ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р МЭК 62138— 2021

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМ КОНТРОЛЯ И УПРАВЛЕНИЯ АТОМНОЙ СТАНЦИИ, ВЫПОЛНЯЮЩИХ ФУНКЦИИ БЕЗОПАСНОСТИ КАТЕГОРИЙ B и C

Общие требования

(IEC 62138:2018, Nuclear power plants — Instrumentation and control systems important to safety — Software aspects for computer-based systems performing category B or C functions, IDT)

Издание официальное

Москва
Российский институт стандартизации
2022

Предисловие

1 ПОДГОТОВЛЕН Акционерным обществом «Русатом Автоматизированные системы управления» (АО «РАСУ») на основе собственного перевода на русский язык англоязычной версии стандарта, указанного в пункте 4

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 20 декабря 2021 г. № 1816-ст

4 Настоящий стандарт идентичен международному стандарту МЭК 62138:2018 «Атомные электростанции. Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории B и C» (IEC 62138:2018 «Nuclear power plants — Instrumentation and control systems important to safety — Software aspects for computer-based systems performing category B or C functions», IDT).

Наименование настоящего стандарта изменено относительно наименования указанного международного стандарта для приведения в соответствие с ГОСТ Р 1.5—2012 (пункт 3.5).

Дополнительная информация, поясняющая текст примененного международного стандарта, приведена в сносках.

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

5 ВЗАМЕН ГОСТ Р МЭК 62138—2010

6 Положения настоящего стандарта действуют в целом в отношении атомных станций, сооружаемых по российским проектам за пределами Российской Федерации

Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «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 (www.rst.gov.ru)

Содержание

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМ КОНТРОЛЯ И УПРАВЛЕНИЯ АТОМНОЙ СТАНЦИИ, ВЫПОЛНЯЮЩИХ ФУНКЦИИ БЕЗОПАСНОСТИ КАТЕГОРИЙ B и C

Общие требования

Software for instrumentation and control systems of a nuclear power plant performing safety functions of categories B and C. General requirements

Дата введения — 2022—09—01

1 Область применения

Настоящий стандарт устанавливает требования к программному обеспечению компьютерных систем контроля и управления (СКУ), выполняющих функции безопасности категории B или C, определяемые в соответствии с МЭК 61226. Данный стандарт дополняет МЭК 60880, который устанавливает требования к программному обеспечению компьютерных СКУ, выполняющих функции безопасности категории A.

Настоящий стандарт взаимосвязан с МЭК 61513 и дополняет его. Действия, относящиеся преимущественно к системному уровню (например интеграция, валидация и монтаж), в настоящем стандарте подробно не рассматриваются. Неспецифичные для программного обеспечения требования изложены в МЭК 61513.

Взаимосвязь между категориями функций и классами систем описана в МЭК 61513. Поскольку классифицированные по безопасности СКУ, входящие в область распространения настоящего стандарта, могут выполнять функции различных категорий, включая функции, не классифицированные по безопасности, требования настоящего стандарта привязаны не к категориям функций, а к классу безопасности СКУ (класс 2 или класс 3).

Настоящий стандарт не предназначен для использования в качестве технического руководства по программному обеспечению общего назначения. Он применим к программному обеспечению СКУ, относящихся к классам безопасности 2 или 3, предназначенных для новых АС, а также для обновляемых или модернизируемых действующих АС.

Для действующих АС применима только часть требований, и эта часть должна быть определена в начале любого проекта.

Цель указаний, представленных в настоящем стандарте, заключается в том, чтобы максимально снизить вероятность скрытых программных дефектов, которые могут привести к системным отказам из-за единичных или множественных программных сбоев (т. е. отказы по общей причине, связанные с программным обеспечением).

Настоящий стандарт не рассматривает напрямую вопросы защиты программного обеспечения от угроз, возникающих в результате вредоносных атак, то есть вопросы кибербезопасности компьютерных систем. Требования к программам безопасности для компьютерных систем приведены в МЭК 62645.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты. Для датированных ссылок применяют только указанное издание ссылочного стандарта, для недатированных — последнее издание (включая все изменения):

Издание официальное

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

IEC 61226[1)], Nuclear power plants — Instrumentation and control important to safety — Classification of instrumentation and control functions (Атомные электростанции. Системы контроля и управления, важные для безопасности. Классификация функций контроля и управления)

IEC 61513:2011, Nuclear power plants — Instrumentation and control important to safety — General requirements for systems (Атомные электростанции. Системы контроля и управления, важные для безопасности. Общие требования)

IEC 62671:2013, Nuclear power plants — Instrumentation and control important to safety — Selection and use of industrial digital devices of limited functionality (Атомные электростанции. Системы контроля и управления, важные для безопасности. Выбор и использование промышленных цифровых устройств ограниченной функциональности)

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.

[МЭК 60880:2006, 3.1]

3.2 **прикладная функция** (application function): Функция СКУ, которая выполняет задачу, связанную в большей степени с контролируемым процессом, а не с работой самой системы.

[МЭК 61513:2011, 3.1][2)]

3.3 **прикладное программное обеспечение** (application software): Часть программного обеспечения СКУ, которое обеспечивает выполнение прикладных функций.

Примечания

1 Прикладное программное обеспечение отличается от системного программного обеспечения.

2 Прикладное программное обеспечение относится к конкретной АС и не должно рассматриваться как ранее разработанное ПО.

[МЭК 61513:2011, 3.2, с изменениями (изменены примечания)]

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

Примечания

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

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.

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

4 См. также термин «универсальный язык».

[МЭК 60880:2006, 3.3, с изменениями (добавлено примечание 4)]

3.5 **отказ по общей причине;** ООП (common cause failure; CCF): Отказ двух или более структур, систем или компонентов, явившийся результатом определенного события или причины[3)].

[1)] Действует IEC 61226:2020 «Nuclear power plants — Instrumentation, control and electrical power systems important to safety — Categorization of functions and classification of systems» (Атомные электростанции. Системы контроля и управления и электроэнергетические системы, важные для безопасности. Категоризация функций и классификация систем).

[2)] В примененном международном стандарте допущена ошибка. Должно быть: «[МЭК 61513:2011, 3.1, с изменениями (исключены примечания)]».

[3)] Согласно НП-001-15, пункт 46: отказы по общей причине — это отказы систем (элементов), возникающие вследствие одного отказа, или ошибки персонала, или внутреннего либо внешнего воздействия (события), или иной причины.

Примечание — Общие причины могут быть внешними или внутренними по отношению к СКУ.

[Глоссарий МАГАТЭ по безопасности, издание 2016 г.]

3.6 **сложность** (complexity): Степень, при которой проектирование, реализация или поведение системы, или элемента является трудной для понимания или верификации.

[МЭК 61513:2011, 3.9]

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

Примечание — Термин включает как традиционные программы, написанные на языках общего назначения, так и программы, написанные на проблемно-ориентированных языках.

[МЭК 60880:2006, 3.10, с изменениями (добавлено примечание)]

3.8 **компьютерный элемент** (computer-based item): Элемент, который основан на программных инструкциях, работающих на микропроцессорах или микроконтроллерах.

Примечания

1 В данной терминологической статье термин «элемент» может быть заменен на термины «система», «оборудование» или «устройство».

2 Компьютерный элемент является видом программируемого цифрового элемента.

3 Этот термин эквивалентен термину «программный элемент».

3.9 **управление конфигурацией** (configuration management): Процесс определения и документального оформления характеристик структур, систем и компонентов (включая компьютерные системы и программное обеспечение), а также процесс обеспечения гарантии того, что разработка, оценка, согласование, выпуск, внедрение, проверка, запись, а также включение изменений данных характеристик в документацию было проведено надлежащим образом.

[Глоссарий МАГАТЭ по безопасности, издание 2016 г.]

3.10 **кибербезопасность** (cybersecurity): Комплекс мероприятий и мер, направленных на предотвращение, обнаружение и реагирование на цифровые атаки, которые могут привести:

- к раскрытию информации, которая может быть использована для совершения злонамеренных действий, могущих привести к возникновению аварии, небезопасной ситуации или снижению производительности АС (нарушению конфиденциальности);
- вредоносной модификации функций, которая может поставить под угрозу нормальную работу HPD, отвечающих за передачу и целостность сигналов при обслуживании компьютеризированных СКУ (включая потерю контроля), что может привести к возникновению аварии, небезопасной ситуации или снижению производительности АС (нарушению целостности);
- злонамеренному препятствованию или блокировке доступа к информации или передачи информации, данных или ресурсов (включая потерю визуализации), что может привести к нарушению нормальной работы СКУ, что, в свою очередь, может привести к возникновению аварии, небезопасной ситуации, или снижению производительности АС (нарушению доступности).

Примечание — Данное определение адаптировано к области применения МЭК 62645 с упором на предотвращение, обнаружение и реагирование на злонамеренные воздействия на СКУ компьютеризированных программных устройств, отвечающих за язык описания аппаратных средств. Известно, что в других стандартах и руководствах термин «кибербезопасность» имеет более широкое значение, принимая во внимание непреднамеренные угрозы, человеческие ошибки и защиту от стихийных бедствий, которые в область применения МЭК 62645 не входят.

[МЭК 62645:2019, 3.7 с изменениями (исключено примечание 2)]

3.11 **специализированная функциональность** (dedicated functionality): Свойство устройств, предназначенных для выполнения только одной ясно определенной функции или только очень узкого диапазона функций, например таких, как снятие значения технологического параметра и оповещение о нем или переключение источника электропитания переменного тока на постоянный ток.

Примечания

1 Данная функция (или узкий диапазон функций) является внутренне присущей устройству, а не полученной в результате программирования пользователем.

2 Вспомогательные функции (например, самоконтроль, самокалибровка, передача данных) могут также быть реализованы внутри устройства, но они не изменяют фундаментальную узкую область применимости устройства.

3 Слово «специализированный» в том смысле, в котором оно используется в МЭК 62671, относится к проектированию для одной конкретной функции, которую нельзя изменить в производственных условиях.

[МЭК 62671:2013, 3.7, с изменениями (добавлено примечание 1)]

3.12 **проектная спецификация** (design specification): Документ или комплект документов, который описывает устройство и работу объекта и используется в качестве основы для применения и интеграции объекта.

3.13 **документация по безопасности** (documentation for safety): Документ или комплект документов, который указывает, каким образом продукт может быть безопасно использован для решения прикладных задач, важных для безопасности.

Примечание — Данное определение следует использовать в контексте ранее разработанного ПО (см. 6.3).

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

[МЭК 60880:2006, 3.15]

3.15 **электрическое/электронное/программируемое электронное изделие;** Э/Э/ПЭ изделие (electrical/electronic/programmable electronic item; E/E/PE item): Изделие, основанное на электрической (Э) и/или электронной (Э) и/или программируемой электронной (ПЭ) технологии.

Примечание — В данном термине и его определении слово «изделие» может быть заменено на слова: система, оборудование или устройство.

[МЭК 61508-4:2010, 3.2.13, с изменениями (добавлено понятие «изделие» и внесены изменения в примечание)]

3.16 **комплекс оборудования** (equipment family): Набор аппаратных и программных компонентов, которые могут работать совместно в одной или более определенных архитектурах или конфигурациях[1)].

Примечания

1 Комплекс оборудования может быть как продуктом от определенного производителя, так и набором продуктов, соединение и адаптация которых выполнены поставщиком.

2 Термин «платформа оборудования» иногда используется как синоним термина «комплекс оборудования».

[МЭК 61513:2011, 3.17, с изменениями (исключено примечание 1)]

3.17 **ошибка** (error): Различие между рассчитанным, наблюдаемым или измеренным значением или состоянием и истинным, установленным или теоретическим значением или состоянием.

Примечание — См. также термины: «ошибка персонала», «дефект», «отказ».

[МЭК 61513:2011, 3.18, с изменениями (добавлено примечание)]

3.18 **исполняемая программа** (executable code): Программное обеспечение, которое включено в специализированную систему.

Примечание — Исполняемая программа обычно включает в себя команды, которые должны выполняться техническим обеспечением специализированной системы, и сопутствующие данные.

3.19 **отказ** (failure): Неспособность конструкции, системы или компонента функционировать в пределах критериев приемлемости.

Примечания

1 Под вышедшим из строя оборудованием понимается то оборудование, которое не может выполнять свои функции, вне зависимости требуются ли они в данный момент или нет. Например, отказ резервной системы может не проявляться до тех пор, пока не потребуется ее функции, либо во время тестирования, либо при отказе системы, в качестве резерва которой она выступает.

2 Отказ является результатом сбоя аппаратного или программного обеспечения, системного сбоя, ошибки оператора, либо технического обслуживания, а также связанной с ним траектории сигнала, которая приводит к отказу.

3 См. также термины: «ошибка персонала», «дефект», «ошибка».

[Глоссарий МАГАТЭ по безопасности, издание 2016 г.]

[1)] Разработка специальной конфигурации для АС и соответствующего прикладного программного обеспечения может поддерживаться программными средствами. Комплекс оборудования обеспечивает набор стандартных операций (например, библиотеку прикладных функций), которые могут быть объединены, образуя специальное прикладное программное обеспечение.

3.20 **дефект** (fault): Неисправность в аппаратуре, программном обеспечении или в компоненте системы.

Примечания

1 Дефекты могут быть результатом случайных отказов, которые возникают, например, из-за деградации аппаратуры в результате старения; возможны систематические дефекты, например, дефекты в программном обеспечении, возникающие из-за ошибок при проектировании.

2 Дефект (особенно дефекты, связанные с проектированием) может оставаться незамеченным пока сохраняются условия, при которых он не отражается на выполнении функции, т. е. пока не произойдет отказ.

3 См. также термины: «ошибка персонала», «ошибка», «отказ».

[МЭК 61513:2011, 3.21, с изменениями (примечание 3 изменено)][1)]

3.21 **аппаратно-программное обеспечение** (firmware): Программное обеспечение, которое тесно связано с особенностями аппаратных средств, на которые оно установлено.

Примечание — Наличие аппаратно-программного обеспечения, как правило, очевидно пользователю аппаратного средства и может фактически рассматриваться как неотъемлемая часть разработки аппаратных средств (хорошим примером такого программного обеспечения является микрокод процессора). Как правило, аппаратно-программное обеспечение может изменяться только пользователем посредством замены компонентов аппаратных средств (например, микросхемы процессора, карты, программируемого постоянного запоминающего устройства), которые используют это программное обеспечение с компонентами, использующими измененное программное обеспечение (аппаратное программное обеспечение). В этом случае конфигурационное управление компонентами аппаратных средств пользователями оборудования фактически обеспечивает конфигурационное управление аппаратно-программным обеспечением. Аппаратно-программное обеспечение фактически рассматривается как программное обеспечение, встроенное в аппаратное средство.

[МЭК 60987:2007, 3.4]

3.22 **функциональная валидация** (functional validation): Верификация правильности спецификаций прикладных функций относительно функциональных и рабочих требований верхнего уровня.

Примечание — Функциональная валидация дополняет системную валидацию, которая верифицирует соответствие системы спецификации функций.

[МЭК 61513:2011, 3.23]

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

Примечания

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

2 Примеры: Ада, Си, Паскаль.

3 См. также термин «проблемно-ориентированный язык».

[МЭК 60880:2006, 3.20, с изменениями (добавлено примечание 3)]

3.24 **ошибка персонала**[2)] [human error (or mistake)]: Действие персонала, которое приводит к непредвиденному результату.

Примечание — См. также термины: «дефект», «ошибка», «отказ».

[МЭК 61513:2011, 3.26, с изменениями (добавлено примечание)]

3.25 **архитектура контроля и управления** (I&C architecture): Организационная структура систем контроля и управления станции, которые являются важными для безопасности.

[МЭК 61513:2011, 3.27][3)]

[1)] В примененном международном стандарте допущена ошибка. Должно быть: «[МЭК 61513:2011, 3.21, с изменениями (исключены примечания 1 и 4, добавлено примечание 3)]».

[2)] Согласно НП-001–15, пункт 47: «ошибка персонала — единичное непреднамеренное неправильное действие или единичный пропуск правильного действия при управлении системами и элементами АС, или единичное непреднамеренное неправильное действие, или пропуск правильного действия при техническом обслуживании или ремонте систем и элементов АС».

[3)] В примененном международном стандарте допущена ошибка. Должно быть: «[МЭК 61513:2011, 3.27, с изменениями (исключены примечания)]».

3.26 **система контроля и управления;** СКУ (I&C system): Система, основанная на применении электрической и/или электронной, и/или программируемой электронной технологии, выполняющая функции СКУ, а также функции обслуживания и наблюдения, связанные с эксплуатацией самой системы.

Примечания

1 Термин используется как обобщающий, охватывающий все элементы системы, включая внутренние источники питания, датчики и другие входные устройства, скоростные линии передачи данных и другие связи, интерфейсы исполнительных устройств и других выходных устройств (см. примечание 2). Различные функции системы могут использовать как выделенные, так и разделенные ресурсы.

2 Элементы, входящие в состав определенной системы контроля и управления, определяют границы этой системы.

3 См. также термин «Э/Э/ПЭ изделие» и примечания к нему.

4 В соответствии с их типовой функциональностью МАГАТЭ устанавливает различие между системами автоматического и ручного управления, системами взаимодействия «человек — машина», системами защиты и блокировки.

3.27 **интеграция** (integration): Последовательная сборка компонентов и их проверка внутри завершенной системы.

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

[МЭК 60880:2006, 3.24]

3.29 **режим работы** (mode of operation): Функциональное состояние элемента, которое обеспечивает определенный рабочий режим.

Пример — Режим инициализации, нормальный режим, неполные режимы в случае наличия ошибки в элементе.

3.30 **операционное программное обеспечение системы** (operational system software): Программное обеспечение, которое функционирует на основном процессоре во время работы системы.

Пример — Операционная система, входные/выходные драйверы, обработчик исключительных ситуаций, программное обеспечение коммуникаций, библиотеки прикладного программного обеспечения, самодиагностика, управление резервированием и смягченной деградацией.

3.31 **параметр** (parameter): Элемент данных, управляющий поведением СКУ и/или ее программного обеспечения, который может быть изменен операторами во время работы АС.

3.32 **ранее разработанное программное обеспечение;** РПО (pre-developed software; PDS): Часть программного обеспечения, которая уже существует, доступна как коммерческий или запатентованный продукт и предлагается к использованию.

Примечания

1 В данном стандарте ранее разработанное программное обеспечение делят на 2 различных вида:

a) комплексное операционное программное обеспечение системы

b) компоненты программного обеспечения.

2 Ранее разработанное программное обеспечение можно разделить на программное обеспечение, которое не разрабатывалось для определенного оборудования, и программное обеспечение, интегрированное в компоненты оборудования и используемое совместно с этим оборудованием.

3 В настоящем стандарте данный термин не включает инструментальное программное обеспечение, даже если оно было ранее разработано.

4 Прикладное программное обеспечение относится к конкретной АС и не должно рассматриваться как ранее разработанное программное обеспечение.

[МЭК 60880:2006, 3.28 с изменениями (добавлены примечания)]

3.33 **существующие узлы** (pre-existing items): Аппаратное или программное обеспечение или оборудование с программным обеспечением, которые уже существуют как коммерческий или специализированный продукт и доступны для применения.

Примечание — Данная статья включена для совместимости с терминами и определениями, приведенными в МЭК 61513:2011. В настоящем стандарте, который рассматривает программное обеспечение, используется термин «ранее разработанное программное обеспечение».

[МЭК 61513:2011, 3.36 с изменениями (изменено примечание)]

3.34 **программируемый цифровой элемент**[1)] (programmable digital item): Элемент для выполнения функции, основанный на программных инструкциях или программируемой логике.

Примечания

1 В данном случае термин «элемент» может быть заменен на термины «система», «оборудование» или «устройство».

2 К основным видам программируемых цифровых элементов относятся компьютерные элементы и элементы программируемой логики.

3 Данный термин, используемый МЭК ПК 45А, является эквивалентом термина «программируемое электронное изделие» (ПЭ изделие), определяемого согласно МЭК 61508.

3.35 **элемент программируемой логики** (programmable logic item): Элемент, который основан на логических компонентах с интегральной схемой, состоящей из логических элементов со схемой взаимного соединения, части которой программирует пользователь.

Примечания

1 В данном случае термин «элемент» может быть заменен на термины «система», «оборудование» или «устройство».

2 Элемент программируемой логики является видом программируемого цифрового элемента.

3 См. также термин «Э/Э/ПЭ изделие» и примечания к нему.

3.36 **самоконтроль** (self-supervision): Автоматическое испытание производительности системного оборудования и непротиворечивости программного обеспечения компьютеризированных систем контроля и управления.

[МЭК 60671:2007, 3.8]

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

[МЭК 61513:2011, 3.51]

3.38 **компонент программного обеспечения** (software component): Одна из составных частей программного обеспечения, которая должна быть интегрирована для формирования комплексного программного обеспечения.

Примечание — В данном стандарте ранее разработанный элемент ПО может рассматриваться в качестве компонента ПО только если он внедрен в более крупное ПО для формирования операционной системы. В частности, верификацию и валидацию комплексного программного обеспечения операционной системы следует выполнять со встроенными программными компонентами. Интеграция может быть осуществлена в рамках программного обеспечения, работающего на одном процессоре, например, для операционных систем или библиотек реального времени. Интеграция может быть также выполнена в рамках программного обеспечения, которое работает в тесной взаимосвязи на нескольких процессорах, например, аппаратно-программное обеспечение коммуникационных модулей или модулей ввода/вывода.

3.39 **разработка программного обеспечения** (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.

3.40 **модификация программного обеспечения** (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.

[МЭК 60880:2006, 3.36]

3.41 **жизненный цикл программного обеспечения безопасности** (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.

[1)] Согласно НП-026–16: «Программируемые цифровые устройства — это элементы управляющих систем, использующие программное обеспечение, включая аппаратно-программные устройства».

[МЭК 60880:2006, 3.37]

3.42 **валидация программного обеспечения** (software validation): Тестирование и оценка интегрированной системы на соответствие спецификациям функциональных, эксплуатационных характеристик и интерфейсов, содержащихся в требованиях к СКУ.

Примечание — В настоящем стандарте валидацию ПО рассматривают как часть валидации системы.

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

[МЭК 60880:2006, 3.40]

3.44 **системное программное обеспечение** (system software): Программное обеспечение, спроектированное для определенной компьютерной системы или семейства компьютерных систем с целью упрощения эксплуатации и обслуживания компьютерной системы и связанных программ, например, операционные системы, ЭВМ, утилиты. Системное программное обеспечение обычно состоит из программного обеспечения операционной системы и инструментальных программ.

Примечания

1 Программное обеспечение операционной системы: программы, исполняемые в заданном процессоре в течение времени работы системы, такие как, операционная система, драйверы ввода/вывода, обработчик исключений, коммуникационное программное обеспечение, библиотеки прикладного программного обеспечения, самоконтроль, управление избыточностью и амортизацией отказов (мягкой деградацией).

2 Инструментальные программы: программное обеспечение, применяемое для разработки, тестирования или обслуживания другого программного обеспечения и систем, такое как компиляторы, генераторы кода, графический редактор, офлайн-диагностика, средства верификации и валидации и т. д.

3 См. также «прикладное программное обеспечение».

[МЭК 61513:2011, 3.58 с изменениями (добавлены примечания 2, 3 и 4)][1)]

3.45 **валидация системы** (system validation): Подтверждение путем проверки и предоставления других свидетельств того, что система полностью удовлетворяет спецификации требований, как запланировано (функциональность, время отклика, устойчивость к дефектам, надежность).

Примечание — Глоссарий МАГАТЭ, издание 2016 года, приводит два следующих определения:

Валидация — процесс определения пригодности продукта или услуги для удовлетворительного выполнения определенных функций. Валидация по своему содержанию шире, чем верификация, и может включать более значительный элемент суждения.

Валидация компьютерной системы — процесс испытаний и оценки интегрированной компьютерной системы (аппаратные средства и программное обеспечение) с целью обеспечения соблюдения функциональных, эксплуатационных и интерфейсных требований.

Во-первых, определение «валидация системы» является частным случаем валидации. Это относится к определенному продукту, а именно к валидации СКУ. Это согласуется с определением МАГАТЭ. Во-вторых, определение МЭК устанавливает базу для валидации, а именно спецификацию требований, тогда как определение МАГАТЭ ссылается только на «определенную функцию».

[МЭК 61513:2011, 3.59]

3.46 **систематический дефект** (systematic fault): Дефект, связанный детерминированным образом с какой-либо причиной, который может быть исключен только путем модификации проекта или производственного процесса, эксплуатационных процедур, документации, либо других важных факторов.

[МЭК 61513:2011, 3.60]

3.47 **верификация**[2)] (verification): Подтверждение исследованием и представлением объективного доказательства того, что результаты деятельности соответствуют целям и требованиям, определенным для этой деятельности.

[МЭК 61513:2011, 3.62]

[1)] В примененном международном стандарте допущена ошибка. Должно быть: «[МЭК 61513:2011, 3.58, с изменениями (исключено примечание 4)]».

[2)] Согласно НП-026–16, пункт 6: «верификация — это подтверждение на основе представления объективных свидетельств того, что результат деятельности на стадии жизненного цикла управляющей системы АС, важной для безопасности, получен с соблюдением требований, предъявляемых к этой системе на данной стадии жизненного цикла системы».

4 Символы и аббревиатуры

СВ	—	компьютеризированный
EPROM	—	стираемое программируемое постоянное запоминающее устройство
HDL	—	язык описания аппаратных средств
HPD	—	HDL-программируемое устройство
I&C	—	контроль и управление
АС (NPP)	—	атомная станция
ООП (CCF)	—	отказ по общей причине
ЧМИ (HMI)	—	человеко-машинный интерфейс

5 Ключевые концепции и допущения

5.1 Общие положения

В разделе 5 представлены некоторые ключевые концепции и допущения, которые касаются характера ПО и вопросов его разработки для СКУ классов безопасности 2 или 3, на которых базируется нормативный документ.

5.2 Типы программного обеспечения

На рисунке 1 представлены различные виды работ, выполняемых программным обеспечением в типичной СКУ или в архитектуре контроля и управления. Программное обеспечение часто может быть определено как системное или прикладное ПО. Системное ПО может также быть подразделено на операционное ПО системы, встроенное в СКУ, важную для безопасности, и поддерживающее ПО (или инструментальные программные средства), которое является автономным или встроенным в системы поддержки, классифицируемые как не важные для безопасности. ПО может также находиться в специализированных устройствах, таких как датчики и исполнительные механизмы, устройства связи и устройства бесперебойного электропитания (УБЭП).

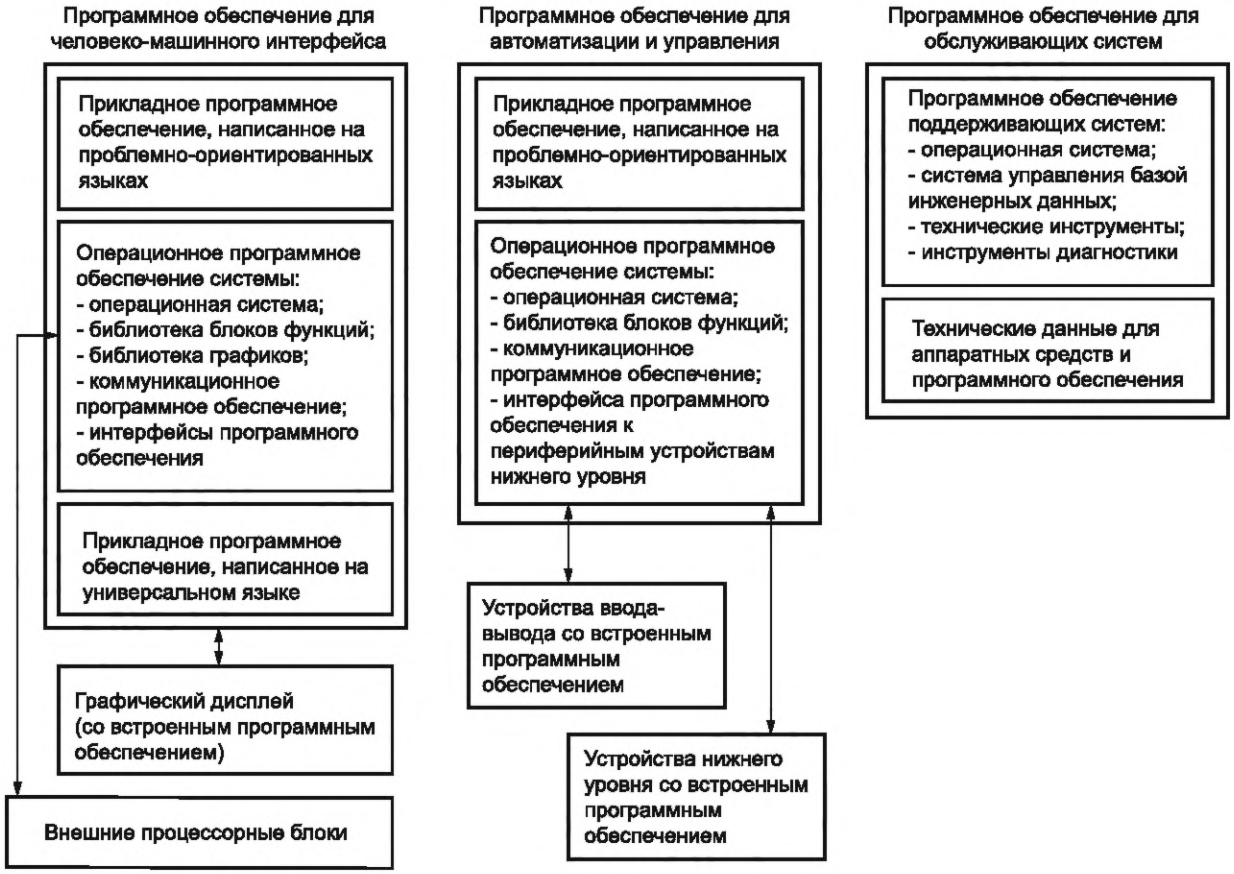


Рисунок 1 — Типичные части ПО в компьютерных СКУ

Программное обеспечение СКУ также подразделяют на ранее разработанное ПО (которое обычно обеспечивает функции, полезные для ряда СКУ) и новое ПО (которое разработано для конкретных задач СКУ). Требования настоящего стандарта, которым должно соответствовать новое ПО, могут также быть предъявлены и к ранее разработанному ПО. Однако в ряде случаев настоящий стандарт предусматривает альтернативные требования, которым должно, в частности, соответствовать ранее разработанное ПО.

Многие современные комплексы оборудования обеспечены ориентированными на решение широкого спектра прикладных задач инструментальными программными средствами (инструментальным ПО), которые позволяют инженерам АС или системным инженерам устанавливать свои требования к ПО с использованием графических методов. Инструментальное ПО может автоматически преобразовывать представленные графически программы в исполняемое прикладное ПО. При надлежащем качестве инструментального ПО считают, что данный подход приводит к уменьшению риска проявления дефектов.

5.3 Типы данных конфигурации

Во многих проектах систем широко используют данные конфигурации. Данные конфигурации могут быть связаны с операционным ПО системы или с прикладным ПО. Данные конфигурации, связанные с прикладным ПО, состоят в основном из технических данных АС, вытекающих из проекта АС, и обычно бывают подготовлены проектировщиками станции, которые не обязаны иметь навыки работы с программным обеспечением. Данные конфигурации могут быть разделены следующим образом:

- элементы данных, которые не предназначены для изменения операторами АС в режиме реального времени и подчиняются тем же требованиям, что и остальная часть программного обеспечения;

- параметры, т. е. элементы данных, которые могут быть изменены операторами при эксплуатации АС (например пределы срабатывания аварийной сигнализации, заданные значения, данные, необходимые для калибровки аппаратуры) и к которым предъявляют специфические требования.

5.4 Жизненные циклы программного обеспечения и системы безопасности

Программное обеспечение призвано всячески содействовать выполнению функций СКУ. Оно может также поддерживать дополнительные функции, необходимые для работы самой системы (например, инициализацию и контроль средств технического обеспечения, связь между подсистемами и синхронизацию подсистем). Таким образом, жизненный цикл программного обеспечения безопасности в большинстве случаев тесно связан с жизненным циклом системы безопасности. В частности, спецификация требований к ПО является частью или непосредственно вытекает из технических характеристик и проекта системы.

Хотя верификация программного обеспечения несомненно является частью жизненного цикла ПО безопасности, часто нет отдельной и четко установленной границы между интеграцией ПО и интеграцией системы. Настоящий стандарт рассматривает интеграцию ПО как часть интеграции системы. Валидацию ПО также рассматривают как часть валидации системы.

На рисунках 2 и 3 представлено соотношение между мероприятиями жизненного цикла ПО безопасности и мероприятиями жизненного цикла системы безопасности.

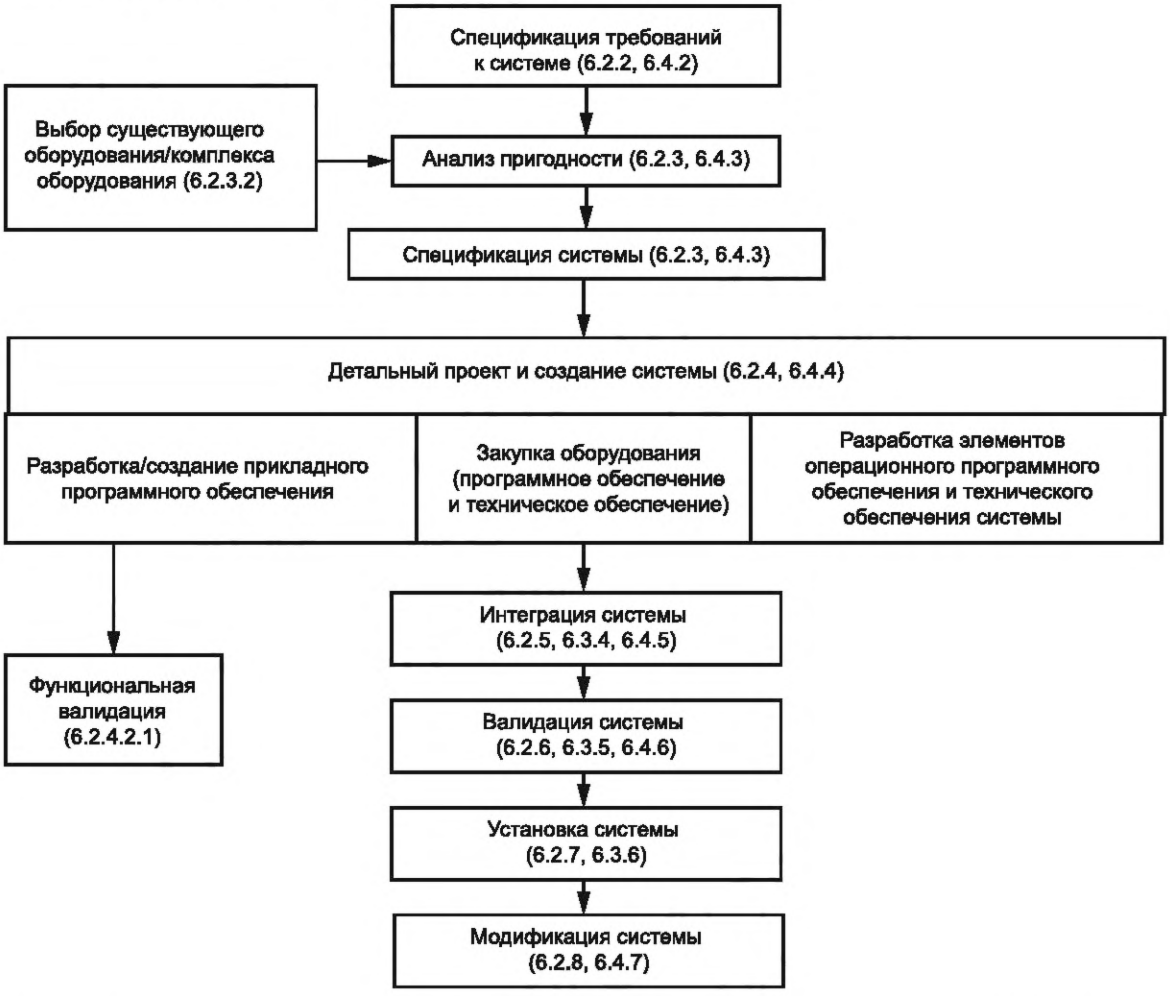
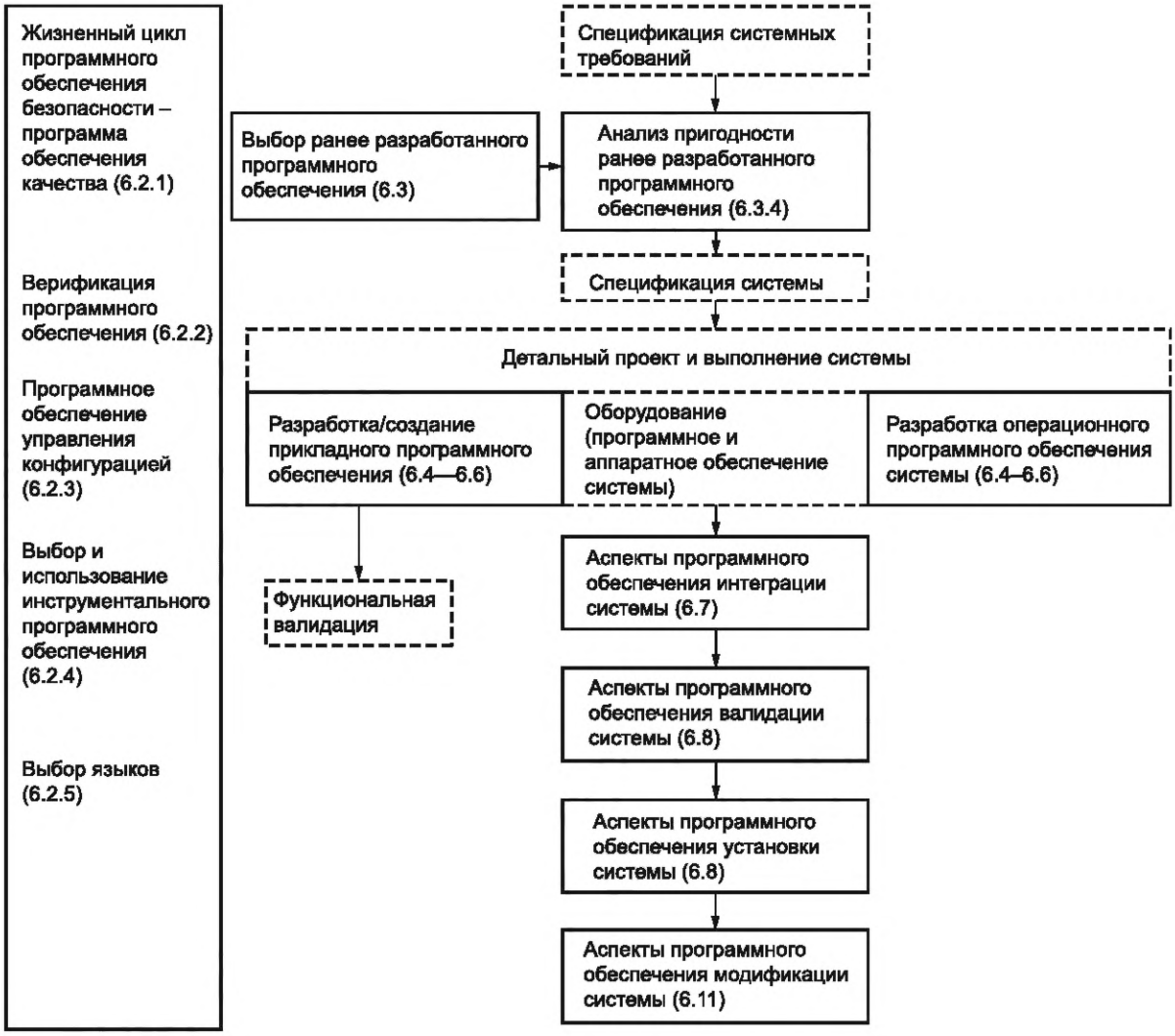


Рисунок 2 — Мероприятия жизненного цикла системы безопасности (в соответствии с МЭК 61513:2011)



Примечание — Блоки, выделенные пунктиром, относятся к мероприятиям, которые в настоящем стандарте не рассматриваются.

Рисунок 3 — Мероприятия жизненного цикла системы безопасности, относящиеся к программному обеспечению системы

Необходимо отметить, что хотя МЭК 61513:2011 устанавливает два различных направления создания программного обеспечения (прикладное ПО и операционное ПО системы, см. рисунки 2 и 3), настоящий стандарт делит требования к созданию ПО на четыре подгруппы:

- требования, применимые независимо от технологии создания (см. 6.6.1);
- требования, специфичные для конфигурации ранее разработанного ПО и устройств, включающих ПО, в частности, связанных с установлением параметров и другими данными конфигурации (см. 6.6.2);
- требования, специфичные для реализации и верификации ПО, написанного на проблемно-ориентированных языках (см. 6.6.3);
- требования, специфичные для реализации и верификации ПО, написанного на универсальных языках (см. 6.6.4).

Блоки, названные на рисунке 3 «Разработка/создание прикладного программного обеспечения» и «Разработка операционного программного обеспечения системы», представляют большую и важную часть жизненного цикла ПО безопасности. На рисунке 4 более подробно показана деятельность, осуществляемая между составлением спецификации требований к ПО и валидацией ПО с четким представлением трех различных направлений реализации проекта ПО (конфигурация ранее разработанного

ПО и устройств, использование проблемно-ориентированных языков и использование универсальных языков).

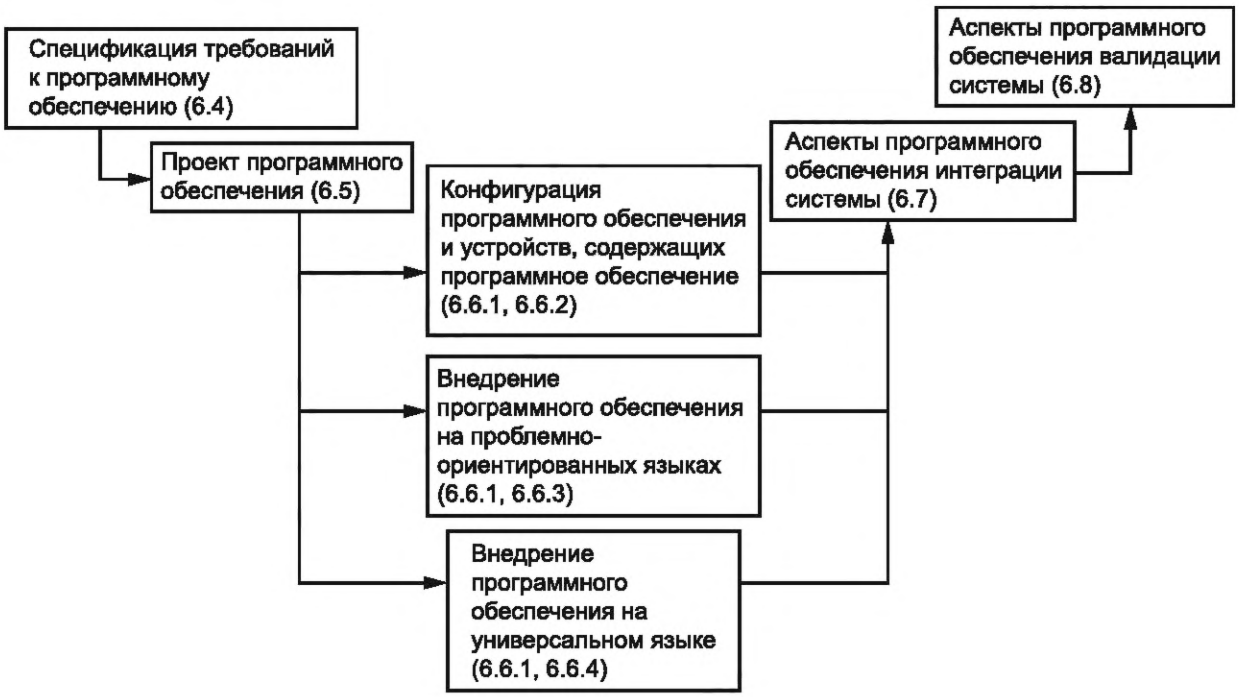


Рисунок 4 — Виды деятельности по разработке программного обеспечения безопасности как часть его жизненного цикла (согласно настоящему стандарту)

5.5 Принципы градации

Вследствие распределения функций, имеющих отношение к безопасности, по категориям A, B и C (см. МЭК 61226) соответствующая градация была введена и для требований к программному обеспечению СКУ, относящихся к классам безопасности 1, 2 и 3.

Информация о программном обеспечении СКУ, относящихся к классу безопасности 1, приведена в МЭК 60880.

Применение требований настоящего стандарта для класса безопасности 3 обеспечивает базовый уровень надежности, соответствующий программному обеспечению СКУ, важных для безопасности. При этом руководствуются следующими принципами:

- необходимо иметь доверие к контролю качества;
- уделить особое внимание обеспечению того, что ПО:

1) по мере необходимости способствует выполнению функций, важных для безопасности, и не оказывает отрицательного влияния на эти функции;

2) соответствует положениям спецификации требований к ПО, которые устанавливают ограничения, важные для безопасности;

- обеспечить как можно более раннее (в разумных пределах) информирование операторов СКУ об ошибках и отказах ПО, которые могут повлиять на функции, важные для безопасности, чтобы можно было предпринять необходимые действия;
- иметь документально оформленные спецификации требований к ПО, спецификации проекта, спецификации интеграции, спецификации валидации (т. е. комплексное функциональное тестирование) и спецификации модификации.

Для класса безопасности 2 в дополнение к принципам, установленным для класса 3, из настоящего стандарта следуют принципы:

- иметь основанное на испытаниях и проекте подтверждение того, что требуемая рабочая характеристика, связанная с безопасностью (например время реакции системы), будет полностью соответствовать заданным условиям;

- предъявлять более жесткие требования к выбору ранее разработанного ПО;
- предъявлять более жесткие требования к верификации, управлению конфигурацией ПО, выбору и использованию инструментального ПО и языков программирования;
- разработать четко сформулированные требования к простоте, четкости, прецизионности, проверяемости, тестируемости и модифицируемости ПО.

Если требование применимо к обоим классам безопасности, то степень его обоснованности, необходимая для подтверждения соответствия настоящему стандарту, может быть смягчена согласно классу безопасности. В частности, степень обоснованности требования для класса 3 может быть понижена по сравнению с классом 2. Вместе с тем, для определения степени обоснованности функций, не относящихся к важным для безопасности в рамках систем класса 2 или 3, необходимо лишь понять, насколько проект гарантирует, что такие функции не могут отрицательно влиять на функции, определяемые как важные для безопасности.

6 Требования к программному обеспечению СКУ классов 2 и 3

6.1 Применимость требований

В разделе 6 изложены требования и рекомендации настоящего стандарта. При этом требования и рекомендации, для которых не указано специальное назначение, применимы к системам классов безопасности 2 и 3, в то время как для требований и рекомендаций, применимых только к системам класса 2 или только к системам класса 3, такая специализация указана, и они отмечены курсивом.

Все требования и рекомендации приведены в виде отдельных нумерованных абзацев. Все остальные абзацы носят информативный характер. В сущности, ненумерованные абзацы содержат пояснения, которые относятся к предыдущему нумерованному абзацу, если не указано иное. Если абзац без номера содержит пояснения, касающиеся не только предыдущего нумерованного абзаца, то в таких ненумерованных абзацах приводят пометку «Относится к xxx и yyy, ...».

Цель настоящего стандарта состоит не в том, чтобы установить определенный комплект документов, а скорее в том, чтобы определить информацию, которая должна быть документально оформлена. Иерархия и формат принимаемой документации могут быть разными при условии соблюдения принципов, изложенных в настоящем стандарте. В справочном приложении А представлен типовой перечень документации на ПО.

6.2 Требования общего характера

6.2.1 Жизненный цикл программного обеспечения безопасности. Гарантия качества программного обеспечения

6.2.1.1 Общие положения

В пункте 6.3.2 МЭК 61513:2011 приведены требования к обеспечению качества на уровне СКУ. Данный подраздел содержит дополнительные специфичные или особо важные требования к ПО.

6.2.1.2 Разработку ПО следует проводить в соответствии с жизненным циклом ПО безопасности. Средства обеспечения этого жизненного цикла должны быть определены в программе обеспечения качества.

Программа обеспечения качества может быть частью программы обеспечения качества системы или отдельной программой обеспечения качества ПО.

6.2.1.3 При использовании отдельной программы обеспечения качества ПО эта программа должна быть совместима с программой обеспечения качества системы. В программе обеспечения качества ПО необходимо учитывать требования пункта 6.3.2 МЭК 61513:2011, поскольку они относятся к ПО.

6.2.1.4 В программе обеспечения качества этап разработки жизненного цикла ПО безопасности должен быть разделен на конкретные мероприятия. Эти мероприятия должны включать действия, необходимые для достижения требуемого качества ПО, а также действия по верификации с получением объективных доказательств того, что это качество достигнуто.

6.2.1.5 В спецификации мероприятий должны быть указаны:
- их цели;
- взаимоотношения и взаимодействие с другими мероприятиями;
- исходные данные и результаты;
- сведения об организации мероприятий и ответственных за их осуществление.

6.2.1.6 Должны быть также указаны необходимое содержание и свойства исходных данных и результатов.

6.2.1.7 Программа обеспечения качества должна содержать требование о том, что выполнение каждого мероприятия должно быть возложено на компетентных лиц, обеспеченных соответствующими ресурсами.

6.2.1.8 Программа обеспечения качества должна содержать требование о том, чтобы изменения в уже утвержденной документации были идентифицированы, проанализированы и утверждены уполномоченными лицами.

6.2.1.9 Программа обеспечения качества должна содержать требование о том, что используемые методы, языки, инструменты, правила и стандарты должны быть идентифицированы и задокументированы, что они должны быть известны лицам, имеющим отношение к разработке ПО, и входить в область их компетенции.

6.2.1.10 Программа обеспечения качества должна содержать требование о том, что в случае использования нескольких методов, языков, инструментов, правил и/или стандартов необходимо четко указывать, какие из них следует использовать для каждого мероприятия.

6.2.1.11 Программа обеспечения качества должна содержать требование о том, чтобы специфичным для проекта терминам, выражениям, сокращениям и условным обозначениям было дано четкое объяснение.

6.2.1.12 Программа обеспечения качества должна содержать требование о том, чтобы возникающие проблемы были отслежены и решены.

6.2.1.13 Программа обеспечения качества должна содержать требование о предоставлении отчетов о результатах ее применения. В частности, должно быть требование о размещении в отчетах результатов верификации и анализа вместе с областью их применения, сделанных заключений и согласованных решений. Любое отклонение от программы обеспечения качества должно быть обосновано и задокументировано.

6.2.1.14 Программа обеспечения качества должна содержать требование о представлении итоговой документации в виде комплекта документов, которые взаимно согласованы и содержат перекрестные ссылки, благодаря чему обеспечивается возможность отследить соответствие окончательного проекта изначально предъявленным требованиям.

6.2.2 Верификация

6.2.2.1 План верификации должен определять область верификации ПО и осуществляемые при этом мероприятия.

6.2.2.2 План верификации должен учитывать требования, изложенные в подпункте 6.3.2.2 МЭК 61513:2011, поскольку они относятся к ПО.

6.2.2.3 Мероприятия по верификации и анализу должны быть выполнены в соответствии с документально оформленными положениями. План верификации должен обеспечивать следующее:

- результаты верификации рассматривают в рамках управления конфигурацией;
- все мероприятия по верификации основаны на строго определенных исходных данных и результаты мероприятий согласуются с этими исходными данными;
- мероприятия реализуют указанные для них цели, а результаты мероприятий имеют требуемые содержание и свойства и соответствуют каждому из согласованных решений;
- результаты являются очевидными, точными и актуализированными;
- результаты подчиняются всем соответствующим правилам;
- результаты отвечают всем соответствующим требованиям настоящего стандарта.

Слова «строго определенный» в данном контексте означают, что текст имеет однозначную трактовку. Определение «очевидный» означает, что лица, читающие документ, могут полностью понять его без чрезмерных усилий, даже если они ранее не участвовали в проекте, при условии, что они обладают необходимыми знаниями. Слово «точный» означает отсутствие неопределенности.

Объем мероприятий по верификации и анализу может зависеть от масштаба и характера ПО, масштаба и характера результатов, подвергаемых верификации и анализу, а также от применяемых методов и инструментального ПО. Объем мероприятий может быть уменьшен в отношении указанных требований, которые не идентифицированы в качестве важных для безопасности (см. 6.4.4.7) и не могут отрицательно влиять на функции, идентифицированные как важные для безопасности.

6.2.2.4 План верификации должен обеспечивать такое ведение записей, при котором процесс верификации может быть проверен в полном объеме, т. е. должна быть обеспечена возможность проведения независимого подтверждения исполнения плана верификации.

6.2.2.5 Верификацию результатов мероприятий должны проводить компетентные лица, не принимавшие участия в этих мероприятиях.

Это не означает, что лицо, являющееся автором одного документа, не может верифицировать другие документы.

6.2.2.6 В число лиц, привлекаемых к верификации результатов мероприятий, должны входить представители структур, использующих эти результаты, а также, при необходимости, другие эксперты.

6.2.2.7 Верификации подлежат спецификация требований к ПО, спецификация проекта ПО и план валидации ПО.

6.2.2.8 *Для класса 2: верификации подлежат применимость и правила реализации проекта.*

6.2.2.9 Верификацию ПО должны проводить лица, не принимавшие участие в разработке ПО, подлежащего верификации.

6.2.2.10 *Для класса 2: лица, проводящие верификацию, должны быть независимы в административном плане от разработчиков ПО.*

6.2.3 Управление конфигурацией

6.2.3.1 Общие положения

Подпункт 6.3.2.3 МЭК 61513:2011 содержит требования по управлению конфигурацией на уровне СКУ. В настоящем стандарте приведены дополнительные требования к ПО, специфичные или особенно важные для безопасности.

6.2.3.2 Управление конфигурацией ПО следует осуществлять в соответствии с положениями плана управления конфигурацией или программы обеспечения качества. Эти положения должны быть совместимы с положениями плана управления конфигурацией на уровне системы.

6.2.3.3 Управление конфигурацией применимо к элементам, отвечающим за корректность работы ПО. В плане управления конфигурацией должно быть определено, какие элементы ПО или какие типы элементов ПО следует контролировать в рамках управления конфигурацией. В частности, в план должны быть включены:

- ключевые документы жизненного цикла ПО безопасности (в особенности документы, требующие верификации);
- компоненты ПО, необходимые для построения рабочей программы, и сама рабочая программа;
- инструментальные программные средства, влияющие на правильность работы ПО.

6.2.3.4 В плане управления конфигурацией должны быть определены технические средства для идентификации элементов ПО, рассматриваемых в рамках управления конфигурацией, и их версий.

6.2.3.5 План управления конфигурацией должен обеспечивать однозначное определение версии ПО, используемой в данной версии системы или оборудования, а также однозначное определение составляющих элементов ПО.

6.2.4 Выбор и использование инструментального ПО

6.2.4.1 Общие положения

Инструментальное программное обеспечение (вспомогательные программы) может играть важную роль в предотвращении появления дефектов в ПО и обнаружении уже существующих дефектов. В частности, вспомогательные программы могут помочь в проектировании архитектуры СКУ и нового прикладного ПО или автоматизировать процесс проектирования.

6.2.4.2 Вспомогательные программы должны способствовать тем видам деятельности по разработке, которые обеспечивают корректность работы ПО.

Следует уделять внимание не только качеству и использованию отдельных вспомогательных программ, но и их совместимости с другими применяемыми вспомогательными программами, чтобы они вместе составляли гармонично связанное инструментальное программное обеспечение. В большинстве случаев предпочтительно использовать программы с обширным положительным опытом практического применения. Использование других вспомогательных программ может быть оправдано требованиями конкретного процесса разработки.

6.2.4.3 *Для класса 2: комплексы оборудования, используемые для создания СКУ, должны быть связаны с инструментальным ПО, снижающим риск внесения дефектов в новое прикладное ПО.*

6.2.4.4 *Для класса 3: комплексы оборудования, используемые для создания СКУ, должны быть связаны с инструментальным ПО, снижающим риск внесения дефектов в новое прикладное ПО.*

Что касается 6.2.4.3 и 6.2.4.4, инструментальное ПО, о котором идет речь, обычно включает в себя поддержку проблемно-ориентированных языков, позволяя инженерно-техническому персоналу и операторам АС определять или верифицировать прикладные функции. Другими существенными

функциями таких инструментальных ПО могут быть анимация, генерация кодов и помощь в создании спецификаций функциональных испытаний.

6.2.4.5 Комплексы оборудования, используемые при разработке СКУ, должны быть связаны с инструментальным ПО, что может снизить риск появления дефектов конфигурации использованных ранее разработанного программного обеспечения и проекта системы.

Такое инструментальное ПО может, например, помочь проектировщикам системы:

- в организации системы в виде соответствующего набора связанных подсистем;
- распределении прикладных функций между подсистемами;
- *создании конфигурации подсистем, их коммуникаций и операционного ПО системы;*
- обеспечении необходимых ресурсов для всех режимов работы системы;
- учете существующих ограничений при проектировании и реализации, особенно тех, которые нацелены на обеспечение корректности и устойчивости системы.

6.2.4.6 Программа обеспечения качества должна точно идентифицировать вспомогательные программы, которые могут повлиять на корректность ПО.

6.2.4.7 Для таких вспомогательных программ должна быть предусмотрена документация пользователя, чтобы гарантировать их применение надлежащим образом.

6.2.4.8 В программе обеспечения качества должно быть проведено различие между вспомогательными программами, которые могут привести к дефектам ПО и вспомогательными программами, которые могут привести только к тому, что уже существующий дефект будет пропущен.

Например, генераторы кодов и компиляторы относятся к первой категории вспомогательных программ, тогда как статические анализаторы кодов и генераторы контрольных примеров относятся ко второй категории вспомогательных программ.

6.2.4.9 *Для класса 2: выбор и использование вспомогательных программ, которые могут привести к дефектам ПО, следует осуществлять согласно документально оформленных процедур и правил, направленных на уменьшение или смягчение такого риска. Должно быть представлено подтверждение их качества и способности приводить к правильным результатам. При использовании вспомогательных программ для генерации заданного элемента или информации, их использование должно быть задокументировано для возможности идентификации данных программ.*

6.2.4.10 *Для класса 3: должно быть представлено подтверждение качества вспомогательных программ, которые могут привести к дефектам ПО, а также подтверждение их способности приводить к правильным результатам.*

6.2.4.11 Подтверждение качества вспомогательной программы и ее способности приводить к правильным результатам должно базироваться на опыте практического применения, квалификации или сертификации вспомогательной программы, сертификации поставщика программы на компетентность в соответствующем виде деятельности, гарантирующей применение надлежащих процессов разработки и/или тестирования программы. Требуемую строгость подтверждения определяют исходя из условий использования вспомогательной программы, объема получаемых с ее помощью результатов, подлежащих верификации, вероятности обнаружения ошибок и серьезности последствий ошибочных результатов, которые не были выявлены. И наоборот, строгость подтверждения (например, квалификация вспомогательной программы в соответствии с МЭК 60880) может заменить проверку некоторых результатов.

6.2.4.12 *Для класса 2: выбор и использование вспомогательных программ, которые могут не выдать информацию о дефектах ПО, следует осуществлять таким образом, чтобы снизить этот риск.*

6.2.4.13 *Для класса 2: использование вспомогательных программ, которые могут не выдать информацию о дефектах ПО, должно быть задокументировано.*

6.2.4.14 *Для класса 2: при замене вспомогательной программы или ее версии, применение которых может привести к дефектам ПО, на другую программу/версию, необходимо принять меры предосторожности, чтобы эта замена не оказала отрицательного влияния на корректность работы ПО.*

Например, помимо качества и способности новой вспомогательной программы приводить к правильным результатам, может потребоваться оценка ее совместимости с предыдущей вспомогательной программой.

6.2.5 Выбор языков

6.2.5.1 Используемые для создания ПО языки (проблемно-ориентированные или универсальные) должны иметь точные документально оформленные синтаксическую структуру и семантику.

6.2.5.2 По возможности, предпочтение отдают проблемно-ориентированным языкам.

6.2.5.3 *Для класса 2: для конкретных компьютерных программ допустимо использовать машинно-ориентированные универсальные языки низкого уровня (например, ассемблерный язык), однако такое использование должно быть обосновано.*

6.2.5.4 Если для создания программы используют более одного языка, интерфейсы между языками должны быть оформлены документально.

Интерфейс между языками включает в себя схемы передачи параметров и представление структур данных.

6.2.5.5 *Для класса 2: используемые универсальные языки должны иметь свойства, упрощающие инструментальное ПО, которое поддерживает статический анализ компьютерных программ.*

6.2.5.6 Используемые универсальные языки должны поддерживать прямой ввод переменных и их статическую типизацию.

6.2.5.7 *Для класса 2: следует использовать прямой ввод переменных и их статическую типизацию.*

6.2.5.8 *Для класса 2: используемые языки и соответствующие библиотеки рабочих программ должны обеспечивать предопределенное выполнение программ.*

Например, обычно не приемлемо случайное прерывание нормального выполнения программы сбора освобожденной памяти.

6.3 Выбор ранее разработанного программного обеспечения

6.3.1 Общие положения

В подпункте 6.2.3.2 МЭК 61513:2011 изложены общие требования, которыми следует руководствоваться при выборе ранее разработанных компонентов (не обязательно компонентов ПО). Настоящий подраздел 6.3 вводит дополнительные требования, специфичные или особенно важные для ПО.

6.3.1.1 Прикладное ПО относится к конкретной АС и потому не может считаться ранее разработанным ПО.

Примечание — Одинаковое прикладное ПО может быть использовано на различных энергоблоках, созданных по одному проекту с учетом одинаковых требований к безопасности. В этом случае обоснования, выработанные для изначального энергоблока, применимы и для последующих энергоблоков.

6.3.2 Документация по безопасности

6.3.2.1 Цели

6.3.2.1.1 Ранее разработанное ПО должно сопровождаться документами, содержащими информацию, необходимую для беспроблемного использования этого ПО в СКУ.

В настоящем стандарте сопроводительный документ или пакет документов назван документацией по безопасности. Если ранее разработанное ПО является частью оборудования или комплекса оборудования, то его документация может быть частью документации по безопасности на оборудование или комплекс оборудования.

Как правило, документация по безопасности включает в себя не только документацию пользователя, которую предоставляет поставщик ранее разработанного ПО. Она может также содержать информацию, полученную в результате дополнительного тестирования, измерений и/или анализов, а также из опыта эксплуатации.

6.3.2.2 Содержание

6.3.2.2.1 Документация по безопасности должна включать в себя описание:

- предусмотренных функций;
- интерфейсов с прикладным ПО;
- ролевых имен, типов, форматов, диапазонов и пределов входных и выходных данных, а также сигналов о нарушениях, параметров и данных конфигурации там, где это необходимо;
- различных режимов работы и соответствующих условий перехода;
- любых ограничений, относящихся к использованию ранее разработанного ПО.

6.3.2.2.2 *Для класса 2: при необходимости ограничения должны:*

- обеспечивать достаточную уверенность в правильности интегрированного ПО и проекта системы (например, пределов, устанавливаемых при использовании динамически распределяемых ресурсов, таких как память, вычислительная мощность, пропускная способность каналов связи, ресурсы операционной системы);

- усиливать способность интегрированного ПО и СКУ обнаруживать отказы и оповещать о них, быть устойчивым к отказам, работать в указанных режимах и восстанавливаться после отказов;

- обеспечивать достаточную уверенность в том, что ошибки операторов и отказы других систем или оборудования, с которыми интегрированное ПО взаимодействует или использует общие ресурсы, будут приводить к определенным режимам эксплуатации;

- гарантировать, что среда ранее разработанного ПО представит все необходимые ресурсы в рамках всех условий работы СКУ.

6.3.2.2.3 Там, где это необходимо, документация по безопасности должна также содержать информацию, касающуюся выполнения функций (например, в виде времени срабатывания).

Функции, реализуемые ПО, включая те, которые относятся к интерфейсам системы, могут различаться в зависимости от условий эксплуатации АС.

6.3.2.2.4 *Для класса 2: документация по безопасности должна также предоставлять информацию:*

- о выполненном самоконтроле, устойчивости к дефектам и видах отказа;

- требованиях ранее разработанного ПО к среде выполнения (например, к техническим средствам или другим компонентам ПО);

- взаимосвязях и интерфейсах ранее разработанного ПО с техническим обеспечением, насколько это необходимо для определения безопасного функционирования системы.

6.3.2.2.5 *Для класса 2: документация по безопасности операционного ПО системы для ранее разработанного комплекса оборудования должна содержать информацию, позволяющую (в сочетании с данными по конкретному применению) делать правильные прогнозы относительно ключевых элементов системы, важных для безопасности, включая, в частности, максимальное время отклика и максимальное использование ресурсов.*

Такая информация может быть представлена в виде данных, формул и/или моделей, позволяющих рассчитать наихудшее значение времени отклика и использование ресурса приложений. Если ПО предусматривает наличие широкого диапазона функций, интерфейсов и возможностей конфигурации, то без сведений о принципах работы ПО может оказаться трудным с достаточной уверенностью оценить правильность информации.

6.3.2.3 Свойства

6.3.2.3.1 Документация по безопасности должна быть точной, не допускающей двоякой интерпретации.

6.3.3 Доказательство корректности

6.3.3.1 Общие требования

6.3.3.1.1 Корректность предварительно разработанного программного обеспечения с учетом его документации по безопасности должна быть подтверждена.

Обычно подтверждение является качественным, т. к. общепризнанных методов количественной оценки не существует. На рисунках 5 и 6 представлен типичный процесс подтверждения, который может быть использован. Однако, это не единственный подход, могут быть использованы и другие.

6.3.3.1.2 При использовании дополнительных способов предоставления свидетельств правильности на ранних этапах жизненного цикла ПО безопасности должны быть определены и обоснованы приемочные критерии. Эти критерии должны быть обоснованы с учетом требований настоящего стандарта, соответствие которым в достаточной мере не установлено.

6.3.3.1.3 Различают два вида ранее разработанного ПО:

a) комплексное операционное ПО системы;

b) компоненты ПО (операционная система, работающая в режиме реального времени, библиотека, аппаратно-программное обеспечение).

Примечание — Прикладное ПО относится к конкретной АС и потому не может считаться ранее разработанным ПО (см. 6.3.1.1).

Мотивация такого разделения состоит в том, что для формирования комплексного операционного ПО системы компоненты ПО должны быть интегрированы в более крупное ПО. Это означает, что компоненты ПО получают преимущества в процессе разработки комплексного операционного ПО системы, с которым их объединяют. Верификация и валидация функциональных возможностей компонентов ПО в этом случае может быть осуществлена в рамках их использования в комплексном операционном ПО системы. Таким образом, рекомендуемый подход к подтверждению правильности комплексного операционного ПО системы с точки зрения его документации по безопасности (см. рисунок 5) является более трудоемким, чем рекомендуемый подход к подтверждению правильности компонентов ПО (см. рисунок 6).

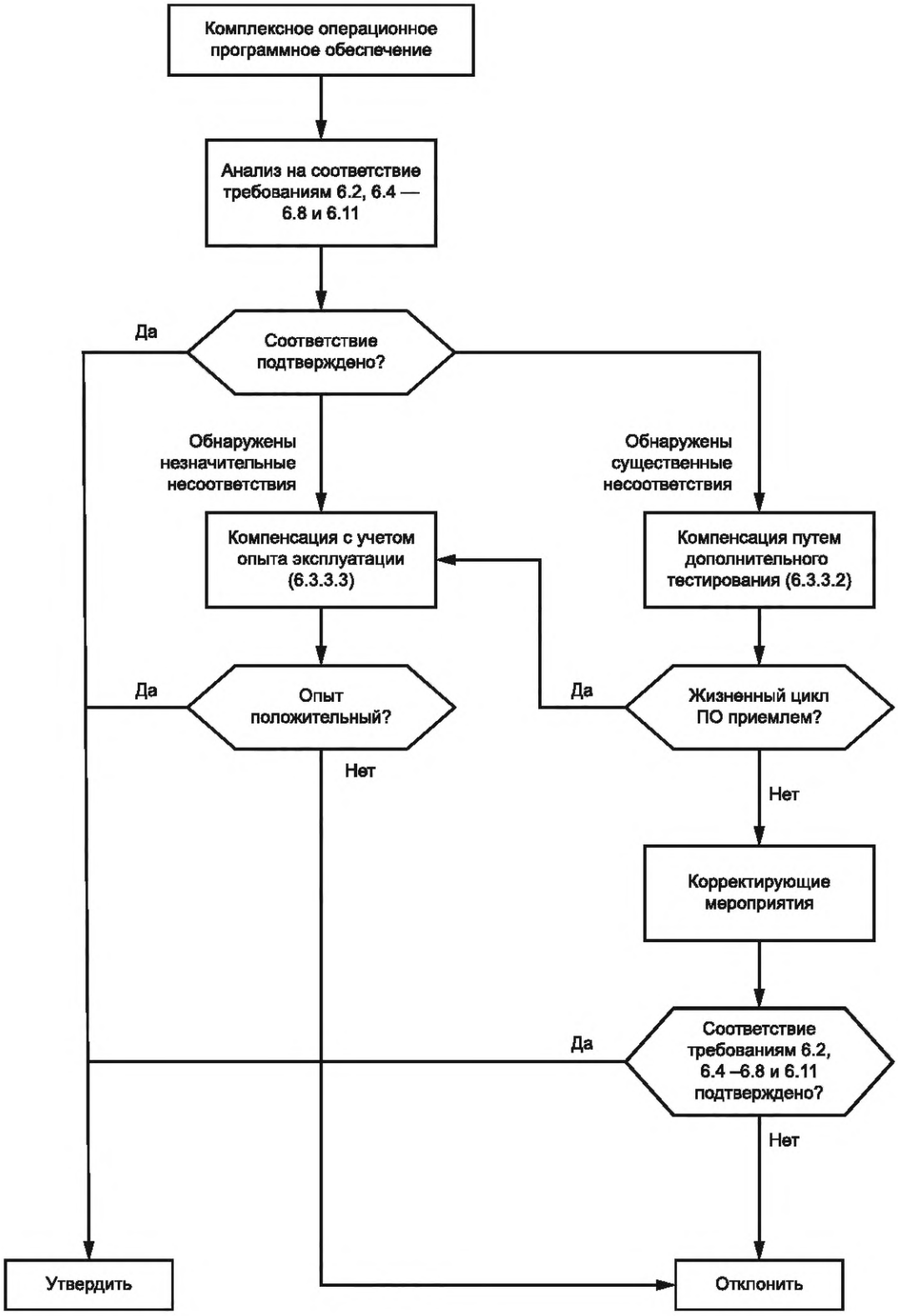


Рисунок 5 — Общая схема типичного процесса квалификации ранее разработанного комплексного операционного ПО системы

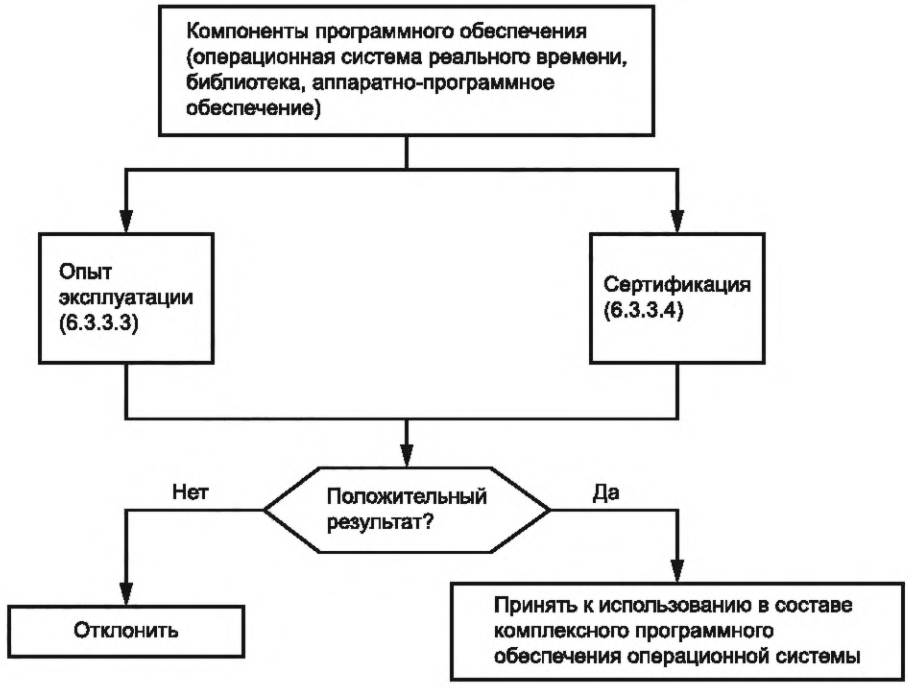


Рисунок 6 — Общая схема типичного процесса квалификации компонентов ранее разработанного ПО

6.3.3.1.4 Элемент ранее разработанного ПО считают компонентом ПО только в том случае, если он интегрирован в более крупное ПО для формирования комплексного операционного ПО системы. Кроме того, элемент ранее разработанного ПО считают компонентом ПО только если более позднее изменение конфигурации операционного ПО не может привести к реализации компонента способом, отличным от его изначального использования (поскольку это значит, что при квалификации комплексного операционного элемента компонент ПО не будет квалифицирован должным образом).

6.3.3.1.5 Верификацию и валидацию комплексного операционного ПО системы следует проводить со встроенными компонентами ПО. Интеграция может быть выполнена в рамках ПО, функционирующего на одном процессоре. Примерами такого ПО являются операционные системы, работающие в режиме реального времени, или библиотеки. Интеграция может быть также выполнена в рамках ПО, работающего в тесной взаимосвязи на нескольких процессорах, например, аппаратно-программного обеспечения коммуникационных модулей или модулей ввода-вывода.

6.3.3.1.6 *Для класса 2: процесс квалификации компонентов ПО (см. рисунок 6) применяют только к компонентам ранее разработанного ПО, которые являются неавтономно исполняемыми программами.*

Автономно исполняемая программа — это ПО, которое может функционировать самостоятельно без каких-либо дополнительных кодов.

К типичным автономно исполняемым программам относятся операционные системы общего назначения, разработанные преимущественно для использования на рабочих местах, поскольку после их установки они автоматически выполняют множество задач, не назначенных пользователем.

Для функционирования библиотек (например, библиотеки С) необходимо вводить дополнительный код. Библиотека может быть скомпилирована и загружена на процессор, но она не будет функционировать без кода, написанного для вызова ее функций. Таким образом, библиотека не относится к автономно исполняемым программам.

Операционные системы, работающие в режиме реального времени, изначально созданные для работы встроенного ПО, обычно не относятся к автономно исполняемым программам, поскольку пользователь должен четко определять каждую задачу, но это необходимо проверять в каждом конкретном случае.

6.3.3.1.7 Корректность компонента ПО с учетом его документации по безопасности должна быть основана на соответствующем положительном опыте работы достаточного объема (см. 6.3.3.3) или на сертификации (см. 6.3.3.4) (см. рисунок 6).

6.3.3.1.8 Для подтверждения корректности комплексного операционного ПО системы с учетом его документации по безопасности вначале необходимо провести оценку соответствия требованиям общих подразделов настоящего стандарта (6.2, 6.4—6.8 и 6.11).

6.3.3.1.9 Комплексное операционное ПО системы принимают, если проведенная оценка установила соответствие ПО требованиям общих подразделов настоящего стандарта (6.2, 6.4—6.8 и 6.11).

Если в ходе оценки выявлено, что комплексное операционное ПО системы имеет незначительные несоответствия требованиям указанных подразделов, то данные несоответствия могут быть компенсированы исходя из соответствующего положительного опыта эксплуатации достаточного объема (см. 6.3.3.3). При отсутствии положительного опыта эксплуатации или недоступности его данных могут быть проведены испытания.

К незначительным несоответствиям относятся случаи, когда полный жизненный цикл ПО системы безопасности прослежен и документально оформлен как для комплексного операционного ПО, но на различных этапах исполнения были соблюдены не все требования общих подразделов (6.2, 6.4—6.8 и 6.11).

Для этапа спецификации требований к ПО (6.4) примером незначительного несоответствия может быть случай, когда определены и документально оформлены требования к программному обеспечению СКУ, но они содержат не весь объем информации, который требуется согласно 6.4.4.

6.3.3.1.10 Если в ходе оценки установлено, что комплексное операционное ПО системы имеет несоответствия требованиям, указанным в 6.2.4, то такие несоответствия компенсируют исходя из соответствующего положительного опыта эксплуатации достаточного объема (см. 6.3.3.3).

6.3.3.1.11 Если в ходе оценки выявлены существенные несоответствия комплексного операционного ПО системы требованиям, указанным в 6.2.2, 6.7 или 6.8, то такие несоответствия компенсируют дополнительными испытаниями (6.3.3.2).

6.3.3.1.12 Если в ходе оценки установлено, что комплексное операционное ПО системы имеет существенные несоответствия требованиям, указанным в 6.2.1, 6.2.3, 6.2.5, 6.4, 6.5, 6.6 или 6.11, то такой жизненный цикл ПО признают неприемлемым. В таких ситуациях для принятия ПО должны быть проведены успешные корректирующие мероприятия. Целью корректирующих мероприятий должно быть достижение соответствия требованиям общих подразделов настоящего стандарта (6.2, 6.4—6.6 и 6.11). Если проведение корректирующих мероприятий невозможно, то необходимо отказаться от использования комплексного операционного ПО системы.

Существенные несоответствия, изложенные в 6.3.3.1.11 и 6.3.3.1.12, относятся к случаям, когда полный жизненный цикл ПО безопасности не прослежен и документально не оформлен. Случай, когда жизненный цикл прослежен, но документально не оформлен, должен быть интерпретирован как существенное несоответствие.

Также примером существенного несоответствия является случай, когда документально не оформлена валидация.

6.3.3.1.13 Стратегия подтверждения корректности ранее разработанного ПО с учетом его документации по безопасности должна быть определена и одобрена всеми участниками процесса разработки СКУ на раннем этапе.

Данная стратегия не может быть полностью определена до завершения процесса оценки соответствия комплексного операционного ПО системы требованиям общих подразделов настоящего стандарта (6.2, 6.4—6.8 и 6.11), поскольку она зависит от недостатков, которые должны быть выявлены в ходе такой оценки.

6.3.3.2 Дополнительные испытания

6.3.3.2.1 Общие положения

Дополнительные испытания могут быть использованы для обоснования подтверждения корректности ранее разработанного ПО при следующих условиях:

6.3.3.2.2 Дополнительные испытания ранее разработанного ПО, выполненные при создании СКУ, должны быть оформлены документально.

6.3.3.2.3 Дополнительные испытания должны подтвердить, что в условиях применения в СКУ ранее разработанное ПО и его работа соответствуют документации по безопасности.

Условия применения могут касаться таких аспектов, как конфигурация ранее разработанного ПО (особенно установка параметров и данных конфигурации), использование функций и интерфейсов, аппаратных средств, процессора и загрузки по требованию.

6.3.3.2.4 *Для класса 2: правила разработки плана дополнительных испытаний должны быть документально оформлены и обоснованы.*

6.3.3.2.5 Документация по дополнительным испытаниям должна содержать следующую информацию:

- версия и конфигурация ранее разработанного ПО;
- описание проведенных испытаний и использованных технических средств, чтобы можно было повторить эти испытания в идентичных условиях;
- принятые при проведении испытаний допущения и доказательство их правомочности;
- полученные результаты и подтверждение их правильности;
- сделанные выводы и согласованные решения.

6.3.3.3 Опыт эксплуатации

6.3.3.3.1 Общие положения

Может быть принят во внимание опыт эксплуатации систем более низкого класса безопасности или систем, не классифицированных по безопасности. Опыт эксплуатации может быть использован как аргумент для подтверждения корректности ранее разработанного ПО при следующих условиях:

6.3.3.3.2 Принятый во внимание опыт эксплуатации должен быть отражен в документации.

6.3.3.3.3 *Для класса 2: учитываемый опыт эксплуатации должен соответствовать точно идентифицированным версиям ранее разработанного ПО и оборудованию, в котором его используют, если данное ПО специально предназначено для этого оборудования.*

6.3.3.3.4 *Для класса 2: если весь опыт эксплуатации или его часть соответствуют другим версиям ранее разработанного ПО и/или оборудованию, то отличия от версий, используемых в СКУ, должны быть оценены и целесообразность применения данного эксплуатационного опыта должна быть обоснована.*

6.3.3.3.5 *Для класса 2: необходимо предоставить документально оформленное обоснование того, что принятый во внимание опыт эксплуатации соответствует условиям использования данной системы контроля и управления (предполагаемая конфигурация ПО является одним из условий использования). В случаях аналогичных условий эксплуатации может быть учтен опыт эксплуатации систем, относящихся к более низкому классу безопасности, либо систем, не классифицированных по безопасности.*

6.3.3.3.6 *Для класса 2: методы, используемые для накопления принимаемого во внимание опыта эксплуатации, должны быть оформлены документально. В частности, необходимо предоставить документальное подтверждение того, что отказы (если таковые имели место), которые были вызваны ранее разработанным ПО в течение учтенного опыта эксплуатации, были правильно обнаружены и зафиксированы.*

6.3.3.3.7 *Для класса 2: должно быть предоставлено доказательство того, что эти отказы были надлежащим образом проанализированы и соответствующие дефекты ПО исправлены.*

6.3.3.4 Сертификация

6.3.3.4.1 Общие положения

Ранее разработанное ПО, уже действующее в системах, важных для безопасности (хотя не обязательно в СКУ АС), может быть аттестовано на соответствие некоторым документам по безопасности. Свидетельство, обеспеченное такой сертификацией, может быть использовано в качестве убедительного аргумента для подтверждения корректности ранее разработанного ПО при следующих условиях:

6.3.3.4.2 Документ по безопасности, использованный при сертификации ранее разработанного ПО, должен содержать подробное описание процесса разработки ПО.

6.3.3.4.3 Принятая во внимание сертификация должна иметь документальное подтверждение.

6.3.3.4.4 Должна быть задокументирована точная идентификация ранее разработанного сертифицированного ПО. Если данное ранее разработанное ПО сертифицировано как часть более крупного продукта (например, как часть оборудования или комплекса оборудования), должна быть также задокументирована точная идентификация этого продукта.

6.3.3.4.5 *Для класса 2: доказательства, подтверждающие сертификацию, подлежат оценке, в частности:*

- условия сертификации (например, условия использования и принятые допущения);

- методы и инструменты, используемые при сертификации;

- полученные результаты (например, сертифицированные свойства и/или параметры).

6.3.3.4.6 *Для класса 2: значимость условий и результатов сертификации для подтверждения корректности должна быть обоснована.*

6.3.3.4.7 *Для класса 2: эффективность методов и инструментов, которые были использованы для сертификации, должна быть подтверждена.*

6.3.3.4.8 *Для класса 2: орган, проводивший сертификацию, должен быть идентифицирован, и он должен быть компетентным в отношении аттестуемых свойств и/или параметров.*

6.3.3.4.9 *Для класса 2: версия сертифицированного ранее разработанного ПО должна быть аналогичной версии, используемой в СКУ.*

6.3.3.5 Модификация

6.3.3.5.1 Общие положения

Если в ранее разработанное ПО, для которого уже существует подтверждение корректности, внесены точно идентифицированные ограниченные модификации, то для обновления или завершения подтверждения корректности следует вместо требований, приведенных в 6.3.3.1–6.3.3.4, использовать следующие ниже требования. Изменение в конфигурации ранее разработанного ПО не является модификацией при условии, что его новая конфигурация остается в пределах, охватываемых подтверждением корректности.

6.3.3.5.2 Модификация ранее разработанного ПО должна быть оформлена документально.

6.3.3.5.3 *Для класса 2: документация по модификации должна содержать:*

- четкую идентификацию измененного ПО;

- специфику модификации, если ПО является частью более крупного продукта (например, оборудования или комплекса оборудования);

- цели, спецификацию и ограничения модификации;

- изменения, внесенные в документацию по безопасности.

6.3.3.5.4 *Для класса 3: документация по модификации должна содержать:*

- четкую идентификацию измененного ПО;

- специфику модификации, если ПО является частью более крупного продукта (например, оборудования или комплекса оборудования);

- цели, спецификацию и ограничения модификации;

- изменения, внесенные в документацию по безопасности.

В 6.3.3.5.3 и 6.3.3.5.4 под спецификой модификации подразумевают:

- четкую идентификацию измененного более крупного продукта;

- цели, спецификацию и ограничения модификации продукта;

- модификации, которые должны быть внесены в остальную часть продукта или которые могут иметь воздействие на ранее разработанное ПО;

- верификацию и валидацию, выполняемые на уровне продукта.

6.3.3.5.5 *Для класса 2: документация по модификации должна также содержать изменения, внесенные в проект ранее разработанного ПО.*

6.3.3.5.6 Документально оформленное доказательство (основанное, например, на экспертизе руководства пользователя, анализе инструментальных средств и/или испытаниях), касающееся модифицированного ПО, а возможно, и более крупного продукта, должно подтверждать:

- что цели модификации достигнуты;
- дефекты не были внесены;
- модифицированное ПО соответствует его обновленной документации по безопасности.

6.3.3.5.7 *Для класса 2: достаточность доказательства должна быть подтверждена, по возможности, с учетом сделанных модификаций и условий использования в рамках СКУ.*

6.3.3.5.8 Документация по безопасности должна быть актуализирована для соблюдения ее точности в части модификаций ПО, которые могут влиять на действия пользователя по установке, эксплуатации и технической поддержке системы, частью которой является ПО.

6.3.4 Функциональное соответствие

6.3.4.1 Общие положения

Положения настоящего пункта стандарта имеют цель гарантировать, что ранее разработанное ПО отвечает потребностям СКУ и не является слишком сложным для удовлетворения этих потребностей.

6.3.4.2 Документация по безопасности ранее разработанного ПО должна быть сопоставлена со спецификацией системы и проектом системы (если это возможно). Несоответствия должны быть устранены.

6.3.4.3 *Для класса 2: следует идентифицировать те функции ранее разработанного ПО, которые не являются необходимыми для поддержки требований к системе. Должно быть предоставлено подтверждение того, что эти функции не оказывают негативного воздействия на безопасность.*

6.3.5 Выбор и использование цифровых устройств ограниченной функциональности

Для цифровых устройств ограниченной функциональности в качестве альтернативы настоящему стандарту может быть использован МЭК 62671, в котором указаны четкие критерии применимости его положений к конкретным устройствам.

6.4 Спецификация требований к программному обеспечению

6.4.1 Общие положения

Настоящий подраздел 6.4 дополняет и уточняет требования, изложенные в подпункте 6.2.3.4 МЭК 61513:2011.

6.4.2 Цели

6.4.2.1 Требования к программному обеспечению СКУ должны быть определены и оформлены документально.

В настоящем стандарте соответствующий документ или пакет документов назван спецификацией требований к ПО. В основном назначение спецификации состоит в определении задач ПО без указания путей их решения. Однако, если это потребуется для проекта СКУ или архитектуры СКУ, возможно установление ограничений проекта и его реализации.

6.4.2.2 *Для класса 2: в спецификации требований к ПО следует избегать излишней сложности проекта ПО.*

6.4.2.3 Спецификация требований к ПО должна быть такой, чтобы:

- способствовать уверенности в корректности проекта СКУ;
- могло быть показано соответствие СКУ требованиям МЭК 61513:2011.

Требования МЭК 61513:2011, относящиеся к спецификации требований к ПО, содержатся главным образом в 6.2.2.3, 6.2.2.4, 6.2.2.5, 6.2.3.3, 6.2.3.5 и 6.2.4.

6.4.2.4 Спецификация требований к ПО должна быть основой проекта ПО, валидации ПО и возможных модификаций ПО.

6.4.3 Исходная информация

6.4.3.1 *Для класса 2: исходная информация для спецификации требований к ПО должна включать спецификацию системы и документацию проекта системы.*

Исходная информация может также включать в себя другие документы, например:

- специфичные ограничения проекта;
- применяемые правила и стандарты;
- такие требования, как независимость функций;
- такие требования к целостности, как самоконтроль, чтобы привести выходы в безопасное состояние в случае обнаружения отказов.

6.4.3.2 *Для класса 2: структура спецификации требований к ПО должна облегчать проведение верификации, обеспечивая ее согласованность и полноту по отношению к исходным документам.*

Спецификация требований к ПО может содержать непосредственные ссылки на исходные документы, чтобы избежать ненужных дублирований и минимизировать риск несогласованности. Она может также ссылаться на другие ранее существовавшие документы, например, на документацию ранее разработанного ПО.

6.4.3.3 Спецификация требований к ПО должна обеспечивать сопоставимость с его исходной документацией.

6.4.3.4 В ходе верификации спецификации требований к ПО (см. 6.2.2) необходимо в первую очередь проверять ее согласованность и полноту по отношению к исходным документам.

6.4.3.5 Ссылки в спецификации требований к ПО на другие документы (при их наличии) должны быть точными и однозначными.

6.4.3.6 *Для класса 2: в спецификации требований к ПО следует избегать необязательной функциональности.*

В принципе, желательно, чтобы возможности программного обеспечения не превышали необходимого, чтобы свести к минимуму его сложность. Тем не менее, поскольку существующая промышленная практика опирается на использование ранее разработанных компонентов ПО, может быть обосновано также включение невостребованных возможностей.

6.4.4 Содержание

6.4.4.1 Спецификация требований к ПО должна определять:

- прикладные функции, выполняемые ПО;

- различные режимы работы ПО и соответствующие условия переходов;
- интерфейсы и взаимодействие ПО с его средой (например, с операторами, остальной частью СКУ, другими системами и оборудованием, с которыми оно взаимодействует или разделяет ресурсы), включая ролевые имена, типы, форматы, диапазоны и ограничения входных и выходных данных;
- параметры ПО, которые могут быть изменены операторами во время эксплуатации (если таковые есть), их ролевые имена, типы, форматы, диапазоны и ограничения, а также проверки, осуществляемые ПО в случае изменения этих параметров;
- требуемые рабочие характеристики (при необходимости);
- то, чего ПО не должно делать или чего должно избегать (при необходимости);
- требования или допущения, устанавливаемые ПО к его среде (при необходимости).

6.4.4.2 В спецификации требований к ПО следует также устанавливать параметры (например, загрузка по запросу), подверженные воздействию среды ПО, особенно для наихудших условий.

Что касается положений 6.4.4.1 и 6.4.4.2, требования к функциям, интерфейсам и характеристикам могут зависеть от режима работы, значений параметров, данных конфигурации и условий, создаваемых для ПО.

6.4.4.3 В спецификации требований к ПО следует определять режимы его работы при обнаружении ошибок или отказов. При необходимости периодических тестирований СКУ спецификация требований к ПО должна также определять требования к режиму выполнения таких тестирований.

6.4.4.4 Спецификация требований к ПО должна содержать ограничения, которые необходимо учитывать при проектировании и внедрении ПО, чтобы обеспечить его корректность и работоспособность.

В число ограничений могут входить следующие:

- обеспечивающие уверенность в корректности ПО и проекта системы (например, устанавливающие пределы при использовании динамически размещаемых ресурсов, таких как память, вычислительная мощность, пропускная способность канала связи, ресурсы операционной системы);
- увеличивающие устойчивость ПО и СКУ к дефектам, способность обнаруживать ошибки и отказы и оповещать о них, способность работать в заданных режимах и восстанавливаться после отказов;
- обеспечивающие уверенность в том, что ошибки операторов и отказы других систем или оборудования, с которыми ПО взаимодействует или использует общие ресурсы, не будут приводить к неприемлемым результатам.

6.4.4.5 Спецификация требований к ПО должна содержать прогнозы, которые необходимо учитывать при проектировании и внедрении ПО, чтобы обеспечить его корректность и работоспособность.

6.4.4.6 Спецификация требований к ПО должна определять возможности ПО обеспечивать своевременное информирование операторов об ошибках или отказах, касающихся функций СКУ, важных для безопасности. Информация, предоставляемая операторам, должна позволять им предпринимать любое необходимое действие.

6.4.4.7 Спецификация требований к ПО должна определять функции и требования, относящиеся к категориям безопасности С или В.

6.4.5 Свойства

6.4.5.1 *Для класса 2: обозначения, правила и документы, используемые при разработке спецификации требований к ПО, должны способствовать ее ясности и точности. Эти атрибуты следует выбирать с учетом их использования во входных данных, а также при проектировании и реализации ПО.*

Так как конкретный формат спецификации не всегда обеспечивает ясное, точное и проверяемое выражение всего того, что требует определения, в одной и той же спецификации требований к ПО могут быть использованы различные дополняющие друг друга форматы. Например, для определения прикладных функций могут быть использованы форматы, отличные от тех, которые используются для других функций.

6.4.5.2 *Для класса 2: требования спецификации должны быть выражены таким способом, чтобы их соблюдение могло быть объективно оценено.*

6.5 Проект программного обеспечения

6.5.1 Цели

6.5.1.1 Проект ПО должен быть оформлен документально.

Соответствующий документ или пакет документов именуют спецификацией проекта ПО. Если используют ранее разработанное ПО, то в спецификации проекта ПО может быть дана ссылка на соответствующие документы.

6.5.1.2 Спецификация проекта ПО должна содержать общие сведения об организации и функционировании ПО (см. также 6.5.3.3).

6.5.1.3 *Для класса 2: спецификация проекта ПО должна способствовать достижению уверенности в качестве проекта ПО и его корректности с точки зрения спецификации требований к ПО.*

6.5.1.4 Спецификация проекта ПО должна свидетельствовать о том, что положения спецификации требований к ПО, важному для безопасности, учтены для всех условий.

6.5.1.5 *Для класса 2: в спецификации проекта ПО следует в документальной форме представлять характеристики ПО, гарантирующие раннее обнаружение любой ошибки или отказа и их нераспространение за установленные пределы. В спецификации следует также указывать действия, которые необходимо предпринимать при обнаружении ошибки или отказа.*

6.5.1.6 Спецификация проекта ПО должна обеспечивать (в случае необходимости) устранение неблагоприятных побочных эффектов, связанных с ошибками и отказами ПО, до его возвращения к нормальному режиму работы.

6.5.1.7 Спецификация проекта ПО должна быть направлена на реализацию модульного принципа построения, возможности проводить тестирование и техническое обслуживание.

6.5.1.8 *Для класса 2: если это не приводит к чрезмерной сложности, проект программного обеспечения СКУ должен содействовать:*

- анализу и тестированию ПО и его компонентов;

- локализации дефектов;

- идентификации результатов модификации.

6.5.1.9 Спецификация проекта ПО должна служить основой для реализации и интеграции ПО, а также для возможных его модификаций.

6.5.2 Исходные данные

6.5.2.1 Исходные данные для проектирования ПО должны включать в себя спецификацию требований к ПО и документацию по безопасности на ранее разработанное ПО.

Исходные данные могут также включать другие документы, такие как специфичные проектные ограничения и/или применяемые правила и стандарты.

6.5.3 Содержание

6.5.3.1 Спецификация проекта ПО должна включать в себя спецификацию:

- полной организации ПО;

- полного функционирования ПО при условиях и режимах работы в соответствии со спецификацией требований к ПО.

6.5.3.2 Полная организация ПО предусматривает информацию, касающуюся:

- четкой идентификации и конфигурации ранее разработанного ПО;

- распределения ресурсов, компонентов ПО и задач по подсистемам;

- распределения (под)функций ПО по определенным для него задачам;

- основных внутренних взаимосвязей, в частности, интерактивных связей между задачами ПО.

6.5.3.3 Спецификация полного функционирования предусматривает информацию, касающуюся:

- взаимодействий, протоколов связи и информационных потоков;

- последовательности выполнения функций и временных ограничений;

- использования ресурсов;

- синхронизации, особенно при использовании общих ресурсов.

6.5.3.4 *Для класса 2: спецификация проекта ПО должна содержать информацию о порядке соблюдения важных для безопасности требований к ПО при всех заданных условиях. При использовании ранее разработанного ПО информация, касающаяся свойств ПО, важных для безопасности, должна быть основана в том числе на прогнозной информации, содержащейся в соответствующей документации по безопасности (см. 6.3.2.2.5).*

6.5.3.5 *Для класса 2: в спецификации проекта ПО и в документации по проектированию системы должны быть установлены и обоснованы меры, предпринимаемые для уменьшения влияния известных и ожидаемых отказов любого ранее разработанного ПО, для которого применены дополнительные меры по подтверждению корректности (см. 6.3.3).*

6.5.3.6 *Для класса 2: в спецификации проекта ПО должны быть установлены правила реализации ПО.*

6.5.3.7 *Для класса 2: в спецификации проекта ПО также должны быть установлены правила конфигурирования и использования ранее разработанного ПО с тем, чтобы обеспечить его использование контролируемым образом, согласующимся с соответствующей документацией по безопасности.*

6.5.3.8 *Для класса 2, спецификация проекта ПО должна включать в себя детальный проект любого нового ПО, реализованного на универсальном языке.*

6.5.3.9 В рамках спецификации проекта ПО спецификация компонентов любого ПО, реализованного на универсальном языке, должна указывать:

- функции, обеспечиваемые компонентом, с их взаимосвязями, ролями, типами, форматами, диапазонами и ограничениями входных и выходных данных, сигналами об аномальном состоянии, данными о конфигурации;
- рабочие характеристики, например, время отклика, точность (если это необходимо);
- требования компонентов к их окружению, например, потребность в динамически распределяемой памяти, ресурсах операционной системы и т. п. (если это необходимо);
- любая другая информация, о которой пользователи компонента должны быть осведомлены;
- любые важные ограничения по реализации.

6.5.3.10 *Для класса 2: спецификация проекта ПО должна содержать информацию, позволяющую сделать корректные прогнозы по поводу ключевых параметров системы, важных для безопасности, включая, в частности, максимальное время отклика и максимальное использование ресурсов.*

Такая информация может быть представлена в форме данных, формул и/или моделей, которые можно использовать для расчета наихудшего времени отклика и использования ресурсов приложений.

6.5.4 Свойства

6.5.4.1 *Для класса 2: спецификация проекта ПО должна представлять проект ПО ясно и точно.*

6.5.4.2 *Для класса 3: спецификация проекта ПО должна представлять проект ПО ясно и точно.*

Что касается 6.5.4.1 и 6.5.4.2, основной подход может базироваться на принципе «сверху-вниз», но некоторые документы могут также содержать информацию, которая обращает внимание на особо важные аспекты (например, устойчивость к отказам), учитываемые для всех ПО или СКУ.

6.5.4.3 *Для класса 2: формат и синтаксис, используемые в спецификации проекта, должны способствовать ясности и точности спецификации проекта ПО.*

6.6 Реализация нового программного обеспечения

6.6.1 Общие требования

6.6.1.1 Общие положения

Требования настоящего подраздела применимы ко всем видам ПО, например, к конфигурации ранее разработанного ПО, к программам, написанным на проблемно-ориентированных или универсальных языках.

6.6.1.2 Использование ранее разработанного ПО должно быть верифицировано с точки зрения соответствия документации по безопасности и ограничениям, установленным спецификацией проекта ПО.

6.6.1.3 Процедуры, используемые для трансляции новых программ в рабочую программу, должны быть оформлены документально и верифицированы.

Обычно процедуры описывают, каким образом должен быть задействован набор компилирующих инструментальных программ или генератор кода для трансляции новых программ в рабочую программу. Этот процесс часто бывает автоматизированным.

6.6.1.4 *Для класса 2: обновление рабочей программы после внесения изменений в программы должно быть выполнено автоматизированными средствами.*

6.6.2 Конфигурация программного обеспечения и устройств, содержащих программное обеспечение

6.6.2.1 Общие положения

Требование настоящего пункта является специфичным для конфигурации настраиваемого ПО. Такое ПО может быть ранее разработанным или новым. Однако в случаях, когда данные конфигурации представляют собой последовательность обработки данных, выполняемую ПО или системой (т. е. когда они фактически являются программой), то применяют требования по 6.6.3.

6.6.2.2 Конфигурация настраиваемого ПО и устройств с встроенным настраиваемым ПО должна быть оформлена документально.

6.6.3 Реализация с помощью проблемно-ориентированных языков

6.6.3.1 Общие положения

Требования настоящего пункта являются специфичными для программ, написанных на проблемно-ориентированных языках. В общем случае проблемно-ориентированные форматы (такие как логические диаграммы или диаграммы функциональных блоков) могут быть использованы для отражения всей спецификации требований к ПО или спецификации проекта ПО или их части. Затем необходимы только до некоторой степени детализированный проект и небольшие усилия по реализации для преобразования спецификации в программы, которые могут быть автоматически транслированы в рабочую программу.

6.6.3.2 Части спецификации требований к ПО и/или спецификации проекта ПО, использованные для получения рабочей программы автоматизированными способами, следует рассматривать как программы, написанные на проблемно-ориентированных языках.

6.6.3.3 *Для класса 2: программы, написанные на проблемно-ориентированных языках, должны быть верифицированы на предмет функциональной корректности и согласованности. Верификация должна подтверждать, что:*

- все элементы проекта поняты правильно (т. е. при всех указанных условиях неожиданного поведения ПО не будет);

- заданное поведение согласуется с целями, установленными спецификацией требований к ПО.

Для улучшения понимания спецификаций и верификации их функциональной корректности и согласованности могут быть применены анимация, тестирования, осмотры, сквозной контроль, официальный анализ и освидетельствование.

6.6.3.4 *Для класса 3: программы, написанные на проблемно-ориентированных языках и относящиеся к функциям, важным для безопасности, должны быть верифицированы на предмет функциональной корректности и согласованности.*

6.6.3.5 Должны быть разработаны методы тестирования, касающиеся не только внутренней структуры, но и функциональных требований к объекту испытаний.

6.6.3.6 *Для класса 2: диапазон функций, подлежащих тестированию, должен быть определен предварительно, чтобы в случае успешного проведения испытаний подтвердить, что объект соответствует всем требованиям к его поведению.*

6.6.3.7 *Для класса 2: в ходе тестирования необходимо контролировать тестируемое структурное покрытие на предмет его соответствия критериям, отвечающим за подтверждение (например таким, как спецификация, состояние, ветвление, поток данных), чтобы убедиться в отсутствии нежелательного поведения. Если структурное покрытие не соответствует указанным критериям, то должно быть приведено объяснение.*

6.6.3.8 *Для класса 2: написанные на проблемно-ориентированных языках программы должны соответствовать документально оформленным правилам, разработанным для придания им ясности, модифицируемости и проверяемости. Несоответствия должны быть обоснованы.*

Набор правил может быть специфичным для языка или пакета программ. Простота, ясность, стандартные расположение и представление, модульный принцип построения, наличие значимых комментариев, избежание небезопасных особенностей языка и его инструментов — это примеры свойств, которые в общем случае облегчают понимание, верификацию, тестирование и последующую модификацию программ.

6.6.4 Реализация с помощью универсальных языков

6.6.4.1 Общие положения

Требования настоящего пункта относятся к программам, написанным на универсальных языках.

6.6.4.2 *Для класса 2: документально оформленная верификация должна свидетельствовать, что программы, написанные на универсальных языках, соответствуют их спецификации, как это определено спецификацией проекта ПО.*

Верификация может заключаться в комбинации просмотра описаний, анализа с помощью инструментальных программ и/или тестирований.

Проверки кодов, изучение пошагового руководства, контрольных таблиц и другие подобные методы часто являются высокоэффективными способами просмотра описаний для выявления дефектов ПО.

Анализы с помощью инструментальных программ могут быть использованы для формального доказательства того, что программа имеет (или не имеет) заданные свойства. Например, анализы могут обеспечить уверенность в том, что при данных условиях (например, при условии

нахождения исходных данных в пределах заданных диапазонов) программа или определенные части программы не содержат дефектов определенного вида (например, неинициализированных переменных, арифметического переполнения или исчезновения значащих разрядов).

Тестирования могут быть выполнены на главном компьютере или на средствах поддержки программных разработок.

6.6.4.3 *Для класса 2: в документации по верификации должны быть зафиксированы:*

- идентичность и версия проверяемых программ;

- вся информация, необходимая для повторения верификации в аналогичных условиях;

- принятые допущения и обоснование их справедливости;

- полученные результаты и подтверждение их корректности;

- выводы и, в случае выявления ошибок, согласованные решения;

- обоснование соответствия критериям приемлемости.

6.6.4.4 Тестирование должно быть разработано с учетом не только требований к внутренней структуре испытуемого объекта, но и функциональных требований к нему.

6.6.4.5 *Для класса 2: диапазон функций должен быть определен до проведения тестирования, чтобы успешное его проведение могло служить подтверждением, что объект соответствует всем необходимым требованиям к его поведению.*

6.6.4.6 *Для класса 2: в ходе тестирования необходимо контролировать тестируемое структурное покрытие на предмет его соответствия критериям, отвечающим за подтверждение (например, таким, как спецификация, состояние, ветвление, поток данных), чтобы убедиться в отсутствии нежелательного поведения. Если структурное покрытие не соответствует указанным критериям, то должно быть приведено объяснение.*

6.6.4.7 Написанные на универсальных языках программы должны соответствовать документально оформленным правилам программирования, обеспечивающим ясность, возможность модификации и тестирования.

Набор правил может быть специфичным для языка или пакета программ. Простота, структурное программирование, модульный принцип построения, инкапсуляция, скрытие данных (чтобы пользователи программного продукта имели дело лишь с предоставляемым им сервисом, но не с внутренней работой продукта), наличие значимых комментариев, избежание опасных особенностей языка и его инструментов являются примерами свойств, которые могут облегчить понимание, верификацию, тестирование и модификацию ПО.

6.6.4.8 *Для класса 2: правила программирования должны быть такими, чтобы их можно было верифицировать, а их целью являлось раннее обнаружение ошибок и защита от ошибок.*

6.6.4.9 *Для класса 2: если для анализа сложности кода могут быть использованы инструменты статического анализа, то правила должны устанавливать допустимые метрические пределы.*

6.6.4.10 *Для класса 2: программы, написанные на универсальных языках, должны быть верифицированы на соответствие применяемым правилам и стандартам. Несоответствия должны быть обоснованы и, при необходимости, должны быть предприняты, документально оформлены и обоснованы соответствующие контрмеры.*

Контрмеры в случае несоответствия могут представлять собой, например, более полную верификацию.

6.7 Программные аспекты интеграции системы

6.7.1 Общие положения

Интеграцию ПО рассматривают как часть интеграции системы. Настоящий подраздел дополняет пункты 6.2.5, 6.3.4 и 6.4.5 МЭК 61513:2011, устанавливая дополнительные требования, специфичные или особенно важные для ПО.

6.7.2 Интеграция ПО и/или ее проверки должны показать, что интегрированные система и ПО:

- соответствуют положениям проекта, которые обеспечивают выполнение тех пунктов спецификации требований к ПО, которые определены как важные для безопасности;

- следуют ограничениям, установленным спецификацией требований к ПО в отношении корректности и работоспособности.

6.7.3 *Для класса 2: если достаточность валидационного тестирования ПО вызывает сомнения, то необходимо получить доказательство корректности его работы либо путем дополнительного тестирования интеграции ПО, либо проведением более полной верификации.*

6.7.4 Интеграция ПО должна быть выполнена согласно положениям плана интеграции системы или плана интеграции ПО.

6.7.5 Должны быть составлены отчеты по результатам применения плана, использованного для интеграции ПО, например, по результатам тестирований. В случае необходимости модификации ПО или системы может потребоваться повторение всего объема или части интеграционных тестирований для оценки масштабов вероятных изменений в работе программы.

6.7.6 *Для класса 2: необходимо обеспечить единство измерений между спецификацией проекта ПО и соответствующими интеграционными тестированиями.*

6.7.7 *Для класса 3: должно быть обеспечено единство измерений между спецификацией проекта ПО и соответствующими интеграционными тестированиями.*

6.8 Программные аспекты валидации системы

6.8.1 Общие положения

Тестирование функциональных возможностей ПО проводят при валидации системы. Настоящий подраздел дополняет пункты 6.2.6, 6.3.5 и 6.4.6 МЭК 61513:2011, устанавливая дополнительные требования, специфичные или особенно важные для ПО. В случае выявления несоответствий валидация может быть продолжена с указанием обоснований или остановлена для устранения несоответствий перед проведением повторной валидации.

6.8.2 *Для класса 2: валидация ПО должна показать, что интегрированное в завершенную СКУ программное обеспечение соответствует каждому положению спецификации требований к ПО, касающемуся функций, характеристик и интерфейса, и, согласно проекту, способствует соответствию требованиям, содержащимся в спецификации требований к системе. Валидация также должна подтверждать, что:*

- установленные функции ПО выполняются правильно, если их параметры и исходные данные находятся в диапазонах, указанных в спецификации требований к ПО, а условия выполнения соответствуют указанным в этой спецификации;

- функции системы, реализация которых осуществляется с помощью ПО, выполняются правильно в условиях, установленных в спецификации требований к системе;

- ПО обеспечивает защиту от ошибок операторов и отказов других систем и оборудования в соответствии со спецификацией требований к ПО;

- в различных режимах ПО выполняет функции так, как запланировано;

- эксплуатационно-технические данные АС, используемые СКУ или интегрированные в нее, являются правильными. В частности, валидация ПО должна показать, что эти данные определяют интерактивную связь между системами и оборудованием АС, с которыми ПО взаимодействует или разделяет ресурсы;

- защита от ошибок операторов и отказов других систем, которую должна обеспечивать проверяемая система в соответствии со спецификацией требований к ней, осуществляется корректно с участием ПО.

При валидации тестирование обычно проводят для ПО, интегрированного в завершенную СКУ. Если представлено соответствующее обоснование, то при валидации допускается использовать инструментальный комплекс завершенной СКУ.

Условия применения функций, важных для безопасности, могут включать в себя одновременное выполнение функций, не являющихся важными для безопасности, то есть эксплуатацию при высокой коммуникационной нагрузке.

6.8.3 *Для класса 3: в ходе валидации ПО должно быть показано, что ПО, интегрированное в завершенную СКУ, соответствует требованиям к функциям, характеристикам и интерфейсам, важным для безопасности. Валидация также должна подтверждать, что:*

- соответствующие функции ПО, важные для безопасности, реализуются должным образом, если их параметры и исходные данные находятся в диапазонах, указанных в спецификации требований к ПО, а условия реализации соответствуют указанным в этой спецификации;

- важные для безопасности функции системы, выполняемые с участием ПО, реализуются должным образом в условиях, установленных в спецификации требований к системе;

- ПО обеспечивает защиту от ошибок операторов и отказов других систем и оборудования, которая установлена в спецификации требований к ПО;

- ПО функционирует должным образом в различных режимах эксплуатации;

- эксплуатационно-технические данные АС, используемые СКУ или интегрированные в нее для выполнения функций, важных для безопасности, являются правильными. В частности, валидация ПО должна показать, что эти данные определяют интерактивную связь между системами и оборудованием АС, с которыми ПО взаимодействует или разделяет ресурсы.

При валидации тестирование обычно проводят для ПО, интегрированного в завершенную СКУ. Если представлено соответствующее обоснование, то при валидации допускается использовать инструментальный комплекс завершенной СКУ.

Условия применения функций, важных для безопасности, могут включать эксплуатацию при высокой коммуникационной нагрузке.

6.8.4 Валидация ПО должна быть выполнена в соответствии с положениями плана валидации системы или ПО.

6.8.5 *Для класса 2: план валидации ПО должен устанавливать необходимые действия по валидации, а также показывать, что эти действия корректно учитывают все положения спецификации требований к ПО, касающиеся функциональности, характеристик и интерфейса. В плане также должны быть установлены основные этапы валидации ПО (например, этап, проводимый вне рабочего места, за которым следуют действия на рабочем месте) и соответствующие средства, методы и инструменты, которые следует использовать.*

6.8.6 *Для класса 2: план валидации ПО должен обеспечивать соответствие спецификации требований к ПО и валидационных мероприятий.*

6.8.7 *Для класса 3: план валидации ПО должен устанавливать необходимые действия по валидации, а также показывать, что эти действия корректно учитывают все положения спецификации требований к ПО, касающиеся функциональности, характеристик и интерфейса, определяемых как важные для безопасности. В плане также должны быть установлены основные этапы валидации ПО (например, этап, проводимый вне рабочего места, за которым следуют действия на рабочем месте) и соответствующие средства, методы и инструменты, которые следует использовать.*

6.8.8 *Для класса 3: план валидации ПО должен обеспечивать соответствие спецификации требований к ПО и валидационных мероприятий.*

6.8.9 Должны быть составлены отчеты, свидетельствующие о том, какой план валидации ПО использован. В случае необходимости модификации ПО или системы следует обеспечить возможность повторения всех или части валидационных тестирований для оценки масштабов вероятных изменений поведения.

6.8.10 *Для класса 2: необходимо, чтобы результаты валидации ПО были проверены лицами, компетентными в данной области, но не участвовавшими непосредственно в процессе валидации.*

6.8.11 *Для класса 3: необходимо, чтобы результаты валидации ПО могли быть проверены лицами, компетентными в данной области, но не участвовавшими непосредственно в процессе валидации.*

6.8.12 В отчетах должны быть документально оформлены конфигурация ПО и конфигурация среды (например, аппаратные и инструментальные средства, если использовались), подвергнутые валидации,

6.8.13 Группа, которая составляет план валидации ПО, должна включать в себя, по крайней мере, одного человека, не участвовавшего в разработке проекта и его реализации.

6.9 Установка программного обеспечения на штатном месте

6.9.1 Общие положения

Пункт 6.2.7 МЭК 61513:2011 содержит требования к установке СКУ на штатном месте. В настоящем подразделе приведены дополнительные требования, специфичные или особо важные для установки ПО.

6.9.2 Процедура установки ПО на штатном месте должна быть оформлена документально. Процедура должна обеспечивать гарантию установки правильной и полной версии ПО.

6.9.3 Процедура установки ПО на штатном месте должна включать и описывать проверки и тестирования, которые должны быть выполнены на месте до начала полномасштабной эксплуатации СКУ. В частности, должно быть верифицировано соблюдение условий, необходимых для правильной работы ПО.

Эти условия могут касаться, например, средств технического обеспечения, на которых установлено ПО, или других систем, с которыми ПО взаимодействует или разделяет ресурсы.

6.10 Протоколы отклонений от нормы

6.10.1 Если после принятия ПО в эксплуатацию наблюдается его неожиданное, очевидно неправильное, необъяснимое или ненормальное поведение, то должен быть составлен протокол отклонений от нормы.

6.10.2 Протокол отклонений от нормы должен содержать подробности работы программы, конфигурацию ПО и технических средств, а также управляющие действия, предпринимаемые во время работы ПО. В протоколе также следует указать составителя, место и дату составления, а также идентификационные данные протокола.

6.10.3 Протоколы отклонения от нормы подвергают анализу. Обнаруженные проблемы оформляют документально, отслеживают и устраняют.

6.10.4 Информация об отклонениях от нормы должна быть передана разработчику и пользователям.

6.11 Модификация программного обеспечения

6.11.1 Общие положения

Решение о модификации ПО зависит от степени его влияния на СКУ, поэтому такие решения принимают с учетом требований пунктов 6.2.8 и 6.4.7 МЭК 61513:2011. В настоящем подразделе приведены дополнительные требования, специфичные или особенно важные для ПО.

6.11.2 Модификации ПО должны быть разработаны с последующей верификацией так, чтобы сохранялась неизменность требований, приведенных в 6.2, 6.3, 6.4, 6.5 и 6.6. Модификации следует проводить на месте в соответствии с требованиями 6.9.

6.11.3 Интеграцию и валидацию модификаций ПО следует проводить в соответствии с 6.7 и 6.8.

6.11.4 Если масштаб модификации не требует полной проверки соблюдения требований, приведенных в 6.7 и 6.8, интеграцию модифицированного ПО проводят согласно первоначальному плану интеграции, а валидацию — согласно первоначальному плану валидации ПО. Адекватность и полнота этих планов должны быть подтверждены с учетом масштаба любых модификаций и привлечением спецификации требований к ПО и спецификации проекта ПО. Должны быть составлены отчеты по результатам реализации этих планов.

6.11.5 *Для класса 2: при регрессивном подходе использование первоначальных планов интеграции и валидации ПО должно обеспечивать достаточную уверенность в том, что модифицированное ПО полностью соответствует спецификации требований к модифицированному ПО, а также:*

- что цели модификации достигнуты;

- дефекты не внесены;

- модифицированное и/или заново вводимое ранее разработанное ПО работает в соответствии с документацией по безопасности и модифицированной спецификацией проекта ПО;

- другие модифицированные и/или новые компоненты ПО соответствуют их спецификации.

6.11.6 Модификации ПО должны быть задокументированы в полном объеме. В частности, все, относящиеся к ПО документы, должны быть обновлены.

6.11.7 В документации по модификации ПО следует указывать:

- цели модификации ПО, включая любые цели на уровне системы;
- компоненты ПО, затронутые или созданные в процессе модификации;
- идентификацию версий этих компонентов до и после модификации.

Цели модификации на уровне системы документируют в соответствии с требованиями пункта 6.4.7 МЭК 61513:2011.

6.11.8 *Для класса 2: в дополнение к информации, указанной выше, в документации по модификации ПО указывают:*

- любые изменения, последовавшие в его спецификации;

- любые ограничения, которые следует соблюдать после модификации;

- ссылки на модифицированный проект и/или документы по реализации.

6.11.9 *Для класса 2: степень детализации документации по модификации ПО должна быть такой, чтобы:*

- способствовать уверенности в корректности модифицированного программного обеспечения и СКУ;

- показывать соответствие СКУ требованиям МЭК 61513:2011, относящимся к теме.

Относящиеся к данной теме требования МЭК 61513:2011 приведены главным образом в 6.2.2.3–6.2.2.5, 6.2.3.3, 6.2.3.5 и 6.2.4.

6.11.10 Должна быть проведена оценка влияния модификации ПО на остальные части СКУ и другие системы, с которыми ПО взаимодействует или разделяет ресурсы. Необходимо предпринять все меры для обеспечения правильной работы СКУ.

6.12 Защита от отказов по общей причине, возникших вследствие работы программного обеспечения

Из-за ошибок человека в процессе разработки и внедрения проекта могут возникать системные дефекты. Такие дефекты могут быть обусловлены ошибками или упущениями в спецификации требований к системе/ПО либо появляться позже при разработке и внедрении ПО (или в разрабатываемой части, или в интегрированном уже существующем проекте). Системные дефекты могут также возникать из-за инструментальных программ, когда они тоже подвержены системным дефектам, возникшим в процессе их разработки и внедрения. Таким образом, ПО потенциально может иметь скрытые системные дефекты, которые при определенных условиях, могут привести к ООП множества объектов, созданных по проекту ПО.

Информация по поводу возможных ООП на уровне системы содержится в стандартах ПК 45А более высокого уровня, а именно:

- МЭК 61513:2011, 5.4.2.6, где описана защита от ООП;
- МЭК 61513:2011, 5.4.4.2, где описана оценка надежности и защита от ООП.

Настоящий стандарт устанавливает процессы разработки и верификации ПО, а также требования к нему, следование которым минимизирует возможность ПО иметь системные дефекты, и потому минимизирует вероятность ООП, которые могут возникать вследствие работы программного обеспечения.

Приложение А
(справочное)

Стандартный перечень документации на ПО

В таблице А.1 приведен стандартный перечень документации на программное обеспечение.

Таблица А.1 — Стандартный перечень документации на ПО

Документ	Номер структурного элемента настоящего стандарта
Документы, которые относятся к изготовлению ПО	
Программа обеспечения качества ПО*	6.2.1
План верификации ПО	6.2.2
План управления конфигурацией ПО*	6.2.3
Документация по безопасности ранее разработанного ПО	6.3
Спецификация требований к ПО	6.4
Спецификация проекта ПО	6.5
Правила программирования	6.6.3.8, 6.6.4.7
Отчет о верификации ПО	6.2.2, 6.6.3, 6.6.4
План интеграции ПО*	6.7
Отчет об интеграции ПО*	6.7
План валидации ПО*	6.8
Отчет о валидации ПО*	6.8
Процедура установки ПО на штатном месте*	6.9
Документы, которые относятся к отклонениям от нормы	
Отчет об отклонениях от нормы	6.10
Документы, которые относятся к модификации ПО	
Документация по модификации ПО	6.11
Регрессивный (первоначальный) план интеграции ПО**	6.11
Отчет об интеграции по регрессивному плану**	6.11
Регрессивный (первоначальный) план валидации ПО**	6.11
Отчет о валидации по регрессивному плану**	6.11

* Данные документы можно не предоставлять, если их содержание включено в документацию системы, например, в программу обеспечения качества системы, план управления конфигурацией системы, план интеграции системы, отчет об интеграции системы, план валидации системы, отчет о валидации системы или в процедуру по установке системы на штатном месте.

** Если масштаб модификации не требует полной проверки соблюдения требований, приведенных в 6.7 и 6.8. Требования подразделов 6.2, 6.3, 6.4, 6.5, 6.6 и 6.9 всегда применимы к модификациям ПО и, следовательно, документы, относящиеся к данным подразделам, должны быть всегда в актуальном состоянии и отражать любые модификации.

Приложение В
(справочное)

Соотношение настоящего стандарта и МЭК 61513:2011

В таблице В.1 показано соответствие структурных элементов настоящего стандарта и МЭК 61513:2011.

Таблица В.1 — Соотношение настоящего стандарта и МЭК 61513:2011

Структурный элемент МЭК 61513:2011		Структурный элемент настоящего стандарта
5.4.2.5	Инструментальные средства	6.2.4
5.4.2.6	Защита от ООП	6.12
5.4.4.2	Оценка надежности и защиты от ООП	
5.6.2	Документация по проекту архитектуры	6.2.4
6	Жизненный цикл системы безопасности, рисунок 5	5.4
6.2.2.3.3	Внутреннее поведение системы	6.3.2, 6.5
6.2.2.7	Квалификация	6.2.4
6.2.3.2	Выбор ранее существующих компонентов	6.3
6.2.3.4	Спецификация программного обеспечения	6.4
6.2.4	Детальное проектирование и реализация системы	6.5, 6.6
6.2.5	Интеграция системы	6.7
6.2.6	Валидация системы	6.8
6.2.7	Монтаж системы	6.9
6.2.8	Модификация проекта системы	6.11
6.3.2	План по обеспечению качества	6.2.1
6.3.2.3	План по управлению конфигурацией системы	6.2.3
6.3.4	План по интеграции системы	6.7
6.3.5	План по валидации системы	6.8
6.4.4	Документация детального (технорабочего) проекта СКУ	6.5, 6.6
6.4.5	Документация по интеграции системы	6.7
6.4.6	Документация по валидации системы	6.8
6.4.7	Документация по модернизации и модификации системы	6.11
6.5.3.3	Анализ и оценка программного обеспечения	Все
8.2	Цели, которые должны быть достигнуты	Все

Приложение С
(справочное)

Взаимосвязь настоящего стандарта и МЭК 61508

С.1 Общие положения

Данное приложение устанавливает соответствия между настоящим стандартом и МЭК 61508-3:2010.

На системном уровне МЭК 61513:2011 (приложение D) устанавливает соответствия с МЭК 61508-1:2010, МЭК 61508-2:2010 и МЭК 61508-4:2010.

С.2 Сравнение областей применения и концепций

МЭК 61508 в целом относится к системам, связанным с безопасностью, тогда как данный стандарт учитывает практику МАГАТЭ и относится к системам, важным для безопасности (важным для ядерной безопасности).

МЭК 61508 классифицирует уровни полноты безопасности, необходимой для компьютерных систем, в соответствии со снижением рисков, который должна обеспечивать система. Для этого определяют серьезность риска, связанную с опасностью, оценивают частоту возникновения и последствия опасности, а также защиту, которую должна обеспечить система для снижения риска возникновения опасности до приемлемого уровня.

Как правило, в атомной энергетике традиционно используют детерминистский метод, чтобы определить значимость безопасности системы и ее влияние на серьезность риска, связанную с вероятностью выброса активности.

МЭК 61508 содержит требование о независимой оценке функциональной безопасности со стороны лиц и организаций, опыт и независимость которых должны расти с повышением уровня целостности безопасности (см. часть 1).

В атомной отрасли операторы АС (которые прежде всего несут ответственность за обеспечение ядерной безопасности) согласно общей практике отвечают за проведение адекватной оценки функциональной безопасности, однако данный процесс часто регулируется национальными нормами в области атомной энергетики.

С.3 Соответствие настоящего стандарта и МЭК 61508-3:2010

Таблица С.1 — Соотношение настоящего стандарта и МЭК 61508-3:2010

Настоящий стандарт		МЭК 61508-3:2010	
5.4	Жизненные циклы программного обеспечения и системы безопасности	7.1	Общие положения
6.2.1	Жизненный цикл программного обеспечения безопасности. Гарантия качества программного обеспечения		
6.2.2	Верификация	7.9	Верификация программного обеспечения
6.2.3	Управление конфигурацией	6.2.3	Управление конфигурацией программного обеспечения
6.2.4	Выбор и использование инструментального ПО	7.4.4	Требования к инструментальным средствам поддержки, включая языки программирования
6.2.5	Выбор языков		
6.3	Выбор ранее разработанного программного обеспечения	7.4.2	Общие требования
6.3.2	Документация по безопасности		Приложение D (нормативное) Руководство по безопасности для применяемых изделий. Дополнительные требования к элементам программного обеспечения
6.4	Спецификация требований к программному обеспечению	7.2	Спецификация требований к программному обеспечению системы безопасности
6.5	Проект программного обеспечения	7.4	Проектирование и разработка программного обеспечения
6.6	Реализация нового программного обеспечения		
6.7	Программные аспекты интеграции системы	7.5	Интеграция программируемой электроники (аппаратных средств и программного обеспечения)

Окончание таблицы С.1

Настоящий стандарт		МЭК 61508-3:2010	
6.8	Программные аспекты валидации системы	7.3	Планирование подтверждения соответствия безопасности системы для аспектов программного обеспечения
		7.7	Подтверждение соответствия аспектов программного обеспечения безопасности системы
6.9	Установка программного обеспечения на штатном месте		Не входит в область применения МЭК 61508-3, поскольку относится к МЭК 61508-1
6.10	Протоколы отклонений от нормы		Не входит в область применения МЭК 61508-3, поскольку относится к МЭК 61508-1
6.11	Модификация программного обеспечения	7.6	Процедуры эксплуатации и модификации программного обеспечения
		7.8	Модификация программного обеспечения
6.12	Защита от отказов по общей причине, возникших вследствие работы программного обеспечения		МЭК 61508-3 содержит информацию о защите от отказов по общей причине, возникающих вследствие работы ПО, в частности, в приложениях С и F
	В атомном секторе данная оценка связана с процессом лицензирования и зависит от органов, которые занимаются вопросами безопасности, а также национальных норм	8	Оценка функциональной безопасности МЭК 61508-1 устанавливает требования к техническим знаниям и независимости лица (лиц), проводящего оценку функциональной безопасности, в зависимости от уровня инноваций, технической новизны и возможных последствий отказов. Кроме того, большинство организаций, участвующих в оценке безопасности (в лице своих сотрудников) и сертификации продукции, имеют аккредитацию от национальных аккредитующих органов

Примечание — Справочные приложения настоящего стандарта и МЭК 61508 в таблице С.1 не рассматриваются.

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
IEC 60880:2006	IDT	ГОСТ Р МЭК 60880—2010 «Атомные электростанции. Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории А»
IEC 61226[1)]	IDT	ГОСТ Р МЭК 61226—2011 «Атомные станции. Системы контроля и управления, важные для безопасности. Классификация функций контроля и управления»
IEC 61513:2011	IDT	ГОСТ Р МЭК 61513—2020 «Системы контроля и управления, важные для безопасности атомной станции. Общие требования»
IEC 62671:2013	—	*

* Соответствующий национальный стандарт отсутствует. Текст стандарта на русском языке доступен на http://www.iaea.org/

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

- IDT — идентичные стандарты.

[1)] Действует IEC 61226:2020.

Библиография

IEC 61508-3:2010, Functional safety of electrical/electronic/ programmable electronic safety-related systems — Part 3: Software requirements

IEC 61508-4:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 4: Definitions and abbreviations

IEC 61511-1—2016, Functional safety — Safety instrumented systems for the process industry sector — Part 1: Framework, definitions, system, hardware and application programming requirements

IEC 62645:2014[1)], Nuclear power plants — Instrumentation and control systems — Requirements for security programmes for computer-based systems

ISO/IEC 12207:2008, Systems and software engineering — Software life cycle processes

ISO 9001:2015, Quality management systems — Requirements

ISO 90003:2014[2)], Software engineering — Guidelines for the application of ISO 9001:2008 to computer software

IAEA Safety Standard Series No. SSR-2/1:2016, Safety of Nuclear Power Plant: Design

IAEA Safety Guide SSG-39:2016, Design of instrumentation and control systems in Nuclear Power Plants

IAEA Safety Glossary:2016, Terminology used in nuclear safety and radiation protection

IAEA Safety Standard Series No. GS-G-3.5:2009, The Management System for Nuclear Installations

IEEE Std 7-4.3.2:2010, IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations

DO–178 revision C:2012, Software Considerations in Airborne Systems and Equipment Certification

[1)] Действует IEC 62645:2019 «Nuclear power plants — Instrumentation, control and electrical power systems — Cybersecurity requirements».

[2)] Заменен на ISO/IEC/IEEE 90003:2018 «Software engineering — Guidelines for the application of ISO 9001:2015 to computer software».

УДК 621.311.3.049.75:006.354 ОКС 27.120.20

Ключевые слова: атомная станция; системы контроля и управления, важные для безопасности; программное обеспечение систем контроля и управления; типы программного обеспечения; конфигурация программного обеспечения; спецификация требований

Редактор *Г.Н. Симонова*
Технический редактор *И.Е. Черепкова*
Корректор *С.И. Фирсова*
Компьютерная верстка *И.Ю. Литовкиной*

Сдано в набор 21.12.2021. Подписано в печать 18.01.2022. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 5,12. Уч-изд. л. 4,60.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении в ФГБУ «РСТ»
для комплектования Федерального информационного фонда стандартов,
117418 Москва, Нахимовский пр-т, д. 31, к. 2.
www.gostinfo.ru info@gostinfo.ru