LDRA Software Technology

экспонента
ЦЕНТР ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И МОДЕЛИРОВАНИЯ

ISO 26262 a Pain in the ASIL

Экономичная сертификация программного обеспечения в соответствии с ISO 26262 (ГОСТ Р ИСО 26262)

ldra.exponenta.ru

Введение

Существует постоянно расширяющийся спектр автомобильных электрических и/или электронных (E/E/PE) систем, таких как адаптивные системы помощи водителю, антиблокировочные системы торможения, рулевое управление и подушки безопасности. Их растущие уровни интеграции и подключенности[1] обеспечивают почти столько же проблем, сколько и их распространение, причем некритические системы, такие как развлекательные системы, используют одну и ту же инфраструктуру связи, как системы управления, торможения и управления. Конечным результатом является необходимость в требовании процессов разработки функциональной безопасности, от спецификации требований, проектирования, внедрения, интеграции, проверки и до конфигурации.

ISO 26262 «Дорожные транспортные средства — функциональная безопасность» был опубликован как реакция на возросшую сложность систем автомобилей и связанных с этим рисков для общественной безопасности[2]. Подобно железнодорожным, медицинским и обрабатывающим секторам перед этим, автомобильный сектор основывал свой функциональный стандарт в значительной степени на отраслевом стандарте функциональной безопасности IEC 615082[3], который, в свою очередь, в значительной степени основывался на руководящих принципах аэрокосмических стандартов, таких как DO-178B[4]/C[5]. Конечным результатом является то, что инструменты доступные для реализации ISO 26262, были проверены ранее, чем сам стандарт.

ISO 26262: 2011 состоит из 10 частей, три из которых ориентированы на разработку продукта: уровень системы (часть 4)[6], аппаратный уровень (часть 5)[7] и уровень программного обеспечения (часть 6)[8]. Он содержит подробные отраслевые рекомендации для производства всего программного обеспечения для автомобильных систем и оборудования, независимо от того, является ли это критичным для безопасности или нет.

ISO 26262: 2011 определяет ряд уровней классификации опасностей, известных как ASIL (уровни полноты безопасности автомобиля, УПБА). ASIL варьируются от A до D, так что накладные расходы, связанные с созданием критически важной системы ASIL D (например, автоматическое торможение), больше, чем требуется для создания системы ASIL A с небольшими последствиями для безопасности (например, в развлекательной системе в автомобиле). ASIL назначается как свойства каждой отдельной функции безопасности, а не как свойство всей системы или системного компонента, и на каждую назначенную ASIL влияет частота ситуации («воздействие»), потенциальное воздействие на нее («серьезность»), и насколько легко ей можно будет управлять («управляемость»).

Безопасность в явном виде не идентифицируется как соображение в стандарте ISO 26262, возможно, отражая тот факт, что встроенные в автомобильные приложения традиционно изолированы. Статические, фиксированные функции, специфические для устройства реализации, а также методы и процессы используют этот статус. Подключение к внешнему миру резко меняет ситуацию, поскольку делает возможным удаленный доступ, не требуя никакой физической модификации систем автомобиля, наиболее известный в работе Миллера и Валасека «Удаленная эксплуатация неизмененного пассажирского автомобиля»[10].

Однако, как и для любого другого риска, так как уязвимости безопасности, угрожают надежности, ISO 26262 требует целей безопасности и требований для решения этих проблем. Короче говоря, действия, которые необходимо предпринять для решения каждой проблемы безопасности, связанной с надежностью, должны быть пропорциональны риску (и, следовательно, ASIL).

[1] Т. е. степени подключения к «интернету вещей» (IoT)

[2] https://www.iso.org/news/2012/01/Ref1499.html

[3] IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems

[4] EUROCAE ED-12B December 1992 DO-178B/C, Software Considerations in Airborne Systems and Equipment Certification

[5] RTC DO-178C,2011,Software Considerations in Airborne Systems and Equipment Certification

[6] ISO 26262-4:2011 Road vehicles -- Functional safety -- Part 4: Product development at the system level

[7] ISO 26262-5:2011 Road vehicles -- Functional safety -- Part 5: Product development at the hardware level

[8] SO 26262-6:2011 Road vehicles -- Functional safety -- Part 6: Product development at the software level

[9] Далее используется сокращение ASIL

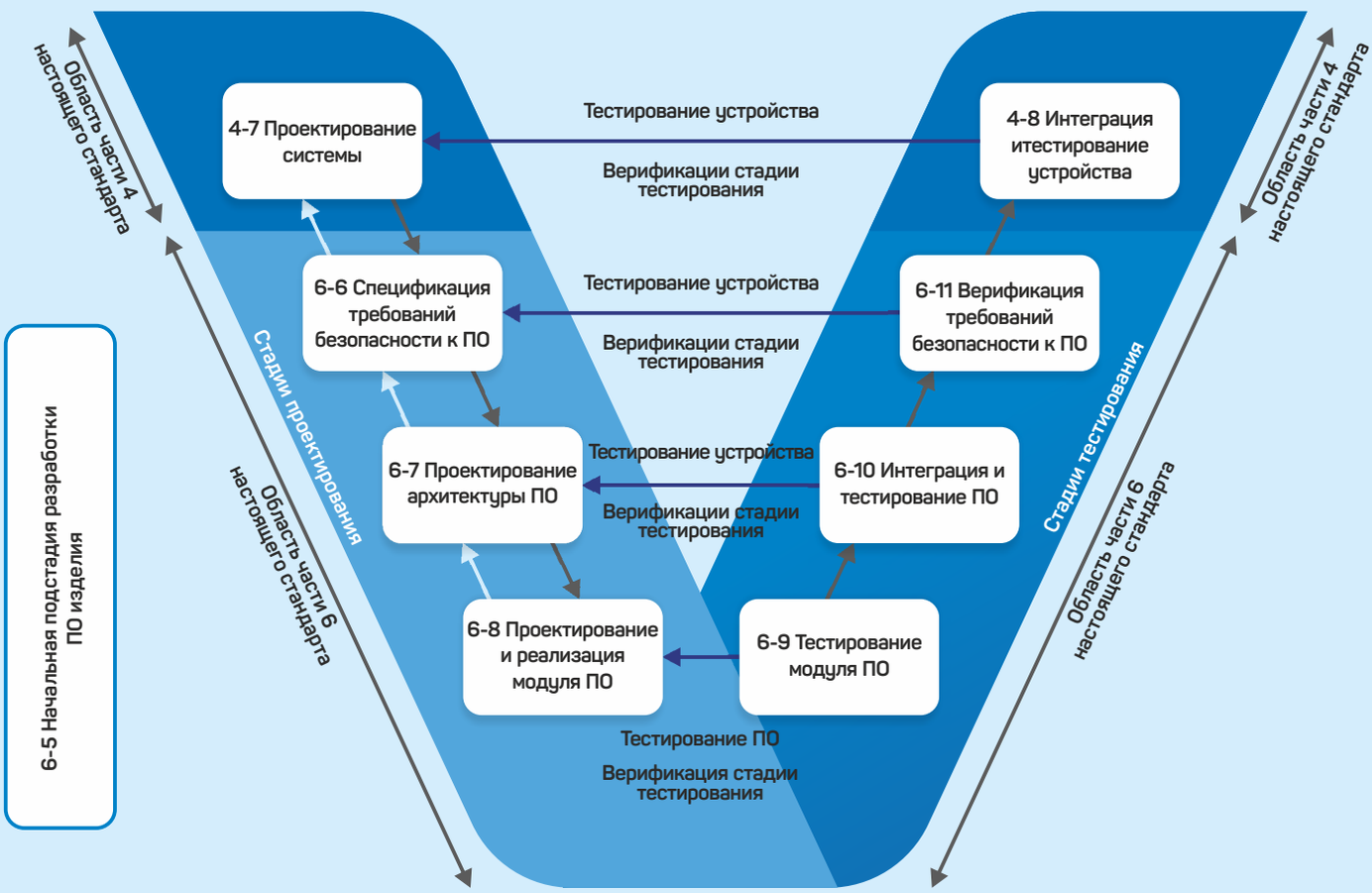
[10] http://illmatics.com/Remote%20Car%20Hacking.pdf Remote Exploitation of an Unaltered Passenger Vehicle, Dr. Charlie Miller & Chris Valasek, August 2015

Цели процесса ISO 26262

Ключевым элементом ISO 26262-4: 2011 является практика распределения технических требований надежности в спецификации проектирования системы и разработка этого проекта для получения плана интеграции и тестирования модулей.

Он применяется ко всем аспектам системы, включая программное обеспечение, с явным разделением методов разработки аппаратного и программного обеспечения, которые рассматриваются в дальнейшем на протяжении всего жизненного цикла.

Взаимосвязь между общесистемным стандартом ISO 26262-4: 2011 и конкретными подфазами программного обеспечения, найденными в ISO 26262-6: 2011, может быть представлена в V-образной модели. Каждый из этих шагов объясняется ниже.



Проектирование системы (ISO 26262-4: 2011, раздел 7)

Продукты этой общесистемной фазы проектирования могут включать в себя чертежи САПР, электронные таблицы, текстовые документы и многие другие артефакты, и в их производстве может быть задействован целый ряд инструментов. На этом этапе также рассматриваются требования к технической безопасности, которые были уточнены и распределены на аппаратное и программное обеспечение. Поддержание трассируемости между этими требованиями и продуктами последующих этапов обычно приводит к сложностям управления проектами.

Идеальный инструмент для управления требованиями может варьироваться от простой электронной таблицы или документа Microsoft Word до специализированного инструмента управления требованиями, такого как IBM Rational

DOORS Next Generation[11] или Siemens Polarion PLM[12].Выбор соответствующих инструментов поможет в поддержании двунаправленной трассируемости между этапами разработки, как обсуждается ниже.

Спецификация требований безопасности программного обеспечения (ISO 26262-6: 2011 Раздел 6)

Эта подфаза фокусируется на спецификации требований безопасности программного обеспечения для поддержки последующих этапов проектирования, учитывая любые ограничения, налагаемые аппаратным обеспечением.

Она обеспечивает интерфейс между системным уровнем ISO 26262-4: 2011 и уровнем программного обеспечения ISO 26262-6: 2011 и описывает процесс развития требований более низкого уровня, связанных с программным обеспечением. Это потребует дальнейшего использования инструментов управления требованиями, обсуждаемой ранее.

Разработка архитектуры программного обеспечения (ISO 26262-6: 2011, раздел 7)

Существует множество инструментов для создания архитектуры программного обеспечения, причем графическое представление архитектуры становится все более популярным. Подходящими инструментами являются MathWorks® Simulink®[13], IBM® Rational® Rhapsody®[14] и ANSYS® SCADE[15].

Инструменты статического анализа LDRA способствуют верификации проекта с помощью анализа потоков управления и данных кода, полученных из этих инструментов, обеспечивая графическое представление взаимосвязи между компонентами кода для сравнения с предполагаемой архитектурой.

Аналогичный подход может быть также использован для генерации графического представления унаследованного системного кода, обеспечивая возможность его дополнений, которые должны быть разработаны и проверены в соответствии с принципами ISO 26262.

Разработка модулей ПО и их интеграция (ISO 26262-4: 2011, раздел 8)

Правила кодирования. Иллюстрация является типичным примером таблицы из ISO 26262-6: 2011. В ней показаны стандарты кодирования и моделирования, которые должны быть соблюдены во время реализации, с указанием того, где соответствие может быть подтверждено набором инструментов LDRA.

Эти рекомендации объединяются, чтобы сделать полученный код более надежным, менее подверженным ошибкам, более тестируемым и/или сопровождаемым. Регулярные рассмотрения представляют собой традиционный подход к обеспечению соблюдения таких рекомендаций, и, хотя они все еще играют важную роль, автоматизация более утомительных проверок с использованием набора инструментов LDRA намного эффективнее, менее подвержена ошибкам, и повышает их повторяемость и доказуемость.

[11]http://www-03.ibm.com/software/products/en/ratidoor

[12]https://polarion.plm.automation.siemens.com/

[13]https://mathworks.com/products/simulink.html

[14]http://www-03.ibm.com/software/products/en/ratirhapfami

[15]http://www.ansys.com/products/embedded-software/ansys-scade-suite

	Методы	A	B	C	D
1a	Принудительное применение низкой сложности	++✓	++✓	++✓	++✓
1b	Использование подмножеств языка	++✓	++✓	++✓	++✓
1c	Принудительное применение строгой типизации	++✓	++✓	++✓	++✓
1d	Использование защитных методов реализации	o	+	++	++
1e	Использование установленных принципов проектирования	+✓	+✓	+✓	++✓
1f	Использование однозначного графического представления	+	++	++	++
1g	Использование руководства по стилю оформления программной документации	+✓	++✓	++✓	++✓
1h	Использование соглашений об именовании	++✓	++✓	++✓	++✓

"++" The method is highly recommended for this ASIL.
"+" The method is recommended for this ASIL.
"o" The method has no recommendation for or against its usage for this ASIL.
✓ Satis ed by the LDRA tool suite

ISO 26262-6: 2011 выделяет подмножество языка, оговариваемого стандартом кодирования MISRA в качестве примера того, что можно использовать. Существует множество различных стандартов кодирования, но вполне допустимо использовать собственный стандарт или настраивать и расширять один из уже существующих стандартов, чтобы сделать его более подходящим для конкретного приложения. Набор инструментов LDRA обеспечивает такую гибкость.

Проектирование архитектуры и разработка программного обеспечения:

Установление соответствующих стандартов проекта для кодирования, проектирования архитектуры и разработки модулей - это, очевидно, три дискретных задачи, но разработчики программного обеспечения, отвечающие за реализацию проекта, должны помнить, что эти задачи выполняются одновременно.

Касательно стандартов кодирования, то рекомендации, касающиеся архитектурного проектирования и реализации программного обеспечения, основаны на представлении о том, что они делают код более надежным, менее подверженным ошибкам, более простым в тестировании и/или более простым в сопровождении. Например, архитектурные рекомендации включают:

- Ограниченный размер программных компонентов и ограниченный размер интерфейсов рекомендуются не в последнюю очередь потому, что большие, бессвязные функции трудно читать, поддерживать и тестировать — и, следовательно, более подвержены ошибкам.

- Высокая степень связанности в каждом программном компоненте. Высокая связность является результатом тесной взаимосвязи между модулями программного обеспечения, что, в свою очередь, влияет на то, как быстро он может выполнять различные задачи, назначенные им.

Набор инструментов LDRA предоставляет показатели для обеспечения соответствия стандарту, такие как показатели сложности, по результатам анализа интерфейсов, показатели связности, оцениваемые с помощью анализа объектов данных, и показатели связности посредством анализа связей по данным и управлению.

Более обобщенно, набор инструментов LDRA может гарантировать, что надлежащие практики, требуемые ISO 26262: 2011, соблюдаются, являются ли они правилами кодирования, принципами разработки или принципами проектирования архитектуры программного обеспечения.

На практике для разработчиков, которые являются новичками в ISO 26262, роль инструмента часто развивается из механизма обнаружения нарушений, и подтверждения, что их нет.

Тестирование компонент программного обеспечения (ISO 26262-6: 2011, раздел 9) и Интеграция и тестирование программного обеспечения (ISO 26262-6: 2011 раздел 10)

Так же, как методы статического анализа (автоматизированные «инспекции» исходного кода) применимы на этапах кодирования, проектирования архитектуры и реализации блоков, методы динамического анализа (включая выполнение какой-либо части или всего кода) применимы к покомпонентному, интеграционному и системному тестированию. Модульное тестирование предназначено для того, чтобы сосредоточиться на отдельных процедурах или функциях программного обеспечения изолированно, тогда как интеграционное тестирование обеспечивает соблюдение требований безопасности и функциональных требований, когда устройства работают вместе в соответствии с архитектурой программного обеспечения.

Таблицы ISO 26262-6: 2011 содержат методы и показатели для выполнения модульных и интеграционных тестов на целевом оборудовании для обеспечения соблюдения требований безопасности и функциональности, а программные интерфейсы проверяются на уровнях модулей и интеграции. Внесение ошибки и тестирование ресурсов дополнительно подтверждают надежность и устойчивость, и, где это применимо, повторное тестирование модели и кода помогает доказать правильную интерпретацию проекта. Артефакты, связанные с этими методами, предоставляют как справочную информацию для их управления и доказательства их завершения. Они включают спецификацию программного обеспечения, процедуры верификации, план верификации и спецификацию верификации. При завершении каждой тестовой процедуры документируется ее прохождение/неудача, а соответствие требованиям проверяется соответствующим образом.

В этом примере показано, как интерфейс программного обеспечения отображается в области функций, позволяя пользователю вводить входы и ожидаемые выходные данные для формирования тестовой обвязки. Затем обвязка компилируется и выполняется на целевом вычислителе, а фактические и ожидаемые результаты сравниваются.

Модульные тесты становятся интеграционными испытаниями, когда модули вводятся как часть дерева вызовов, а не «заглушек». Точно такие же тестовые данные могут использоваться для проверки кода в обоих случаях.

Анализ граничных значений может быть автоматизирован с использованием средства «экстремального теста» в наборе инструментов LDRA для автоматического создания серии единичных тестовых векторов. Это же средство также предоставляет средство для определения граничных значений эквивалентности, таких как минимальное значение, значение ниже нижнего значения раздела, более низкое значение раздела, верхнее значение раздела и значение выше верхней границы раздела.

Если изменения станут необходимыми - возможно, в результате неудавшегося теста или в ответ на изменение требования от клиента - тогда все тестируемые модульные и интеграционные тесты должны быть запущены повторно (проверена регрессия). Набор инструментов LDRA предоставляет средства для автоматического повторного применения этих тестов для обеспечения того, что изменения не нарушали никаких установленных функций.

ISO 26262: 2011 не требует, чтобы какой-либо из этих тестов способствовал развертыванию инструментов тестирования программного обеспечения.

Однако, как и для статического анализа, инструменты динамического анализа помогают сделать процесс тестирования намного более эффективным, особенно для больших проектов.

Метрики структурного покрытия. Помимо того, что программное обеспечение функционирует правильно, динамический анализ LDRA используется для создания показателей структурного покрытия. В сочетании с покрытием требований на уровне программного обеспечения эти показатели предоставляют необходимые данные для оценки полноты тестовых векторов и для демонстрации отсутствия непредвиденных функциональных возможностей.

Метрики, рекомендованные ISO 26262: 2011 и предоставляемые набором инструментов LDRA, включают функциональное покрытие, покрытие вызовов, операторов, ветвей и модифицированное покрытие условий/решений (MC/DC)[16], средства модульного и системного тестирования могут работать совместно, - так, что, например, данные о покрытии могут быть сгенерированы для большей части исходного кода посредством динамического системного теста, а затем дополняться с помощью модульных тестов для проверки таких конструкций, которые недосягаемы во время нормальной работы системы.

Тестирование программного обеспечения и Модельно-ориентированное проектирование: набор инструментов LDRA можно интегрировать с несколькими различными инструментами разработки на основе моделей, такими как MathWorks Simulink, IBM Rational Rhapsody и ANSYS SCADE. Фаза разработки включает в себя создание модели обычным способом, причем интеграцию инструментов лучше проводить после того, как исходный код был автоматически сгенерирован из модели.

В качестве примера, используем продукты MathWorks. Маршрут совместного тестирования будет таков: Сначала выполняется разработка и проверка моделей в Simulink. Затем из Simulink генерируется код, инструментируется набором инструментов LDRA, и исполняется либо в режиме "программа-в-контуре" (SIL)[17], либо в режиме "процессор-в-контуре" (PIL)[18]. Отчеты о структурном покрытии представлены на уровне исходного кода Simulink и набором инструментов LDRA.

В дополнение к тестированию «спина к спине» такая интеграция обеспечивает возможности для обеспечения того, чтобы сгенерированный исходный код соответствовал стандарту кодирования, например, MISRA AC ACG14, было выполнено динамическое тестирование на уровне исходного кода, проверено соответствие требованиям и все написанные вручную дополнения к автоматически сгенерированному коду были протестированы.

Двунаправленная трассируемость (ISO 26262-4: 2011 и ISO 26262-6: 2011)

Двунаправленная трассируемость является фундаментальной концепцией на протяжении всего IS026262: 2011, причем каждый этап разработки должен точно отражать предыдущий. Теоретически, если придерживаться точной последовательности V-модели, требования никогда не изменятся, и тесты никогда не вызовут проблемы. Но жизнь не такая.

Что произойдет, если произойдет изменение кода, как ответ на неудачный тест интеграции, не пройденный, возможно, из-за несоответствия требований или наличия ошибки кодирования. Какие другие модули программного обеспечения зависят от модифицированного кода?

Такие сценарии могут быстро привести к ситуациям, когда пропадает трасссируемость между артефактами разработки программного обеспечения. Разумеется, можно поддерживать трассируемость вручную, но автоматизация поддержания трассируемости более предпочтительна.

Проект программного обеспечения может принимать различные формы - возможно, в форме документа с подробным описанием на естественном языке или, возможно, на основе моделей. В любом случае, все элементы проекта должны быть двунаправленно прослеживаться как для требований безопасности программного обеспечения, так и для архитектуры программного обеспечения. Затем программные компоненты должны быть реализованы в соответствии с указанными требованиями, а затем трассироваться на спецификации проекта.

[16]См. https://youtu.be/wsazZB3xGQs — Инструменты LDRA для верификации ПО: Сбор покрытия MC/DC

[17]Исполнение на хосте

[18]Исполнение на целевом вычислителе

Набор инструментов LDRA можно использовать для установления политики трассируемости между требованиями и тестовыми векторами из разных областей, что позволяет оценивать покрытие тестами. Влияние непройденных тестов можно оценить и устранить, как и их влияние на изменение требований и пробелы в покрытии требованиями. Также артефакты, такие как матрицы трассируемости, могут автоматически генерироваться, чтобы представить доказательства соответствия ISO 26262: 2011.

На практике, первоначальное структурное покрытие обычно вычисляется как результат выполнения функциональных тестов на инструментированном коде, оставляя неисполненные части кода, которые требуют дальнейшего анализа. Это, в конечном итоге, приводит к добавлению или модификации тестовых векторов, изменениям требований и/или удалению мертвого кода. Как правило, итеративная последовательность рассмотрений, исправлений и анализа обеспечивает соответствие спецификациям.

Уверенность в использовании инструментального программного обеспечения (ISO 26262-8: 2011, раздел 11)

Этот вспомогательный процесс определяет механизм, обеспечивающий доказательство того, что набор инструментов применим для работы. Необходимый уровень уверенности в программном инструменте зависит от обстоятельств его развертывания, как с точки зрения возможности того, что неисправный программный инструмент может вводить или не обнаруживать ошибки в связанном с безопасностью компоненте, а также вероятность того, что такие ошибки могут быть предотвращены или обнаружены.

Набор инструментов LDRA был квалифицирован для использования в системах, совместимых с ISO 26262, до ASIL D, что приводит к значительному снижению накладных расходов пользователей в предоставлении доказательств этой уверенности.

В зависимости от оценки пользователем своего приложения, набору инструментов LDRA будет присвоен «Уровень доверия инструмента» либо TCL1, либо TCL2.Во всех случаях, кроме случаев, когда набору инструментов назначается TCL2, а продукт обозначается как ASIL D, наличие сертификата TUV достаточно для установления достаточной уверенности в инструменте. В противном случае, инструмент должен быть подвергнут процессу валидации, чтобы показать, что инструмент способен анализировать эталонные образцы программного обеспечения в соответствующей целевой среде.

LDRA предоставляет пакет поддержки квалификации инструмента (Tool Qualification Support Package, TQSP) с такими эталонами.

+7 (495) 009-65-85
info@exponenta.ru
ldra.exponenta.ru