

Вебинар, 20.11.2025



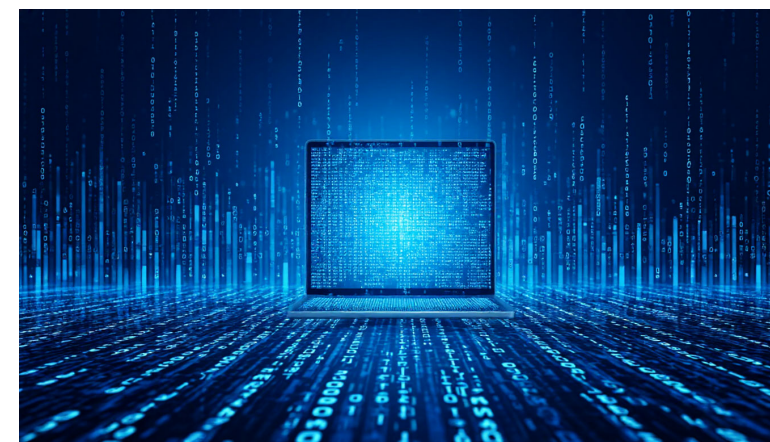
Особенности разработки встроенного ПО по требованиям ФБ

Елена Рогова

Ведущий специалист по функциональной безопасности,
к.т.н., ООО «ФанкСэйфети»

Андрей Рогов

Генеральный директор, к.т.н., ООО «ФанкСэйфети»



- **Цель функциональной безопасности (ФБ):** СНИЗИТЬ уровень риска, связанного с опасностями, вызванными некорректным поведением Э/Э/ПЭ систем, до приемлемого уровня
- **Для чего:** для защиты людей, оборудования и окружающей среды.

SIL

ФБ в промышленной автоматизации



ГОСТ Р МЭК 61508 **/разработка/**

ФБ Э/Э/ПЭ систем, связанных с безопасностью

Фокус на проектировании, разработке и изготовлении оборудования

(а также ГОСТ Р МЭК 61784-3, ГОСТ ISO 13849-1, ГОСТ Р МЭК 62061, ГОСТ Р МЭК 61131-6 и др.)

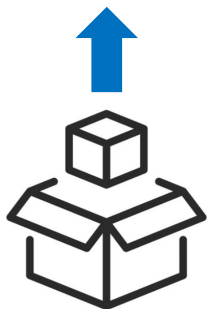


ГОСТ Р МЭК 61511 **/эксплуатация/**

ФБ приборных систем безопасности для промышленных процессов

Фокус на интеграции, проектировании и эксплуатации систем безопасности

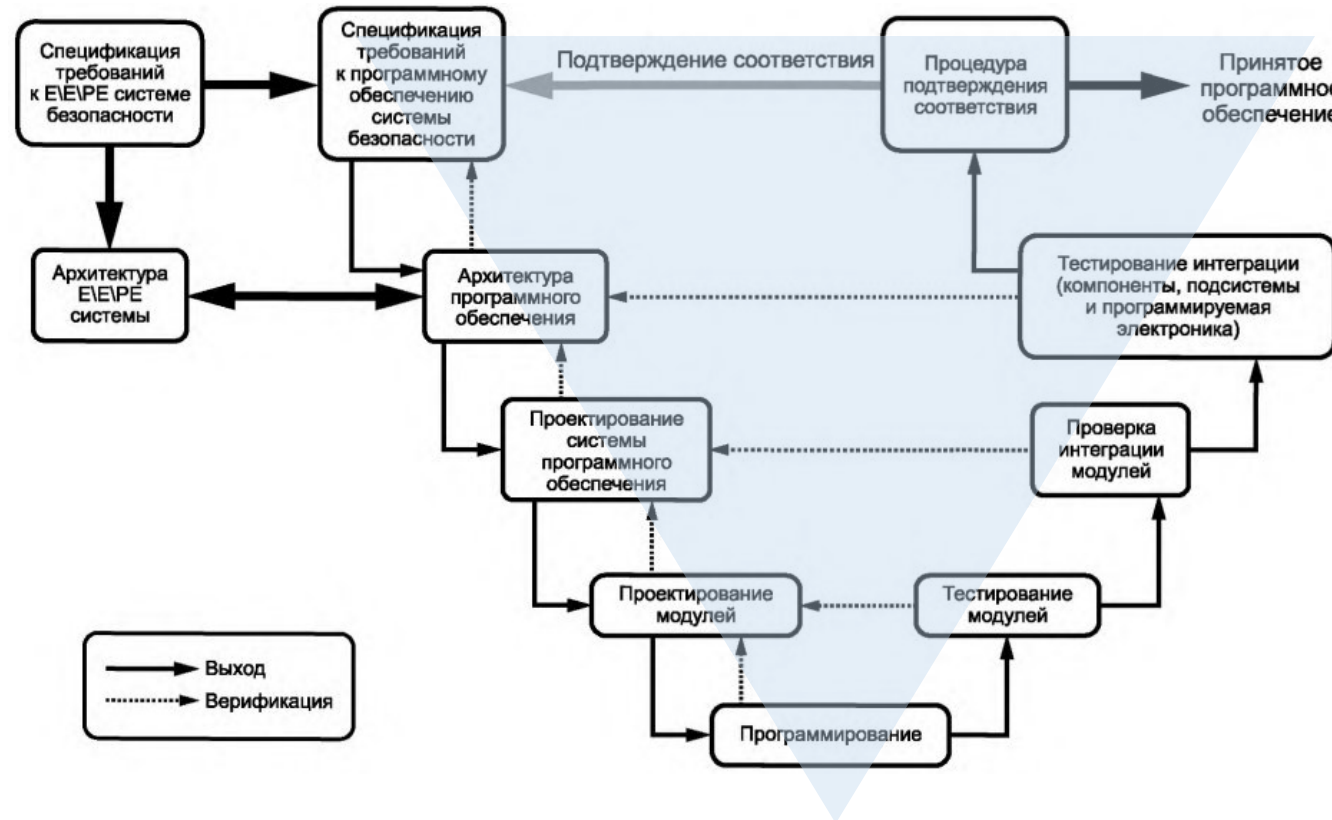
МЭК 61508



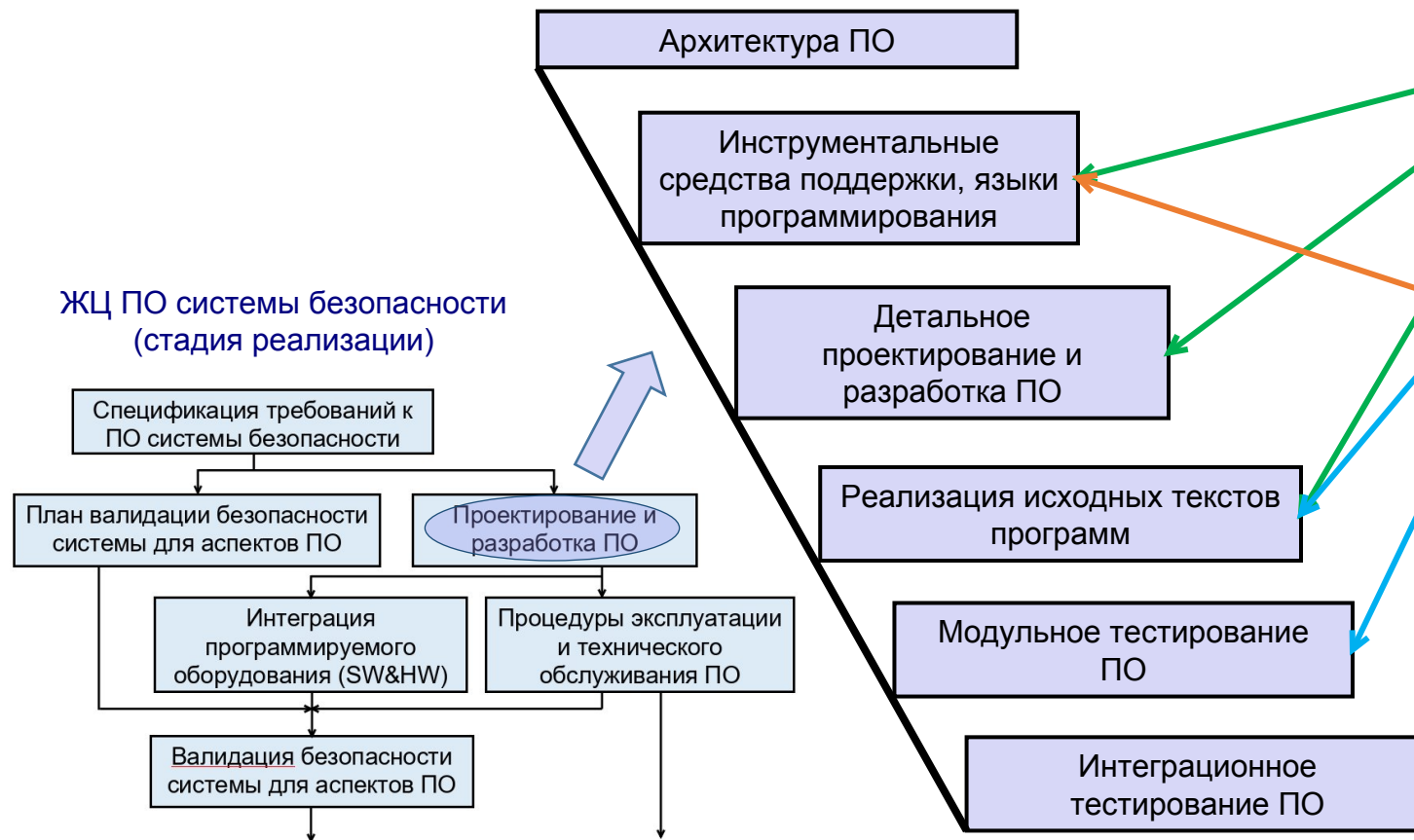
- ✓ Системный уровень
- ✓ Аппаратная часть (HW)
- ✓ Программная часть (SW)
- ✓ Коммуникационная часть
- ✓ Менеджмент функциональной безопасности

- системное ПО
- ОС
- прошивки
- ПО в коммуникационных сетях
- прикладное ПО
- интерфейсы пользователей

ГОСТ IEC 61508-3-2018: При разработке ПО необходимо следовать V-модели, которую, при необходимости, можно адаптировать с учетом полноты безопасности и сложности проекта



Проектирование и разработка ПО (п.7.4) включает :



Сегодня на вебинаре обсудим:

1. Стандарты кодирования
2. Верификация ПО и статический анализ
3. Инструментальные средства поддержки и их квалификация

Случайный отказ аппаратных средств (random hardware failure): Отказ, возникающий в случайный момент времени, который является результатом одного или нескольких возможных механизмов ухудшения характеристик в аппаратных средствах

Систематический отказ (systematic failure): Отказ, связанный детерминированным образом с какой-либо причиной, которая может быть исключена только путем модификации проекта либо производственного процесса, операций, документации, либо других факторов

Случайный отказ
/только HW/

Систематический отказ
/и HW, и SW/

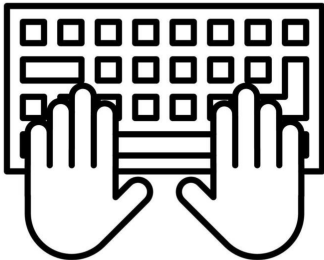
Руководство по выбору методов и средств (Приложение А, ГОСТ IEC 61508-3)

1. Спецификация требований к программному обеспечению системы безопасности
2. Проектирование и разработка ПО: проектирование архитектуры программного обеспечения
3. Проектирование и разработка ПО: инструментальные средства поддержки и языки программирования
4. Проектирование и разработка ПО: детальное проектирование (включая проектирование системы ПО, проектирование модуля ПО и кодирование)
5. Проектирование и разработка ПО: тестирование и интеграция программных модулей
6. Интеграция программируемых электронных устройств (SW & HW)
7. Подтверждение соответствия безопасности системы аспектов программного обеспечения
8. Модификация
9. Верификация ПО
10. Оценка ФБ

ГОСТ IEC 61508-3, п.7.4.4.12: Языки программирования для разработки всего ПО, связанного с безопасностью, должны использоваться в соответствии со стандартами составления программ для таких языков.

Таблица В.1 — Стандарты для проектирования и кодирования

Метод/средство ¹⁾	Ссылка	УПБ1	УПБ2	УПБ3	УПБ4
1 Использование стандартов кодирования для сокращения вероятности ошибок	C.2.6.2	HR	HR	HR	HR
2 Не использовать динамические объекты	C.2.6.3	R	HR	HR	HR
3a Не использовать динамические переменные	C.2.6.3	—	R	HR	HR
3b Проверка создания динамических переменных в неавтономном режиме	C.2.6.4	—	R	HR	HR
4 Ограниченное использование прерываний	C.2.6.5	R	R	HR	HR
5 Ограниченное использование указателей	C.2.6.6	—	R	HR	HR
6 Ограниченное использование рекурсий	C.2.6.7	—	R	HR	HR
7 Не использовать неструктурированное управление в программах, написанных на языках высокого уровня	C.2.6.2	R	HR	HR	HR
8 Не использовать автоматическое преобразование типов	C.2.6.2	R	HR	HR	HR



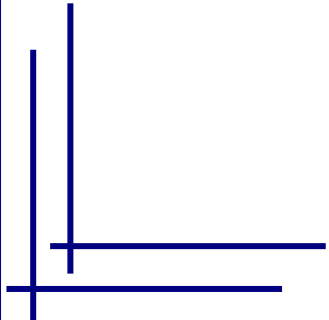
Стандарт кодирования (англ. coding standard) — набор правил, соглашений, лучших практик, рекомендаций, используемых при написании исходного кода на некотором языке программирования.

Стандарт кодирования позволяет обеспечить качества ПО:

- Безопасность (Safe): можно использовать без вреда.
- Защищённость (Secure): невозможно (сложно) взломать.
- Надёжность (Reliable): всегда работает так, как задумано.
- Тестируемость (Testable): можно протестировать на уровне кода.
- Поддерживаемость (Maintainable): можно поддерживать даже по мере роста кодовой базы.
- Переносимость (Portable): одинаково работает в разных средах.

Причины использования стандарта кодирования:

- Соответствие отраслевым стандартам (ГОСТ Р МЭК 61508 и др.).
- Стабильное качество кода — независимо от того, кто его пишет.
- Безопасность программного обеспечения с самого начала.
- Снижение затрат на разработку и поддержку, ускорение вывода продукции на рынок.



- В индустрии встраиваемых систем стандарты кодирования обязательны (или настоятельно рекомендуются) для соответствия. Это особенно актуально для стандартов функциональной безопасности, включая:
 - ГОСТ Р МЭК 61508:
«Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью.»
 - ГОСТ Р ИСО 26262:
«Дорожные транспортные средства. Функциональная безопасность.»
 - ГОСТ Р МЭК 62279:
«Системы связи, сигнализации и обработки данных. Программное обеспечение систем управления и защиты на железных дорогах.»
- Связанные с безопасностью правила стандартов кодирования помогают выполнять конкретные нормативные требования стандартов ФБ.
- Стандарты кодирования позволяют (частично) автоматизировать проверку на соответствие требованиям ФБ.
- Каждый стандарт кодирования написан для определенного языка программирования. Большинство разработчиков встраиваемых систем используют C или C++.

Таблица С.1 - Рекомендации по конкретным языкам программирования (ГОСТ Р МЭК 61508-7-2012)

Язык программирования	УПБ1	УПБ2	УПБ3	УПБ4
Java	NR	NR	NR	NR
Java с подмножеством	R	R	NR	NR
ADA	HR	HR	R	R
ADA с подмножеством	HR	HR	HR	HR
C	R	-	NR	NR
C с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR
C++	R	-	NR	NR
C++ с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR
Ассемблер	R	R	-	-
Ассемблер с подмножеством и стандартом кодирования	R	R	R	R

- **MISRA C** — набор рекомендаций (“guidelines”) или стандарт кодирования по разработке ПО для языка программирования C, разработанный консорциумом MISRA — Motor Industry Software Reliability Association (Ассоциация надежности программного обеспечения для автомобильной промышленности).
- **Цель MISRA C** — обеспечить безопасность, защищенность, надёжность и переносимость кода во встраиваемых системах, в частности, в системах, написанных на стандартах ISO C/C90/C99.
- **MISRA C** определяет *подмножество* языка C, которое может быть использовано для создания программных приложений, требующих высокого уровня безопасности
- Издания: 1998, 2004, 2012, 2023, 2025



Требование ГОСТ IEC 61508-3	Описание требования	Правила MISRA C 2012
B.1-2	Не использовать динамические объекты	Dir 4.12, Rule 21.3
B.1-5	Ограниченное использование указателей	Dir 4.1, Rule 11.x, Rule 18.x
B.1-6	Ограниченное использование рекурсий	Rule 17.2
B.1-7	Не использовать неструктурированное управление в программах, написанных на языках высокого уровня	Rule 15.x
B.1-8	Не использовать автоматическое преобразование типов	Rule 10.x

Категории рекомендаций МЭК 61508-3 для уровня полноты безопасности (УПБ):

HR	Настоятельно рекомендуется применять этот метод или средство для данного уровня полноты безопасности. Если метод или средство не используется, то на этапе планирования системы безопасности этому должно быть дано подробное объяснение со ссылкой на приложение С, и это объяснение должно быть согласовано с экспертом
R	Метод или средство рекомендуется применять для данного уровня полноты безопасности, но степень обязательности рекомендации ниже, чем в случае рекомендации HR
-	Для данного метода или средства рекомендации ни за, ни против не приводятся
NR	Данный метод или средство не рекомендуется для этого уровня полноты безопасности. Если данный метод или средство применяют, то на стадии планирования системы безопасности этому должно быть дано подробное обоснование со ссылкой на приложение С, которое следует согласовать с экспертом

Категории директив/правил MISRA C:

Mandatory - Обязательные	Должно соблюдаться для соответствия MISRA C. Нарушение не допускается.
Required - Требуемые	Должно соблюдаться для соответствия MISRA C. Каждое нарушение должно быть записано и согласовано в соответствии с процедурой отступлений от правил.
Advisory - Рекомендательные	Рекомендации. Официальная процедура отступления не требуется. Организация или проект вправе приравнять любое рекомендательное указание к статусу обязательного или требуемого.

- Нет однозначного соответствия между категориями рекомендаций МЭК 61508-3 для УПБ и категориями директив/правил MISRA C.

— Если код удовлетворяет требованиям MISRA C, значит ли это, что он удовлетворяет требованиям МЭК 61508?

— **НЕТ!**

- MISRA C покрывает лишь **часть** требований (стандарт кодирования) к ПО согласно ГОСТ Р МЭК 61508.
- Остальные требования:
 - Требования к процессам (жизненный цикл)
 - Верификация
 - Управление изменениями
 - Квалификация средств поддержки ПО
 - Оценка функциональной безопасности
 - ...



— *Когда можно сказать, что MISRA C «достаточно»?*

— ***Только если:***

- Процессы разработки уже соответствуют требованиям МЭК 61508
- MISRA C используется для удовлетворения требованиям МЭК 61508 в части стандарта кодирования.

«Ограничение ответственности»

(можно увидеть в отчётах инструментов статического анализа):

- *«Соответствие MISRA C способствует выполнению требований МЭК 61508 в части безопасного программирования и стандарта кодирования, однако само по себе соответствие MISRA C недостаточно для демонстрации соответствия МЭК 61508.»*

Верификация ПО (Таблица А.9, ГОСТ IEC 61508-3)

Метод/средство ¹⁾	Ссылка	УПБ1	УПБ2	УПБ3	УПБ4
1 Формальное доказательство	С.5.12	—	R	R	HR
2 Анимация спецификации и тестирования	С.5.26	R	R	R	R
3 Статический анализ	В.6.4, таблица В.8	R	HR	HR	HR
4 Динамический анализ и тестирование	В.6.5, таблица В.2	R	HR	HR	HR
5 Прямая прослеживаемость между спецификацией проекта программного обеспечения и планом верификации (включая верификацию данных) программного обеспечения	С.2.11	R	R	HR	HR
6 Обратная прослеживаемость между планом верификации (включая верификацию данных) программного обеспечения и спецификацией проекта программного обеспечения	С.2.11	R	R	HR	HR
7 Численный анализ в автономном режиме	С.2.13	R	R	HR	HR
Тестирование и интеграция программного модуля	См. таблицу А.5				
Проверка интеграции программируемых электронных устройств	См. таблицу А.6				
Тестирование программной системы (подтверждение соответствия)	См. таблицу А.7				



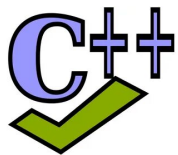
Статический анализ (Таблица В.8, ГОСТ ИЕС 61508-3)

ГОСТ ИЕС 61508-3, п.7.9.2.12: Исходный текст **должен быть верифицирован статическими методами** для того, чтобы гарантировать соответствие спецификации проекта программных модулей, необходимым стандартам кодирования и плану подтверждения соответствия безопасности системы для аспектов программного обеспечения.

Метод/средство ¹⁾	Ссылка	УПБ1	УПБ2	УПБ3	УПБ4
1 Анализ граничных значений	C.5.4	R	R	HR	HR
2 Таблица контрольных проверок	B.2.5	R	R	R	R
3 Анализ потоков управления	C.5.9	R	HR	HR	HR
4 Анализ потоков данных	C.5.10	R	HR	HR	HR
5 Предположение ошибок	C.5.5	R	R	R	R
6a Формальные проверки, включая конкретные критерии	C.5.14	R	R	HR	HR
6b Сквозной контроль (программного обеспечения)	C.5.15	R	R	R	R
7 Тестирование на символьном уровне	C.5.11	—	—	R	R
8 Анализ проекта	C.5.16	HR	HR	HR	HR
9 Статический анализ выполнения программы с ошибкой	B.2.2, C.2.4	R	R	R	HR
10 Временной анализ выполнения при наихудших условиях	C.5.20	R	R	R	R



- **Статический Анализ (СА)** — анализ кода без его выполнения.
- **Инструменты СА** могут работать с исходным кодом, с скомпилированным кодом и промежуточными представлениями (AST, CFG, DFG).
- СА выполняет проверки:
 - небезопасные вызовы функций или использование устаревших библиотек;
 - определение и доступность переменных/функций/объектов;
 - недостижимый код, бесконечные циклы, непредсказуемые граничные условия, возможные утечки памяти или необработанные исключения, неоднозначные и неоптимальные выражения.
- СА может проверять на соответствие стандарту кодирования (MISRA)
- Инструменты СА:
Cppcheck, Clang Static Analyzer, PVS-Studio, Coverity, CodeSonar, PC-Lint



- Большинство правил MISRA C можно проверить с помощью инструментов статического анализа кода («разрешимые правила»).
- «Разрешимое правило» (“Decidable Rule”):
можно всегда однозначно проверить код на соответствие правилу
- «Неразрешимое правило» (“Undecidable Rule”):
невозможно всегда однозначно проверить код на соответствие правилу (используя статический анализ)
- Правило обычно «неразрешимо», если обнаружение нарушений зависит от динамических свойств кода: значений данных, достижение потоком управления определённой точки в программе, внешних данных и периферии
- Статический анализатор может выдавать ложные (false positives) или пропускать нарушения (false negatives) для «неразрешимых» правил
- Некоторые правила требуют использования динамического анализа кода



Средства поддержки программного обеспечения

On-line

В режиме реального времени

Программное средство, имеющее непосредственный доступ к системе, связанной с безопасностью, в процессе ее функционирования.

Примечание: такие же требования, как и к любой части ПО системы, связанной с безопасностью - это элемент ПО системы, связанной с безопасностью

В автономном режиме

Off-line

Программное средство, поддерживающее этап разработки ЖЦ ПО, которое не имеет непосредственного доступа к системе, связанной с безопасностью, в процессе ее функционирования.

T1 - не генерирует программ, которые явно или неявно включаются в рабочую программу (включая данные) системы, связанной с безопасностью

T2 - включает в себя средства тестирования или верификации проекта либо рабочей программы, причем такие, ошибки в которых могут привести к сбою при обнаружении ошибок в рабочей программе, но эти средства не могут создавать ошибки в самой рабочей программе.

T3 - генерирует программы, которые явно или неявно включаются в рабочую программу системы, связанной с безопасностью

Средства поддержки программного обеспечения в автономном режиме

Примеры:

- T1** - текстовый редактор или средства поддержки проектирования, написанные не на автокоде
- T2** - генератор тестовых программ, средства измерения тестового охвата, **средства статического анализа**
- T3** - оптимизирующий компилятор, связь между исходным кодом программы и сгенерированным объектным кодом которого не очевидна, компилятор, который включает исполнимый пакет программ в рабочую программу

ГОСТ IEC 61508-3, п.7.4.4.16 Управление конфигурацией должно гарантировать, что для инструментальных средств в классах T2 и T3 используются только **квалифицированные версии**.

ГОСТ IEC 61508-3, п.7.4.4.18 Каждая новая версия инструментального средства поддержки, работающего в автономном режиме, **должна быть квалифицирована**.

ГОСТ IEC 61508-3, п.7.4.4.5 Для того, чтобы определить уровень доверия к инструментальным средствам и возможные механизмы отказа инструментальных средств, которые могут повлиять на исполняемое ПО, должна быть выполнена **оценка инструментальных средств поддержки классов T2 и T3**. Если механизмы отказа идентифицированы, то должны быть использованы соответствующие меры по их ослаблению.



Что такое квалификация средств поддержки ПО?

ГОСТ IEC 61508-3, п.7.4.4.7 Результаты подтверждения соответствия инструментальных средств должны быть документально оформлены и содержать:

- а) хронологическую запись действий по подтверждению соответствия;
- б) версию используемого руководства по инструментальному средству;
- в) функции инструментального средства, для которых проводится процедура подтверждения соответствия;
- г) используемые инструментальные средства и оборудование;
- д) результаты действия по подтверждению соответствия (валидации); документально оформленные результаты подтверждения соответствия должны установить, что подтверждено соответствие ПО или существуют причины для отказа;
- е) контрольные примеры и их результаты для последующего анализа;
- ж) несоответствия между ожидаемыми и фактическими результатами.



Что такое квалификация средств поддержки ПО?

Пример квалификационного пакета от Вендора инструментального ПО:

- Функциональные требования ИПО (которые нужно квалифицировать)
- Руководство пользователя (вкл. руководство по установке и конфигурации, руководство по интерпретации результатов, ограничения на использование)
- Описание процесса разработки ИПО (QA процедуры, известные аномалии и ограничения со списком выявленных багов, несоответствий с объяснением, почему это не влияет / влияет на безопасное использование ИПО)
- V&V План (на каждое требование - один или несколько тестовых кейсов)
- V&V пакет (все тестовые кейсы из в V&V плана; скрипты для выполнения тестов и оценки результатов)



Примеры несоответствий ПО требованиям ФБ:

- Отсутствие применения стандартов кодирования (например, MISRA C)
- Пренебрежение принципом модульности
- Средства поддержки ПО используются без квалификации
- Верификация на разных этапах разработки ПО выполняется частично
- Отсутствие прослеживаемости требований к ПО и разделения на safety и не safety требования
- Отсутствие исходного кода исполнительной системы (для ПЛК)
- Отсутствие описания архитектуры ПО, спецификаций, РФБ и др.
- Отсутствие системы управления изменениями ПО



Спасибо!

ООО «ФанкСэйфети»



- ✓ Подготовка к сертификации по стандартам ФБ
- ✓ Оценка функциональной безопасности, аудит ФБ
- ✓ FMEA, FMEDA анализ сложных систем "под ключ"
- ✓ Разработка КУБ коммуникационных протоколов. Расчёт интенсивности остаточных ошибок
- ✓ Расчёт надёжности
- ✓ Повышение ФБ компетенций специалистов, задействованных в Ваших проектах
- ✓ В штате сертифицированные TÜV SÜD инженеры и эксперт по ФБ

8(800)222-89-85
info@func-safety.ru