Николай Бодунов, Игорь Починок, Auriga Inc., НИВЦ МГУ

Обзор стандартов жизненного цикла ПО для систем с высокими требованиями к надёжности и безопасности

Статья посвящена обзору наиболее распространённых стандартов, описывающих процессы создания надёжного и безопасного программного обеспечения. Описываются базовые подходы к разработке такого ПО и особенности отраслевых стандартов в приложении к медицине, авиации, автомобильному транспорту, атомной энергетике и космической деятельности.

Введение

В современном мире всё больше устройств содержат микропроцессоры и, соответственно, функциональность этих устройств во многом зависит от управляющего ими программного кода. В этой связи безопасность и надёжность программного кода выходят на передний план – ведь, в зависимости от типа устройства, выход его из строя может повлиять как на комфорт одного человека (отказ стиральной машины), так и на безопасность жителей всей страны (взрыв на реакторе в Фукусиме). Вполне естественно, что для разработки надёжных программных продуктов появляются стандарты, специфичные для отраслей их будущего применения (транспорт, медицина, производство, атомная энергетика). Статья посвящена обзору наиболее распространённых стандартов, описывающих процессы создания надёжного и безопасного программного обеспечения (ПО).

Общие положения

Мы будем использовать термин *safety-critical* для систем, которые, вследствие их краха или сбоя могут:

- привести к смерти или серьёзному ущербу для здоровья людей;
- привести к разрушению или серьёзным повреждениям дорогостоящего оборудования;
- нанести ущерб окружающей среде.

Если крах или сбой ПО в этих системах может привести к вышеперечисленным последствиям, то ПО классифицируется как *safety-critical* и должно разрабатываться в соответствии с регулирующими эту отрасль стандартами.

В данной статье будут описываться стандарты только на жизненный цикл ПО, иногда ограниченного до разработки программного кода и тестов, в то время как аппаратные (электронные и механические) особенности рассматриваемых систем остаются за пределами рассмотрения.

В интересах статьи вначале определим отрасли человеческой деятельности – достаточно крупные и с уже существующими формализованными стандартами на разрабатываемое программное обеспечение. К этим отраслям относятся:

- медицина (IEC 62304);
- транспорт, в том числе:
  - авиация (DO-178);
  - автомобильный транспорт (ISO 26262);
- атомная энергетика (IEC 60880, IEC 62138);
- космос (ECSS-E-ST-40C).

Некоторые отрасли имеют также собственные стандарты, например железнодорожный транспорт (IEC 62279) (не рассматривается в данной статье, вследствие малой распространённости этого стандарта). Судостроение вообще не имеет пока установившегося отраслевого стандарта и использует общий IEC 61508, описанный ниже.

Общие черты для большинства стандартов ПО класса safety-critical

Базовым стандартом для ПО класса safety-critical является стандарт IEC 61508: «Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems». Этот стандарт описывает общие требования к ПО для систем класса safety-critical и применяется к

системам, связанным с безопасностью в случаях, когда одна или несколько таких систем включают в себя электрические и/или электронные программируемые устройства. Если в какой-то отрасли ещё не созданы специфические стандарты, то используется «классический» IEC 61508. Таким образом, IEC61508 выступает базой для разработки производных от него отраслевых стандартов и диктует общие для них свойства:

1. Отношение к стороннему ПО. Основное отличие ПО класса safety-critical от остального – строго управляемый процесс разработки, нацеленный на минимизацию рисков внесения ошибки в требования, ПО и тесты. Если ПО приобретается на стороне и используется затем для совместной работы с создаваемым ПО класса safety-critical (или его разработки – компиляторы, runtime-библиотеки, генераторы тестов и пр.), стандарты класса safety-critical рассматривают его как SOUP (Software Of Unknown Pedigree), PDS (Previously Developed Software) или COTS (Commercial Off-The-Shelf). Разумеется, для стороннего ПО, за редким исключением, недоступны как исходный код, так и любые сведения о процессе его жизненного цикла. Поэтому стороннее ПО перед использованием в приложениях класса safety-critical обязательно подвергается функциональному тестированию и квалификации по его итогам.

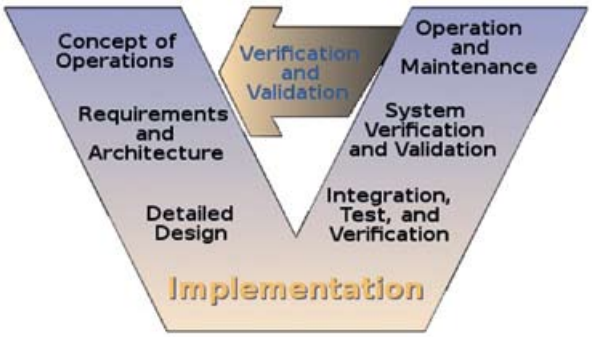

Рис. 1. Простой вариант V-модели разработки ПО (Источник: Википедия)

2. Функционирование ПО класса safety-critical не на PC-совместимых компьютерах, а во

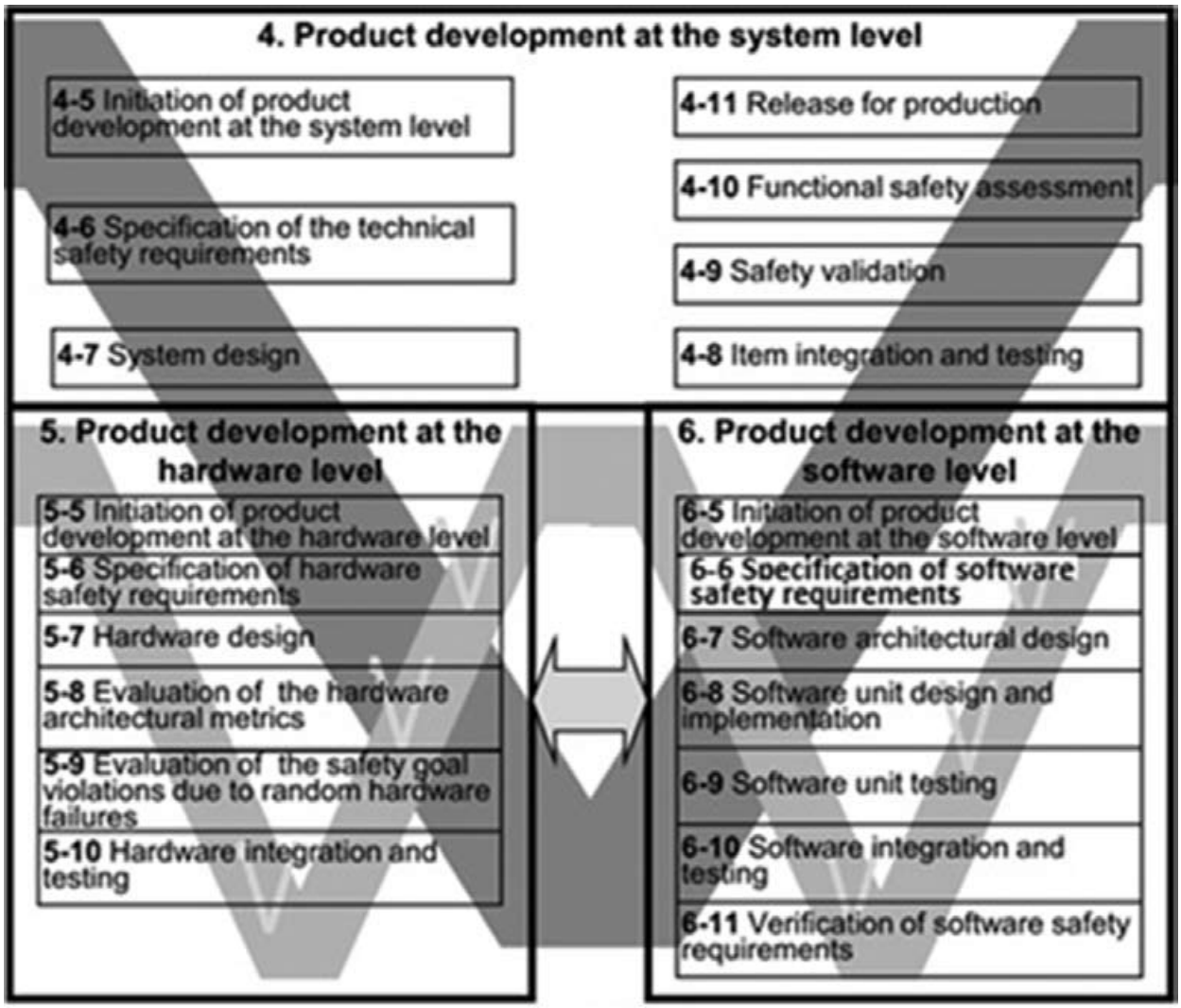

Рис. 2. Более сложный вариант V-модели разработки ПО

встраиваемых специализированных системах – в автомобилях, самолётах, контроллерах управления производственным оборудованием, которые не являются изделиями COTS, а разрабатываются, как и соответствующее ПО, под конкретную задачу. Это связано с повышенными требованиями к аппаратуре в системах класса safety-critical – по температуре, влажности, работе в агрессивной среде, иногда радиации. Поэтому важной особенностью ПО класса safety-critical является его сертификация в качестве неотъемлемой части программно-аппаратного комплекса. ПО, которое сертифицировано на одной аппаратной системе, не является автоматически сертифицированным для всех подобных аппаратных систем. Сертификацию (и предшествующее ей тестирование) придётся проходить полностью для новой аппаратной системы.

3. Отношение к ПО, сертифицированному по предыдущим версиям стандартов. Большинство стандартов не требует ресертификации при переходе к новой версии стандарта («работает – не надо трогать»), но есть и исключения – например, DO-178B требует ресертификации систем, сертифицированных по более ранним версиям стандарта (DO-178, DO-178A).

4. **Использование так называемой V-модели разработки**, которая позволяет эффективно отслеживать и поддерживать непрерывную взаимосвязь между разработкой и тестированием. В простом случае (если стандарт позволяет сертифицировать ПО отдельно от аппаратуры) модель разработки выглядит так, как показано на рис. 1.

Если же, как часто бывает, ПО и аппаратная часть разрабатываются и сертифицируются в связке, модель выглядит сложнее, в этом случае букву V образуют также аппаратная и программная части (ISO 26262) (рис.2).

V-модель не запрещает использовать итерационный подход к разработке, а также опускать отдельные этапы – в той мере, в какой это позволяет строгость соответствующего стандарта.

5. Каждый стандарт определяет несколько классов критичности систем, на которых работает ПО – в зависимости от последствий, к которым приведёт отказ системы. Систему относят к одному из классов и процессы разработки для неё могут быть более или менее строгими в пределах одного стандарта. К какому именно классу отнести систему, зависит от многих обстоятельств. Так, например, искажение показаний электронного термометра может привести к инъекции сильнодействующего лекарства, и как один из возможных исходов, к смерти пациента, или же, если медсестра опытная, всего лишь к повторному замеру температуры другим термометром. Отнесение ПО к тому или иному классу, вместе с ответственностью за это отнесение, полностью лежит на плечах **заказчика** разработки.

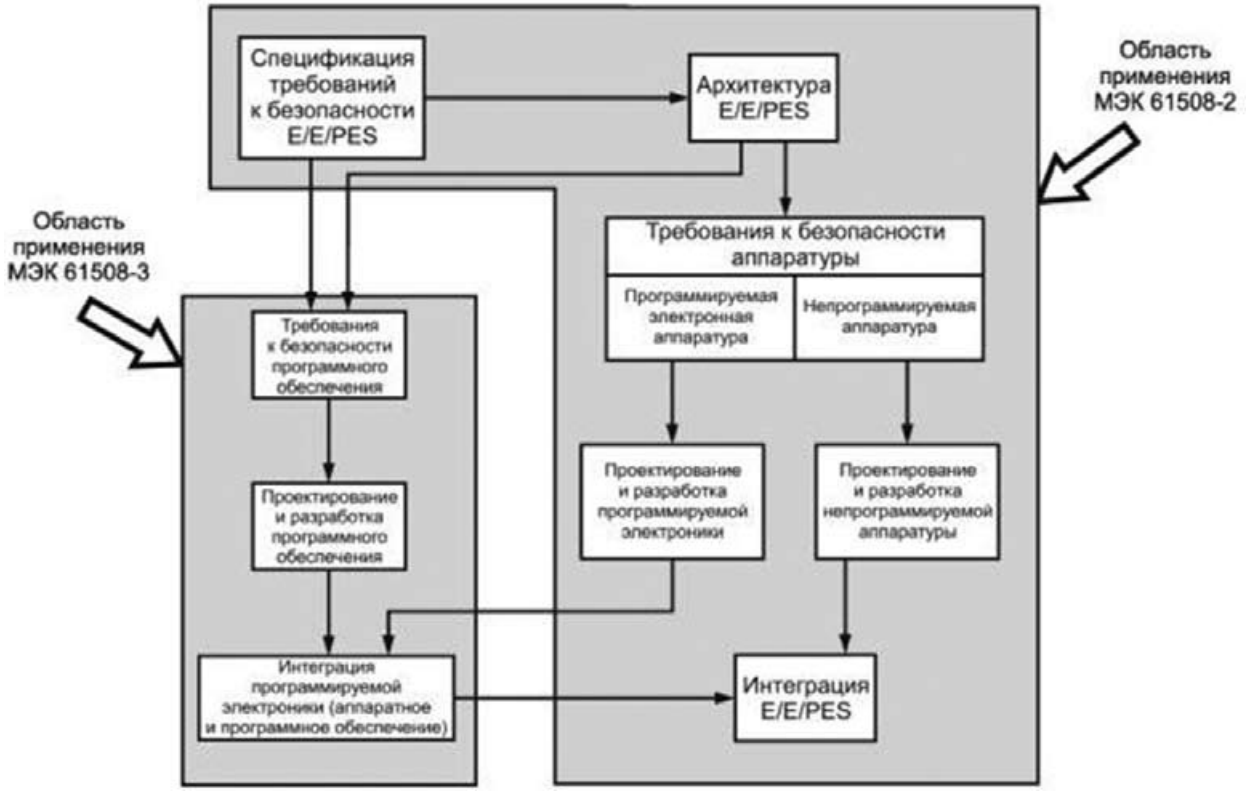

Рис. 3. Соотношение между второй (Part 2) и третьей (Part 3) частями стандарта IEC 61508

Рис. 4. V-модель, которая используется в стандарте IEC 61508

Обзор стандартов

Стандарт IEC 61508

Стандарт IEC 61508 «Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems» (E/E/PE, or E/E/PES), Part 3 «Software requirements». (отечественный аналог данного стандарта – ГОСТ Р МЭК 61508) регулирует разработку любого ПО, которое является частью системы, связанной с безопасностью, либо используется для разработки системы, связанной с безопасностью, в рамках области применения IEC 61508 Part 1 (общие требования) и IEC 61508 Part 2 (требования к системам).

Соотношение между второй и третьей частями стандарта показано на рис. 3.

Стандарт определяет:

- спецификацию требований к безопасности ПО;
- планирование подтверждения требований к безопасности ПО;
- проектирование и разработку ПО;
- интеграцию ПО и аппаратных средств, на которых оно выполняется;
- работу ПО и процедуры модификации;
- подтверждение соответствия безопасности;
- модификацию ПО;
- верификацию ПО.

Стандарт IEC 61508 описывает интегральные уровни безопасности (SIL0–SIL3)(Safety Integrity Level), которые используются при выборе процессов разработки ПО и аппаратной составляющей. Уровень SIL3 характеризует наибольшую полноту безопасности.

Стандарт устанавливает требования к стадиям жизненного цикла ПО и действиям, которые должны предприниматься в процессе проектирования и разработки ПО. Стандарт использует V-модель, описанную на рис. 4.

Каждая из стадий жизненного цикла, приведённых на рис. 4, подробным образом описана в стандарте с точки зрения назначения, области применения, входных и выходных данных.

Приложения A и B к стандарту определяют обязательные и необязательные методы и средства, используемые в каждой из вышеописанных стадий, в зависимости от интегрального уровня безопасности SIL0–SIL3.

Отраслевые стандарты, которые рассматриваются ниже, – производные от IEC 61508.

Стандарт IEC 62304

Стандарт IEC 62304 «Medical device software – Software life cycle processes» адаптирует требования IEC 61508 к жизненному циклу разработки и поддержки ПО для медицинских приложений, работающих в медицинских устройствах, и ПО, которое квалифицируется само по себе как медицинское устройство (описано в документе Европейской Комиссии «MEDDEV 2.1/6. Qualification and Classification of stand alone software»).

Требования к системе менеджмента качества, usability, клиническим испытаниям и валидации ПО в данном стандарте не определяются. Они могут быть предметом рассмотрения других стандартов, в частности требования к валидации медицинского встраиваемого ПО определены в другом стандарте, EN 60601-1.

Стандарт IEC 62304 описывает три класса безопасности:

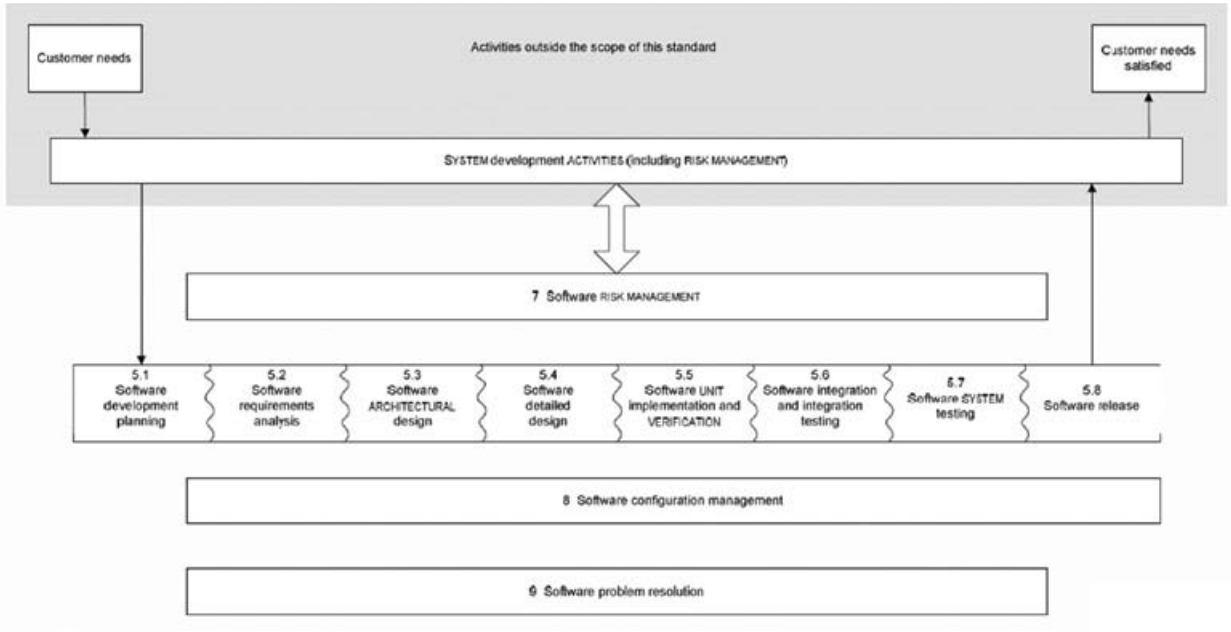

Рис. 5. Общая картина жизненного цикла разработки ПО согласно IEC 62304

– класс A – сбой ПО не может привести к травмам или ущербу здоровью;
– класс B – сбой ПО может привести к незначительным травмам;
– класс C – сбой ПО может привести к смерти или значительному ущербу здоровью.

При разработке ПО класса A могут быть применены лишь несколько требований IEC 62304, для ПО класса C – все требования.

Общая картина жизненного цикла разработки ПО согласно IEC 62304 отражена на рис. 5.

Стандарт не предъявляет требований к организации самого процесса разработки и допускает отдавать разработку ПО на аутсорсинг. Тем не менее процессы управления рисками обязательны для всех классов A–C и должны осуществляться в соответствии с ISO 14971.

Стандарт DO-178

Стандарт DO-178 «Software Considerations in Airborne Systems and Equipment Certification» описывает требования к созданию встраиваемого ПО для авиационных бортовых систем (авионики). Не надо объяснять, что к надёжности бортовых систем самолётов предъявляются самые жёсткие требования в силу специфики области. ПО для наземных систем описывается другими стандартами.

Стандарт рассматривает ПО как часть экосистемы, поскольку в авиационной области ПО, электроника и механика сертифицируются вместе как единая сущность.

Поскольку в авиации отказ бортовых систем может привести к более серьёзным последствиям, чем отказ медицинского оборудования, то и уровней критичности отказа в DO-178 имеется больше. Как и в IEC 62304, уровни критичности отказа рассматриваются в первую очередь по влиянию на безопасность экипажа и пассажиров. Стандарт DO-178 определяет пять уровней критичности отказов систем и сертифицируют ПО для работы на этих уровнях:

A – катастрофический: отказ может привести к крушению летательного средства;

B – угрожающий: отказ может привести к проблемам с управлением летательного средства, серьёзным травмам или смерти пассажиров;

C – важный: значительный, но имеющий меньшее влияние, чем B; например, отказ может вызвать

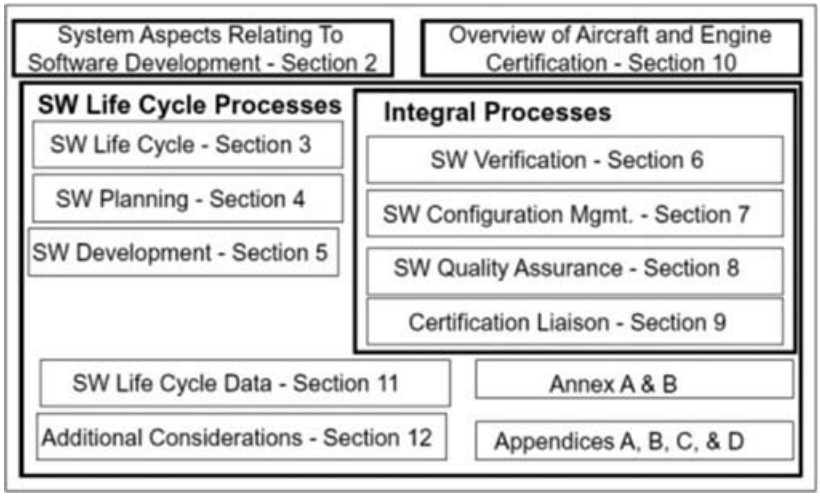

Рис. 6. Структура стандарта DO-178

серьёзные неудобства для пассажиров, не приводящие, тем не менее, к травмам;

D – малозначительный: ощущаемый пассажирами, но терпимый (например, задержка рейса или смена маршрута);

E – незначительный: не имеет влияния на экипаж и пассажиров (не рассматривается в стандарте DO-178, за исключением определения).

Структура стандарта показана на рис. 6.

Процессы жизненного цикла ПО включают процесс планирования, процессы разработки ПО и интегральные процессы, которые протекают во время всего жизненного цикла ПО. Стандарт описывает требования (objectives) к каждому из этих процессов и набор активностей, которые должны включать процессы, чтобы следовать этим требованиям.

Стандарт ISO 26262

Стандарт ISO 26262 «Road vehicles – Functional safety» описывает требования к жизненному циклу систем, влияющих на безопасность пассажирских автомобилей массового производства с весом до 3500 кг. Помимо требований к ПО, стандарт ISO 26262 содержит также требования к электронике и электрике автомобилей.

Область применения стандарта:
- системы, встроенные в автомобиль;
- системы, имеющие влияние на безопасность пассажира в автомобиле.

Так, например, система ABS подпадает под действие этого стандарта, а магнитола – нет.

Структура стандарта ISO 26262 показана на рис. 7.

Стандарт специфицирует следующие процессы в части, касающейся разработки ПО:
- инициализация разработки продукта на программных уровнях;
- спецификация требований к безопасности ПО;
- разработка архитектуры ПО;
- разработка блоков ПО;
- тестирование блоков ПО;
- интеграция и тестирование ПО;
- верификация требований к безопасности ПО.

Так же, как и в остальных стандартах, вводится определение уровней безопасности, но в данном стандарте этот вопрос рассмотрен более глубоко.

Стандарт описывает *Automotive Safety Integrity Level(ASIL)*, общий уровень безопасности, который классифицирует различные возникающие события в автомобиле (или его элементах) по нескольким па-

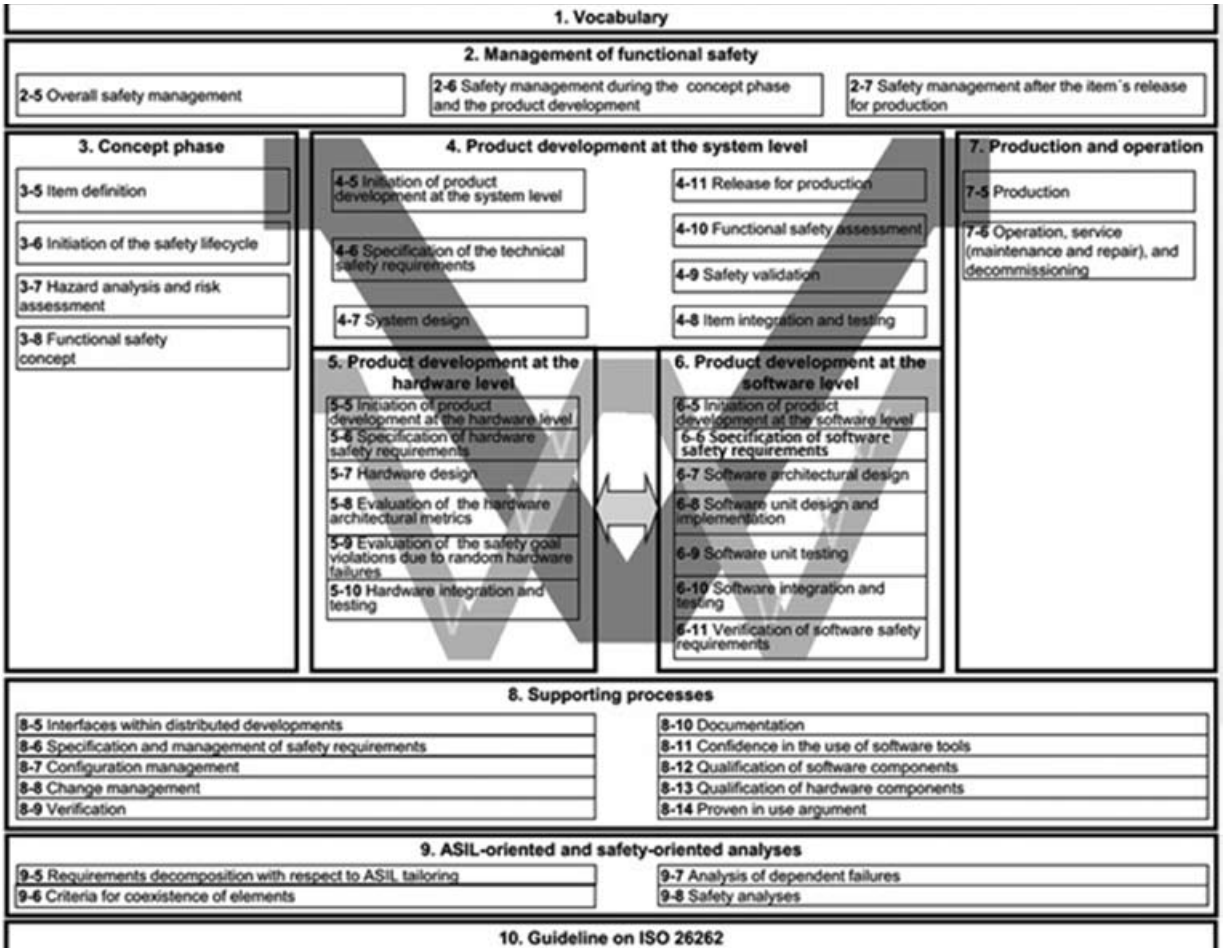

Рис. 7. Структура стандарта ISO 26262

раметрам. Такое многомерное разделение позволяет создать гибкий набор методологий, достаточно легко адаптируемый к большинству задач.

Следует также заметить, что стандарт не предусматривает раздельных требований к безопасности программной и аппаратной части подсистем, они разрабатываются совместно и параллельно как экосистема, в целях «подстраховки» той и другой стороны (усиление либо замена слабых мест одной системы за счёт использования другой).

В заключение описания стандарта ISO 26262 стоит упомянуть ещё одну опцию – *Proven in use*. Она позволяет избежать больших затрат на верификацию уже существующего ПО (и электроники). Эта опция применяется, когда известно о бессбойном функционировании ПО на автомобилях, находящихся в массовом производстве в течение предшествующего времени.

Стандарты создания ПО для атомных электростанций

Всего существует два стандарта ПО на АЭС, в зависимости от подсистем, для которых оно предназначено. Для того чтобы понять, чем различаются эти подсистемы, вначале посмотрим на их определения в стандарте верхнего уровня IEC 61226 «Nuclear

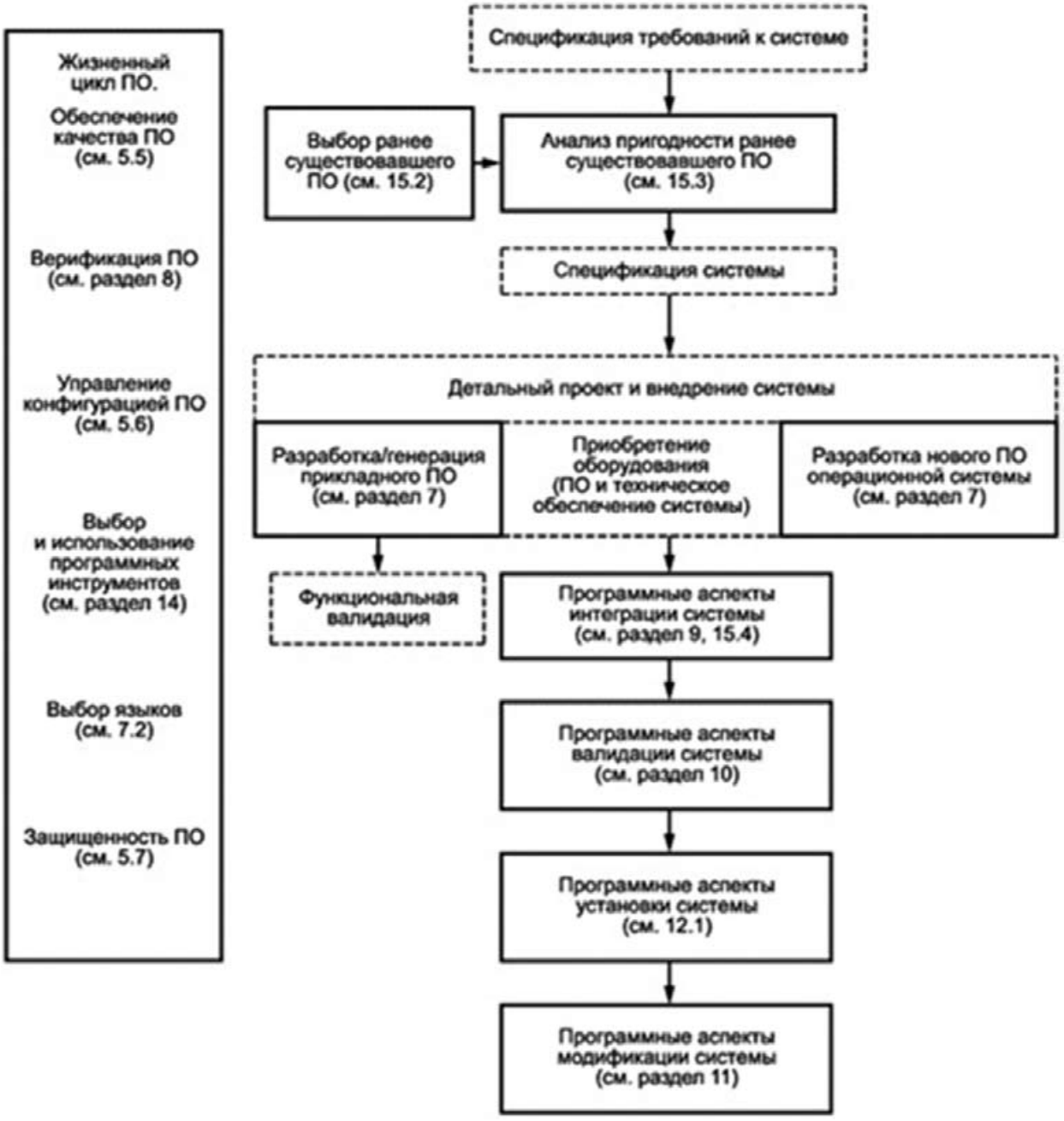

Рис. 8. Техническое и программное обеспечение для АЭС разрабатываются параллельно, исходя из общей спецификации стандарта IEC 60880

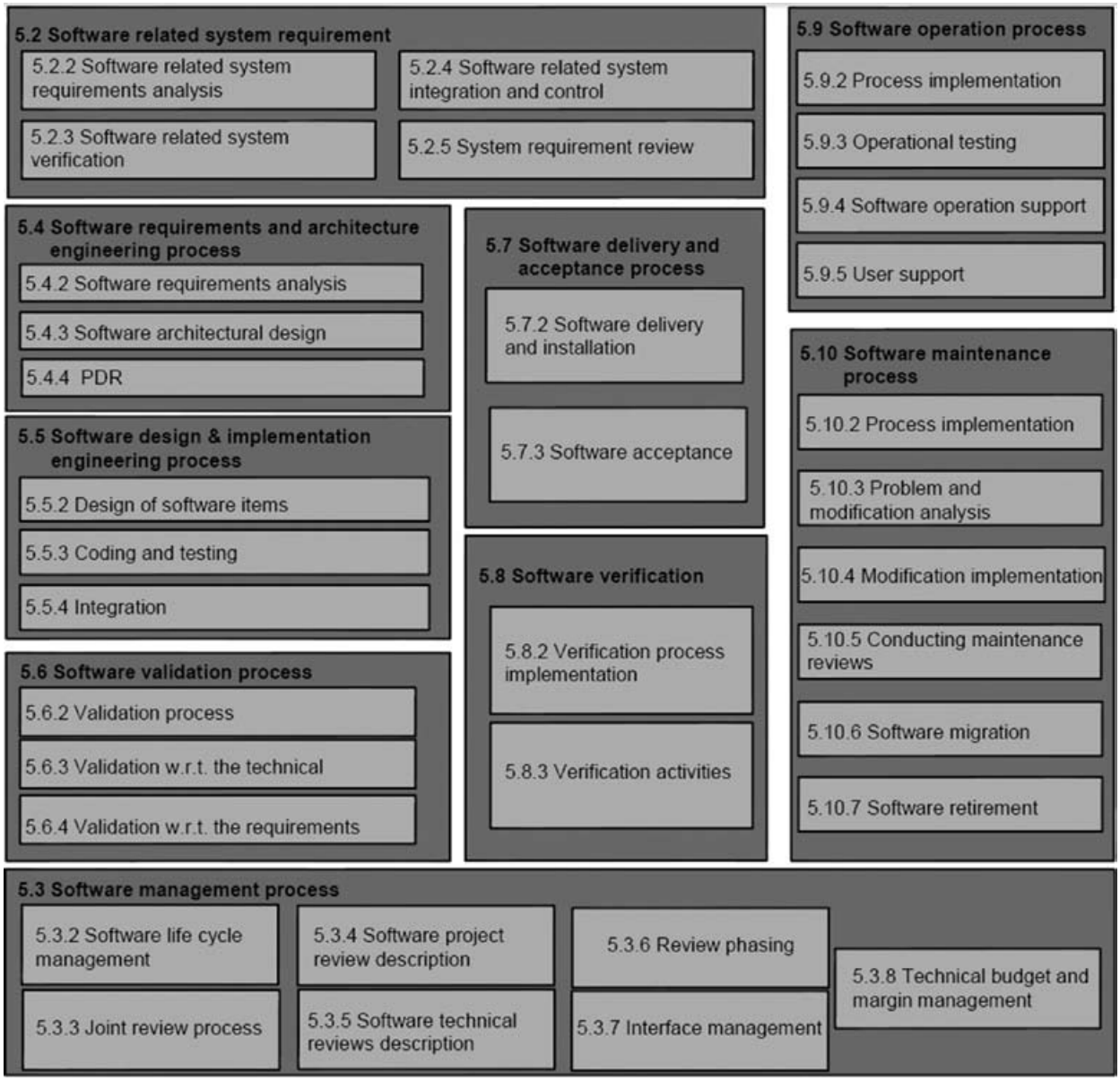

Рис. 9. Структура процессов разработки в соответствии со стандартом ECSS-E-ST-40C

power plants – Instrumentation and control important to safety – Classification of instrumentation and control functions» (аналог ГОСТ Р МЭК 61226-2011), который даёт интерпретацию требований общего стандарта IEC 61508 применительно к атомной области.

Стандарт IEC 61226 определяет три категории оборудования:

– К категории А относят функции, которые играют основную роль в достижении или поддержании безопасности АЭС с целью предотвращения развития аварии до недопустимых последствий.
– К категории В относят функции, которые играют дополнительную роль по отношению к функциям категории А в достижении или поддержании безопасности АЭС.
– К категории С относят функции, которые играют вспомогательную или косвенную роль в достижении или поддержании безопасности АЭС.
– Категория С включает в себя те функции, которые имеют определённое значение для обеспечения безопасности, но не относятся к категории А или В.

Теперь можно перейти к рассмотрению стандартов на создание ПО для АЭС.

Начнём рассмотрение со стандарта IEC 60880 «Nuclear power plants. Instrumentation and control systems important for safety. Software aspects for computer-based systems performing category A functions» (аналог – ГОСТ Р МЭК 60880 – 2011).

Область действия: каждый этап разработки ПО и документации, включая спецификацию требований, проектирование, разработку, верификацию, валидацию и эксплуатацию. К ПО стандарт относит ПО операционной системы, прикладное ПО и конфигурационные данные для настройки ПО.

Стандарт ориентирован на исключение «человеческого фактора» при взаимодействии рабочих групп и перехода с одного этапа разработки на следующий. Так, например, жёстко регламентирована даже такая часть жизненного цикла ПО, как обучение оператора.

Жизненный цикл безопасности системы включает жизненный цикл безопасности ПО. Подобно большинству стандартов класса safety-related, техническое и программное обеспечение разрабатываются параллельно, исходя из общей спецификации, а затем объединяются на этапах жизненного цикла, соответствующих интеграции и установке, как показано на рис. 8.

Верификация ПО выделяется в отдельную категорию. Помимо регламентирования технических процедур, в стандарте затрагиваются также организационные. В частности, стандарт особо подчёркивает, что группы разработки и верификации должны иметь независимое подчинение, общаться только формальным образом и по заранее определённой процедуре. Верификация разделяется на верификацию проекта разработки на этапе его инициализации (по сути, верификацию требований) и верификацию созданного кода.

Валидация ПО производится в составе системы. Стандарт подробно описывает программные аспекты установки и эксплуатации, включая установку ПО на месте эксплуатации, его адаптацию, а также защищённость ПО на месте эксплуатации (пренебрежение последним требованием, очевидно, было причиной проблем с вирусами на иранской АЭС в Бушере).

Стандарт IEC 62138 «Nuclear power plants – Instrumentation and control important for safety – Software aspects for computer- based systems performing category B or C functions» (аналог – ГОСТ Р МЭК 62138-2010) рассматривает требования к менее ответственным подсистемам АЭС. Поэтому и требования не так строги, как в IEC 60880. Например, подраздел описания верификации разработанного ПО в IEC 62138 включает описание всего четырёх пунктов, вместо 90 пунктов в IEC 60880. Похожая ситуация и с остальными подразделами. Условно можно считать, что IEC 62138 сохранил суть IEC 60880, но сильно сократил детализацию процессов жизненного цикла ПО. Это связано с тем фактом, что стоимость ПО играет не последнюю роль, и там, где не требуется максимально возможная безопасность, можно задуматься об экономии денег и времени.

Стандарты в области космической деятельности

Стандарт ECSS-E-ST-40C «Space engineering. Software» описывает требования (с различиями, описываемыми процессами адаптации), применим ко всем элементам космической экосистемы – космических кораблей, пусковых площадок и других наземных систем.

Область действия стандарта – весь цикл разработки, то есть процессы сбора требований, разработки, производства, верификации и валидации, передачи заказчику, функционирования и поддержки.

Фундаментальный принцип стандарта ECSS-E-ST-40C – это постоянное взаимодействие между поставщиком и заказчиком в течение всего процесса разработки. Центральной точкой взаимодействия между заказчиком и поставщиком является ревью. Ревью также синхронизируют процесс разработки. Такое взаимодействие наводит на мысли о возможном использовании гибких методологий разработки, но на деле их сложно применять при работе по ECSS-E-ST-40C, поскольку помимо ревью существует большое количество переходных документов, сводящих гибкость на нет.

Уровни критичности систем, частью которых является ПО, описаны в смежном стандарте, ECSS-Q-ST-80C, и описывают 4 категории критичности A–D, в зависимости от степени тяжести последствий.

Если разработка ПО включена в комплексный проект разработки, фазы разработки ПО могут иметь взаимосвязь с фазами вышестоящего проекта.

Структура процессов разработки по стандарту ECSS-E-ST-40C показана на рис. 9.

Стандарт не является строго фиксированным, при необходимости его положения могут быть адаптированы под конкретный проект, в зависимости от его сложности, количества уровней иерархии, а также уровня критичности блока, на котором работает ПО.

Заключение

Несмотря на целевое применение описанных стандартов в системах класса safety-critical, в обычных системах (программы для типового пользователя PC) использование этих стандартов не только возможно, но даже желательно. Основным препятствием к этому является повышение стоимости разработки (на 25–40% и выше, по данным IBM – http://public.dhe.ibm.com/software/dw/ru/download/DO-178b_RU-AV-01_s4.pdf), поэтому для разработки «обычного» ПО применяются менее жёсткие процессы.

Следует отметить, что стандарты на ПО класса safety-critical явно не указывают на методологию разработки, допуская MSF, RUP и даже Agile. Но при этом требования обезличенности кода и тестов, жёстко прописанных условий документирования, строго формальной передачи результатов между этапами и (например, в IEC 60880) существование прямого запрета на личный контакт разработчиков из разных групп нарушают фундаментальные принципы гибких методологий, что приводит к фактической невозможности их использования в разработке ПО класса safety-critical.