирований для каждого параметра останова или защиты как в отдельности, так и в соответствующих комбинациях.

10.2.3 Тестирования должны:

- охватывать полностью репрезентативным образом все диапазоны сигналов и расчетных параметров;
- исчерпывающим образом охватывать голосование, а также другую логику и логические комбинации;
- проводиться для всех сигналов останова или защиты в окончательной конфигурации установки;
- обеспечивать подтверждение точности и времени отклика, а также то, что при любом отказе или комбинации отказов осуществляются правильные действия;
- проводиться для всех других функций, которые непосредственно влияют на безопасность реактора (например, запреты, блокировки).

10.2.4 Кроме того, в плане валидации системы должны быть указаны необходимые входные сигналы и их значения, ожидаемые выходные сигналы и критерии приемки.

10.2.5 Используемое при валидации оборудование должно быть соответствующим образом откалибровано и конфигурировано (параметры технического и программного обеспечения).

10.2.6 Следует показать, что используемое при валидации оборудование соответствует цели валидации системы.

10.3 Программные аспекты отчета о валидации системы

10.3.1 В отчете о валидации системы должны быть отражены результаты программных аспектов валидации системы.

10.3.2 В отчете должны быть указаны техническое обеспечение, программное обеспечение и конфигурация использованной системы, а также использованное оборудование и его калибровка и использованные модели при моделировании.

10.3.3 В данном отчете также должны быть указаны любые отклонения.

10.3.4 В данном отчете должны быть обобщены результаты валидации системы.

10.3.5 В данном отчете должна быть дана оценка соответствия системы всем требованиям.

10.3.6 Результаты должны сохраняться в виде, позволяющем осуществление проверки лицами, непосредственно не участвовавшими в валидации.

10.3.7 Использованные в процессе валидации инструментальные программы должны быть указаны в отдельном пункте отчета.

10.3.8 Должно быть документально оформлено использование моделирования станции и ее систем.

10.4 Процедуры устранения дефектов

10.4.1 В плане валидации системы должны быть установлены процедуры отчета о дефектах и устранения дефектов, выявленных во время валидации.

10.4.2 Эти процедуры отчета о дефектах и устранения дефектов должны применяться ко всем дефектам, обнаруженным во время валидации системы и требующим модификации проекта системы или программного обеспечения.

10.4.3 В этих процедурах должно быть предусмотрено проведение повторной верификации проекта системы, технического и программного обеспечений в соответствии с планом управления конфигурацией системы.

10.4.4 Эти процедуры должны обеспечивать проведение модификации проекта системы и программного обеспечения в соответствии с процедурой модификации раздела 11 и планом управления конфигурацией системы.

10.4.5 Должна проводиться оценка каждого указанного в отчете дефекта с целью определения, не был ли характер дефекта таким, чтобы он мог быть обнаружен на более раннем этапе.

10.4.6 Если установлено, что дефект мог быть обнаружен на более раннем этапе, то должен быть проведен анализ этого более раннего этапа с целью обнаружения возможного систематического дефекта.

11 Модификация программного обеспечения

Модификацией программного обеспечения считается внесенное в программное обеспечение изменение, которое обычно влияет как на рабочую программу, так и на документацию.

Модификация программного обеспечения может потребоваться по следующим причинам:

- изменения функциональных требований;
- изменения в программных средствах;
- изменения в техническом обеспечении;
- отклонения, обнаруженные во время тестирований или эксплуатации.

Модификация программного обеспечения может потребоваться на этапе разработки или после поставки:

a) До реализации любой модификации программного обеспечения должна быть установлена и документально оформлена формализованная процедура управления модификацией, которая должна включать в себя требования по верификации и валидации.

b) В этой процедуре должно быть установлено, каким образом учитывать требования данного пункта.

11.1 Процедура запроса на модификацию

11.1.1 При рассмотрении вопроса модификации должны быть предприняты следующие шаги:

- составление запроса на модификацию;
- оценка запроса;
- решение.

11.1.2 В составленном запросе на модификацию должны быть однозначно определены:

- причины запроса;
- цель;
- функциональная область применения;
- инициатор;
- дата возникновения.

11.1.3 Запрос на модификацию должен быть включен в документацию по модификации программного обеспечения в качестве ее составной части. Если программное обеспечение модифицируется в рамках модификации проекта системы, то документация по модификации программного обеспечения должна входить в документацию по модификации проекта системы в качестве ее составной части.

11.1.4 Должна быть проведена независимая оценка запроса на модификацию.

11.1.5 В оценке запроса на модификацию должна быть проверена ее целесообразность с целью обеспечения того, чтобы:

- предлагаемые изменения были четко и однозначно определены;
- предлагаемые изменения исправляли причины отклонений, если изменение вытекает из отчета об отклонениях;
- предлагаемые изменения не ухудшали способность программного обеспечения обеспечивать требуемые функции категории А;
- преимущества от реализации любых изменений не превышались нарушениями, которые может внести их реализация (неправильно спроектированные и реализованные изменения могут вызвать отказы системы).

11.1.6 В запросе на модификацию должны также проверяться:

- техническая достижимость;
- влияние на другую часть системы (например, расширение памяти) или другое оборудование (например, испытательные системы); в этом случае должен оформляться запрос на модификацию, касающийся этих областей влияния;
- влияние на возможные изменения в методах, инструментальных программах или стандартах, которые должны использоваться при выполнении модификации (по сравнению с теми методами, инструментальными программами и стандартами, которые применялись при разработке версии программного обеспечения, подлежащей модификации);
- влияние на само программное обеспечение, включая перечень подверженных влиянию модулей;
- влияние на эксплуатационные характеристики (быстродействие, точность и т.п.);
- стратегия и усилия, необходимые для верификации и валидации с целью удостоверения в сохранении корректности существующего программного обеспечения; анализ необходимости проведения повторной верификации должен быть оформлен документально в поддающейся проверке форме;
- набор рассмотренных документов.

Процесс оценки может состоять из нескольких этапов.

Исходный запрос может быть рассмотрен на предмет целесообразности и достижимости до того как будет проводиться какая-либо детальная оценка влияния.

После проведения общей оценки влияния может быть проведена повторная более подробная оценка.

Запрос на проведение модификации находится на рассмотрении до принятия решения, которое может:

- отклонить запрос; в этом случае он отсылается обратно с обоснованием отказа;
- потребовать проведения детального анализа, результатом которого будет отчет об анализе модификации программного обеспечения;
- одобрить запрос и начать процесс модификации.

11.1.7 В случае, если затребован отчет об анализе модификации программного обеспечения, этот отчет должен составляться персоналом, хорошо осведомленным о программном обеспечении системы.

11.2 Процедура осуществления модификации программного обеспечения

11.2.1 Для модификаций, реализуемых на работающем оборудовании, когда невозможно провести необходимые тестирования из эксплуатационных соображений, поставщик программного обеспечения должен иметь доступ к испытательной конфигурации, идентичной реальной системе во всех значимых аспектах (включая установленную ЭВМ, транслятор, испытательные инструменты, имитатор станции и т.п.), для получения подтверждения обоснованности модификаций.

11.2.2 Процедура модификации, соответствующая какому-либо конкретному изменению, будет зависеть от этапов процесса разработки, подверженных влиянию этих изменений:

- при изменении в спецификации требований к программному обеспечению весь процесс разработки программного обеспечения для любой части системы СКУ, подверженной влиянию этого изменения, должен быть подвергнут повторной проверке;
- изменение в процессе разработки должно быть проанализировано на предмет его потенциального влияния на соответствующие нижние уровни;
- модификация должна проводиться в соответствии с правилами, приведенными в разделе 7.

11.2.3 После проведения модификации весь процесс или часть процесса верификации и валидации, описанные в 8.1 и разделе 10, должны быть вновь проведены в соответствии с анализом влияния модификации программного обеспечения (см. 11.3).

11.2.4 Все документы, подверженные влиянию модификации, должны быть откорректированы, и в них должна быть дана ссылка на запрос на модификацию программного обеспечения.

11.2.5 В отчете о модификации программного обеспечения должны быть обобщены все действия, предпринятые с целью модификации.

11.2.6 Все эти документы должны иметь дату, номер и быть зарегистрированы в архиве по управлению модификациями проекта.

11.2.7 Стратегия разработки модифицированного программного обеспечения на работающей станции должна быть оценена на предмет ее влияния на стратегию голосования, технического обслуживания и передачи данных. В зависимости от результатов этой оценки модификации программное обеспечение может вводиться постепенно, позволяя по очереди проводить тестирования нового программного обеспечения на каждой резервной линии, с параллельной работой немодифицированных линий.

11.3 Модификация программного обеспечения после поставки

Причинами такой модификации могут быть:
- отчет об аномалии;
- изменение функциональных требований после поставки;
- технологические изменения;
- изменение условий эксплуатации.

11.3.1 В случае аномалии программного обеспечения должен быть составлен отчет об аномалии, в котором должны быть приведены признаки, внешние условия для системы и состояние системы на момент обнаружения аномалии, а также указаны ее предполагаемые причины.

11.3.2 В случае возникновения после запуска неожидаемого, очевидно, неправильного необъяснимого поведения программного обеспечения обслуживающий персонал должен составить отчет об аномалии, в котором должны быть указаны подробности поведения, конфигурации технического и программного обеспечений и действия, предпринятые в это время.

11.3.3 В отчете должны быть приведены данные о составителе отчета, месте, дате, обстоятельствах и регистрационном номере. Отчеты должны рассматриваться разрабатывающей группой, назначаемой в соответствии с категориями важности и распускаемой после ответа обслуживающему персоналу.

Предпочтительно, чтобы отчет об аномалии и процедура урегулирования разрабатывались на основании аналогичного процесса, утвержденного при верификации и валидации.

11.3.4 Для устранения дефектов необходимо составление запроса на модификацию программного обеспечения, реализация которого должна проводиться с помощью процедуры, описанной в 11.1.

11.3.5 В случае изменения в спецификации требований к программному обеспечению должна быть проведена повторная проверка всего процесса разработки программного обеспечения для той части системы, которая подвержена влиянию изменения.

11.3.6 Любые новые требования к техническому обеспечению и возможности технического обеспечения должны быть проверены в отношении их возможного влияния на программное обеспечение.

11.3.7 Эта проверка должна включать в себя все суждения, касающиеся технического обеспечения и рассмотренные при проектировании первоначального программного обеспечения.

Если можно показать, что модифицированная система не влияет на спецификацию требований к программному обеспечению, то может быть применена упрощенная процедура осуществления модификации на этапах проектирования или кодирования.

В перечислении b) подпункта 6.3.6.1 МЭК 61513 рекомендуется, чтобы после завершения модификации такая модификация отражалась в комплекте документов на внесение изменений. В нем также требуется, чтобы в документах этого комплекта были описаны средства реализации модификации на работающем оборудовании либо была дана ссылка на утвержденную существующую процедуру.

11.3.8 Во всех случаях после осуществления модификации на работающем оборудовании должен быть составлен документ, в котором указывают дату осуществления модификации и результаты любых конкретных тестирований или наблюдений, необходимых в соответствии с процедурой реализации.

11.3.9 Этот документ должен быть помещен в архив управления модификацией программного обеспечения проекта.

12 Программные аспекты установки и эксплуатации

В настоящем разделе представлены требования к взаимодействию между операторами и компьютерными системами класса 1 в процессе установки и эксплуатации. Эти требования касаются:
- установки программного обеспечения на месте эксплуатации;
- защищенности программного обеспечения на месте эксплуатации;
- адаптации программного обеспечения к условиям на месте эксплуатации;
- обучения.

12.1 Установка программного обеспечения на месте эксплуатации

Должна быть предусмотрена процедура тестирований для проверки работоспособности программного обеспечения, касающейся отклика, калибровки, функциональной работы и взаимодействия с другими системами.

12.2 Защищенность программного обеспечения на месте эксплуатации

12.2.1 Должна быть проведена оценка конфигурации на месте эксплуатации и присвоение параметров с целью проверки того, что соответствующие контрмеры предприняты в отношении потенциальных угроз защищенности.

12.2.2 Если это целесообразно, то программное обеспечение должно быть конфигурировано и параметризовано так, чтобы собирать важную информацию для периодической проверки защищенности и составления отчета о СКУ.

12.2.3 Действия по модификации программного обеспечения должны систематически подготавливаться с учетом потенциальных угроз защищенности.

12.2.4 Исходя из концепции защищенности работы станции на нормальной мощности, должны быть определены особые режимы, осуществляемые при вводе в эксплуатацию и модификации программного обеспечения.

Эти режимы могут включать в себя специальные возможности программного обеспечения, которые блокируются во время работы на мощности, функции включения аварийной сигнализации, которые блокируются во время модификации, использование интерфейсов, которые блокируются или выключаются во время работы на мощности, использование станций обслуживания и инструментальных программ, а также необходимость проведения операции оператором из установленного места.

12.2.5 Любые аномалии должны быть компенсированы дополнительными мерами по обеспечению качества, такими как дополнительные административные и аналитические меры во время и/или после осуществления действий по модификации, осуществляемые с целью обеспечения работоспособности программного обеспечения.

12.2.6 Инструментальные программы и оборудование, используемое при модификации программного обеспечения на месте эксплуатации, должны выбираться в зависимости от уровня их потенциальной угрозы защищенности системы.

12.2.7 Новые массивы данных или новые версии программного обеспечения, осуществляющие связанные с защищенностью изменения, должны быть верифицированы для подтверждения того, что требования к защищенности учтены должным образом.

12.2.8 Процедура инсталляции программного обеспечения или данных на месте эксплуатации должна включать в себя и устанавливать проверки работоспособности программного обеспечения, которые проводятся до полномасштабного введения СКУ в эксплуатацию.

12.3 Адаптация программного обеспечения к условиям эксплуатации

Для предотвращения неосторожного или неправильного изменения параметров оператором, способного повлиять на уставки или другие изменяемые данные системы класса 1, должна быть предусмотрена соответствующая процедура и/или блокирующее устройство.

12.4 Обучение оператора

12.4.1 Программа обучения

Для достижения безопасной работы станции действия оператора так же важны, как и надежность оборудования.

12.4.1.1 Поэтому должна быть предусмотрена программа обучения оператора для применения систем безопасности и его программного обеспечения, предназначенная как для операторов станции, так и специалистов по контролю и управлению; при этом программа должна соответствовать сложности системы и реализуемым защитным функциям.

12.4.1.2 Программа обучения должна касаться действий при нормальных и аномальных условиях эксплуатации станции.

12.4.1.3 Программа обучения должна касаться всех важных устройств связи оператора с компьютерной системой.

12.4.1.4 В программу следует включать специальное обучение по распознаванию аномалий в работе технического и программного обеспечения.

12.4.2 План обучения

12.4.2.1 Должен быть составлен учебный план, согласующийся с принципами программы обучения.

12.4.2.2 Должно быть разработано руководство пользователя СКУ по эксплуатации и обслуживанию для применения персоналом.

12.4.2.3 В руководстве пользователя следует определить каждое устройство связи оператора. Каждая функция каждого устройства должна быть объяснена и проиллюстрирована в соответствии с ее сложностью.

12.4.3 Обучающая система

12.4.3.1 Обучение оператора должно проводиться на обучающей системе, эквивалентной реальной системе технического и программного обеспечения.

12.4.3.2 Должны быть обеспечены задающие сигналы станции с помощью испытательной системы, способной моделировать нормальные и аномальные состояния реактора.

13 Защита от отказов по общей причине, вызываемых программным обеспечением

В настоящем разделе приведены требования к защите от дефектов программного обеспечения и кодирования, способных привести к отказам по общей причине (ООП) функций, классифицированных по категории А в соответствии с МЭК 61226.

13.1 Общие сведения

ООП могут произойти в системах контроля и управления и в оборудовании, реализующих различные схемы защиты от тех же ИПС (см. пункт 5.3.1 МЭК 61513). Само по себе программное обеспечение не вызывает ООП. ООП относятся к отказам системы, возникающим в результате дефектов в функциональных требованиях, проектах системы или программного обеспечения.

МАГАТЭ требует применения «защиты в глубину» (см. 2.9 Требований МАГАТЭ NS-R-1) при всех действиях, будь то организационные, поведенческие или проектирующие действия с целью обеспечения перекрывающих друг друга защит, так чтобы в случае возникновения отказа в подсистеме он компенсировался или корректировался в интегральной системе.

В соответствии с критерием единичного отказа (см. 5.34 — 5.39 Требований МАГАТЭ NS-R-1) требуется, чтобы комплекс систем безопасности был способен соответствовать своему назначению, несмотря на единичный случайный отказ, произошедший в любой части комплекса.

13.1.1 Дефекты программного обеспечения являются систематическими, а не случайными, поэтому критерий единичного отказа не может быть применен к программному обеспечению системы так же, как он применяется к техническому обеспечению. При применении концепции защиты в глубину следует рассматривать возможные последствия ООП, вызванные программным обеспечением внутри каждого уровня защиты и между резервными уровнями защиты, и следует принимать соответствующие контрмеры во время проведения разработки и оценок (если ООП программного обеспечения является потенциальной причиной отказа), например:

1) при проектировании и реализации, верификации каждого отдельного слоя защиты и

2) при оценке независимости и разнообразия резервных защитных слоев.

Средством улучшения надежности некоторых систем и снижения вероятности определенных ООП является разнообразие (см. 4.23 — 4.31 Руководства по безопасности МАГАТЭ NS-G-1.3).

Обоснованием необходимости защиты от дефектов программного обеспечения является тот факт, что любой дефект программного обеспечения остается незамеченным в соответствующей системе или соответствующем канале до тех пор, пока не будет зарегистрирован и устранен, и он может вызвать отказ в случае, если осуществляются обращения к нему при определенной сигнальной траектории. Если две или более системы либо два или более канала, реализующие различные уровни защиты для одного и того же ИПС (см. пункт 5.3.1 МЭК 61513), имеют дефект и оказываются под воздействием определенной сигнальной траектории в период времени, когда они чувствительны к такому воздействию, то отказ способен произойти в обеих (или всех) системах и в обоих (или всех) каналах, и это называется ООП. Более подробное описание этих условий приведено в разделе G.1 приложения G.

13.1.2 Возможность возникновения ООП из-за программного обеспечения должна поэтому рассматриваться уже при проектировании. Если постулированные условия возникновения ООП могут быть спрогнозированы, то для защиты от ООП из-за программного обеспечения может потребоваться изменение проекта и применение специальных средств защиты, включая разнообразие программного обеспечения.

Степень повышения защиты от ООП, а также степень повышения надежности, которые могут быть достигнуты, не могут быть определены количественно. Решение должно приниматься на основании качественной оценки надежности, достижимой для программного обеспечения.

Если ошибки совершаются до начала проектирования программного обеспечения, то они могут привести к дефектам требований и возможным отказам системы, защита от которых не может быть обеспечена лишь средствами программотехники. Защита от таких ООП рассматривается в подпункте 5.3.1.5 МЭК 61513.

Если ошибки совершаются человеком во время процесса разработки программного обеспечения, то они могут привести к дефектам программного обеспечения и потенциальным отказам системы. Там, где такие дефекты приводят к отказам более одного уровня защиты, отказы рассматриваются как ООП из-за программного обеспечения.

13.2 Проектирование программного обеспечения с учетом ООП

Основной и наиболее важной защитой от ООП из-за программного обеспечения является создание программного обеспечения высочайшего качества, т.е. насколько это возможно свободного от ошибок. Другим важным фактором, ограничивающим возможность возникновения ООП из-за программного обеспечения, является расширение самоконтроля с помощью таких действий, как проверка диапазонов параметров и подсчет времени циклов с целью проверки достоверности данных и т.п., как это указано в 6.2, 7.1 и А.2.2 приложения А.

Использование для разработки и верификации программного обеспечения развитых методов программирования при поддержке инструментальных программ помогает уменьшить число принимаемых человеком решений и, таким образом, уменьшить число дефектов в разработанном программном обеспечении.

13.3 Источники и последствия ООП из-за программного обеспечения

13.3.1 Анализ возможности возникновения ООП из-за программного обеспечения должен быть проведен и документально оформлен на системном уровне и/или на уровне общей архитектуры контроля и управления СКУ, важных для безопасности АЭС.

Примечание 1 — Требования к архитектуре контроля и управления приведены в 5.3.1 МЭК 61513.

Примечание 2 — Требования к архитектуре отдельных систем контроля и управления приведены в 6.1.2 МЭК 61513.

13.3.2 Анализ должен включать в себя следующие шаги:

1) идентификация компонентов программного обеспечения, используемых в системе или в архитектуре контроля и управления;

2) анализ возможности возникновения ООП из-за этих компонентов в системе или архитектуре контроля и управления;

3) анализ возможных последствий этих ООП.

Примечание — Проведение анализа возможных последствий дефектов не избавляет от необходимости проведения верификации и валидации в соответствии с требованиями раздела 8 настоящего стандарта. Цель такого анализа состоит в обнаружении слабых мест в проекте и (затем) во внесении изменений в него и/или в повышении доверия к проекту программного обеспечения.

13.3.3 Если общие модули используются более чем в одной системе, то эти модули должны быть идентифицированы и должна быть проведена оценка обеспечения надежности таких общих модулей. Методы подтверждения правильности приведены в разделе G.4 приложения G.

13.3.4 Данные, передаваемые внутри компьютерной системы или между компьютерными системами, должны быть идентифицированы. Для определения того, могут ли дефектные данные привести к ООП в принимающих компьютерах или системах, должен быть проведен анализ.

13.3.5 Должна быть учтена возможность возникновения таких условий на станции, когда одно и то же программное обеспечение, функционирующее в различных технических средствах, будет подвержено воздействию идентичных или одновременных сигнальных траекторий и, следовательно, обнаружится один и тот же дефект в нескольких каналах или функциональных частях.

Примечание — Отказы могут быть вызваны сигнальными траекториями, которые не рассматривались во время проектирования, верификации и валидации программного обеспечения отдельных каналов или систем.

13.3.6 Деятельность по модификации программного обеспечения (см. раздел 11) может стать причиной ООП, поэтому процессы, используемые для оценки изменения программного обеспечения или данных, должны обеспечивать уверенность в отсутствии вносимых дефектов.

13.3.7 Анализ возможности возникновения ООП из-за программного обеспечения должен быть проведен и документально оформлен как часть оценки защиты от ООП проекта архитектуры контроля и управления (см. 5.3.3 МЭК 61513).

13.3.8 В случае, если в результате анализа обнаружена неприемлемая угроза ООП, возникающая из-за программного обеспечения, проект программного обеспечения или архитектура контроля и управления должны быть исправлены. Методы реализации защиты от ООП приведены в разделе G.3 приложения G, а методы реализации элементов разнообразия — в разделе G.5 приложения G.

13.4 Реализация разнообразия

13.4.1 При реализации разнообразия следует использовать независимые системы с функциональным разнообразием. Если функциональное разнообразие неуместно или невозможно, то следует рассмотреть системное разнообразие, разнообразие элементов программного обеспечения и разнообразие подходов к проектированию. Признаки важности разнообразия приведены в разделе G.5 приложения G. Методы, выбранные для защиты от ООП, должны быть оформлены документально и обоснованы с помощью соответствующего анализа.

13.4.2 На уровне программного обеспечения защиту от ООП следует основывать на выборе соответствующих методов, таких как:

1) обеспечение различных условий работы программного обеспечения;

2) защита от ошибки и распространения отказа;

3) снижение отрицательного воздействия ООП;

4) использование программного обеспечения, имеющего различные спецификации для различных реализаций одних и тех же функциональных требований.

Примечание 1 — Различия в методах проектирования и реализации следует рассмотреть, но это не является обязательным требованием.

Примечание 2 — N-версионное программирование не рекомендуется.

13.5 Баланс недостатков и преимуществ, связанных с использованием разнообразия

Если в программном обеспечении используется разнообразие, то следует обосновать и задокументировать недостатки и преимущества, касающиеся общей надежности программного обеспечения, на основе вышеуказанного анализа (см. 4.27 Руководства по безопасности МАГАТЭ NS-G-1.3 и 3.81 — 3.85 Руководства по безопасности МАГАТЭ NS-G-1.2). Аспекты потенциальных преимуществ, недостатков, а также аспекты обоснований приведены в разделе G.6 приложения G.

14 Инструментальные программы для разработки программного обеспечения

14.1 Общие сведения

В настоящем подразделе существующие требования настоящего стандарта представлены в расширенном виде с тем, чтобы охватить инструментальные программы, используемые при разработке программного обеспечения компьютеров в системах безопасности атомных электростанций.

Использование соответствующих инструментальных программ уменьшает число ошибок в процессе разработки и, следовательно, повышает надежность программного продукта путем снижения риска внесения дефектов во время этого процесса. Использование инструментальных программ может также иметь экономические выгоды, поскольку уменьшает время и усилия, необходимые для получения программного обеспечения. Инструментальные программы могут применяться для автоматической проверки соблюдения правил структурирования и требований стандарта, формирования соответствующих записей и согласованной документации в стандартных форматах, а также для поддержания управления изменениями. Инструментальные программы могут также уменьшить усилия, необходимые для тестирований, а также выполнения автоматизированных записей. Необходимость в инструментальных программах может возникнуть при специфичной методологии разработки, требующей их применения.

14.1.1 Инструментальные программы особенно эффективны в тех случаях, когда они работают совместно. Следует соблюдать осторожность и не возлагать на инструментальные программы выполнение задач, находящихся за пределами их возможностей, например, они не могут заменить человека при принятии решений. В некоторых случаях использование инструментальных программ дает больше, чем полная автоматизация процесса. При использовании инструментальных программ должен быть найден баланс

между выгодами и рисками, связанными с их применением, а также баланс между выгодами и рисками, связанными с отказом от их применения. Важный принцип состоит в том, чтобы при выборе инструментальной программы ограничивалась вероятность совершения ошибки и внесения дефекта, а также обеспечивалась максимальная вероятность регистрации дефектов.

В число инструментальных программ, рассматриваемых в настоящем стандарте, включены те из них, которые используются для фиксирования требований, а также те, что применяются для реализации требований в коде и данных конечной системы (могут существовать промежуточные шаги). В настоящем стандарте рассматриваются также инструментальные программы, непосредственно используемые при проведении верификации, валидации и тестирований, инструментальные программы для подготовки прикладных данных и управления ими (см. 14.3.5), а также инструментальные программы для контроля и управления процессами и продукцией, используемыми при разработке программного обеспечения.

Автономные инструментальные программы, используемые для расчета важных переменных, применяемых при проектировании и проведении анализа оборудования, важного для безопасности, не входят в область применения настоящего стандарта. В область применения настоящего стандарта также не входят текстовые процессоры, инструментальные программы для контроля проектирования и другие офисные программы, косвенно связанные с разработкой программного обеспечения.

14.2 Выбор инструментальных программ

14.2.1 Инструментальные программы для разработки программного обеспечения систем класса 1 должны выбираться так, чтобы обеспечить процесс их программирования. Критерии и процесс выбора описаны в 14.3.1. Должна быть определена и документально оформлена область применимости всех инструментальных программ. Инструментальные программы и их выходные данные не должны использоваться без предварительного обоснования, вне заявленной области их применения.

14.2.2 Инструментальные программы, используемые при разработке программного обеспечения систем класса 1, должны быть верифицированы и оценены в той степени, которая соответствует требованиям к надежности инструментальных программ, их типу [см. 14.2.3, перечисления 1) — 5)] и вероятности внесения дефектов.

14.2.3 Инструментальные программы должны иметь достаточную надежность для того, чтобы не ухудшать надежность конечной программы. Например, инструментальная программа может отрицательно влиять на разработку программного обеспечения путем внесения ошибок, выработки искаженных выходных данных, неспособностью зарегистрировать уже существующий дефект.

Для снижения требований к надежности отдельных инструментальных программ при их выборе могут быть рассмотрены принципы «защиты в глубину» и разнообразие, принятые для архитектуры контроля и управления.

Степень требуемых верификации и оценки также зависит от типа инструментальной программы и от того, возможна ли полная верификация и валидация выходных данных инструментальной программы. Существуют следующие типы инструментальных программ:

1) преобразующие инструментальные программы, такие, например, как генераторы кода, компиляторы программы, преобразующие текст или диаграмму из одного уровня абстракции в другой, обычно более низкий;

2) инструментальные программы для верификации и валидации, такие, например, как статические анализаторы кода, мониторы тестового покрытия, вспомогательные средства для доказательства теорем и имитаторы;

3) диагностические инструментальные программы, используемые для поддержания и контроля нахождения программного обеспечения в рабочих условиях;

4) инструментальные программы инфраструктуры, такие, например, как системы поддержки разработки;

5) инструментальные программы управления конфигурацией, такие, например, как инструментальные программы управления версией.

14.3 Требования к инструментальным программам

Требования к инструментальным программам представлены по следующим темам:

a) средства разработки программ;

b) аттестация инструментальной программы;

c) управление конфигурацией инструментальных программ;

d) трансляторы/компиляторы;

e) инструментальные программы для прикладных данных;

f) автоматизация тестирований.

14.3.1 Средства разработки программ

14.3.1.1 Инструментальные программы следует использовать для поддержания всех аспектов жизненного цикла программного обеспечения, если существует выгода от их использования и если инструментальные программы имеются в наличии. Должен быть проведен анализ средств разработки программ и процесса разработки программного обеспечения для определения стратегии обеспечения инструментальной поддержкой. Результаты анализа должны быть оформлены документально. При отсутствии инструментальных программ может возникнуть потребность в рассмотрении возможности разработки новых инструментальных программ.

Ниже приведены примеры процессов и операций, для которых может оказаться выгодным применение инструментальных программ:

1) создание и проверка спецификации, проектирования и реализации (см. приложение Н);

2) инструментальные программы, работающие на языке или его сокращенном варианте (см. 14.3.4);

3) подготовка, верификация и валидация прикладных данных, а также управление ими (см. 14.3.5);

4) автоматизация тестирований (см. 14.3.6).

14.3.1.2 Следует разработать критерии и приоритеты для выбора и оценки инструментальных программ для разработки программного обеспечения с тем, чтобы обеспечить возможность их альтернативного выбора. Критерии следует структурировать по характеристикам качества программного обеспечения, как это определено в ИСО/МЭК 9126: функциональность, надежность, удобство использования, эффективность, модифицируемость и компактность. Критерии могут включать в себя другие характеристики, такие как затраты на лицензирование и ресурсы, необходимые для использования инструментальной программы, строгость плана обеспечения качества, по которому разрабатывалась инструментальная программа, информация об инструментальной программе от поставщика и альтернатива применению инструментальной программы.

14.3.1.3 Поддержка средств разработки программ с помощью инструментальных программ должна быть проанализирована и документально оформлена с выяснением следующих аспектов:

1) каким образом каждый процесс поддерживается или не поддерживается инструментальными программами;

2) точная идентификация инструментальных программ (например, название, номер версии) и, по возможности, их конфигурации;

3) как каждая инструментальная программа будет использоваться в проекте;

4) каким образом выходные данные каждой инструментальной программы будут проходить верификацию и/или валидацию по отношению к входным данным;

5) каким образом другие инструментальные программы или процессы смягчают последствия дефекта в инструментальной программе, включая смягчение вероятных ошибок при формировании и подготовке данных для применения в режиме «онлайн»;

6) как данные инструментальные программы связаны с другими инструментальными программами, т.к. для них может потребоваться применение, обработка и передача информации, используемой другими инструментальными программами или частью архива данных;

7) каким образом инструментальная программа осуществляет согласованный интерфейс с пользователем и остальными средствами разработки программ;

8) насколько инструментальные программы соответствуют выбранным методам разработки программного обеспечения;

9) способность инструментальной программы регистрировать ошибки и реагировать на особые ситуации;

10) насколько инструментальные программы соответствуют конкретным условиям использования, включая пользователей, оборудование, среду и задачи пользователя для достижения максимальной эффективности и минимального влияния ошибок пользователя;

11) каким образом инструментальные программы препятствуют несанкционированному или неправильному использованию и изменениям.

14.3.1.4 Стратегия модификации, обновления или замещения инструментальных программ должна быть документально оформлена и обоснована. Эта стратегия является частью стратегии модификации операционного программного обеспечения, которая должна обеспечивать возможность адаптации или корректировки операционного программного обеспечения в течение всего периода его применения на АЭС.

Эта стратегия должна также обеспечивать обоснованность перехода к новой версии инструментальной программы, а также соответствующую аттестацию, т.е. оценку в соответствии с требованиями настоящего стандарта.

14.3.1.5 Для инструментальных программ, используемых для обеспечения разнообразия, т.е. для компиляторов, применяемых для разработки многоверсионных разнородных систем программного обеспечения, следует подтвердить их разнородность. Этого можно достичь путем подтверждения того, что:

1) все инструментальные программы были получены от различных поставщиков (например, одна программа может быть разработана, а другая — приобретена в готовом виде) или

2) инструментальные программы имеют различные языки на входе и/или на выходе или

3) инструментальные программы имеют различные требования и процессы проектирования.

14.3.2 Аттестация инструментальных программ

14.3.2.1 Должна быть составлена стратегия аттестации инструментальных программ, и аттестация этих программ должна проводиться в соответствии с данной стратегией. В стратегии должны быть учтены требования к надежности инструментальных программ и тип инструментальных программ.

14.3.2.2 Должны быть определены качественные требования по надежности с учетом:

1) последствий дефекта в инструментальной программе;

2) вероятность того, что инструментальная программа вызовет или активизирует дефекты в программном обеспечении, реализующем функцию безопасности;

3) какие другие инструментальные программы или процессы смягчают последствия дефектов в данной инструментальной программе.

Примечание — Принципы «защиты в глубину» и разнообразие могут способствовать снижению требований к надежности.

14.3.2.3 В стратегии аттестации инструментальных программ должны быть учтены:

1) анализ процесса разработки инструментальной программы и информация о ней от поставщика;

2) адекватность документации по инструментальной программе, позволяющая проведение верификации ее выходных данных и обеспечивающая простоту изучения;

3) тестирования и валидация инструментальной программы;

4) оценка инструментальной программы за период ее применения;

5) информация по опыту применения инструментальной программы.

Примечание — Раздел 15 содержит требования к применению ранее разработанного программного обеспечения, что также следует учитывать в стратегии аттестации инструментальных программ.

14.3.2.4 Выходные данные инструментальной программы должны систематически подвергаться верификации (например, посредством тестирований, анализа или сравнения с выходными данными функционально аналогичных инструментальных программ), если эти выходные данные включаются в конечное программное обеспечение.

14.3.2.5 Инструментальные программы должны быть объектом оценки в соответствии с тем, как это описано в разделе 15, либо они должны разрабатываться в соответствии с требованиями к обеспечению качества в соответствии с разделами с 1 — 12, за исключением случаев, когда:

- инструментальная программа не может внести дефекты в программное обеспечение (например, текстовый редактор для документации) либо

- имеются средства смягчения всех возможных дефектов инструментальных программ (например, путем разнообразия процессов или проектирования системы [см. перечисление 5) пункта 14.3.1.3], либо

- выходные данные инструментальной программы подвергаются систематической верификации [см. перечисление 4) пункта 14.3.1.3]. В процессе аттестации может быть учтен опыт предшествующего использования инструментальной программы, где была подтверждена ее адекватность применению, важному для безопасности.

14.3.3 Управление конфигурацией инструментальной программы

14.3.3.1 Все инструментальные программы должны находиться под управлением конфигурацией для обеспечения полной идентификации выбранных инструментальных программ (включая название, версию, вариант и, возможно, конфигурацию) и параметры инструментальной программы, используемой для формирования основного программного обеспечения (см. 5.6).

Примечание — Это полезно не только для согласованности конечного программного обеспечения, но также помогает оценивать источник возникновения дефекта, который может находиться в исходном коде, инструментальной программе или параметрах инструментальной программы. Это может также оказаться необходимым при оценке вероятности возникновения ООП из-за инструментальных программ.

14.3.3.2 Записи, в которых документально оформляются произошедшие ошибки и ограничения на инструментальные программы, должны сохраняться в течение всего срока службы инструментальной программы, чьи выходные данные способны внести дефект в конечное программное обеспечение.

14.3.3.3 Любые изменения в инструментальной программе должны быть верифицированы и оценены.

14.3.4 Трансляторы/компиляторы

В данном пункте представлены требования, относящиеся к трансляторам/компиляторам. Размеры и сложность многих компиляторов таковы, что трудно бывает продемонстрировать правильность его работы. Однако обширный опыт использования может повысить степень доверия к тому, что компилятор работает правильно.

14.3.4.1 Трансляторы и компиляторы следует выбирать, руководствуясь существенными для трансляторов/компиляторов критериями данного пункта (которые дополняют рекомендации приложения D).

14.3.4.2 Трансляторы/компиляторы не должны без предупреждения удалять защитные программы или функции проверки ошибок, вводимые программистом.

14.3.4.3 Следует избегать использования оптимизации компилятора. Она не должна использоваться, если в результате получается объектный код, чрезмерно сложный для понимания, отладки, тестирования и валидации.

Примечание — Оптимизация кода может использоваться для удовлетворения требований к эксплуатационным характеристикам, возникающим в результате ограничений быстродействия технических средств и пределов памяти. В исключительных случаях альтернативное использование ассемблерного кода может рассматриваться в дополнение к изменению платформы технических средств.

14.3.4.4 Если была применена оптимизация, то для оптимизированного кода должны быть проведены тестирования, верификация и/или валидация.

14.3.4.5 Библиотеки, которые используются в конечной системе, должны рассматриваться как наборы ранее разработанных компонентов программного обеспечения. Используемые компоненты библиотеки должны быть оценены, аттестованы и использоваться в соответствии с требованиями раздела 15, касающимися аттестации ранее разработанного программного обеспечения.

14.3.4.6 Для обеспечения правильности любого дополнительного кода (команд ассемблера), вводимого транслятором, который непосредственно не отслеживается до оператора исходной строки (например, код проверки ошибок, код обработки ошибок и исключительных ситуаций), должны быть проведены верификация и/или валидация.

14.3.5 Инструментальные программы прикладных данных

Для компьютерных систем безопасности обычно требуются прикладные данные для определения сигналов, адресов и функциональных параметров прикладных функций и функций обслуживания. Данные могут быть обширными и обычно содержат следующую информацию:

- указатели признаков сигналов, описания сигналов, положение источников и нумерация кабелей, типы измерений, электрические диапазоны или состояния, технические устройства, определения состояний, вызывающих сигнал тревоги, аварийные уровни и уровни блокировки;
- места подключения внешнего сигнала, адреса и указатели баз данных, адреса и указатели информации, адреса и характеристики технических средств, расположение приборов в системе отображения информации, информация о символах на экране дисплея, информация о цветовых параметрах на экране дисплея, информация о содержании сигнала на экране дисплея, форматы записей и внутренних блоков передаваемой информации, а также детали ее содержания;
- коды защитных действий, приоритеты или логика аварийной сигнализации, выходные данные для действий, идентификация логических операций и таймеры, состояния на выходах, которые должны возникать при отказе.

Данные могут быть взяты из проектных чертежей, перечней и спецификаций операций и технологической контрольно-измерительной аппаратуры на станции. Данные будут транслированы для загрузки в специализированный процессор системы, а затем использованы для управления работой программного обеспечения в режиме «онлайн».

Требования, относящиеся к подготовке, верификации и валидации и управлению данными для их использования в режиме «онлайн», приведены ниже:

14.3.5.1 Проект прикладных данных пакета программ и метод их получения из прикладных данных станции должны быть определены и документально оформлены.

14.3.5.2 Прикладные параметры, которые могут быть изменены во время действий оператора, должны быть идентифицированы вместе с методами, используемыми для управления изменениями таких параметров.

14.3.5.3 Изменения, внесенные в модифицируемые данные, не должны искажать другие данные и код в исполняющей системе.

14.3.5.4 Формальная сторона процедур верификации данных должна быть аналогичной формальной стороне процедур верификации и валидации программного обеспечения, включая идентификацию и выявление ошибок. Должны быть проведены сквозные верификационные проверки, и эти проверки должны включать в себя все этапы преобразования данных, начиная с извлечения данных из информации о проекте станции и до включения структур данных в программное обеспечение, работающее в режиме «онлайн», в том числе использование средств передачи.

14.3.5.5 Данные, загружаемые в программное обеспечение, работающее в режиме «онлайн», следует представлять в форме, которая позволяет провести их распечатку и верификацию, либо должна использоваться инструментальная программа, которая воспринимает данные и хранит их в форме, допускающей их верификацию.

14.3.5.6 Должно быть обеспечено наличие оборудования, позволяющего провести верификацию всех загруженных данных конфигурации на месте применения.

14.3.5.7 Если данные определяют интерфейс между двумя системами, рекомендуется, чтобы предоставляемые данные генерировались из одной и той же базы данных (см. 5.3.1.4 МЭК 61513).

14.3.5.8 В некоторых случаях, когда работа программного обеспечения, включая процессы, поток данных и входные и выходные соединения, управляется или модифицируется данными конфигурации, специальное, документально оформленное обоснование должно подтверждать соответствующий уровень оценки, после чего к данным должны быть применены тестирования. После изменения таких данных могут потребоваться обширные повторные тестирования системы.

14.3.6 Автоматизация тестирований

Автоматизация увеличивает число тестирований, которые могут быть проведены за данный период времени. Этого можно достичь, выполняя следующие требования:

14.3.6.1 Рекомендуется, чтобы инструментальные программы автоматизированной валидации, которые вырабатывают испытательные данные, передают или преобразуют эти данные и результаты тестирований, а также оценивают результаты тестирований, осуществляли полное составление протокола. Это применимо как к тестированиям модулей, так и к моделированиям станции.

14.3.6.2 Соответствующие инструментальные программы следует использовать при тестировании и/или моделированиях поведения рабочей программы, загружаемой в целевую систему.

14.3.6.3 Соответствующие инструментальные программы должны использоваться для обеспечения или верификации того, что нужная рабочая программа правильно загружается в конечную систему.

14.3.6.4 Следует рассмотреть применение следующих дополнительных инструментальных программ:

1) тестовые генераторы, анализаторы тестового покрытия и тестовые драйверы;

2) диагностические программы, работающие в режиме «онлайн», с проверкой состояния памяти и оборудованием для трассировки;

3) отладочные программы с отладочным оборудованием на уровне исходного кода;

4) автоматизированные тестовые наборы для упрощения регрессивного тестирования.

15 Аттестация ранее разработанного программного обеспечения

15.1 Общие сведения

В настоящем подразделе представлены требования к использованию ранее разработанного программного обеспечения (РПО) в компьютерных СКУ. Эти требования установлены как часть требований к аттестации систем, в которые интегрировано РПО (см. 6.4 МЭК 61513).

РПО для СКУ может быть представлено от небольших компонентов программного обеспечения (например, библиотечные модули прикладных функций) до больших и сложных программных продуктов (например, части операционных систем или драйверов коммуникаций). По отношению к техническим средствам РПО можно разделить на два типа:

a) универсальное РПО, которое не разрабатывалось специально для конкретных технических средств, и

b) РПО, интегрированное в технические средства, которое должно использоваться совместно с этими техническими средствами.

Компоненты РПО называются «компонентами многократного использования», если они могут быть использованы в различных компьютерных программах или системах, например, как часть комплекса оборудования (платформы оборудования). Компоненты, не зависящие от деталей конкретного применения на станции, могут рассматриваться как аттестованные для применения в системе, выполняющей функции категории А.

В спецификациях новых систем безопасности часто ссылаются на использование ранее разработанного оборудования, включая РПО, для построения части или всей «новой системы» (см. 6.1.2.1 МЭК 61513). Использование ранее разработанного оборудования может оказаться выгодным с точки зрения продуктивности и надежности систем, если эти элементы имеют соответствующее качество и вводятся должным образом. Если элементы РПО использовались во многих приложениях, аналогичных тому, для которого они предназначены, то этот опыт работы может быть заявлен при их оценке. В частности, повторное использование РПО, прошедшего валидацию, способно повысить уверенность в надежности системы.

15.2 Общие требования

15.2.1 РПО, предполагаемое к использованию, как и любой другой компонент программного обеспечения, как часть системы, выполняющей функции категории А, должны соответствовать всем требованиям настоящего стандарта.

15.2.2 При оценке РПО следует:

1) определить соответствие РПО требованиям по функциональности, эксплуатационным характеристикам и архитектуре, входящим в спецификацию требований к системе (см. 6.1.1 МЭК 61513), и определить ее конечную пригодность;

2) определить любые модификации, необходимые для корректировки или адаптации РПО;

3) оценить качество РПО и

4) оценить опыт эксплуатации РПО, если это необходимо для указанных выше оценок.

15.2.3 Выводы по оценке должны быть оформлены документально.

Примечание — В соответствии с требованиями настоящего стандарта применяют оценку для определения действий, проводимых организацией, ответственной за разработку компьютерной системы (или осуществляемых от имени этой организации); ни в коем случае не подразумевается и не требуется, чтобы оценка проводилась разрешительными органами, хотя они и имеют право ее проводить.

15.3 Процесс проведения оценки

Процесс проведения оценки должен включать в себя:

a) оценку функциональных и эксплуатационных особенностей РПО и существующей аттестационной документации (см. 15.3.1).

Примечание — Для РПО, интегрированной в программу, эти особенности могут быть выражены как свойства программы в соответствии с МЭК 61069-2;

b) оценку качества процесса разработки программного обеспечения (см. 15.3.2).

Примечание — Для ранее разработанного программного обеспечения многоразового использования необходимо проводить лишь оценку пригодности; подразумевается, что оценка качества осуществляется при валидации;

c) оценку опыта эксплуатации, если необходимо компенсировать слабости результатов по перечисленниям a) и b) (см. 15.3.3) и

d) комплексную итоговую документально оформленную оценку выводов из вышеуказанных оценок и связанную с этим дополнительную работу, которая дает возможность принять РПО для использования в системе.

Взаимосвязи между различными этапами процесса оценки РПО показаны на рисунке 4.

Взаимосвязи между процессом оценки РПО и аттестацией системы показаны на рисунке 5.

Примечание — Процесс, описанный в настоящем подразделе, является упрощенным и не отражает всех повторений или пересечений между действиями по оценке и действиями по разработке компьютерной системы.

1 Оценка пригодности (см. 15.3.1)

Необходимая входная документация (см. 15.3.1.1)	
Документация по спецификации системы	Спецификация РПО и документация пользователя

Требования по оценке (см. 15.3.1.2)	
Сравнение спецификаций системы и РПО	Идентификация модификаций и пропущенных пунктов

Выводы		
РПО пригодна	Необходима дополнительная работа	Должно быть отвергнуто

2 Оценка качества (см. 15.3.2)

Необходимая входная документация (см. 15.3.2.1)		
Проектная документация	Документация по жизненному циклу	(архивные эксплуатационные документы)

Требования к оценке (см. 15.3.2.2)		
Анализ проекта	Анализ обеспечения качества	Идентификация пропущенных пунктов

Выводы		
Качество жизненного цикла РПО приемлемо либо необходимые модификации выполнимы	Требуются дополнительное тестирование и документация либо требуется дополнительная оценка опыта эксплуатации	РПО должно быть отвергнуто

3 Оценка опыта эксплуатации (см. 15.3.3)

Необходимая входная документации (см. 15.3.3.1)		
Набор данных	Время эксплуатации	Архивные данные о дефектах

Требования по оценке (см. 15.3.3.2)

Выводы		
Достаточный опыт эксплуатации	Опыт эксплуатации еще не достаточен	РПО должно быть отвергнуто

4 Комплексная итоговая оценка (см. 15.3.4)

Качество РПО приемлемо	Необходимые модификации проведены

5 Интеграция в систему и обслуживание (см. 15.4)

Рисунок 4 — Схема процесса аттестации ранее разработанного программного обеспечения

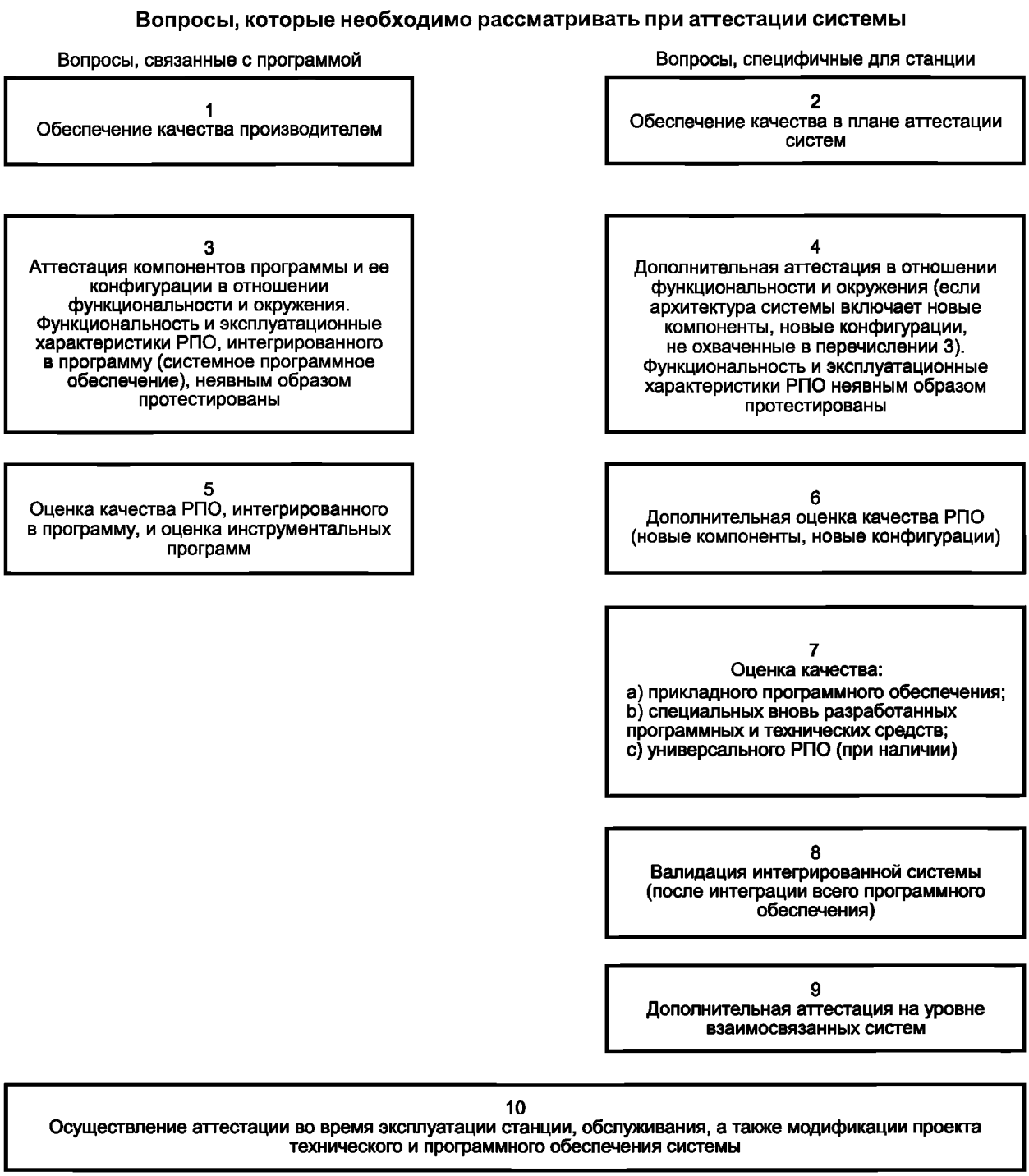


Рисунок 5 — Связь оценки РПО с планом аттестации системы, в которую оно интегрировано

15.3.1 Оценка пригодности

Цель этого процесса заключается в подтверждении того, что спецификация РПО по функциональности, эксплуатационным характеристикам и архитектуре соответствует спецификации требований. В этом процессе определяются компоненты, непосредственно пригодные для использования в системе станции, а также компоненты, требующие модификации.

Примечание — Оценка основана на анализе спецификаций и рабочей документации.

Проведение оценки пригодности следует начинать на раннем этапе системной спецификации (см. 6.2 МЭК 61513), чтобы:

- помочь проектантам при архитектурном проектировании системы;

- получить проверяемое свидетельство того, что функциональность и эксплуатационные характеристики РПО соответствуют требованиям системы.

15.3.1.1 Необходимая входная документация

15.3.1.1.1 Должны быть доступны:

- документация по системной спецификации, определяющая требования к функциональности, интерфейсу и эксплуатационным характеристикам, которым должна соответствовать РПО в рамках системной архитектуры (см. 6.1.1 и 6.1.2 МЭК 61513);

- описание РПО и документация пользователя. В этих документах должны быть в явном виде определены характеристики, важные для соответствия спецификациям системы по функциональности и эксплуатационным характеристикам. Если характеристики в явном виде не определены, то должны быть проведены анализ или тестирование.

15.3.1.1.2 РПО должна находиться под управлением конфигурации; должны быть точно известны версия и конфигурация РПО.

15.3.1.2 Требования к оценке пригодности

15.3.1.2.1 Спецификации РПО должны быть оценены в отношении спецификации требований системы (см. 6.1.1 МЭК 61513). В случае расхождений РПО должна быть отвергнута либо модифицирована, либо спецификация требований должна быть адаптирована так, чтобы разрешить возникшие расхождения при условии сохранения функций, важных для безопасности.

15.3.1.2.2 При необходимости модификации РПО должна быть проведена оценка, основанная на проектной документации РПО, с целью определения возможности проведения изменений способом, согласующимся с требованиями настоящего стандарта. Если изменения не могут быть внесены, то использование РПО должно быть отвергнуто.

Примечание — Оценка качества РПО показывает выполнимость этих модификаций (см. 15.3.2.2). Соответствующую реализацию проводят в рамках жизненного цикла системы (см. раздел 6 МЭК 61513).

15.3.1.2.3 Для РПО, содержащего библиотеку, за исключением случаев, когда оценке подлежит вся библиотека, должна иметься возможность перестройки библиотеки с образованием ограниченной библиотеки, удовлетворяющей потребностям программного обеспечения, и возможность соединения программы с этой ограниченной библиотекой, которая должна состоять из компонентов, прошедших оценку.

15.3.1.2.4 При оценке пригодности должны быть определены функции, которые включены в РПО, не предназначенные для системы и не востребованные ею, и меры, обеспечивающие беспрепятственное выполнение этих функций безопасности.

15.3.1.2.5 По завершении оценки должен быть составлен документ, в котором записывают:

1) согласуются ли спецификации РПО по функциональности и эксплуатационным характеристикам со спецификацией требований системы и

2) если РПО непригодна — основания для ее отклонения.

15.3.2 Оценка качества

Цель данной оценки состоит в получении свидетельства того, что элементы проекта РПО соответствуют системе, выполняющей функции категории А, и что должное обеспечение качества осуществляется в течение жизненного цикла РПО. Эта оценка основана на документации по проекту и плану качества для РПО, но для нее может также потребоваться анализ архива по эксплуатации.

15.3.2.1 Входная документация

15.3.2.1.1 Должна быть доступна в дополнение к перечисленной в 15.3.1.1:

- документация по спецификации системы, в которой определена важность для безопасности функций, реализуемых с помощью РПО в архитектурном проекте системы (см. 6.1.2 и приложение А МЭК 61513);

Примечание — Необходимый для достижения с помощью оценки качества доверительный уровень того, что РПО будет работать, как предписано, будет различным для трех категорий, при этом для категории А требуется более высокий доверительный уровень (см. 8.2.1 МЭК 61226).

- аттестационная документация РПО, включая документацию по предыдущим сертификациям или независимым оценкам РПО, если она должна использоваться при оценке.

15.3.2.1.2 Должна быть представлена относящаяся к РПО либо обоснованно использована следующая информация:

- план качества программного обеспечения (разделение на простые задачи и соответствующие действия), используемый в жизненном цикле РПО (см. раздел 5), и соответствующие записи для задач и процедур по обеспечению качества (в особенности, планирование верификации);
- документы по спецификации, проекту, реализации и модификации и соответствующие верификационные документы;
- план интеграции технического и программного обеспечений и соответствующая верификация;
- план валидации и тестирования, осуществляемые над программой поставщиком или заказчиком.

Примечание — Для последних двух пунктов перечисления эта документация плана необходима только в случае, если РПО интегрировано в компоненты технического обеспечения.

15.3.2.1.3 Документация по опыту эксплуатации РПО должна быть доступна в случаях компенсации недостатка указанной документации или обоснования использования опыта, отличающегося от указанного в настоящем стандарте.

15.3.2.1.4 Рекомендуется, чтобы в документации была представлена информация о производственных факторах, таких, например, как распространение РПО и поддержка поставщиком РПО.

15.3.2.2 Требования к оценке качества

15.3.2.2.1 Требования плана качества программного обеспечения для РПО и соответствующие верификация и документация должны быть оценены на соответствие требованиям настоящего стандарта. Этот анализ соответствия требует интерпретации с целью определения требований, применимых в контексте использования РПО в системе.

15.3.2.2.2 Проект РПО должен быть согласован с ограничениями по архитектуре и детерминированному внутреннему режиму работы системы.

15.3.2.2.3 Если при разработке РПО используются методы, отличные от приведенных в приложениях к настоящему стандарту, то их пригодность должна быть проанализирована и обоснована в соответствии с 5.5. Их важность для обеспечения качественных характеристик программного обеспечения должна быть оценена во взаимосвязи с требованиями системы. Результаты оценки и анализа должны быть зафиксированы для независимого рассмотрения.

15.3.2.2.4 При расхождении с требованиями настоящего стандарта свойства, которые не могут быть верифицированы, слабости или пропущенные шаги в процессе верификации или оформления документации должны быть выявлены. Каждому выявленному элементу должен быть присвоен ранг, соответствующий его важности в обеспечении качественных характеристик программного обеспечения, а также важности для безопасности функций, реализованных в системе. Руководство по ранжированию несоответствий приведено в пункте I.1 приложения I.

15.3.2.2.5 Если системы, выполняющие функции категории А, включают в себя функции более низкой категории, которые должны выполняться РПО, а архитектурный проект системы таков, что РПО может потенциально подвергнуть опасности функции категории А (см. 6.2 МЭК 61513), то к этому РПО должны быть применены критерии оценки, применимые к РПО, выполняющему функции категории А.

15.3.2.2.6 В документации по аттестации должны быть представлены свидетельства того, что РПО, интегрированная в компоненты технических средств, прошла валидацию с целью подтверждения ее соответствия функциональным и эксплуатационным спецификациям.

Примечание — Функциональное и эксплуатационное поведение компонентов РПО может быть в неявном виде аттестовано с помощью функциональной аттестации отдельного оборудования, в которое эти компоненты интегрированы (см. 2 на рисунке 5). Однако существуют свойства, которые могут быть аттестованы только с использованием конфигурирования оборудования.

15.3.2.2.7 Когда компоненты РПО содержат элементы, которые не могут быть подвергнуты валидации вне окончательной конфигурации системы, валидация этих элементов должна осуществляться в окончательной конфигурации системы.

15.3.2.2.8 Качество и степень тестового покрытия при валидации, осуществляемой для РПО, должны быть оценены с учетом требований разделов 9 и 10 и, при необходимости, должны быть проведены дополнительные валидационные тестирования.

15.3.2.2.9 Когда оценка проекта и жизненного цикла завершена, должен быть подготовлен документ, в котором была бы зафиксирована верность одного из следующих утверждений:

1) качество РПО подтверждено и никаких дополнительных тестирований или анализа опыта эксплуатации не требуется;

2) при наличии оценки для данной конфигурации системы должна быть проведена дополнительная аттестация;

3) во время оценки был зафиксирован недостаток информации, но он может быть компенсирован проведением дополнительной верификации и валидации, тестированием или анализом кода и документацией;

4) во время оценки был зафиксирован недостаток информации, который может быть компенсирован использованием опыта эксплуатации;

5) РПО (или часть РПО) требует модификации для данного использования в системе (см. 5.3.3), и поскольку оно имеет соответствующий уровень качества, то желательно, но не обязательно, чтобы модификации проводились в соответствии с требованиями настоящего стандарта.

Примечание — После проведения установленных модификаций необходима дополнительная оценка в рамках аттестации системы (см. 5 на рисунке 5);

6) могут возникнуть серьезные проблемы при перенесении РПО на новые технические средства;

7) качество РПО не соответствует необходимому уровню и РПО должно быть отвергнуто на основании того, что имеющиеся недочеты слишком велики или имеющаяся информация недостаточна для эффективной их компенсации, и

8) установлена/неустановлена независимость аттестованных функций/свойств РПО от тех функций/свойств, которые не были аттестованы.

15.3.3 Оценка опыта эксплуатации

Цель данной оценки состоит в получении свидетельства того, что соответствующий опыт эксплуатации РПО может повысить доверие к РПО в случае наличия недостатков, зафиксированных во время оценки качества.

Должны быть определены функции/свойства РПО, которые подлежат оценке на основании опыта эксплуатаций, и:

1) методы сбора данных об опыте эксплуатации;

2) методы регистрации времени эксплуатации версии РПО и формирования эксплуатационного архива;

3) эксплуатационный архив отчетов о полученных данных, дефектах и ошибках и

4) эксплуатационный архив о модификациях, выполненных из-за дефектов или по другим причинам.

15.3.3.1 Валидация входных данных и методы получения эксплуатационного архива

15.3.3.1.1 Оценка опыта эксплуатации РПО основывается на данных, полученных от поставщика, и, если возможно, от пользователей систем, в которых работает данная РПО. Для того, чтобы опыт эксплуатации был признан пригодным для оценки РПО, должны быть оценены методы сбора данных и формирования эксплуатационного архива.

15.3.3.1.2 Должна использоваться только информация, полученная с помощью хорошо определенного и управляемого процесса сбора данных.

15.3.3.1.3 Процедура сбора должна быть оценена с целью валидации данных на их полноту и достоверность. Руководство по сбору и валидации данных приведено в разделе I.2 приложения I.

15.3.3.1.4 Опыт эксплуатации должен считаться пригодным только в случае, если он отслеживался при условиях, аналогичных тем, которые будут существовать при планируемой эксплуатации.

15.3.3.1.5 Должно быть установлено суммарное время работы оцениваемого РПО. Оно может быть рассчитано сложением значений времени работы на каждой из установок, для которых собран опыт эксплуатации и проведена его валидация. Значение времени, в течение которого РПО не работало в типовом режиме, не учитывают.

15.3.3.1.6 Время работы берут для той версии РПО, которая будет использоваться. Если в расчет включают время работы других версий, проводят анализ различий и архива для этих версий. Анализ с целью подтверждения пригодности архива должен определить отличающиеся части и функции РПО, а также части и функции, на которые не влияют модификации.

15.3.3.1.7 Проблемы, отказы и их устранение в различных версиях РПО должны быть проанализированы и классифицированы в соответствии с их значимостью. Должно быть оценено их влияние на выполняемые функции.

15.3.3.1.8 На аттестованные функции не должны влиять ошибки или модификации, обнаруженные или сделанные для других функций той же РПО.

15.3.3.1.9 При оценке коммуникационных функций следует учитывать опыт эксплуатации. Следует определить рабочие пределы в сравнении с прогнозами для нормальных условий, условий с пиковой нагрузкой и условий отказа оборудования.

15.3.3.1.10 Опыт эксплуатации следует считать пригодным, если удовлетворяются следующие критерии:

1) для РПО накоплено достаточное время работы (см. приложение I).

Примечание — Достаточное время работы следует определять для каждого случая отдельно на основе инженерной оценки. В этой оценке следует учитывать, главным образом, ожидаемый уровень надежности, требующийся на уровне системы для функций, в которых используется РПО;

2) не было осуществлено никаких существенных модификаций и не было зафиксировано никаких ошибок в течение значительного времени работы в нескольких местах применения и

3) предпочтительно, чтобы РПО работало на нескольких установках.

15.3.3.1.11 После завершения оценки опыта эксплуатации должен быть оформлен документ, в котором фиксируется, что:

1) либо опыт эксплуатации пригоден для установленных функций/свойств РПО с обоснованием того, почему этот опыт эксплуатации может быть использован для оценки качества,

2) либо опыт эксплуатации непригоден или недостаточно обоснован.

15.3.3.2 Критерии приемки, применяемые при использовании опыта эксплуатации в качестве компенсирующего фактора

Соответствующий опыт эксплуатации может быть использован в качестве компенсирующего фактора для приемки РПО, если удовлетворяются следующие критерии:

15.3.3.2.1 Оценка данных по опыту эксплуатации не должна полностью заменять саму оценку проекта программы и связанной с ним документации (см. 15.3.2.2).

15.3.3.2.2 Пригодный опыт эксплуатации должен быть принят как часть обоснования, если он используется только для компенсации слабых мест в оценке РПО в отношении рекомендаций перечисления c) раздела B.2 приложения B, касающихся операционных систем и стандартных программ.

15.3.3.2.3 Строгость анализа данных опыта эксплуатации должна соответствовать категории безопасности функций системы, а свидетельства технической правильности, получаемые из этого анализа, должны согласовываться со свидетельствами, получаемыми при применении разделов 1 — 12.

15.3.3.2.4 Строгость анализа данных опыта эксплуатации должна соответствовать категории безопасности функций системы, а свидетельства технической правильности, получаемые из этого анализа, должны согласовываться со свидетельствами, получаемыми при применении разделов 1 — 12.

15.3.3.2.5 По результатам проведенной оценки должен быть оформлен документ, в котором фиксируют:

1) насколько удовлетворительно данные опыта эксплуатации компенсируют любые слабые места, определенные в оценке проекта и жизненного цикла РПО, или

2) использование РПО отклоняется, т.к. результаты оценки отрицательные или опыт эксплуатации недостаточен для компенсации слабых мест, определенных в разработке.

15.3.4 Комплексная оценка

15.3.4.1 Когда оценки и вся необходимая дополнительная работа (модификации РПО, дополнительные тестирования, дополнительная документация) завершены, должен быть подготовлен документ по комплексному обоснованию использования РПО для реализации в системе.

15.3.4.2 В этом документе, основанном на выводах из оценок, указанных в 15.3.1, 15.3.2 и 15.3.3, должна быть зафиксирована оценка, которая подтверждает факт, что РПО (или часть РПО) пригодна и имеет уровень качества, соответствующий предполагаемому ее использованию в системе, и что никаких дальнейших модификаций не требуется.

15.4 Требования к интеграции в систему и модификации РПО

15.4.1 После комплексной оценки должно быть принято официальное решение об использовании РПО, которое должно быть документально оформлено в рамках программы, следующей за формализованным рассмотрением проекта.

15.4.2 Процедуры интеграции РПО должны быть описаны в планах обеспечения качества системы и интеграции системы (см. 6.2.1 и 6.2.3 МЭК 61513).

15.4.3 После приемки РПО должна быть введена под управление конфигурацией системы (см. 6.2.1.2 МЭК 61513) и должен использоваться только вариант, подвергшийся указанной аттестации и некоторым необходимым модификациям, определенным в процессе приемки.

15.4.4 В плане качества системы должны быть предусмотрены процедуры обновления РПО, когда необходимым становится использование нового варианта.

15.4.5 Информацию об ошибках и отказах из-за РПО на других установках и в других применениях, а также информацию о соответствующих модификациях РПО следует оценить и проанализировать в течение последующей эксплуатации.

Приложение А
(обязательное)

Жизненный цикл безопасности программного обеспечения и детализация требований к программному обеспечению

А.1 Жизненный цикл программного обеспечения

Процесс разработки программного обеспечения представлен на рисунке 3, на котором показаны деятельность по жизненному циклу, начиная со спецификации, включая проектирование и реализацию и заканчивая верификацией и валидацией.

Детализация требований к программному обеспечению описана в разделе А.2.

А.2 Детализация требований к программному обеспечению

А.2.1 Описание взаимных ограничений между техническим и программным обеспечениями

А.2.1.1 Должны быть описаны следующие аспекты:

- общие рабочие характеристики (разрядность, типы обмена, быстродействие и т.п.); во многих случаях достаточно бывает ссылки на справочники производителя оборудования;
- рабочие характеристики специального оборудования (конкретные драйверы, оборудование для передачи данных и т.п.);
- числовые ограничения;
- требования к комплектам стандартных программ;
- требования к самоконтролю технических средств;
- должна быть дана, по крайней мере, ссылка на документ с требованиями к техническим средствам.

А.2.1.2 Должны быть определены взаимные требования к техническим и программным обеспечениям с учетом регистрации отказа и реакции на выявление отказа.

Критерии, установленные в Руководстве МАГАТЭ NS-G-1.3, не допускают никакого дополнительного расширения, касающегося технического обеспечения компьютерных систем. Однако необходима документация ко всем требованиям к техническому обеспечению, влияющим на программное обеспечение, для того чтобы обеспечить основу интеграции технического и программного обеспечений и валидацию компьютерной системы.

А.2.1.3 Должно быть описано взаимодействие между функциями, принадлежащими к разным уровням безопасности (например, функциями категории А и функциями других категорий), и выполняемым программным обеспечением.

А.2.2 Самоконтроль

А.2.2.1 Рекомендуется, чтобы при отказах программного обеспечения автоматически предпринимались соответствующие действия с учетом следующих факторов:

- отказы должны быть определены с разумной степенью детализации;
- должен быть гарантирован максимально возможный отказоустойчивый выход;
- если такой гарантии нельзя предоставить, то несоответствие выхода системы должно касаться только менее существенных для безопасности требований;
- последствия отказов должны быть минимизированы;
- должны быть рассмотрены возможности включения корректирующих программ, таких, например, как возвращение в предыдущее состояние, восстановление системы;
- обслуживающему персоналу должна быть представлена достаточно подробная информация об отказах.

А.2.2.2 Система должна быть спроектирована так, чтобы был возможен соответствующий самоконтроль. Принципы проектирования, используемые для осуществления этой возможности, включают в себя:

- разбиение на модули;
- промежуточные проверки достоверности;
- использование резервирования и разнообразия; разнообразие может быть реализовано в качестве функционального разнообразия или разнообразия программного обеспечения;
- обеспечение достаточного времени исполнения и объема памяти для целей самопроверки;
- для подтверждения адекватности элементов самоконтроля может быть использовано моделирование отказов.

А.2.2.3 В любых обстоятельствах самоконтроль не должен препятствовать своевременному реагированию системы.

А.2.3 Представление спецификаций программного обеспечения

А.2.3.1 Спецификации программного обеспечения должны быть легко понимаемыми всеми группами пользователей.

А.2.3.2 Представление должно быть достаточно подробным, свободным от противоречий и, по возможности, не содержать повторений.

А.2.3.3 Документ должен быть свободен от деталей по реализации, быть полным, последовательным и современным.

А.2.3.4 В документе по спецификации программного обеспечения должны четко различаться существенные требования и менее обязательные цели.

Документ по спецификации программного обеспечения предназначен для использования:

- его авторами;
- заказчиком, клиентом и конечным пользователем;
- группой по разработке программного обеспечения;
- группой по верификации программного обеспечения;
- персоналом по оценке и лицензированию.

Приложение В
(обязательное)

Детализированные требования и рекомендации по проектированию и реализации

В настоящем приложении приведены таблицы, каждая из которых начинается с основного требования (должно быть), такого как в таблице В.1а, за которым помещен перечень связанных с ним рекомендаций (следует), таких как в таблице В.1а (пункт В.1аа).

Для каждого требования в проектах следует отобрать рекомендации, которым будут следовать при проектировании и реализации. Следует обосновать отклонение оставшихся рекомендаций (например, «не применимо» или «охвачено другими мерами» и т.п.).

Требования и рекомендации перечислены в таблицах В.1а — В.5f.

В.1 Процесс проектирования

Таблица В.1а — Модифицируемость

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
В.1а	Процесс проектирования должен сделать ПО легко модифицируемым	Риска внесения дефектов при реализации изменений /
В.1аа	Характеристики разрабатываемого ПО и его функциональные требования, которые, вероятно, будут изменяться в течение жизненного цикла, следует определить на раннем этапе проекта	/ достижения безопасности и экономически эффективной гибкости
В.1аb	Модули следует выбирать так, чтобы воздействие ожидаемых изменений на ПО было ограничено на последующих этапах проектирования	/ минимизации риска возникновения дефектов из-за изменений
В.1ас	Модифицируемость следует тщательно уравновесить с ее сложностью, временем выполнения и объемом памяти	Получения слишком сложной системы/

Таблица В.1b — Подход «сверху вниз»

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
В.1b	При проектировании необходимо использовать подход сверху вниз	Ошибок проекта / полноты проекта
В.1ba	Общие вопросы предшествуют специальным	Риска несогласованности / логически последовательной работы разработчиков от предъявления требований до окончания проекта
В.1bb	На каждом этапе проектирования всю систему следует полностью описать и подвергнуть верификации	/ согласованности и полноты проекта; наиболее раннего обнаружения участков, создающих проблемы
В.1bc	На возможно более раннем этапе проектирования следует определить все проблемные участки	/ включения проблемных участков в качестве входной информации для принятия решений по проектированию
В.1bd	Принципиальные решения следует обсуждать и документально оформлять как можно раньше	/ наиболее ранней оценки достижимости проекта. Снижения вероятности изменений на более поздних этапах разработки ПО
В.1be	После принятия любого крупного решения, влияющего на другие части системы, следует рассмотреть альтернативы и документировать их факторы риска	Дублирования работ / тщательного проектирования

Окончание таблицы В.1b

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.1bf	Следует определить последствия, оказываемые отдельными решениями на другие части системы	Дублирования работ / тщательного проектирования
B.1bg	Разрыв между уровнями проектирования ПО должен быть таким, чтобы те, кто проводит анализ, могли понять каждый уровень проектирования в его взаимосвязи с предыдущим уровнем	Проектов, трудных для понимания / ясности проекта ПО и согласованности проекта, подлежащего верификации
B.1bh	Следует проводить проектирование и разработку ПО, используя одно или несколько формализованных описаний высокого уровня (там, где это целесообразно и эффективно), подобно тому, как это делается в математической логике, теории множеств, а также использовать псевдокод, таблицы решений, логические схемы, другие графические средства или проблемно-ориентированные языки	Неправильной интерпретации или неточности /
B.1bi	Следует использовать автоматические средства разработки	/ сокращения области ошибок человека
B.1bj	Документацию следует составлять так, чтобы автор спецификации был в состоянии понять и проверить реализацию функций в проекте	/ модификаций и соответствия спецификациям

Таблица B.1c — Верификация промежуточных результатов проекта

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.1c	Промежуточные результаты проекта должны верифицироваться	/ наиболее быстрого обнаружения ошибок, полноты проекта
B.1ca	Следует показать полноту и самосогласованность каждого уровня проекта	Необходимости последующих изменений/
B.1cb	Следует показать, что каждый уровень проекта согласуется с предыдущим уровнем и со спецификацией требований к ПО, существенных для данного уровня	Пропущенных аспектов / отсутствия пропущенных требований
B.1cc	Проверки согласованности следует проводить персоналу, не вовлеченному в процесс разработки	—
B.1cd	Этому персоналу следует только отмечать недостатки, но не давать никаких рекомендаций	Привязки конкретных лиц к программе / сохранения критического отношения
B.1ce	Где необходимо, им следует давать четкие пояснения, чтобы помочь разработчикам правильно понять обнаруженные недочеты	/ повысить интерес к деятельности по верификации и общую эффективность работы группы

Таблица B.1d — Управление модификаций в процессе разработки

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.1d	Внесение необходимых изменений в процессе разработки программы должно начинаться на самом раннем этапе проектирования, на котором еще можно вносить изменения	Внесения новых дефектов в результате изменений/ отсутствия скрытых, имеющих отдаленные последствия дефектов
B.1db	Если какой-либо модуль изменяется, то для него следует провести повторное тестирование в соответствии с описанными ранее принципами (см. 11.2) до его повторной интеграции в системе	Скрытых дефектов в модуле, вызванных изменением/

Окончание таблицы В.1d

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.1dc	Кроме того, связанные с данным модулем другие модули, которые к нему обращаются и к которым он обращается, следует подвергнуть повторному тестированию, поскольку на них также влияет изменение	Скрытых дефектов в связанных модулях, вызванных изменением/
B.1dd	В документацию по существенным изменениям следует включать требования, части программы, области данных, характеристики управляющей логики, временные аспекты, подверженные влиянию	/ отслеживаемости результатов изменения
B.1de	Изменения, влияющие на уже протестированные части или на работу других людей, должны быть оценены и проанализированы до их внесения	/ скрытых, имеющих отдаленные последствия эффектов
B.1def	Примечание — Эта процедура применима для изменений, влияющих на работу только одного человека, и для модификаций, влияющих на всю систему. В последнем случае дополнительно применяются рекомендации раздела 11.	—

B.2 Структура программного обеспечения

Таблица B.2a — Структуры управления и оценки

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.2a	Программы и их части должны систематически группироваться	/ облегчения проведения оценки и модификации
B.2aa	Рекомендуется, чтобы специальные операции системы выполнялись специальными частями	/ тестируемости
B.2ab	ПО следует разбивать на части так, чтобы аспекты, связанные с такими функциями, как: - внешние интерфейсы компьютера (например, управление устройствами, обработка прерываний); - сигналы реального времени (например, часы); - параллельная обработка (например, блок оперативного управления); - размещение памяти; - специальные функции (например, утилиты); - размещение стандартных функций на технических средствах конкретного компьютера, были отделены от прикладных программ с хорошо определенными интерфейсами между ними	/ улучшения тестируемости, ясности проекта
B.2ac	Рекомендуется, чтобы структура программы позволяла реализовывать ожидаемые изменения при минимуме усилий (см. также таблицу B.1a)	/ адаптируемости системы
B.2ad	Следует четко формулировать используемые методы структурирования	/ понятности
B.2ae	Насколько возможно, последовательность выполнения программы на одном процессоре	Путаницы из-за временных проблем или различных последовательностей прерывания /
B.2af	Однозначная и четкая модульная структура компьютерной программы	/ понятности, тестируемости

Таблица B.2b — Модули

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.2b	Модули должны быть ясными и понятными	/ понятности
B.2ba	Каждый модуль должен соответствовать определенной функции	/ тестируемости
B.2bb	Модуль должен иметь только один вход. Хотя иногда могут потребоваться множественные выходы, тем не менее рекомендуется использовать один выход	/ простоты верификации
B.2bc	Размер модулей не должен превышать предела для выполняемых операторов. Этот предел устанавливается для конкретной системы и только в специальных случаях допускаются более длинные модули	Громоздких модулей /
B.2bd	Интерфейсы между модулями следует делать насколько возможно простыми, единообразными по всей системе и полностью документально оформленными	Ошибок в интерфейсах /
B.2be	Следует свести к минимуму число параметров интерфейсов модулей	Ошибок в интерфейсах /
B.2bf	Статус параметров интерфейсов модулей (т.е. вход, выход или вход/выход) следует четко устанавливать	Ошибок в интерфейсах /

Таблица B.2c — Операционное системное программное обеспечение

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.2c	Обращение к операционной системе должно быть ограничено	Отказов из-за ошибок в операционной системе / упрощения верификации системы
B.2ca	Следует использовать только тщательно проверенные операционные системы, снабженные соответствующей документацией о верификации; если операционная система является РПО, то см. раздел 15	Скрытых ошибок/
B.2cb	Следует избегать использования универсального системного программного обеспечения (операционных систем)	Использования чрезмерно сложных программных продуктов/
B.2cc	Если универсальная операционная система необходима, то ее применение следует ограничить небольшим числом простых функций	/ обоснованного использования тщательно проверенной операционной системы
B.2cd	Операционная система должна содержать только необходимые функции	«Мертвых» программ/
B.2ce	Функции операционной системы должны всегда называться одинаково	/ простоты верификации
B.2cf	Функции, используемые для управления техническими средствами, следует брать из операционной системы или разрабатывать и верифицировать внутри этой системы	Дополнительных усилий по программированию и тестированию/
B.2cg	Функции операционной системы должны быть строго определены и иметь хорошо определенные интерфейсы	Ошибок при использовании функций/
B.2ch	Условия использования и взаимосвязи функций операционной системы должны быть известными и проверенными	Ошибок при использовании функций/
B.2ci	Необходимо следовать указаниям настоящего стандарта в целом, если операционная система или ее часть разработана для специального применения в целях безопасности	/ простоты верификации

Таблица B.2d — Время выполнения

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.2d	Влияние характера физического процесса на время выполнения следует удерживать на низком уровне	Труднообъяснимых временны́х проблем/
B.2da	Время выполнения любой системы или части системы в условиях пиковой нагрузки должно быть небольшим по сравнению с временем выполнения, после которого нарушаются требования безопасности системы	Необходимости изменений на поздних стадиях/
B.2db	Результаты, связанные с последовательной программой, не должны зависеть: - от времени, необходимого для выполнения программы, а также - от времени (отнесенного к независимым «часам»), когда начинается выполнение программы	Труднообъяснимых временны́х проблем/ определенности
B.2dc	Компьютерные программы следует проектировать так, чтобы операции выполнялись в правильной последовательности, не зависимой от быстроты выполнения	Проблем синхронизации и проблем времени прогона / облегчения анализа
B.2dd	Время прогона не должно существенно изменяться в результате изменений входных данных	/ упрощения оценки времени прогона и верификации времени прогона
B.2de	Значение изменения времени прогона, которое может быть вызвано входными данными, должно быть документально оформлено	/ упрощения оценки времени прогона и верификации времени прогона
B.2df	Части кода, время выполнения которых зависит от входных данных, должны быть короткими	/ достижения цели B.2de
B.2dg	Объем данных, считываемых в течение одного вычислительного цикла, должен быть постоянным	/ поддержания небольшой разницы во времени прогона
B.2dh	Время выполнения программы не должно быть связано с поступлением данных	Проблем синхронизации и проблем времени прогона / облегчения анализа

Таблица B.2e — Прерывания

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.2e	Необходимо ограничивать применение прерываний	Проблем синхронизации и проблем времени прогона / облегчения анализа
B.2ea	Прерывания могут использоваться, если они упрощают проект ПО и не делают верификацию чрезмерно сложной	/ упрощения понимания специальных конфигураций
B.2eb	Критические части ПО (например, критических во времени, критических в отношении изменения данных) следует определить и защитить	Проблем синхронизации и проблем времени прогона / облегчения анализа
B.2ec	ПО может запрещать обработку прерываний во время прохода критических частей. Такие запреты следует обосновывать	Проблем синхронизации и проблем времени прогона / облегчения анализа
B.2ed	Если прерывания используются, то для непрерываемых частей необходимо иметь оценки максимального времени вычислений, чтобы можно было рассчитать максимальное время, в течение которого прерывание запрещено	Проблем со временем прогона/
B.2ee	Применение или блокирование прерываний должно быть тщательно оформлено документально	/ валидации системы

Таблица B.2f — Арифметические выражения

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.2f	По возможности, вместо сложных арифметических выражений необходимо использовать простые	Побочных эффектов / простоты тестирования
B.2fa	Решения не должны зависеть от громоздких арифметических расчетов	/ нахождения реверсивных формул расчета функций
B.2fb	Следует использовать по возможности упрощенные, ранее верифицированные арифметические выражения	/упрощения демонстрации соответствия между реализуемой функцией и ее кодом
B.2fc	Если используются громоздкие арифметические выражения, то их следует кодировать таким образом, чтобы по коду можно было легко показать их соответствие определенному арифметическому выражению	/ облегчения анализа программы

B.3 Самоконтроль

Таблица B.3a — Проверки достоверности

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.3a	Необходимо проводить проверки достоверности (страхующее программирование)	Возможных оставшихся дефектов/
B.3aa	Правильность или достоверность промежуточных результатов следует периодически проверять	Возможных оставшихся дефектов/
B.3ab	Следует проверять области изменения: - входных переменных; - выходных переменных; - промежуточных параметров, включая проверку границы массива	Возможных оставшихся дефектов/
B.3ac	Отрицательные результаты проверки достоверности должны фиксироваться	/ диагностики возможных оставшихся дефектов

Таблица B.3b — Безопасная выходная информация

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.3b	Если обнаружена неисправность, то система должна выдавать четко определенную выходную информацию	Распространения дефектов / безотказной работы
B.3ba	Если можно, то следует использовать методы полного и корректного устранения ошибок	Поломки системы из-за мелких отказов /
B.3bb	Если используются методы устранения ошибок, то необходимо фиксировать появление любой ошибки	Накопления ошибок / раннего устранения ошибок
B.3bc	Необходимо регистрировать появление постоянных ошибок, влияющих на функционирование системы	Накопления ошибок / раннего устранения ошибок

Таблица В.3с — Содержимое памяти

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
В.3с	Содержимое памяти должно быть защищено или контролироваться. Примечание — Защита может иметь целью предотвращение (предотвратить осуществление несвоевременной модификации) или регистрацию (зарегистрировать ненормальное состояние — искажение памяти) с соответствующей быстрой реакцией (ограничение или коррекция ошибки). Регистрация предполагает мониторинг (или его синоним — контроль)	Несанкционированных изменений, возможных оставшихся дефектов /
В.3са	Память, отведенная для констант и команд, должна быть защищена или должен осуществляться контроль изменений	Распространения ошибок адресации или отказов аппаратных средств, включая перемежающиеся отказы /
В.3cb	Не следует допускать неправомочных считываний и записей	—
В.3сс	Систему следует страховать от изменений кода или данных оператором станции	/ сохранения целостности разрешенной системы

Таблица В.3d — Проверка ошибок

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
В.3d	Проверку ошибок необходимо проводить на уровне кодирования	Распространения отказа /
В.3da	Счетчики и ловушки отклонений от допустимости должны гарантировать правильный прогон программы	Специфических дефектов логики управления, повторяющихся дефектов технических средств/
В.3db	Следует проверять правильность передачи любого параметра, включая проверку типа параметров	Дефектов в проектировании интерфейса и потока данных /
В.3dc	При адресации массива следует проверять его границы	Ошибок в потоке данных, слишком большого числа повторений в цикле /
В.3dd	Следует контролировать время прогона критических частей (например, с помощью таймера)	Дефектов проектирования управления, а также слишком большого числа повторов в цикле /
В.3de	Следует использовать логические утверждения (например, в треугольнике, если НЕ (a+b>c), то Ошибка)	/ достоверности промежуточных результатов

В.4 Детальное проектирование и кодирование

Таблица В.4а — Ветвления и циклы

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
В.4а	С ветвлениями и циклами следует обращаться осторожно	/ понятного и верифицируемого алгоритма управления
В.4аа	Следует избегать использования переходов назад, вместо этого следует использовать операторы цикла (только для языков высокого уровня)	Трудностей анализа алгоритма управления / удобочитаемости
В.4ab	Следует исключить переходы в циклы, модули или подпрограммы	Трудностей анализа алгоритма управления / удобочитаемости
В.4ас	Следует избегать переходов из циклов, если они не ведут к полному окончанию цикла. Исключение — выход по отказу	Трудностей анализа алгоритма управления / удобочитаемости

Окончание таблицы В.4а

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4ad	В модулях со сложной структурой для более четкого выделения структуры следует использовать макрокоманды, процедуры или подпрограммы	Трудностей анализа алгоритма управления / удобочитаемости
B.4ae	В качестве особой меры обеспечения доказательства правильности и верификации программы следует избегать вычисляемых операторов GOTO, а также переменных типа «метка»	—
B.4af	В тех случаях, когда используется список альтернативных ветвлений или операторов, управляемых вариантами, список условий ветвления или вариантов должен быть исчерпывающим. Концепция «вариант по умолчанию» должна быть зарезервирована для обработки сбоя	/ внесения ясности, исключающей «или»
B.4ag	Следует использовать циклы только с постоянными максимальными областями значений переменной цикла	Проблем с временем прогона, нарушениями границ массива / обозримости числа проходов

Таблица B.4b — Подпрограммы

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4b	Подпрограммы следует организовывать как можно проще	Ненужной сложности /
B.4ba	Они должны иметь только заранее определенное максимальное число параметров	/ поддержания программ и интерфейсов простыми и короткими
B.4bb	Они должны связываться с окружением исключительно через свои параметры	/ простоты понимания потока данных, анализа потока данных
B.4bc	Подпрограммы должны иметь только одну точку входа	/ простоты понимания алгоритма управления, анализа алгоритма управления
B.4bd	Подпрограммы должны для каждого вызова подпрограммы возвращаться только к одной точке. Исключение — выход по умолчанию	/ простоты понимания алгоритма управления, анализа алгоритма управления
B.4be	Точка возврата должна следовать непосредственно за точкой вызова	/ простоты понимания алгоритма управления, анализа алгоритма управления

Таблица B.4c — Вложенные структуры

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4c	С вложенными структурами следует обращаться осторожно	/ понятности
B.4ca	Следует избегать вложенных макрокоманд	Получения чрезмерно сложного кода /
B.4cb	Следует избегать объединения различных вариантов действия программы посредством оператора вложенного цикла или вложенных процедур, если они скрывают взаимосвязь между структурой задачи и структурой программы	/ получения подходящих структур следования
B.4cc	Следует использовать иерархии процедур и циклов, если они проясняют структуру системы	/ показа различных уровней абстракции при нисходящей (сверху вниз) разработке

Таблица B.4d — Адресация и массивы

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4d	Следует использовать простые методы адресации	/ упрощения анализа потока данных
B.4da	Следует использовать только один метод адресации для каждого типа данных	/ единообразия интерфейса с базой данных
B.4db	Следует избегать громоздких вычислений индексов	/ лучшего понимания потока данных
B.4dc	Массивы должны иметь фиксированную, определенную заранее длину	Трудностей, связанных с временем прогона, усложнения алгоритма управления / упрощения анализа потока данных
B.4dd	Размерность в каждой ссылке на массив должна быть равна размерности в соответствующей декларации этого массива	/ понимания процесса адресации

Таблица B.4e — Структуры данных

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4e	Структуры данных и соглашения о наименовании данных должны использоваться в единообразной форме по всей системе	/ анализа потока данных и интуитивного понимания значения элементов данных
B.4ea	Переменные, массивы и ячейки памяти должны иметь единственное назначение и структуру. Следует избегать использования эквивалентности	Ошибок в использовании данных, искусственных проблем синхронизации / отслеживаемости потока данных
B.4eb	Название каждой переменной должно отражать ее применение	/ интуитивного понимания элемента данных
B.4ec	Константы и переменные величины следует располагать в различных частях памяти	Порчи данных и кода / самоконтроля аппаратных средств
B.4ed	Когда в качестве глобальной структуры используется «база данных» с возможностью универсального доступа или аналогичные ресурсы, то доступ к ним должен обеспечиваться через стандартные процедуры обработки ресурсов или связь со стандартными задачами манипулирования ресурсами	—
B.4ee	Программы, которые получают данные из других программ или передают данные в другие программы, должны обмениваться согласованными наборами данных	Несогласованности данных /

Таблица B.4f — Динамические изменения

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4f	Следует избегать изменений исполняемого кода	/ анализа алгоритма управления

Таблица B.4g — Тестирования устройств и комплексные тестирования

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4g	Тестирования устройств и комплексные тестирования должны проводиться во время разработки программы	/ максимально быстрого обнаружения дефектов

Окончание таблицы В.4g

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.4ga	Подход к тестированиям должен соответствовать подходу к проектированию (например, при проектировании сверху — вниз тестирования следует проводить с использованием моделирования еще не существующих частей системы — фиктивных модулей; после завершения разработки системы должны следовать интеграционные тестирования снизу — вверх)	/ максимально быстрого обнаружения дефектов
B.4gb	Следует тщательно тестировать каждый модуль до включения его в систему и документально оформлять результаты испытаний	Трудностей после интеграции / управления изменениями
B.4gc	Следует составлять формализованное описание входных тестовых данных и результатов (протокол тестирования)	Дублирования работы / ускорения получения лицензии на ПО
B.4gd	Следует регистрировать и анализировать дефекты, обнаруженные во время тестирования программы	/ обнаружения отдельных ошибок проектирования
B.4ge	Следует фиксировать незавершенные испытания	/ ясности
B.4gf	Для облегчения использования результатов тестирования устройств и комплексного тестирования во время окончательной валидации следует фиксировать ту степень, которая была достигнута при прошлых тестированиях (например, все маршруты через испытуемый модуль)	Дублирования работ /

5 Рекомендации, зависящие от языка

Таблица B.5a — Последовательности и оформление

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.5a	Необходимо разработать подробные правила оформления различных языковых конструкций	/ получения программных листингов единообразной понятной формы
B.5aa	В рекомендации следует включать последовательность деклараций, в том числе типы параметров	—
B.5ab	Последовательность инициализаций	—
B.5ac	Последовательность неисполняемого кода / исполняемого кода	—
B.5ad	Последовательность описания форматов (например, для таких языков, как ФОРТРАН)	—

Таблица B.5b — Комментарии

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.5b	Взаимосвязи между комментариями и кодом должны быть зафиксированы в подробных правилах	Трудностей как с написанием, так и с пониманием комментариев / получения осмысленных комментариев
B.5ba	Должен быть ясен предмет комментирования	—
B.5bb	Расположение комментариев должно быть одинаковым	—
B.5bc	Форма и стиль комментариев должны быть единообразными	—

Таблица B.5c — Ассемблер

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.5c	Если используется язык ассемблера, то необходимо следовать расширенным и документально оформленным правилам программирования	Трудностей программирования на ассемблере, применения трюков / простоты и понятности
B.5ca	Нельзя использовать команды ветвления с подстановкой адреса. Содержание таблицы ветвления должно быть постоянным	Ветвлений, цель которых нельзя идентифицировать по коду в точке ветвления/
B.5cb	Вся косвенная адресация должна следовать одной и той же схеме	/ ясного понимания положения ячеек памяти, адресуемых разными способами
B.5cc	Следует избегать косвенных смещений	/ обеспечения возможности сразу видеть, насколько происходит смещение
B.5cd	Следует избегать множественных подстановок или множественного индексирования в пределах одной машинной команды	/ легкого нахождения ячеек памяти
B.5ce	Одни и те же макрокоманды всегда должны вызываться с одним и тем же числом параметров	/ понимания функций макрокоманд
B.5cf	Метки (именованные ячейки) следует использовать для осуществления ссылок внутри программы. Следует избегать численных значений (абсолютных адресов или относительных сдвигов)	/ обеспечения ассоциации с целью ветвления
B.5cg	Соглашения о вызове подпрограммы должны быть единообразными в пределах всей системы и оговариваться правилами	Произвольных типов параметров и адресов ячеек/

Таблица B.5d — Правила кодирования

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.5d	Необходимо выпустить подробные правила кодирования	/ улучшения ясности и согласованности кода
B.5da	Должно быть понятным предназначение строк кода и как это предназначение реализуется	/ идентификации блоков
B.5db	Компоновка модулей должна быть единообразной	/ понимания структуры модулей
B.5dc	Дальнейшие детали следует регулировать в соответствии с потребностями	—

Таблица B.5e — Проблемно-ориентированные языки

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.5e	Использование проблемно-ориентированных языков предпочтительней использования машинно-ориентированных языков	/ понятности, простоты
B.5ea	Функции или часть функций, не осуществимых с помощью проблемно-ориентированных языков, следует разрабатывать в виде независимых модулей	—
B.5eb	Проблемно-ориентированные языки с графическим синтаксисом должны предлагать сопряженный литерный язык	/ использования автоматического анализа кода
B.5ec	Любые элементы проблемно-ориентированных языков, не пригодные для разработки системы класса 1, должны быть определены и, в конечном счете, их следует избегать	—

Таблица B.5f — Автоматическая генерация кода

Пункт	Рекомендация	Позволяет избежать/позволяет добиться
B.5fa	Выходные данные генератора кода должны быть отслеживаемыми до его входа	/ возможности верификации выходных данных генератора кода во время аттестации или использования
B.5fb	Генерируемый код должен быть читаемым	/ обнаружения дефектов из-за генератора кода
B.5fc	Не должно быть никаких изменений на уровне генерированного кода. Если изменения необходимы, то они должны быть внесены и документально оформлены на уровне входа	/ сохранение согласованности выхода со входом / обеспечение качества программного обеспечения, к которому относится генерированный код
B.5fd	Там, где это применимо, язык, используемый для генерированного кода, должен соответствовать рекомендациям приложения D	/ применимости инструментальных программ для компиляции, а также верификации и валидации; удобочитаемости

Приложение С
(справочное)

Примеры технологии прикладного программирования (разработка программного обеспечения с использованием проблемно-ориентированных языков)

С момента публикации первого издания МЭК 61880 в 1986 г. произошло быстрое развитие технологии прикладного программирования.

Комплексы оборудования (платформы систем), предназначенные для задач автоматизации, сейчас широко представлены на рынке.

Неотъемлемой частью этих платформ являются мощные инструментальные программы, разрабатываемые как по экономическим причинам, так и с целью обеспечения качества.

Типичным аспектом прикладного программного обеспечения является то, что наиболее ответственные элементы традиционного процесса разработки программного обеспечения осуществляются автоматически.

В настоящем приложении приведен пример того, как требования настоящего стандарта могут быть применены в контексте типичной разработки программного обеспечения с помощью проблемно-ориентированного языка.

С.1 Принципы применения требований

Качество программного обеспечения является существенной частью достижения общего качества и безопасности компьютерных систем контроля и управления, важных для безопасности.

Основные факторы обеспечения соответствия требованиям настоящего стандарта состоят в:

a) структурированном по этапам процессе разработки программного обеспечения с четким определением входной и выходной информации (см. 5.4);

b) понятной структуре программного обеспечения, разрабатываемой на этапе проектирования программного обеспечения, которая формирует основу для кодирования, а также для оценки программного обеспечения (см. 7.1);

c) правилах кодирования и технологии, соответствующих требованиям и рекомендациям, приведенным в приложении В (см. 7.3.2);

d) отслеживаемости исходных требований вплоть до конечного кода программного обеспечения (см. 7.4).

В приведенном примере процесс разработки программного обеспечения строго определяется инструментальными программами. Тем не менее, процесс структурирован по этапам с четким определением входной и выходной информации.

Генератор кода спроектирован так, чтобы соответствовать техническим требованиям, относящимся к проектированию программного обеспечения. Как следствие, генерируемый код обеспечивает четкую принадлежность кода конкретным функциям.

Аналогично правила кодирования в генераторе кода были реализованы так, чтобы удовлетворялись требования и рекомендации, приведенные в приложении В.

Примененный набор инструментальных программ поддерживает отслеживаемость исходных требований во время всех этапов процесса разработки вплоть до исполняемого кода, интегрированного в целевую систему.

С.2 Применение требований к жизненному циклу программного обеспечения

Использование проблемно-ориентированных языков, поддерживаемое инструментальными программами для автоматической генерации кода, существенно влияет на жизненный цикл программного обеспечения.

Примеры жизненного цикла для технологии разработки прикладного программного обеспечения показаны на рисунке С.1. В отличие от классического жизненного цикла программного обеспечения этапы детального проектирования, кодирования, интеграции модулей и тестирования интегрированы в автоматизированный процесс.

При выполнении проектирования деятельность, связанная со спецификацией, и, частично, с валидацией системы, обычно осуществляется технологами производственного процесса, тогда как деятельность по проектированию системы контроля и управления, функциональной спецификации и тестированию системы — разработчиками контроля и управления.

Прикладное программное обеспечение может быть создано непосредственно после составления спецификации системы контроля и управления и настройки относящихся к ней функций.

Следовательно, имеется возможность оценки установленных функций посредством генерированного кода с использованием модели или конкретных траекторий входных данных.

Эта функциональная оценка, которая может быть проведена даже до изготовления технического обеспечения целевой системы, может улучшить качество проекта путем раннего выявления в нем дефектов.

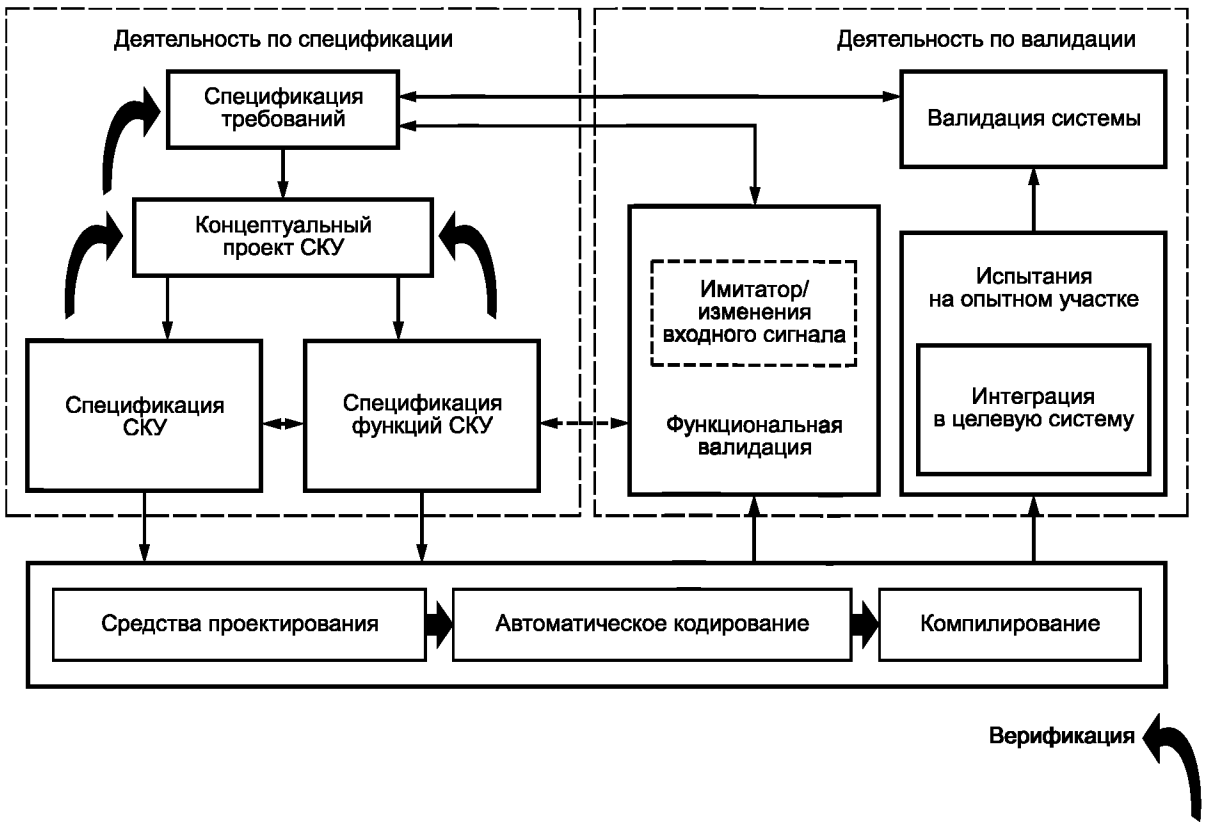


Рисунок С.1 — Жизненный цикл технологии прикладного программирования

Набор инструментальных программ поддерживает весь жизненный цикл программного обеспечения, т.е. спецификацию, проектирование, включая верификацию и валидацию, эксплуатацию системы и ее модификации в случае внесения в последующем изменений в начальные требования.

Форматы для документального оформления спецификации проекта системы и связанные с этим функции поддерживают эффективную верификацию. Этот элемент технологии прикладного программирования согласуется с требованиями разделов 8 — 10.

С.3 Применение требований к автоматической генерации кода

Автоматическая генерация кода является типичным элементом современных комплексов оборудования (платформ системы), обеспечивающих эффективное проектирование систем контроля и управления, а также получение программных продуктов высочайшего уровня качества.

Такие комплексы оборудования включают в себя набор инструментальных программ для поддержки процесса программирования.

При соблюдении требований настоящего стандарта методология автоматической генерации кода обеспечивает получение программного обеспечения высокого качества и снижает вероятность внесения ошибок человеком.

С.3.1 Инструментальные программы для управления проектированием и разработкой

В указанном выше примере инструментальные программы генерируют прикладной исходный код, исходя из формализованной спецификации (например, графической спецификации) на тщательно подобранном языке:

a) использование характерных для конкретного применения обозначений в спецификации архитектуры систем контроля и управления и связанных с ней функций помогает решению проблем взаимосвязи между разработчиками системы и разработчиками контроля и управления;

b) входные обозначения основаны на графическом представлении, определенном для предназначенных функций контроля и управления и сравнимом с классическими проверенными диаграммами функций, что позволяет проводить комплексную проверку установленных функций;

c) разработанная документация позволяет добиться строгой взаимосвязи с результатами разработки.

Набор инструментальных программ оказывает помощь в осуществлении следующей деятельности в области программирования:

- возможность устанавливать архитектуру технического обеспечения в соответствии с требуемой отказоустойчивостью, что приводит к необходимости создания резервной структуры;

- демонстрация правильности синтаксиса для спецификации программного обеспечения с помощью интегрированных проверочных инструментальных программ;

- создание эффективного управления конфигурацией путем безошибочной идентификации всех компонентов программного обеспечения [например, с использованием контрольных сумм CRC (cyclical redundancy check - циклический избыточный код)];

- верификация для обеспечения правильной технологии на каждом этапе проектирования;

- верификация проекта посредством моделирования;

- диагностика и проверка программного обеспечения, работающего в целевой системе, посредством обслуживающих инструментальных программ;

- проектирование СКУ посредством инструментальных программ по оценке работы процессора и шины при наибольших нагрузках;

- управление проектными данными посредством базы данных для всех существенных проектных данных.

С.3.2 Инструментальные программы автоматической генерации

В приведенном выше примере инструментальные программы для автоматической генерации кода обладают следующими характеристиками:

a) автоматическая генерация кода охватывает всю область прикладных функций, прикладных данных и коммуникации между всеми устройствами обработки системы класса 1;

b) установлены входные обозначения для автоматической генерации кода (синтаксис и семантика);

c) прикладные функции установлены с помощью функциональных диаграмм;

d) инструментальные программы позволяют осуществлять назначение проектируемых функций соответствующим устройствам обработки;

e) использованы испытанные правила проектирования программного обеспечения, интегрированные в инструментальную программу, которые обеспечивают:

- ясную структуру генерируемого исходного кода, соответствующего основным требованиям настоящего стандарта,

- алгоритм управления программы не зависит от последовательности обращений к графической спецификации,

- алгебраические проверки для избежания исключительных ситуаций,

- использование статического распределения ресурсов;

f) код генерируется на языке высокого уровня, который позволяет использовать стандартные компиляторы при получении исполняемого объектного кода для целевой системы. Код генерируется отслеживаемым путем, непосредственно связанным с кодами моделирования станции для анализа переходных процессов и возмущений.

Приложение D
(справочное)

Язык, транслятор, редактор связей

Приведенные в таблицах D.1 — D.4 подробные рекомендации для безопасного применения языка, его транслятора и редактора связей дополняют рекомендации основной части настоящего стандарта. Эти рекомендации также применимы к любым другим вспомогательным программам системы. Рекомендации для трансляторов применимы также к интерпретаторам, кросскомпиляторам и эмуляторам. Аналогичным образом аспекты, применимые к трансляторам и редакторам связей, следует учитывать при выборе и разработке формализованной спецификации на средства проектирования и их использование. Для проекта следует отбирать рекомендованные критерии в соответствии с указанными приоритетами.

Таблица D.1 — Общая информация

Пункт	Рекомендации	Приоритет
a	Транслятор, редактор связей и загрузчик должны быть тщательно протестированы перед использованием; эта операция рассматривается как очень важная	1
b	Рекомендуется иметь достаточно качественные и надежные данные о трансляторе, редакторе связей и загрузчике	2
c	В тех случаях, если используются вспомогательные системные программы, такие как вспомогательные средства, системы документации и подобные программы, до использования их следует должным образом протестировать	1
d	Синтаксис языка должен быть полностью и однозначно определен	2
e	Следует установить полную и понятную семантику языка	1
f	Использование языков высокого уровня предпочтительнее использования машинно-ориентированных языков	2
g	Распространенность языка и его адекватность проблеме считаются важными аспектами	2
h	Насколько это возможно, следует выполнять рекомендации, приведенные в приложении B	1
i	Читаемость полученного кода более существенна, чем удобство записи во время программирования	2
j	Синтаксическая нотация должна быть единообразна; допускается не более одной нотации для одного и того же понятия	2
k	В языке следует избегать элементов, которые могут вызвать ошибки	2
l	Получаемые программы должны быть легко модифицируемыми	2
m	Входные, выходные и изменяемые параметры должны быть синтаксически различимы	2
n	На всех этапах процесса трансляции следует обеспечить дополнительный выход для анализа	3

Таблица D.2 — Обработка ошибок

Пункт	Рекомендации	Приоритет
a	Транслятор языка и редактор связей должны обеспечивать регистрацию такого числа ошибок программирования, какое возможно во время трансляции или исполнения в режиме онлайн	2
b	Во время исполнения в режиме онлайн должна существовать возможность обработки исключительных ситуаций	2

Окончание таблицы D.2

Пункт	Рекомендации	Приоритет
c	Язык должен обеспечивать обработку утверждений	3
d	Ошибки, способные вызвать исключительные ситуации во время исполнения, включают в себя:	
	- превышение границ массива	1
	- превышение диапазона величин	1
	- обращение к неинициализированным переменным	3
	- невозможность удовлетворить утверждение	2
	- отбрасывание значащих разрядов числовых величин	2
	- пропуск параметров неправильного типа	1
e	Если при трансляции или редактировании связей обнаружена ошибка, то о ней следует сообщить, не делая попыток исправить	2
f	Если нет ясности в том, что нарушились какие-то правила, то должно быть выдано предупреждение	3
g	Во время трансляции должны проверяться типы параметров	3

Таблица D.3 — Обработка данных и переменных

Пункт	Рекомендации	Приоритет
a	Диапазон каждой переменной должен определяться при трансляции	1
b	Точность каждой переменной с плавающей запятой должна определяться во время трансляции	2
c	Не должно быть неявных преобразований типов	2
d	Тип каждой переменной, массива, элемента записи, выражения, функции и параметра должен определяться во время трансляции	2
e	Переменные массивы, параметры и т.д. должны быть явно декларированы, включая их типы	1
f	Следует различать типы переменных, соответствующие входам, выходам, параметрам процедур и подпрограмм	2
g	Должны быть разрешены имена переменных произвольной длины	2
h	Насколько возможно, проверку типов предпочтительнее проводить на этапе трансляции, чем на этапе исполнения	3
i	Во время трансляции должно проверяться, разрешено ли присвоение для любого частного элемента данных	2

Таблица D.4 — Аспекты режима онлайн

Пункт	Рекомендации	Приоритет
a	Во время оценки выражения не должно допускаться внешнее присваивание для любой переменной, которая доступна в этом выражении	1
b	Время, расходуемое на вычисления, должно быть доступно проверке в режиме «онлайн»	3
c	В режиме «онлайн» должно обеспечиваться фиксирование ошибок (см. таблицу D.2,d)	1

Приложение Е
(справочное)

Верификация и тестирование программного обеспечения

Е.1 Деятельность по верификации и тестированию программного обеспечения

Настоящий раздел содержит руководство по верификации и тестированию программного обеспечения. В зависимости от тестируемой программы используемые для этой цели различные методы являются более или менее эффективными в обнаружении дефектов программы. Следующие основные методы верификации и тестирования программного обеспечения взаимно дополняют друг друга: основанный на инструментальных программах статический анализ кода (например, сравнение модификаций), проверка рабочей программы (визуальный анализ) и динамическое выполнение (например, имитация реальной работы программного обеспечения). Большинство из этих методов представляет собой рекомендуемые систематические подходы. Может применяться ряд дополнительных подходов к тестированию, включая статистическое тестирование, которое может использоваться для поиска случаев, не охваченных систематическим подходом.

Обзор возможных методов приведен в таблице Е.4.1. В случае необходимости для различных частей программы следует выбрать различные методы и различные критерии для подбора данных тестирования с тем, чтобы определить, является ли рассматриваемая часть программы свободной от ошибок, или определить доверительный уровень. Выбор зависит от внутренней структуры части программы, требуемого уровня надежности, запросов, поступающих на нее при эксплуатации станции, и имеющихся средств тестирования.

При тестировании программного обеспечения следует рассматривать различные уровни проекта программного обеспечения (например, уровни модуля, подсистемы и системы).

Е.2 Системный подход

Каждый модуль следует систематически верифицировать и тестировать на основе системного подхода в соответствии с их предназначением и связанными с ними критериями охвата. В дополнение к неавтоматизированным верификации и тестированию следует как можно шире применять автоматизированные средства верификации и тестирования. Результаты следует сверять с ожидаемыми результатами, полученными из спецификации на программный модуль. С входными и выходными данными следует обращаться также как и в системе безопасности.

Таблица Е.4.2 является контрольным списком, который может интерпретироваться согласно требованиям для каждого отдельного случая. Из-за большого разнообразия случаев, встречающихся на практике, невозможно рекомендовать какую-либо комбинацию тестов в качестве нормативной для определенного класса применений. Однако в таблице отмечено, какие тесты следует проводить в любых условиях. С другой стороны, очевидно, что для конкретного приложения не могут быть выполнены все комбинации всех тестов, также как и все тесты в отдельности.

На уровне подсистемы программное обеспечение частично интегрируется в систему. Тестирования на уровне подсистемы подтверждают должную интеграцию программных модулей. Необходимо разграничивать случаи, когда существует зависимость по потоку данных между модулями программного обеспечения, а также случай, когда такой зависимости не существует. Для достижения уверенности в правильности следования всем независимым ветвлениям в подсистеме необходимо провести тестирование. Следует также использовать контрольные тесты на границах входных областей и на пределах рабочей области модуля.

Следует использовать тот же набор тестовых данных, что и на уровне модулей. Результаты следует сверять с предварительно рассчитанными значениями. Параметры должны быть входными и выходными по отношению к подсистеме точно так же, как и в системе безопасности. В тех случаях, когда это практически возможно, испытание на уровне подсистемы следует проводить с использованием конфигурации технического обеспечения, идентичного целевому техническому обеспечению.

На уровне системы программное обеспечение полностью интегрируется аппаратными средствами. Тест на уровне системы подтверждает должную интеграцию подсистем. Тестирование следует проводить прогоном программ с использованием реалистической модели или реалистической версии системы, контролируемой или управляемой системами класса 1.

Настоящее тестирование всей системы необходимо проводить в соответствии с положениями об эксплуатационных характеристиках, приведенных в спецификации требований к программному обеспечению. Помимо описанной выше деятельности по тестированию следует проводить системный временно́й анализ и проверку.

Е.3 Статистические методы

Статистические методы могут применяться в дополнение к системным методам.

Статистические методы имеют следующие характеристики:

- тесты отбирают с помощью независимой выборки из представляющей работу системы в режиме эксплуатации распределения вероятностей;
- последовательность и число тестов не влияет на отдельный тестовый прогон;
- каждый произошедший отказ фиксируется;
- число тестов — большое (см. приведенные ниже примеры);
- отказы происходят редко.

Обычно статистическое тестирование проводится для укрепления уверенности в правильности работы систем класса 1, достигнутой с помощью обширной программы функциональных типовых тестов.

Формулы, выведенные для вероятностной верификации программного обеспечения, могут использоваться для оценки вероятности отказа системы при запросе и верифицируемых допущениях. Формулы дают следующую оценку верхней границы для вероятности отказа при запросе:

допустим, что проводят n статистических тестов и зафиксировано 0 отказов. Тогда для вероятности регистрации отказа *pfd*, которое должно быть меньше или равно некоторой заданной величине *pfd* с вероятностью α, необходимо выполнение следующего условия:

$$pfd \le -\frac{\ln(1-\alpha)}{n}.$$

Таким образом, для $\alpha = 0{,}95$, после n безотказных тестов мы получаем приближение $pfd \le \frac{2{,}99}{n}$ с вероятностью 0,95.

Например, чтобы получить для *pfd* значение 10^{-4} при вероятности 95 %, должно быть без отказа проведено 29 900 тестов.

Для $\alpha = 0{,}99$ мы получим приближение $pfd \le \frac{4{,}6}{n}$ с вероятностью 0,99.

Например, чтобы получить для *pfd* значение 10^{-4} при вероятности 99 %, должно быть без отказа проведено 46 900 тестов.

Пригодность рассчитанных значений *pfd* зависит от аналогичности профиля входных данных теста и профиля реальных входных данных, существующего во время работы системы. Если приведенное выше уравнение используется для нереального профиля работы, оно даст оценку *pfd* для воображаемого профиля применения, т.е. оценка *pfd* может сильно отличаться от реальной надежности системы при ее активной работе. Это является главным недостатком статистического подхода к тестированию, поскольку обычно бывает очень трудно точно определить профиль, существующий при реальной работе системы, особенно для систем с большим количеством входных данных.

Е.4 Методы верификации и тестирования

Таблица Е.4.1 — Выбранные методы верификации

Общее представление								
Порядковый номер	Метод	Цель	Основные преимущества	Проблемы при использовании, основные недостатки	Стоимость и трудозатраты по сравнению со стоимостью разработки	Результат	Связь с другими методами	Оценка
1	Контроль за процедурой тестирования	Получение значения надежности без дополнительных усилий	Небольшие дополнительные усилия при применении Не требуется детальных знаний об объекте испытаний	Получаемые оценки надежности недостаточны Практические случаи лишь приблизительно соответствуют теории	Малые	Вероятностные оценки, например, MTBF, т.е. средняя наработка на отказ	Используется теория вероятности (см. пункт 2 таблицы)	Трудно использовать для целей безопасности
2	Стратегическое тестирование							
2.1	На готовность	Получение значения надежности без рассмотрения деталей кодирования	Не требуется детальных знаний об объекте испытаний	Обеспечить условия теста, при которых различные входные сигналы поступают с той же вероятностью, что и при эксплуатации	Для получения значимых результатов требуется большое число тестирований	Вероятностные оценки, например, готовность на запрос, вероятность отказа за время жизни программы	То же (см. пункт 1 таблицы)	Допускается в сокращенной форме использовать дополнительно к анализу программы
2.2	На отсутствие ошибок			Обеспечить условия теста, при которых различные свойства программы затрагиваются с равной вероятностью	Стоимость теста может быть значительно выше	Вероятностные оценки, например, предел вероятности для все еще имеющихся ошибок ПО		
3	Доказательство правильности программы	Обеспечивает основу для сравнения со спецификацией	Строгое утверждение о достижимой правильности	Нет доступного формализма для обнаружения утверждений цикла или инвариантов цикла; склонность к ошибкам	Такого же порядка величины или больше	Правильно/неправильно, в пределах правильности самого доказательства	Используются некоторые методы доказательства правильности программы	Возможно единственно целесообразный путь для общих циклов ПОКА (WHILE)

Окончание таблицы Е.4.1

Общее представление								
Порядковый номер	Метод	Цель	Основные преимущества	Проблемы при использовании, основные недостатки	Стоимость и трудозатраты по сравнению со стоимостью разработки	Результат	Связь с другими методами	Оценка
4	Анализ программы		Дает хорошие результаты для простых программ	Некоторые конструкции программы сложны для анализа		При достаточно глубоком анализе освобождает от некоторых специфических ошибок	Используются некоторые идеи из доказательства правильности программы	
4.1	Неавторизированный анализ	Обеспечивает основу для значимого теста или сравнения со спецификациями	Возможно выполнения — без дополнительного инструментария	Склонность к ошибкам	Несколько ниже стоимости разработки			Следует использовать в отсутствие автоматизированных систем
4.2	Авторизированный анализ	Как и выше, иногда ограничен несколько нечетким критерием качества	Только ограниченная неавторизированная работа. Результаты получаются быстро	Большинство автоматизированных средств требуют дополнительной ручной работы. Некоторые результаты применения таких средств трудны в интерпретации	Определяется используемыми автоматизированными средствами; значительно ниже стоимости разработки			Следует использовать по возможности шире. Наиболее перспективный подход

Е.4.2 Методы тестирования

Таблица Е.4.2.1 — Общие сведения

Порядковый номер	Вид теста	Тип детектируемых ошибок	Следует выполнять	Примечание
1	Варианты, типичные для поведения программы в целом, ее арифметика, временные аспекты	Все, но без гарантии полноты	Всегда	Предполагается, что система работает правильно, если все варианты выполняются правильно
2	Все отдельные и явно определенные требования	Полностью регистрируются неиспользуемые функции	Всегда в первую очередь, если требуемые функции детально определены	Тест может быть исчерпывающим, если функции разделены. Мало информации о проблемах согласованности действия
3	Все входные переменные для экстремальных случаев (тест на аварийный отказ)	Ошибки синхронизации, но без гарантии. Переполнение, потеря значащих разрядов	Всегда	—
4	Работа всех внешних устройств	Ошибки проектирования интерфейса аппаратных средств и ПО	Всегда	—
5	Статические ситуации и варианты динамического поведения, представительные для протекания технологического процесса	Все, но без гарантии	Всегда	Особенно ценно, когда имеется модель технологического процесса
6	Правильная работа, демонстрируемая при включении и выключении каждой резервной подсистемы/каждого внешнего устройства (некоторые комбинации также следует тестировать, где это существенно)	Ошибки при работе с интерфейсом технического обеспечения	Если поведение технологического процесса хорошо известно и не очень разнообразно	Обеспечивает запас прочности системы

Таблица Е.4.2.2 — Тестирование ветвей

Порядковый номер	Вид теста	Тип детектируемых ошибок	Следует выполнять	Примечание
7	Проверка каждого исполняемого хотя бы один раз оператора	Недоступный код	Всегда	—
8	Проверка каждого результата каждой ветви, исполняемой хотя бы один раз	Ошибки в логике управления без гарантии полноты	Всегда	Содержит тест 5; может быть исчерпывающим, если отсутствуют циклы и нет проблем синхронизации. Чисто комбинаторная проблема, отражаемая в структуре программы

Окончание таблицы Е.4.2.2

Порядковый номер	Вид теста	Тип детектируемых ошибок	Следует выполнять	Примечание
9	Проверка каждого предикатного члена, поступающего на каждую ветвь	Ошибки в логике управления и потоке данных	Если неприменима комбинация тестов 8 и 12	Содержит тест 8; включен в комбинацию тестов 9 и 14
10	Проверка каждого цикла с использованием минимального, максимального и, по крайней мере, одного промежуточного числа повторений	Ошибки в управлении циклом и обработке массива данных	Всегда, если программа содержит циклы	Не применимо к конструкции «бесконечных циклов»
11	Проверка каждой ветви, исполняемой хотя бы один раз	Все ошибки логики управления	Для валидации того, что выбранные проект и кодирование не делают верификацию слишком сложной	Выполняется только для модулей. Содержит тест 8; следует отметить, что каждое новое повторение цикла ведет к новой ветви

Таблица Е.4.2.3 — Тестирование перемещения данных

Порядковый номер	Вид теста	Тип детектируемых ошибок	Следует выполнять	Примечание
12	Проверка каждого объявления каждой области памяти, осуществляемого хотя бы один раз	Ошибка в потоке данных, но без гарантии	Используемые массивы	—
13	Проверка каждой ссылки на каждую область памяти, осуществляемой хотя бы один раз	Ошибки в потоке данных, в особых случаях – все ошибки потока данных	Используемые массивы	В большинстве случаев содержит тест 12. Целесообразно только в связи с тестом 12
14	Проверка всех преобразований данных от входа до выхода, осуществляемых хотя бы один раз	Все ошибки потока данных	Всегда для отдельных сегментов	Осуществимо только для модулей. Возможно, покрывается тестами 7, 8 или 11

Таблица Е.4.2.4 — Тестирование временных характеристик

Порядковый номер	Вид теста	Тип детектируемых ошибок	Следует выполнять	Примечание
15	Проверка всех временных ограничений	Ошибки временных характеристик, слишком долгое время вычислений	Всегда	—
16	Проверка максимально возможного числа комбинаций последовательностей прерываний	Организационные ошибки	Если число комбинаций не слишком большое	—
17	Проверка всех значительных комбинаций последовательностей прерываний	Организационные ошибки	Всегда	—

Таблица Е.4.2.5 — Разное

Порядковый номер	Вид теста	Тип детектируемых ошибок	Следует выполнять	Примечание
18	Проверка правильного положения всех границ области входных данных	Ошибочное деление области входных данных	Если используются аналоговые входные данные	Количество и вид подобластей входных данных устанавливается анализом
19	Проверка точности арифметических расчетов во всех критических точках	Численные ошибки, ошибки в алгоритмах, ошибки округления	Если используется компьютер с короткой длиной слова; если используется сложная арифметика	—
20	Только для программ: проверка взаимосвязи и взаимодействия модулей	Неправильный перенос данных между модулями	Всегда	Желательно использование вспомогательных средств тестирования
21	Проверка каждого модуля, к которому происходит хотя бы одно обращение	Неправильная логика управления и неправильный поток данных между модулями, без гарантии	Всегда	—
22	Проверка каждого обращения к другому модулю, происходящего хотя бы один раз	—	Всегда	—
23	Проверка работы при большой нагрузке	Ошибки синхронизации и срабатывания	Всегда	Обеспечивает запас прочности системы

Приложение F
(справочное)

Перечень документации, требующейся в течение жизненного цикла безопасности программного обеспечения

Таблица F.1

Ссылки на пункты настоящего стандарта	Основная ссылка	Другие ссылки
Документы, связанные с разработкой программного обеспечения	—	—
Спецификация требований к системе	15.2	15.3
Спецификация системы	5.3	6.1, 8.1, 9.3, 15.3
Спецификация требований к программному обеспечению	6.1	3.33 (3.35 и 3.37), 5.3, 6.4, 7, 7.1.3, 8.1, 8.2, 8.2.3.2, 11.3, 15.3.1.2
План обеспечения качества программного обеспечения	5.5	—
Подробные рекомендации (разработка программы)	7.3	Приложение D
План верификации программного обеспечения	8.2.1	—
Программные аспекты плана интеграции системы	9.1	9.3
Спецификация проекта программного обеспечения	7.4	7.1.3, 8.1, 8.2.2, 8.2.3
Отчет о верификации программного обеспечения	8.2.2	7
Спецификация тестирования программного обеспечения	8.2.3.1.2	8.2.3.1.3
Отчет о верификации кода программного обеспечения	8.2.3.1.1	—
Отчет о тестировании программного обеспечения	8.2.3.1.3	—
Программные аспекты отчета о верификации интегрированной системы	9.5	9.1, 9.2, 9.3, 9.4
Программные аспекты плана валидации системы	10.1	10.2, 10.4
Программные аспекты отчета о валидации системы	10.3	—
Руководство пользователя программного обеспечения	12.4.2	—
Программные аспекты плана тестирования при вводе в эксплуатацию	—	—
Программные аспекты отчета о тестировании при вводе в эксплуатацию	—	—
Документы, связанные с модификациями программного обеспечения	—	—
Отчет об аномалиях	11.1	11.3
Запрос на модификацию программного обеспечения	11.1	11.2, 11.3
Отчет о модификации программного обеспечения	11.2	—
Архив управления модификациями программного обеспечения	11.2	11.3

Приложение G
(справочное)

Некоторые аспекты отказа по общей причине (ООП) и разнообразия

ООП может произойти, например:

- если скрытый дефект был введен в два или более компонента или в две или более системы, и все эти компоненты или системы работают в одинаковых или схожих условиях, так что отказ может быть спровоцирован коррелированным во времени способом либо
- если условия для отказа распространяются через обмен данными.

G.1 ООП, вызванные программным обеспечением

Для того, чтобы программное обеспечение вызвало ООП, траектория сигнала должна инициировать дефектный элемент программного обеспечения, который вызывает отказ, воздействующий на две или более системы или на два или более канала (например, два канала защиты, два контроллера замкнутых контуров или две логических подсистемы управления). Снижение вероятности ООП в различных системах из-за программного обеспечения может быть достигнуто снижением вероятности того, что программное обеспечение этих систем содержит общие дефекты, и/или с помощью обеспечения работы этих систем по различным сигнальным траекториям. Дефекты программного обеспечения могут возникнуть из спецификации требований на СКУ или могут быть внесены при разработке программного обеспечения.

Для того, чтобы ООП создали угрозу безопасности, они должны препятствовать выполнению функции безопасности и происходить в тот период времени, когда может возникнуть угроза безопасности, либо сам этот отказ должен представлять угрозу безопасности, связанную, например, с потерей защиты или управления.

При использовании одного и того же или аналогичного программного обеспечения, методов реализации или алгоритмов в резервных или различных системах существует значительный общий элемент.

В настоящее время не существует признанного метода оценки вероятности или частоты отказов, возникающих из-за недостатков программного обеспечения.

G.2 Возможные причины и следствия ООП

G.2.1 Возможность ООП

При использовании общего программного обеспечения или общих модулей существует возможность ООП, вызванного программным обеспечением. Потенциальными источниками скрытых дефектов являются ошибки проектирования, происходящие из спецификации требований к СКУ, архитектуры, алгоритмов, методов разработки, инструментальных программ, методов реализации или обслуживания.

Неправильное понимание требований и неправильное их преобразование могут привести к дефектам в спецификации программного обеспечения и к рискам возникновения ООП из-за проявления этих дефектов в конечном программном обеспечении. Недостатки программного обеспечения могут быть следствием неправильных, неполных, неточных, неправильно понятых требований и спецификаций к программному обеспечению. Ошибки проекта, приводящие к дефектам программного обеспечения, могут быть внесены в разные программы из-за таких общих человеческих факторов, как уровень подготовки, организация, ход мысли и подходы к проектированию.

Другая возможность возникновения ООП кроется в соединении одних систем с другими системами, имеющими более низкое качество программного обеспечения.

G.2.2 Инициирование ООП

Отдельные помехи или входные сигналы могут спровоцировать ООП, если они воздействуют:

- на два или более резервных канала системы, использующих общее программное обеспечение;
- на две системы, функции которых различны, но которые используют общие программные модули.

G.2.3 Аномальные условия и события

Необычные отказы технического обеспечения, условия на станции и события могут вызвать непредсказуемые сигнальные траектории, неожиданные состояния программного обеспечения, переходные процессы или условия перегрузки, которые не были учтены первоначальными требованиями или при проектировании программного обеспечения.

Возможные события, которые могут вызвать ООП, включают в себя:

- деятельность по обслуживанию;
- сбой общего синхронизирующего сигнала, вызывающий потерю синхронности действий;
- переходные процессы в питающей сети, вызывающие остановку в работе программного обеспечения или его перезапуск;
- аварийные остановы на станции, вызывающие перегрузку коммуникационных каналов;
- превышение предела возможностей оператора, вызывающее неверное действие;

- запросы оператора, вызывающие превышение предела возможности системы во время аварийных остановов и при переходных режимах работы;
- состояние, когда задействованы и находятся в работе все функции автоматических контроллеров, и
- анормальные условия во время аварий и пусконаладочных работ.

G.3 Защита от ООП

Возможные меры защиты включают в себя:

- методы, используемые в течение жизненного цикла, цель которых состоит в получении свободного от ошибок программного обеспечения (см. 13.1);
- обоснование и улучшение качества общего программного обеспечения (см. 13.2);
- использование общего программного обеспечения в очень узких и гарантированных условиях;
- ограничение последствий отказов программного обеспечения (см. 13.3);
- проектирование каналов или систем так, чтобы совпадение отказов двух каналов или систем было очень маловероятным из-за того, что имеется очевидное различие сигнальных траекторий для этих систем;
- проектирование каналов или систем с использованием асинхронности их работы; это может быть использовано как защита по отношению к одинаковым процессорам в различных каналах, на которые влияют одни и те же траектории в одно и то же время, и
- разнообразие деталей для некоторых или всех функций и углубление концепции независимости во время всего жизненного цикла (см. 13.4).

G.4 Доказательство корректности

Методы доказательства корректности включают в себя:

- использование проверенных стандартных модулей программного обеспечения с четким и проверенным интерфейсом (см. раздел 15); типичные функции включают в себя, например, управление устройствами, мониторинг процесса и сбор входных данных, основные алгоритмы управления (такие как пропорционально-интегрально-дифференциальный алгоритм, нечувствительная область, гистерезис);
- использование для декодирования загруженной в память программы инструментальных программ и процедур, независимых от процессов проектирования, и демонстрация соответствия спецификации загруженной в память программы;
- использование динамического анализа для тестирования правильного поведения программного обеспечения в моделируемой среде, представляющей основные части станции (см. 8.2.3.2.3);
- использование статического анализа программы для определения управляющей логики и потока данных, а также демонстрации правильности процессов принятия решения и логических процессов;
- использование двух версий программного обеспечения, испытанных поочередно, с применением случайных сигнальных траекторий. Этот метод может использоваться как дополнение к систематическим испытаниям для регистрации дефектов при проектировании и кодировании;
- реализация полной программы испытаний снизу — вверх, когда правильная работа каждой компоненты системы всесторонне проверяется до ее интегрирования в систему.

G.5 Виды разнообразия

a) Разнообразие ПО включает в себя (в порядке важности) следующие аспекты:

- функциональное разнообразие,
- различные спецификации проекта при одних и тех же функциональных требованиях;

b) разнообразие на уровне системы может включать в себя:

- использование независимых систем для различных критериев запуска,
- использование различных базовых технологий, таких как компьютерные технологии и системы с «жесткой» логикой,
- использование различных типов компьютеров, модулей технического обеспечения и основных концепций проектирования,
- использование различных классов компьютерной техники, таких как PLC (контроллеры с программируемой логикой), микропроцессоры или миникомпьютеры;

c) особенности подхода к проектированию и решению проблем, улучшающие разнообразие, включают в себя следующие различия:

- алгоритмы обработки,
- данные для конфигурации, калибровки и выполняемых функций,
- техническое обеспечение для входных сигналов,
- интерфейсы и коммуникации технического обеспечения,
- процессы дискретизации входов,
- временна́я последовательность операций,
- процессы синхронизации,
- использование архивной информации, регистров-защелок и скоростей изменения;

d) различия в проектировании и методах реализации включают в себя:

- языки,
- системы компиляции,
- библиотеки поддержки,

- инструментальные программы,
- методы программирования,
- системное и прикладное программное обеспечение,
- структуры программного обеспечения,
- различное использование одних и тех же модулей,
- данные и структуры данных;

e) разнообразие во время тестирований (тестирование со взаимной нагрузкой);

f) разнообразные аспекты подхода к управлению включают в себя:

- сознательное следование при проектировании двум различным методам разработки (принудительно),
- разделение групп проектантов,
- ограничение общения между группами,
- формализованное общение при разрешении неясностей в требованиях и спецификациях,
- использование различных процессов определения логики,
- различные методы документирования,
- использование различного персонала.

G.6 Недостатки, преимущества и обоснование разнообразия

G.6.1 Недостатки

Вызванные разнообразием недостатки могут включать в себя:

- большую общую сложность,
- возрастание риска случайного срабатывания,
- более сложные спецификации и проект,
- контроль двух поставщиков,
- проблемы, связанные с модификацией, например, обеспечение сохранения разнообразия при модификации,
- увеличение объема документации,
- увеличение пространственного объема, занимаемого системой, дополнительных ресурсов, потребляемых системой, ужесточения требований к контролю окружающей среды,
- стоимость нескольких версий программного обеспечения; может ухудшить экономические показатели, за исключением методов испытаний,
- каждая из произведенных версий может оказаться худшего качества.

G.6.2 Преимущества

Использование функционального или программного разнообразия улучшает защиту от ООП, вызываемых программным обеспечением.

G.6.3 Обоснование

В качестве обоснования может рассматриваться увеличение надежности функций безопасности, достигаемых за счет разнообразия.

Приложение Н
(справочное)

Инструментальные программы для создания и проверки спецификации, проектирования и реализации

Инструментальные программы образуют в настоящее время существенную часть среды разработки программного обеспечения, выполняющего функции безопасности. Неавтоматизированные методы связаны с большим риском совершения ошибок и требуют привлечения высококвалифицированного персонала. Поэтому эти методы нуждаются в инструментальной поддержке с использованием математических методов, раскрывающих структурные и внутренние функциональные взаимосвязи программного обеспечения и проверяющих внутреннюю совместимость, совместимость с некоторой априорной моделью, желательные/нежелательные свойства и т.п.

Конечное подтверждение соответствия программы своей спецификации может осуществляться с помощью анализатора соответствия. Если проверенный генератор кода обеспечивает полное соответствие исполняемой программы описанию ее проекта, то статистический и динамический анализы обеспечивают разнообразие проверки правильности этого описания.

Инструментальные программы для формализированных методов спецификации и проектирования могут быть классифицированы в качестве конструктивных или аналитических инструментов.

Н.1 Конструктивные инструменты

Конструктивные инструменты используются для поддержки разработки спецификации, проектирования, кодирования. Они могут включать в себя:

Н.1.1 Текстовый редактор

Поскольку формализованные методы, основанные на теории множеств, на исчислении предикатов и исчислении высказываний требуют специальных математических символов, то важно иметь соответствующий текстовый редактор, способный отображать эти символы на экране с высоким разрешением и четко их распечатывать.

Н.1.2 Графический интерфейс

Там, где формализованные методы используют графику, требуются соответствующие графические возможности.

Н.1.3 Автоматический генератор кода

После утверждения формализованной спецификации целостность процесса проектирования может быть существенно улучшена путем применения прошедшего валидацию автоматического генератора кода. Такой генератор кода преобразует спецификацию в исполняемый код, снижая таким образом вероятность внесения ошибок. Кроме того, посредством выбора генератора кода может быть реализована безопасная сокращенная версия языка.

Для стандартных функций рекомендуется использовать сертифицированные модули программного обеспечения.

Автоматически генерируемый код должен быть читаемым. Комментарии должны способствовать распознаванию соответствующих частей спецификации. Структура автоматически генерируемого кода должна способствовать автоматической верификации.

Н.1.4 Генератор доказательства правильности

Формализованные методы, основанные на логических умозаключениях, требуют использования генератора доказательства правильности, который автоматически регистрирует доказательства правильности, возникающие на этапах проектирования.

Н.2 Аналитические инструменты

Аналитические инструменты позволяют проводить проверку спецификации, проектирования и реализации. Аналитические инструменты могут включать в себя следующие программы:

Н.2.1 Программа проверки синтаксиса

Программа проверки синтаксиса представляет информацию о структуре программы, по использованию данных программы, зависимости выходных переменных от входных переменных и управляющей логике программы, что позволяет проводить:

a) определение дефектов структуры, таких как множественный запуск, множественное завершение, недостижимый код, избыточный код, отсутствие использования результатов функции;

b) определение иерархии модулей/подпрограмм;

c) определение нарушений стандартов и соглашений по программированию, включая проверку на наличие безусловных переходов в циклах;

d) определение данных, которые считываются до их записи, данных, которые записываются до их чтения, данных, записанных дважды без их промежуточного чтения;

e) проверку информационного потока по спецификации;

f) оказание помощи в проектировании плана динамических испытаний;

g) управление тестовыми данными и, возможно, генерацию тестовых данных.

Н.2.2 Программа семантической проверки

Программа семантической проверки описывает математические соотношения между выходными и входными переменными для каждого семантически достижимого пути в свободных от ветвлений частях программы. Это позволяет проводить проверку того, что программа будет делать при всех обстоятельствах, а также регистрировать дефекты, такие, например, как неожидаемые выходные величины, на которые влияют входные величины, неправильный отклик на неожидаемые входные величины, неправильная последовательность функций и операторов и т.п.

Н.2.3 Генератор формализованных проверок

Формализованные проверки проекта требуют использования интерактивной программы, которая проводит необходимую работу с символами под управлением оператора, для того чтобы выполнить доказательство правильности. Такая программа известна как «помощник в доказательстве теорем» (ПДТ). Это обычно означает применение программы проверки доказательств, вход по такой программе является выходом ПДТ. ПДТ представляют собой большие программы, отсутствие дефектов для которых нельзя доказать. Поэтому необходимо проведение разносторонней верификации. Программа проверки доказательств должна быть основана на формализованной теории доказательств, и эта программа должна верифицироваться по отношению к этой теории.

Н.2.4 Аниматор

По возможности, спецификации рекомендуется анимировать с тем, чтобы конечный пользователь системы мог проверить различные аспекты спецификации или проекта с целью подтверждения соответствия (насколько это возможно) требований спецификации и предлагаемого проекта. Анимация должна с максимальной возможностью представлять проект и может потребовать использования макетов для демонстрации нефункциональных аспектов. Эта оценка проводится на соответствие критериям пользователя, и требования к системе могут быть модифицированы в свете этой оценки.

Н.2.5 Анализатор соответствия

Анализатор соответствия может показать, что программа правильно реализует спецификацию. При этой демонстрации анализатор соответствия использует входные и выходные условия плюс инвариант цикла. Анализатор соответствия систематически подтверждает выполнение в программе каждого условия.

Приложение I
(справочное)

Требования к ранее разработанному программному обеспечению (РПО)

I.1 Руководство для учета несоответствий и компенсирующих факторов

Слабости и несоответствия требованиям МЭК 60880 могут быть различных типов, например:

- требования относительно понятности и полноты документации по программному обеспечению;
- требования по читаемости программ;
- требования, относящиеся к проектированию программного обеспечения, которые улучшают детерминированное поведение при функционировании в реальном времени;
- требования, относящиеся к программному обеспечению, которые обеспечивают самоконтроль компьютерного технического обеспечения, или
- требования, относящиеся к полноте документированных отчетов по валидации.

Слабости и несоответствия редко бывают «черно-белыми», и их рекомендуется учитывать, рассматривая степень их выполнения, и в соответствии с влиянием на качество компьютерной системы, например:

требования МЭК 60880, касающиеся понятности, являются критическими при разработке и модификациях и менее существенны для стабильного и проверенного продукта.

О возможности доверия к опыту эксплуатации или дополнительным испытаниям в качестве компенсирующих факторов следует судить в зависимости от типа РПО и его роли в компьютерной системе, например:

- программное обеспечение операционной системы, обычно доступное как микрокод или двоичный код, может оказаться трудным для анализа, и может быть ограничен доступ к документации процесса его разработки. В некоторых случаях большой опыт эксплуатации может служить важным фактором приемки, особенно для программного обеспечения, выполняющего определенные повторяющиеся функции, например, драйверы коммуникации или части операционных систем. В других случаях, например, для программного обеспечения, выполняющего функции контроля и восстановления компьютерных систем, может отсутствовать большой опыт эксплуатации, поскольку эти функции редко приводятся в действие во время работы компьютерных систем;
- по библиотекам прикладных программ может быть составлена подробная документация на основе доступной информации по проекту данного модуля, процессу разработки и валидационным испытаниям. Данные по опыту эксплуатации на аналогичных станциях или дополнительным испытаниям также могут быть использованы для поддержки оценки соответствия, так что может быть достигнута достаточная степень уверенности в правильности работы данного типа программного обеспечения.

I.2 Сбор и проверка данных по опыту эксплуатации

I.2.1 Сбор данных

Рекомендуется собирать следующие данные:

- информацию с объектов, включая конфигурацию и условия эксплуатации РПО в компьютерных системах, используемые функции, число прогонов РПО;
- время работы на объекте, включая общее время с момента первого запуска, общее время эксплуатации последней версии РПО, общее время с момента последней серьезной ошибки (при наличии), общее время с момента последнего отчета об ошибке (при наличии);
- отчет об ошибках, включая дату ошибки, ее серьезность, устранение и
- историю версии, включая дату и идентификацию версий, а также соответствующие им конфигурацию, исправленные ошибки, функциональные модификации или расширения, неразрешенные проблемы.

I.2.2 Проверка данных

При верификации статистической значимости собранных данных рекомендуется иметь в виду различные факторы:

- данные по времени эксплуатации приемлемы, только если с момента ввода РПО прошло заранее установленное минимальное время;
- время работы РПО может быть принято во внимание, только если оно относится к тем возможностям РПО, которые действительно использовались при работе компьютерной системы, где РПО было инсталлировано.

Примечание — Например, некоторые модули прикладных библиотек, относящиеся, в частности, к дополнительным функциям системы, могут быть установлены в эту систему, но никогда не использоваться на конкретном объекте или использоваться редко;

- полноту данных по опыту эксплуатации (например, действительно ли все сбои корректно документированы) рекомендуется оценивать в зависимости от организации сопровождения на объекте;
- приемлемыми являются лишь данные из реальных, объективных источников информации.

Приложение J
(справочное)

Соответствие между МЭК 61513 и настоящим стандартом

J.1 Общие сведения

МЭК 60880 (1986 г.) был разработан за несколько лет до МЭК 61513 и включал в себя положения, относящиеся к уровню системы в дополнение к положениям уровня программного обеспечения.

В процессе пересмотра положения системного уровня, которые сейчас должным образом отражены в МЭК 61513, и из пересмотренного текста были изъяты. Положения, на которые в МЭК 61513 имеются явные ссылки или которые должным образом там не освещены, сохранены в пересмотренном тексте и представлены в настоящем приложении с указанием номеров пунктов МЭК 60880 (1986 г.) и настоящего МЭК 60880.

J.2 Пункты МЭК 60880 (1986 г.), на которые есть ссылки в МЭК 61513

МЭК 61513 является документом системного уровня, охватывающим все категории функций безопасности СКУ.

Во введении к МЭК 61513 сказано: «При рассмотрении компьютерных систем класса 1 данный стандарт следует применять совместно с МЭК 60880 и МЭК 60987 с тем, чтобы полностью удовлетворить требования к техническому и программному обеспечениям».

Соответствующие положения сохранены в настоящем издании МЭК 60880 и, где необходимо, пересмотрены.

В таблице J.1 приведено соответствие между пунктами МЭК 60880 (1986 г.), обозначенного как «Доп.», на которые есть ссылки в МЭК 61513, и пересмотренными пунктами настоящего стандарта.

Таблица J.1 — Соответствие пунктов МЭК 61513 и пунктов настоящего стандарта

Пункты МЭК 61513, в которых есть ссылки на	Пункты МЭК 60880 (1986 г.)/Доп.	Пункты настоящего стандарта
5.3.1.5.4 Защита от ООП, вызванных систематическими дефектами. Примечание 2 — Цель данного подраздела — дать общие сведения. Детальные требования к защите от отказов по общей причине вследствие ошибок в программном обеспечении для функций категории А приведены в 4.1 МЭК 60880-2*.	4.1 Доп.	13
5.3.3.1 Оценка надежности и защита от ООП. Примечание 2 — Требования к анализу отказов по общей причине вследствие ошибок в программном обеспечении приведены в 4.1.3 МЭК 60880-2*.	4.1.3 Доп.	13.3
5.5.1 Документация по проекту архитектуры Примечание — Требования к инжиниринговым методам и инструментальным средствам создания программного обеспечения для систем класса 1 приведены в 4.2 и 4.3 МЭК 60880-2*.	4.2 Доп. 4.3 Доп.	14 15
6 Жизненный цикл безопасности системы Примечание 1 — Это требование отличается от требования подраздела 6.1 МЭК 60880. Для программного обеспечения оно желательно, но не рассматривается в качестве необходимого требования завершать каждую фазу разработки до начала следующей фазы, выполняемой в соответствии с указанными выше требованиями. Примечание 2 — Для систем класса 1 требования к программному обеспечению на данной стадии определены в МЭК 60880. Примечание 3 — Для систем класса 1 требования к существующему программному обеспечению на данной стадии определены в МЭК 60880-2*.	6.1 Нет Нет	8.1

Продолжение таблицы J.1

Пункты МЭК 61513, в которых есть ссылки на	Пункты МЭК 60880 (1986 г.)/Доп.	Пункты настоящего стандарта
6.1.1.1.1 Прикладная функция ...Количественная оценка надежности прикладных функций может потребоваться при верификации проекта системы и общего проекта энергоблока (см. подраздел А.2.2 приложения А, МЭК 60880). Эту оценку обычно проводят при проектировании оборудования системы, т.к. здесь уже имеется накопленный опыт, однако не существует метода, пригодного для количественной оценки надежности программного обеспечения (см. 6.1.3.1.2)....	А.2.2	Нет
6.1.1.2.1 Архитектура системы Примечание — Отказы из-за программного обеспечения являются систематическими, а не случайными. Поэтому критерий единичного отказа не может применяться при разработке программного обеспечения системы в том виде, в каком это делается при проектировании оборудования. На уровне каждой системы и архитектуры контроля и управления следует рассматривать возможные воздействия отказа по общей причине из-за программного обеспечения на каждом уровне защиты или между резервированными уровнями (см. 4.1.1 МЭК 60880-2*).	4.1.1 Доп.	13.1
6.1.1.2.2 Внутреннее поведение системы c) (Специальное требование) Для того, чтобы обеспечить высокую степень гарантии в детерминированном поведении, системы класса 1 должны разрабатываться с использованием технологий, подобных тем, которые приведены в приложении В МЭК 60880 (особенно B2.d относительно времени реакции и B2.e относительно прерываний). ... Примечание 3 — См. раздел 1 МЭК 60880 относительно роли приложений к стандарту, и что требуется, если практика отличается от того, что приведено в приложении.	B.2d, B.2e 1	B.2d, B.2e 5.5.3
6.1.1.2.3 Самоконтроль и устойчивость к отказам a) Системы должны проектироваться так, чтобы ошибки и отказы регистрировались как можно раньше с целью поддержания требуемой работоспособности системы. Выявление отказов с помощью устройств самодиагностики не должно осуществляться за счет применения слишком сложных устройств. Насколько возможно для каждого класса системы следует соблюдать требования 4.8 и подраздела А.2.8 приложения А МЭК 60880 к самодиагностике	4.8, А.2.8	6.2, А.2.2
6.1.2 Спецификация системы В соответствии с разделом А.1 приложения А МЭК 60880 на этой фазе определяются требования к программному обеспечению, техническому обеспечению и требования по интеграции системы.	А.1	5.3
6.1.2.1 Выбор ранее существующих компонентов Примечание 2 — В подразделе 4.3 МЭК 60880-2* указаны критерии принятия повторно используемого разработанного программного обеспечения для функции категории А.	4.3	15
6.1.2.2.3 Защита от развития отказов и их побочных эффектов. Примечание — Детальные требования, позволяющие избежать склонности программных комплексов к ошибкам, и требования к верификации и тестированию программных модулей приведены в МЭК 60880 и МЭК 60880-2*.	Нет	Нет

Продолжение таблицы J.1

Пункты МЭК 61513, в которых есть ссылки на	Пункты МЭК 60880 (1986 г.)/Доп.	Пункты настоящего стандарта
6.1.2.3 Спецификация программного обеспечения Примечание 1 — Архитектура программного обеспечения определяет его основные компоненты и субсистемы, их взаимодействие между собой и пути достижения требуемых характеристик. Требования к архитектуре программного обеспечения не входят в настоящий стандарт (для систем класса А см. МЭК 60880 и МЭК 60880-2*). a) Для упрощения спецификации, верификации и валидации прикладных функций, архитектура программного обеспечения должна обеспечить четкое разделение между прикладным программным обеспечением и системным программным обеспечением.(см. B.2a МЭК 60880). В таких случаях верификация и валидация прикладного программного обеспечения могут осуществляться отдельно	Нет B.2a	Нет B.2a
6.1.3 Детальное проектирование системы и его реализация Для систем класса 1 требования на разработку программного обеспечения приведены в МЭК 60880 и МЭК 60880-2*, а на оборудование — в МЭК 60987	Нет	Нет
6.1.4 Интеграция системы Целью данного этапа является сборка оборудования и модулей программного обеспечения и проверка совместимости программного обеспечения, установленного на оборудовании (см. раздел 7 МЭК 60880)	7	9
6.1.5 Валидация системы Целью данного этапа является тестирование интегрированной системы для обеспечения соответствия с функциональными, операционными спецификациями и спецификациями интерфейса (см. раздел 8 МЭК 60880). b) (специальное требование) Требования раздела 8 МЭК 60880 должны применяться к валидации функции категории А.0	8	10
6.1.7 Модификация проекта системы e) (специальное требование) Для систем класса 1 процесс модификации программного обеспечения должен осуществляться в соответствии с разделом 9 МЭК 60880, а процесс модификации оборудования — в соответствии с разделом 11 МЭК 60987	9	11
6.2.1 План обеспечения качества системы Примечание — Требования к плану обеспечения качества программного обеспечения систем безопасности указаны в разделе 3 МЭК 60880. e) План обеспечения качества должен быть сформирован на ранней стадии жизненного цикла системы и сформирован, исходя из общего плана других мероприятий жизненного цикла безопасности I&C. План может входить либо в спецификацию систем, либо выпускаться в качестве отдельного документа (см. подраздел 3.2 МЭК 60880)	3 3.2	5.5 5.5
6.2.1.1 План верификации системы h) (специальное требование) Для систем класса 1 план верификации системы должен выполняться лицами, не занятыми в ее проектировании (в соответствии с 6.2.1 МЭК 60880)	6.2.1	8.2.1
6.2.3 План интеграции системы Система интеграции включает в себя работу по интеграции субсистем в систему и интеграции программного обеспечения для СВ систем. В значительной степени, система интеграции охватывает работу, описанную в подразделе 7.4 МЭК 60880	7.4	9.1

Окончание таблицы J.1

Пункты МЭК 61513, в которых есть ссылки на	Пункты МЭК 60880 (1986 г.)/Доп.	Пункты настоящего стандарта
6.2.4 План валидации системы b) (специальное требование) Для функций категории А должен быть разработан план валидации системы, при этом мероприятия по осуществлению валдации должны выполняться группами лиц, не занятых в проектировании, реализации и/или модификации системы (см. раздел 8 МЭК 60880)	8	10
6.2.5 План инсталляции системы b) (специальное требование) Для систем класса 1 план инсталляции системы должен выполняться в соответствии с разделом 8 МЭК 60987 и 10.1.1 МЭК 60880)	10.1.1	Нет
6.3.1.2 Характеристики Примечание 3 — Подробные требования в отношении программных средств для систем класса 1 изложены в подразделе 4.2 МЭК 60880-2*.	4.2 Приложение	14
6.3.3 Детальная проектная документация системы Примечание — Для систем класса 1, требования к оформлению документации на программное обеспечение изложены в МЭК 60880 и МЭК 60880-2*, а на оборудование — в МЭК 60987	Нет	Нет
6.3.4.2 Характеристики b) (специальное требование) Для систем класса 1 применяются требования подраздела 7.7 МЭК 60880	7.7	9.5
6.3.5.2 Характеристики (специальное требование) Для систем класса 1 применяются требования подраздела 8.7 МЭК 60880	8.1	10.3
6.4.1.2 Анализ и оценка программного обеспечения b) (специальное требование) Для систем класса 1 вновь разработанное программное обеспечение должно быть подвергнуто оценке в соответствии с требованиями МЭК 68880. c) (специальное требование) Ранее существующее оборудование, выбранное для систем класса 1, должно быть разработано в соответствии с общепризнанными руководствами и стандартами, обеспечивающими высокий уровень качества, требуемый для функций класса А (см. 8.1.2 МЭК 61226). В частности, должны быть соблюдены требования МЭК 60880-2 на ранее разработанное программное обеспечение и программные средства, а также требования МЭК 60987	Нет Нет Приложение	
Таблица 5 — Требования к спецификации и реализации FSE Валидация: см. МЭК 60880 (6.2.4)	Пункт не определен	
* Заменен. Действует МЭК 6088 (2006).		

J.3 Пункты МЭК 60880 (1986 г.), не отраженные в МЭК 61513

Эти пункты содержатся в настоящем стандарте и были изменены там, где это необходимо. Они должны быть рассмотрены при разработке следующей редакции стандарта.

Таблица J.2 — Пункты МЭК 60880, которые должны быть рассмотрены при следующем пересмотре МЭК 61513

6.3 Периодические испытания
9.2 Интеграция системы
10.2 Валидация системы
12.4 Обучение операторов

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
МЭК 60671	_	*
МЭК 61069-2:1993	_	*
МЭК 61226	IDT	ГОСТ Р МЭК 61226—2011 «Атомные станции. Системы контроля и управления, важные для безопасности. Классификация функций контроля и управления»
МЭК 61508-4	IDT	ГОСТ Р МЭК 61504-4—2007 «Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью. Часть 3. Требования к программному обеспечению»
МЭК 61513	IDT	ГОСТ Р МЭК 61513—2011 «Атомные станции. Системы контроля и управления, важные для безопасности. Общие требования к системам»
ИСО/МЭК 9126	IDT	ГОСТ Р ИСО/МЭК 9126—93 «Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению»
Руководство МАГАТЭ NS-G-1.2	—	*
Руководство МАГАТЭ NS-G-1.3	—	*

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

- IDT — идентичные стандарты.

УДК 004.384.044.4.4:621.311.25:621.039:006.354 ОКС 27.120.20

Ключевые слова: атомная электростанция; архитектура контроля и управления; системы контроля и управления, важные для безопасности; жизненный цикл безопасности; функция контроля и управления категории А; разработка программного обеспечения; инсталляция; функциональная валидация

Редактор *В. Н. Копысов*
Технический редактор *В. Н. Прусакова*
Корректор *Е. Ю. Митрофанова*
Компьютерная верстка *З. И. Мартыновой*

Сдано в набор 19.10.2011. Подписано в печать 12.12.2011. Формат 60×84 1/8. Бумага офсетная. Гарнитура Ариал. Печать офсетная. Усл. печ. л. 10,23. Уч.-изд. л. 10,05. Тираж 84 экз. Зак. 1285

ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru
Набрано и отпечатано в Калужской типографии стандартов, 248021 Калуга, ул. Московская, 256.