Глава 10: Качество программного обеспечения

Из SWEBOK

Содержание

АКРОНИМЫ

CMMI	Модель зрелости интеграции
CoSQ	CoSQ Стоимость качества программного обеспечения
COTC	Готовое коммерческое программное обеспечение
FMEA	Вид отказа и анализ последствий
ЗСТ	Анализ дерева отказов
PDCA	Планируй-Делай-Проверяй-Действуй
ПДСА	Планируй-делай-учись-действуй
КВД	Развертывание функции качества
СПИ	Улучшение программного процесса
SQA	Гарантия качества программного обеспечения
SQC	Контроль качества программного обеспечения
кв.м.	Управление качеством программного обеспечения
УК	Полное управление качеством
ВиВ	Верификация и валидация

ВВЕДЕНИЕ

Что такое качество программного обеспечения и почему так важно, чтобы оно было включено во многие области знаний (KA) руководства *SWEBOK* ?

Одна из причин заключается в том, что термин « *качество программного обеспечения* » перегружен. Качество программного обеспечения может относиться: к желательным характеристикам программных продуктов, в той степени, в которой конкретный программный продукт обладает этими характеристиками, а также к процессам, инструментам и методам, используемым для достижения этих характеристик. На протяжении многих лет авторы и организации по-разному определяли термин «качество». Для Фила Кросби это было «соответствие требованиям» [1]. Уоттс Хамфри называет это «достижением отличного уровня «пригодности к использованию» [2]. Тем временем IBM придумала фразу «качество, определяемое рынком», где «клиент является окончательным арбитром» [3*, стр. 8].

В последнее время качество программного обеспечения определяется как «способность программного продукта удовлетворять заявленные и предполагаемые потребности при определенных условиях» [4] и как «степень, в которой программный продукт соответствует установленным требованиям; однако качество зависит от степени, в которой эти установленные требования точно отражают потребности, желания и ожидания заинтересованных сторон» [5]. Оба определения включают предпосылку

соответствия требованиям. Ни один из них не относится к типам требований (например, функциональные, надежность, производительность, безотказность или любые другие характеристики). Важно, однако, что эти определения подчеркивают, что качество зависит от требований.

Эти определения также иллюстрируют еще одну причину преобладания качества программного обеспечения в этом *Руководстве* : частую двусмысленность *качества программного обеспечения* в сравнении с требованиями к качеству *программного обеспечения* («- *ilities* » является распространенным сокращением). Требования к качеству программного обеспечения на самом деле являются атрибутами (или ограничениями) функциональных требований (того, что делает система). Требования к программному обеспечению могут также указывать использование ресурсов, протокол связи или многие другие характеристики. Этот KA пытается внести ясность, используя *качество программного обеспечения* в самом широком смысле из приведенных выше определений и используя *требования к качеству программного обеспечения* .как ограничения функциональных требований. Качество программного обеспечения достигается соответствием всем требованиям, независимо от того, какая характеристика указана или как требования сгруппированы или названы.

Качество программного обеспечения также учитывается во многих KA SWEBOK, поскольку оно является основным параметром усилий по разработке программного обеспечения. Для всех инженерных продуктов основная цель состоит в том, чтобы обеспечить максимальную ценность для заинтересованных сторон, уравновешивая ограничения стоимости и графика разработки; это иногда характеризуется как «пригодность к использованию». Ценность для заинтересованных сторон выражается в требованиях. Для программных продуктов заинтересованные стороны могут оценивать цену (сколько они платят за продукт), время выполнения (насколько быстро они получают продукт) и качество программного обеспечения.

В этом KA рассматриваются определения и дается обзор практик, инструментов и методов определения качества программного обеспечения и оценки состояния качества программного обеспечения во время разработки, обслуживания и развертывания. Приведенные ссылки содержат дополнительные сведения.

РАЗБИВКА ТЕМ ПО КАЧЕСТВУ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разбивка тем для KA качества программного обеспечения представлена на рисунке 10.1.

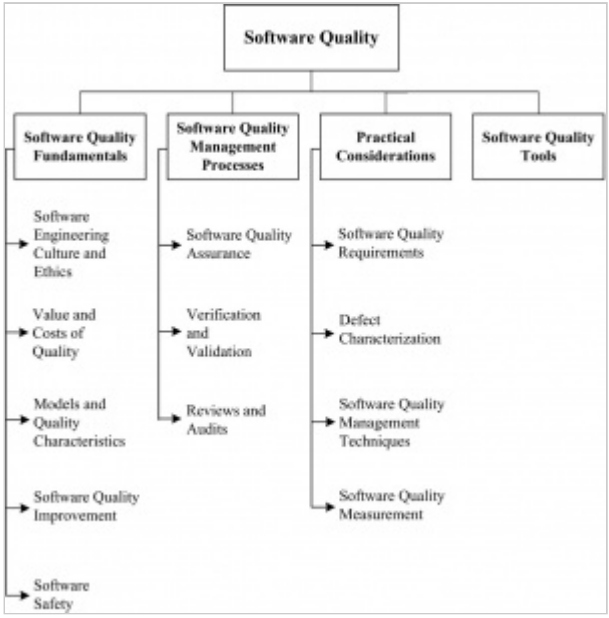

Рисунок 10.1: Разбивка тем для KA качества программного обеспечения

1 Основы качества программного обеспечения

Достижение соглашения о том, что представляет собой качество для всех заинтересованных сторон, и четкое информирование об этом согласии разработчиков программного обеспечения требуют формального определения и обсуждения многих аспектов качества.

Инженер-программист должен понимать концепции качества, характеристики, значения и их применение к разрабатываемому или поддерживаемому программному обеспечению. Важной концепцией является то, что требования к программному обеспечению определяют требуемые атрибуты качества программного обеспечения. Требования к программному обеспечению влияют на методы измерения и критерии приемлемости для оценки степени, в которой программное обеспечение и связанная с ним документация достигают желаемых уровней качества.

1.1 Культура и этика разработки программного обеспечения

[3 , c1s4] [6 , c2s3.5]

Ожидается, что разработчики программного обеспечения разделят приверженность качеству программного обеспечения как часть своей культуры. Здоровая культура разработки программного обеспечения включает в себя множество характеристик, в том числе понимание того, что компромисс между стоимостью, графиком и качеством является основным условием разработки любого продукта. Строгая этика разработки программного обеспечения предполагает, что инженеры точно сообщают информацию, условия и результаты, связанные с качеством.

Этика также играет важную роль в качестве программного обеспечения, культуре и отношении разработчиков программного обеспечения. Компьютерное общество IEEE и ACM разработали кодекс этики и профессиональной практики (см. Кодексы этики и профессионального поведения в профессиональной практике разработки программного обеспечения KA).

1.2 Стоимость и затраты на качество

[7 , c17, c22]

Определить, а затем добиться качества программного обеспечения непросто. Характеристики качества могут требоваться или не требоваться, или они могут требоваться в большей или меньшей степени, и между ними может быть достигнут компромисс. Чтобы помочь определить уровень качества программного обеспечения, т. е. достижение ценности для заинтересованных сторон, в этом разделе представлены затраты на качество программного обеспечения (CoSQ): набор показателей, полученных в результате экономической оценки процессов разработки и сопровождения качества программного обеспечения. Измерения CoSQ являются примерами измерений процесса, которые можно использовать для определения характеристик продукта.

Предпосылка, лежащая в основе CoSQ, заключается в том, что уровень качества программного продукта можно вывести из стоимости действий, связанных с устранением последствий низкого качества. Низкое качество означает, что программный продукт не полностью «удовлетворяет заявленные и подразумеваемые потребности» или «установленные требования». Существует четыре категории затрат на качество: предотвращение, оценка, внутренний сбой и внешний сбой.

Затраты на профилактику включают в себя инвестиции в усилия по улучшению процессов разработки программного обеспечения, инфраструктуру качества, инструменты обеспечения качества, обучение, аудиты и анализы со стороны руководства. Эти затраты обычно не относятся к конкретному проекту; они охватывают всю организацию. Затраты на оценку возникают в результате деятельности по проекту, в ходе которой обнаруживаются дефекты. Эти действия по оценке можно разделить на затраты на обзоры (проектирование, коллегиальные) и затраты на тестирование (единичное тестирование программного обеспечения, интеграция программного обеспечения, тестирование на системном уровне, приемочное тестирование); расходы на оценку будут распространяться на субподрядных поставщиков программного обеспечения. Затраты на внутренние сбои — это затраты, понесенные для устранения дефектов, обнаруженных в ходе оценочной деятельности и обнаруженных до поставки программного продукта заказчику.

Инженеры-программисты должны иметь возможность использовать методы CoSQ для определения уровней качества программного обеспечения, а также должны быть в состоянии представить альтернативы качества и их стоимость, чтобы можно было найти компромисс между стоимостью, графиком и доставкой ценности для заинтересованных сторон.

1.3 Модели и характеристики качества

[3 , c24s1] [7 , c2s4] [8 , c17]

Терминология для характеристик качества программного обеспечения отличается от одной таксономии (или модели качества программного обеспечения) к другой, каждая модель может иметь разное количество иерархических уровней и разное общее количество характеристик. Различные авторы разработали модели характеристик или атрибутов качества программного обеспечения, которые могут быть полезны для обсуждения, планирования и оценки качества программных продуктов. ISO/IEC 25010: 2011 [4] определяет качество продукции и качество при использовании как две взаимосвязанные модели качества. Приложение B в Руководстве SWEBOK содержит список применимых стандартов для каждого KA. Стандарты для этого KA охватывают различные способы характеристики качества программного обеспечения.

1.3.1 Качество программного процесса

Управление качеством программного обеспечения и качество процесса разработки программного обеспечения имеют прямое отношение к качеству программного продукта.

Модели и критерии, которые оценивают возможности организаций, занимающихся разработкой программного обеспечения, в первую очередь являются соображениями организации и управления проектами и, как таковые, рассматриваются в КА «Управление программной инженерией» и «Процесс разработки программного обеспечения».

Невозможно полностью отличить качество процесса от качества продукта, потому что результаты процесса включают продукты. Определить, способен ли процесс стабильно производить продукцию желаемого качества, непросто.

Процесс разработки программного обеспечения, обсуждаемый в разделе Процесс разработки программного обеспечения КА, влияет на характеристики качества программных продуктов, которые, в свою очередь, влияют на качество, воспринимаемое заинтересованными сторонами.

1.3.2 Качество программного продукта

Инженер-программист, прежде всего, должен определить истинное назначение программного обеспечения. В этом отношении требования заинтересованных сторон имеют первостепенное значение, и они включают требования к качеству в дополнение к функциональным требованиям. Таким образом, разработчики программного обеспечения несут ответственность за выявление требований к качеству, которые могут быть неявными с самого начала, и понимание их важности, а также уровня сложности их достижения. Все процессы разработки программного обеспечения (например, выявление требований, проектирование, конструирование, построение, проверка, улучшение качества) разрабатываются с учетом этих требований к качеству и могут повлечь за собой дополнительные затраты на разработку, если важны такие атрибуты, как безопасность, защищенность и надежность. Дополнительные затраты на разработку помогают гарантировать, что полученное качество может быть компенсировано ожидаемыми преимуществами.

Термин «рабочий продукт» означает любой артефакт, являющийся результатом процесса, используемого для создания конечного программного продукта. Примеры рабочего продукта включают спецификацию системы/подсистемы, спецификацию требований к программному обеспечению для программного компонента системы, описание проекта программного обеспечения, исходный код, документацию по тестированию программного обеспечения или отчеты. В то время как некоторые подходы к качеству описываются с точки зрения окончательной производительности программного обеспечения и системы, надежная инженерная практика требует, чтобы промежуточные рабочие продукты, имеющие отношение к качеству, оценивались на протяжении всего процесса разработки программного обеспечения.

1.4 Повышение качества программного обеспечения

[3 , c1c4] [9 , c24] [10 , c11c2.4]

Качество программных продуктов может быть улучшено с помощью превентивных процессов или итеративного процесса постоянного улучшения, который требует управленческого контроля, координации и обратной связи от многих параллельных процессов: (1) процессов жизненного цикла программного обеспечения, (2) процесса обнаружения ошибок/ обнаружение, устранение и предотвращение дефектов и (3) процесс улучшения качества.

Теория и концепции, лежащие в основе повышения качества, такие как повышение качества за счет предотвращения и раннего обнаружения дефектов, непрерывное улучшение и ориентация на заинтересованные стороны, имеют отношение к разработке программного обеспечения. Эти концепции основаны на работе экспертов по качеству, которые заявили, что качество продукта напрямую связано с качеством процесса, используемого для его создания. Такие подходы, как цикл улучшения Деминга «Планируй-Делай-Проверяй-Действуй» (PDCA), эволюционная поставка, кайдзен и развертывание функции качества (QFD), предлагают методы для определения целей в области качества и определения того, выполняются ли они. Другой метод — IDEAL Института программной инженерии [7*]. *Руководство SWEBOK* теперь признает управление качеством важной дисциплиной.

Спонсорство руководства поддерживает оценку процессов и продуктов, а также полученные результаты. Затем разрабатывается программа улучшения, определяющая подробные действия и проекты по улучшению, которые необходимо выполнить в разумные сроки. Поддержка управления подразумевает, что каждый проект улучшения имеет достаточно ресурсов для достижения поставленной перед ним цели. Спонсорство руководства часто запрашивается путем реализации активной коммуникационной деятельности.

1.5 Безопасность программного обеспечения

[9 , c11c3]

Системы, критически важные для безопасности, — это системы, в которых сбой системы может причинить вред жизни человека, другим живым существам, физическим структурам или окружающей среде. Программное обеспечение в этих системах критично с точки зрения безопасности. Растет число приложений критически важного для безопасности программного обеспечения во все большем числе отраслей. Примеры систем с критически важным для безопасности программным обеспечением включают системы общественного транспорта, химические заводы и медицинские устройства. Сбой программного обеспечения в этих системах может иметь катастрофические последствия. Существуют отраслевые стандарты, такие как DO-178C [11], а также новые процессы, инструменты и методы для разработки критически важного для безопасности программного обеспечения. Целью этих стандартов, инструментов и методов является снижение риска внесения ошибок в программное обеспечение и, таким образом, повышение надежности программного обеспечения.

Критичное для безопасности программное обеспечение можно разделить на прямое и косвенное. Прямой — это программное обеспечение, встроенное в критически важную для безопасности систему, такую как компьютер управления полетом самолета. Непрямой включает в себя программные приложения, используемые для разработки критического для безопасности программного обеспечения. Косвенное программное обеспечение включено в среды разработки программного обеспечения и среды тестирования программного обеспечения.

Три дополняющих друг друга метода снижения риска отказа: предотвращение, обнаружение и устранение, а также ограничение ущерба. Эти методы влияют на функциональные требования к программному обеспечению, требования к производительности программного обеспечения и процессы разработки. Повышение уровня риска подразумевает повышение уровня обеспечения качества программного обеспечения и методов контроля, таких как инспекции. Более высокие уровни риска могут потребовать более тщательной проверки требований, дизайна и кода или использования более формальных аналитических методов. Еще один метод управления рисками программного обеспечения и контроля над ними — создание гарантийных случаев. Случай обеспечения уверенности — это обоснованный, проверяемый артефакт, созданный для подтверждения утверждения, что его требование или требования удовлетворены. Он содержит следующее и их взаимосвязи: одно или несколько утверждений о свойствах; аргументы, которые логически связывают доказательства и любые предположения с утверждениями; и совокупность доказательств и предположений, поддерживающих эти аргументы [12].

2 Процессы управления качеством программного обеспечения

Управление качеством программного обеспечения — это совокупность всех процессов, которые гарантируют, что программные продукты, услуги и реализации процессов жизненного цикла соответствуют целям организации в области качества программного обеспечения и удовлетворяют потребности заинтересованных сторон [13, 14]. SQM определяет процессы, владельцев процессов, требования к процессам, измерения процессов и их результатов, а также каналы обратной связи на протяжении всего жизненного цикла программного обеспечения.

SQM включает четыре подкатегории: планирование качества программного обеспечения, обеспечение качества программного обеспечения (SQA), контроль качества программного обеспечения (SQC) и улучшение процесса разработки программного обеспечения (SPI). Планирование качества программного обеспечения включает определение стандартов качества, которые следует использовать, определение конкретных целей в области качества, а также оценку усилий и графика мероприятий по обеспечению качества программного обеспечения. В некоторых случаях планирование качества программного обеспечения также включает определение используемых процессов качества программного обеспечения. Действия SQA определяют и оценивают адекватность программных

процессов, чтобы предоставить доказательства, подтверждающие уверенность в том, что программные процессы подходят и производят программные продукты подходящего качества для их предполагаемых целей [5]. Действия SQC исследуют определенные артефакты проекта (документы и исполняемые файлы), чтобы определить, соответствуют ли они стандартам, установленным для проекта (включая требования, ограничения, проекты, контракты и планы). SQC оценивает промежуточные продукты, а также конечные продукты.

Четвертая категория SQM, связанная с улучшением, имеет различные названия в индустрии программного обеспечения, включая SPI, улучшение качества программного обеспечения и корректирующие и предупреждающие действия в отношении программного обеспечения. Действия в этой категории направлены на повышение эффективности процессов, производительности и других характеристик с конечной целью повышения качества программного обеспечения. Хотя SPI может быть включен в любую из первых трех категорий, все большее число организаций выделяют SPI в отдельную категорию, которая может охватывать множество проектов (см. Процесс разработки программного обеспечения KA).

Процессы обеспечения качества программного обеспечения состоят из задач и методов, показывающих, как реализуются планы программного обеспечения (например, планы управления программным обеспечением, разработки, управления качеством или управления конфигурацией) и насколько хорошо промежуточные и конечные продукты соответствуют заданным требованиям. Результаты выполнения этих задач собираются в отчеты для управления до того, как будут предприняты корректирующие действия. Задача управления процессом SQM состоит в обеспечении точности результатов этих отчетов.

Управление рисками также может играть важную роль в предоставлении качественного программного обеспечения. Включение дисциплинированного анализа рисков и методов управления в процессы жизненного цикла программного обеспечения может помочь улучшить качество продукта (см. KA по управлению программной инженерией для получения соответствующих материалов по управлению рисками).

2.1 Обеспечение качества программного обеспечения

[7 , c4–c6, c11, c12, c26–27]

Чтобы подавить широко распространенное заблуждение, обеспечение качества программного обеспечения не является тестированием. Обеспечение качества программного обеспечения (SQA) — это набор действий, которые определяют и оценивают адекватность программных процессов для предоставления доказательств, подтверждающих уверенность в том, что программные процессы являются подходящими и производят программные продукты надлежащего качества для их предполагаемых целей. Ключевым атрибутом SQA является объективность функции SQA по отношению к проекту. Функция SQA также может быть организационно независимой от проекта; то есть свободным от технического, управленческого и финансового давления со стороны проекта [5]. SQA имеет два аспекта: обеспечение качества продукта и обеспечение качества процесса, которые объясняются в разделе 2.3.

План качества программного обеспечения (в некоторых отраслях промышленности он называется планом обеспечения качества программного обеспечения) определяет действия и задачи, используемые для обеспечения того, чтобы программное обеспечение, разработанное для конкретного продукта, удовлетворяло установленным требованиям проекта и потребностям пользователей в пределах стоимости проекта и ограничений графика, а также соответствовало с проектными рисками. SQAP прежде всего обеспечивает четкое определение и понимание целевых показателей качества.

Мероприятия и задачи плана SQA по обеспечению качества указаны с их затратами, требованиями к ресурсам, целями и графиком по отношению к связанным целям в планах управления разработкой программного обеспечения, разработки программного обеспечения и сопровождения программного обеспечения. План SQA должен соответствовать плану управления конфигурацией программного обеспечения (см. KA управления конфигурацией программного обеспечения). План SQA определяет документы, стандарты, методы и соглашения, регулирующие проект, а также то, как эти элементы проверяются и отслеживаются для обеспечения адекватности и соответствия. План SQA также определяет меры; статистические методы; процедуры сообщения о проблемах и корректирующих действий; ресурсы, такие как инструменты, методы и методологии; безопасность физических носителей; подготовка; отчетность и документация SQA. Более того, план SQA касается действий по обеспечению

качества программного обеспечения любого другого типа деятельности, описанного в планах программного обеспечения, таких как закупка программного обеспечения поставщика для проекта, установка коммерческого готового программного обеспечения (COTS) и обслуживание после поставки программного обеспечения. . Он также может содержать критерии приемки, а также действия по отчетности и управлению, которые имеют решающее значение для качества программного обеспечения.

2.2 Проверка и проверка

[9 , c2s2.3, c8, c15s1.1, c21s3.3]

Как указано в [15],

- Цель V&V — помочь организации-разработчику встроить качество в систему на протяжении всего жизненного цикла. Процессы верификации и верификации обеспечивают объективную оценку продуктов и процессов на протяжении всего жизненного цикла. Эта оценка показывает, являются ли требования правильными, полными, точными, непротиворечивыми и проверяемыми. Процессы верификации и верификации определяют, будут ли продукты разработки данной деятельности

соответствует требованиям этой деятельности, а также соответствует ли продукт предполагаемому использованию и потребностям пользователя.

Верификация — это попытка убедиться, что продукт построен правильно, в том смысле, что выходные продукты деятельности соответствуют спецификациям, наложенным на них в предыдущих действиях. Валидация — это попытка убедиться, что создан правильный продукт, т. е. продукт выполняет свое конкретное предназначение. Как процесс проверки, так и процесс проверки начинаются на ранней стадии разработки или сопровождения. Они обеспечивают изучение ключевых характеристик продукта по отношению как к непосредственному предшественнику продукта, так и к спецификациям, которым необходимо соответствовать.

Цель планирования верификации и верификации состоит в том, чтобы обеспечить четкое распределение каждого ресурса, роли и ответственности. В итоговых документах плана верификации и верификации описываются различные ресурсы, их роли и действия, а также используемые методы и инструменты. Понимание различных целей каждой деятельности по верификации и верификации помогает в тщательном планировании методов и ресурсов, необходимых для достижения их целей. В плане также рассматриваются вопросы управления, коммуникации, политик и процедур деятельности по верификации и верификации и их взаимодействия, а также требования к сообщениям о дефектах и документации.

2.3 Обзоры и аудиты

[9 , c24s3] [16]

Проверки и процессы аудита в широком смысле определяются как статические — это означает, что никакие программы или модели не выполняются — проверка артефактов разработки программного обеспечения в отношении стандартов, которые были установлены организацией или проектом для этих артефактов. Различные виды обзоров и аудитов различаются по их цели, уровням независимости, инструментам и методам, ролям и предмету деятельности. Аудиты обеспечения качества продукции и процессов обычно проводятся специалистами по обеспечению качества программного обеспечения (SQA), независимыми от групп разработчиков. Управленческие обзоры проводятся организационным или проектным руководством. Инженерно-технический персонал проводит технические проверки.

- Анализы со стороны руководства оценивают фактические результаты проекта по сравнению с планами.
- Технические обзоры (включая инспекции, пошаговые и кабинетные проверки) исследуют результаты инженерных работ.
- Аудит обеспечения процесса. Мероприятия по обеспечению качества процесса SQA гарантируют, что процессы, используемые для разработки, установки, эксплуатации и обслуживания программного обеспечения, соответствуют контрактам, соответствуют любым установленным законам, правилам и положениям и являются адекватными, эффективными и действенными для их предполагаемой цели [5].

- Аудит гарантии качества продукции. Деятельность SQA по обеспечению качества продукции гарантирует предоставление доказательств того, что программные продукты и сопутствующая документация указаны в контрактах и соответствуют им; и обеспечить выявление и устранение несоответствий [5].

2.3.1 Обзоры руководства

Как указано в [16*],

- Цель анализа со стороны руководства состоит в том, чтобы отслеживать прогресс, определять статус планов и графиков и оценивать эффективность.

процессов, инструментов и методов управления. Анализы со стороны руководства сравнивают фактические результаты проекта с планами для определения статуса проектов или работ по техническому обслуживанию. Основными параметрами анализа со стороны руководства являются стоимость проекта, график, объем и качество. Анализы со стороны руководства оценивают решения о корректирующих действиях, изменениях в распределении ресурсов или изменениях содержания проекта.

Входные данные для анализа со стороны руководства могут включать в себя аудиторские отчеты, отчеты о ходе работы, отчеты о верификации и проверке и планы многих типов, включая, среди прочего, управление рисками, управление проектами, управление конфигурацией программного обеспечения, безопасность программного обеспечения и оценку рисков. (См. соответствующий материал в документах «Управление разработкой программного обеспечения» и «Управление конфигурацией программного обеспечения».)

2.3.2 Технические обзоры

Как указано в [16*],

- Целью технического обзора является оценка программного продукта группой квалифицированных специалистов для определения его пригодности для использования по назначению и выявления несоответствий спецификациям и стандартам. Он предоставляет руководству доказательства для подтверждения технического состояния проекта.

Хотя любой рабочий продукт может быть рассмотрен, технические обзоры выполняются для основных рабочих продуктов разработки программного обеспечения требований к программному обеспечению и дизайна программного обеспечения.

Цель, роли, виды деятельности и, что наиболее важно, уровень формальности отличают различные типы технических обзоров. Инспекции являются наиболее формальными, обходы менее формальными, а парные обзоры или кабинетные проверки — наименее формальными.

Примеры конкретных ролей включают лицо, принимающее решения (т. е. руководитель программного обеспечения), руководитель проверки, регистратор и проверяющие (технический персонал, который исследует рабочие продукты). Обзоры также различаются по тому, включены ли в процесс встречи (личные или электронные). В некоторых методах проверки проверяющие в одиночку проверяют рабочие продукты и отправляют свои результаты обратно координатору. В других методах контролеры работают совместно на встречах. Технический анализ может потребовать наличия обязательных входных данных для продолжения:

- Заявление о целях
- Конкретный программный продукт
- Конкретный план управления проектом
- Список проблем, связанных с этим продуктом
- Процедура технического обзора.

Команда следует задокументированной процедуре проверки. Техническая проверка завершается после завершения всех действий, перечисленных в проверке.

Технические обзоры исходного кода могут включать в себя широкий спектр вопросов, таких как анализ алгоритмов, использование критических компьютерных ресурсов, соблюдение стандартов кодирования, структура и организация кода для обеспечения тестируемости, а также соображения безопасности.

Обратите внимание, что технические обзоры исходного кода или моделей проектирования, таких как UML, также называются статическим анализом (см. раздел 3, Практические соображения).

2.3.3 Проверки

«Целью инспекции является обнаружение и идентификация аномалий программного продукта» [16*]. Вот некоторые важные отличия инспекций от других видов технических обзоров:

- 1. Правила. Инспекции основаны на проверке рабочего продукта по отношению к определенному набору критериев, установленных организацией. Наборы правил могут быть определены для различных типов рабочих продуктов (например, правила для требований, описания архитектуры, исходный код).
- 2. Отбор проб. Вместо того, чтобы пытаться проверить каждое слово и цифру в документе, процесс проверки позволяет проверяющим оценивать определенные подмножества (выборки) просматриваемых документов.
- 3. Сверстник. В проверке не участвуют лица, занимающие руководящие должности над членами инспекционной группы. Это

ключевое различие между рецензированием коллег и рецензированием со стороны руководства.

- 4. Светодиод. Инспекционные собрания ведет беспристрастный модератор, обученный методам инспекции.
- 5. Встреча. Процесс инспекции включает встречи (личные или электронные), проводимые модератором в соответствии с формальной процедурой, на которых члены инспекционной группы сообщают о выявленных ими аномалиях и других вопросах.

В проверках программного обеспечения всегда участвует автор промежуточного или конечного продукта; других отзывов может и не быть. В состав проверок входят также руководитель проверки, регистратор, считыватель и несколько (от двух до пяти) проверяющих (инспекторов). Члены инспекционной группы могут обладать разным опытом, например, знанием предметной области, знанием методов проектирования программного обеспечения или знанием языков программирования. Контроль обычно проводится на одном относительно небольшом участке продукта (образцах). Каждый член команды изучает программный продукт и другие входные данные для проверки перед обзорным совещанием, возможно, применяя аналитический метод (см. раздел 3.3.3) к небольшому разделу продукта или ко всему продукту, сосредоточив внимание только на одном аспекте — например, интерфейсы. Во время осмотра, модератор ведет сеанс и проверяет, все ли подготовились к проверке и проводит сеанс. Регистратор осмотра фиксирует обнаруженные аномалии. Набор правил с критериями и вопросами, относящимися к интересующим вопросам, является распространенным инструментом, используемым в проверках. Результирующий список часто классифицирует аномалии (см. раздел 3.2, Характеристика дефектов) и проверяется командой на полноту и точность. Решение о выходе из инспекции соответствует одному из следующих вариантов: Характеристика дефектов) и проверяется командой на предмет полноты и точности. Решение о выходе из инспекции соответствует одному из следующих вариантов: Характеристика дефектов) и проверяется командой на предмет полноты и точности. Решение о выходе из инспекции соответствует одному из следующих вариантов:

- 1. Принять без доработок или, в крайнем случае, с незначительной доработкой
- 2. Принять с проверкой доработки
- 3. Повторно осмотрите.

2.3.4 Прохождение

Как указано в [16*],

- Целью систематического обхода является оценка программного продукта. Пошаговое руководство может проводиться с целью информирования аудитории о программном продукте.

Проходки отличаются от инспекций. Основное отличие состоит в том, что автор представляет продукт работы другим участникам встречи (лицом к лицу или в электронном виде). В отличие от осмотра, участники встречи не обязательно видели материал до встречи. Заседания могут проводиться менее официально. Автор берет на себя роль объяснения и показа материала участникам и получения обратной связи. Как и инспекции, пошаговые обзоры могут проводиться для любого типа рабочего продукта, включая план проекта, требования, дизайн, исходный код и отчеты о тестировании.

2.3.5 Аудиты обеспечения качества процесса и качества продукции

Как указано в [16*],

- Целью аудита программного обеспечения является предоставление независимой оценки соответствия программных продуктов и процессов

применимым нормам, стандартам, руководствам, планам и процедурам.

Аудиты обеспечения процесса определяют адекватность планов, графиков и требований для достижения целей проекта [5]. Аудит — это официально организованная деятельность, в которой участники играют определенные роли, такие как ведущий аудитор, другой аудитор, регистратор или инициатор, и в том числе представитель проверяемой организации. Аудиты выявляют случаи несоответствия и составляют отчет, требующий от команды принятия корректирующих действий.

Хотя может быть много формальных названий проверок и аудитов, например, определенных в стандарте [16*], важным моментом является то, что они могут иметь место практически для любого продукта на любом этапе процесса разработки или обслуживания.

3 практических соображения

3.1 Требования к качеству программного обеспечения

[9 , c11s1] [18 , c12] [17 , c15s3.2.2, c15s3.3.1, c16s9.10]

3.1.1 Факторы влияния

Различные факторы влияют на планирование, управление и выбор мероприятий и методов SQM, в том числе:

- домен системы, в которой находится программное обеспечение; системные функции могут быть критически важными для безопасности, критически важными для бизнеса, критически важными для безопасности
- физическая среда, в которой находится программная система
- системные и программные функциональные (что система делает) и качественные (насколько хорошо система выполняет свои функции) требования
- коммерческие (внешние) или стандартные (внутренние) компоненты, которые будут использоваться в системе
- применимые стандарты разработки программного обеспечения
- методы и программные средства, которые будут использоваться для разработки и обслуживания, а также для оценки и улучшения качества
- бюджет, персонал, организация проекта, планы и расписание всех процессов
- предполагаемые пользователи и использование системы
- уровень целостности системы.

Информация об этих факторах влияет на то, как организованы и документированы процессы SQM, как выбираются конкретные действия SQM, какие ресурсы необходимы и какие из этих ресурсов налагают ограничения на усилия.

3.1.2 Надежность

В тех случаях, когда сбой системы может иметь чрезвычайно серьезные последствия, общая надежность (аппаратная, программная, человеческая или операционная) является основным требованием к качеству сверх базовой функциональности. Это происходит по следующим причинам: системные сбои

затрагивают большое количество людей; пользователи часто отвергают ненадежные, небезопасные или небезопасные системы; стоимость отказа системы может быть огромной; а ненадежные системы могут привести к потере информации. Надежность системы и программного обеспечения включает такие характеристики, как доступность, надежность, безопасность и защищенность. При разработке надежного программного обеспечения можно применять инструменты и методы для снижения риска внесения ошибок в промежуточные результаты или конечный программный продукт. Процессы, техники, методы проверки, валидации и тестирования, а инструменты выявляют неисправности, влияющие на надежность, на самых ранних этапах жизненного цикла. Кроме того, в программном обеспечении могут потребоваться механизмы для защиты от внешних атак и предотвращения сбоев.

3.1.3 Уровни целостности программного обеспечения

Определение уровней целостности является методом управления рисками.

- Уровни целостности программного обеспечения — это диапазон значений, которые представляют сложность программного обеспечения, критичность, риск, уровень безопасности, уровень безопасности, желаемую производительность, надежность или другие уникальные для проекта характеристики, которые определяют важность программного обеспечения для пользователя и приобретателя. Характеристики, используемые для определения уровня целостности программного обеспечения, различаются в зависимости от предполагаемого применения и использования системы. Программное обеспечение является частью системы, и уровень его целостности должен определяться как часть этой системы.

Назначенные уровни целостности программного обеспечения могут меняться по мере развития программного обеспечения. Особенности дизайна, кодирования, процедур и технологий, реализованные в системе или программном обеспечении, могут повышать или понижать назначенные уровни целостности программного обеспечения. Уровни целостности программного обеспечения, установленные для проекта, являются результатом соглашений между покупателем, поставщиком, разработчиком и независимыми уполномоченными органами. Схема уровней целостности программного обеспечения — это инструмент, используемый для определения уровней целостности программного обеспечения. [5]

Как отмечено в [17*], «уровни целостности могут применяться во время разработки, чтобы распределить дополнительные усилия по проверке и проверке на компоненты с высокой степенью целостности».

3.2 Характеристика дефектов

[3 , c3c3, c8c8, c10c2]

Методы оценки качества программного обеспечения (т. е. контроля качества программного обеспечения) находят дефекты, ошибки и сбои. Описание этих методов приводит к пониманию продукта, облегчает внесение исправлений в процесс или продукт и информирует руководство и другие заинтересованные стороны о состоянии процесса или продукта. Существует много таксономий, и, хотя были предприняты попытки прийти к единому мнению, литература указывает на то, что используется довольно много. Характеристика дефектов также используется в аудитах и проверках, при этом руководитель проверки часто представляет список вопросов, предоставленных членами группы, для рассмотрения на собрании по проверке.

По мере развития новых методов проектирования и языков, наряду с развитием общих программных технологий, появляются новые классы дефектов, и требуется много усилий для интерпретации ранее определенных классов. При отслеживании дефектов инженера-программиста интересует не только количество дефектов, но и их типы. Одной информации, без какой-либо классификации, может быть недостаточно для выявления основных причин дефектов. Конкретные типы проблем необходимо сгруппировать, чтобы определить тенденции с течением времени. Суть в том, чтобы установить таксономию дефектов, значимую для организации и для разработчиков программного обеспечения.

Мероприятия по контролю качества программного обеспечения обнаруживают информацию на всех этапах разработки и обслуживания программного обеспечения. В некоторых случаях слово « *дефект* » перегружено для обозначения разных типов аномалий. Однако разные инженерные культуры и стандарты

могут использовать несколько разные значения этих терминов. Разнообразие терминов побуждает этот раздел предоставить широко используемый набор определений [19]:

- *Вычислительная ошибка* : «разница между вычисленным, наблюдаемым или измеренным значением или условием и истинным, заданным или теоретически правильным значением или условием».
- *Ошибка* : «Действие человека, приводящее к неверному результату». Промах или ошибка, которую совершает человек. Также называется человеческой ошибкой.
- *Дефект* : «Несовершенство или недостаток рабочего продукта, когда этот рабочий продукт не соответствует его требованиям или спецификациям и нуждается в ремонте или замене». Дефект вызван ошибкой человека.
- *Ошибка* : дефект в исходном коде. «Неверный шаг, процесс или определение данных в компьютерной программе». Кодирование человеческой ошибки в исходном коде. Ошибка — это официальное название ошибки.
- *Отказ* : «событие, при котором система или системный компонент не выполняет требуемую функцию в заданных пределах». Неудача

производится, когда процессор сталкивается с ошибкой при определенных условиях.

Используя эти определения, тремя широко используемыми показателями качества программного обеспечения являются плотность дефектов (количество дефектов на единицу размера документов), плотность сбоев (количество сбоев на 1 тыс. строк кода) и интенсивность сбоев (сбоев на час использования или на час тестирования).). Модели надежности строятся на основе данных об отказах, собранных во время тестирования программного обеспечения или от программного обеспечения в эксплуатации, и, таким образом, могут использоваться для оценки вероятности будущих отказов и для помощи в принятии решений о прекращении тестирования.

Одним из вероятных действий, вытекающих из выводов SQM, является удаление дефектов из исследуемого продукта (например, поиск и исправление ошибок, создание новой сборки). Другие действия направлены на устранение причин дефектов, например, анализ первопричин (RCA). Действия RCA включают анализ и обобщение результатов для выявления основных причин и использования методов измерения для улучшения продукта и процесса, а также для отслеживания дефектов и их устранения. Усовершенствование процесса в первую очередь обсуждается в процессе разработки программного обеспечения KA, а процесс SQM является источником информации.

Данные о несоответствиях и дефектах, обнаруженных методами контроля качества программного обеспечения, могут быть утеряны, если они не будут зарегистрированы. Для некоторых методов (например, технических обзоров, аудитов, инспекций) присутствуют регистраторы для записи такой информации вместе с вопросами и решениями. Когда используются автоматизированные инструменты (см. раздел 4, Инструменты качества программного обеспечения), выходные данные инструмента могут содержать информацию о дефекте. Отчеты о дефектах предоставляются руководству организации.

3.3 Методы управления качеством программного обеспечения

[7 , c7s3] [8 , c17] [9 , c12s5, c15s1, p417][16]

Методы контроля качества программного обеспечения можно разделить на несколько категорий, но простой подход использует только две категории: статические и динамические. Динамические методы включают выполнение программного обеспечения; статические методы включают анализ документов

3.3.1 Статические методы

Статические методы исследуют документацию по программному обеспечению (включая требования, спецификации интерфейса, конструкции и модели) и исходный код программного обеспечения без выполнения кода. Существует множество инструментов и методов для статического исследования рабочих продуктов программного обеспечения (см. раздел 2.3.2). Кроме того, инструменты, анализирующие поток управления исходным кодом и поиск мертвого кода, считаются инструментами статического анализа, поскольку они не требуют выполнения программного кода.

Другие, более формальные типы аналитических методов известны как формальные методы. Они, в частности, используются для проверки требований к программному обеспечению и проектов. В основном они использовались для проверки важных частей критических систем, таких как конкретные требования безопасности и безопасности. (См. также Формальные методы в моделях и методах разработки программного обеспечения KA.)

3.3.2 Динамические методы

Динамические методы включают в себя выполнение программного кода. При разработке и сопровождении программного обеспечения применяются различные виды динамических методов. Как правило, это методы тестирования, но такие методы, как моделирование и анализ моделей, могут считаться динамическими (см. Модели и методы разработки программного обеспечения KA). Чтение кода считается статическим методом, но опытные инженеры-программисты могут выполнять код по мере его чтения. Чтение кода может использовать динамические методы. Это несоответствие в категоризации указывает на то, что люди с разными ролями и опытом в организации могут рассматривать и применять эти методы по-разному.

Различные группы могут выполнять тестирование во время разработки программного обеспечения, в том числе группы, независимые от команды разработчиков. Программа Software Testing KA полностью посвящена этой теме.

3.3.3 Тестирование

Два типа испытаний могут подпадать под V&V из-за их ответственности за качество материалов, используемых в проекте:

- Оценка и тестирование инструментов, которые будут использоваться в проекте
- Тесты на соответствие (или обзор тестов на соответствие) компонентов и продуктов COTS, которые будут использоваться в продукте

Иногда независимой (сторонней или IV&V) организации может быть поручено проведение тестирования или мониторинг процесса тестирования. V&V может быть призвано оценить само тестирование: адекватность планов, процессов и процедур, адекватность и точность результатов.

Третья сторона не является разработчиком и не связана с разработкой продукта. Вместо этого третья сторона является независимым учреждением, обычно аккредитованным каким-либо органом власти. Их цель — протестировать продукт на соответствие определенному набору требований (см. KA «Тестирование программного обеспечения»).

3.4 Измерение качества программного обеспечения

[3 , c4] [8 , c17] [9 , c90]

Измерения качества программного обеспечения используются для поддержки принятия решений. С ростом сложности программного обеспечения вопросы качества выходят за рамки того, работает ли программное обеспечение или нет, до того, насколько хорошо оно достигает измеримых целей в области качества.

Решения, поддерживаемые измерением качества программного обеспечения, включают определение уровней качества программного обеспечения (особенно потому, что модели качества программного продукта включают меры для определения степени, в которой программный продукт достигает целей в области качества); управленческие вопросы об усилиях, стоимости и графике; определение момента прекращения тестирования и выпуска продукта (см. «Прекращение» в разделе 5.1 «Практические соображения» KA по тестированию программного обеспечения); и определение эффективности усилий по совершенствованию процессов.

Стоимость процессов SQM часто возникает при принятии решения о том, как должен быть организован проект или группа разработки и сопровождения программного обеспечения. Часто используются общие модели стоимости, которые основаны на том, когда обнаруживается дефект и сколько усилий требуется для устранения дефекта по сравнению с обнаружением дефекта на более ранних этапах процесса разработки. Данные измерения качества программного обеспечения, собранные внутри компании, могут дать лучшее представление о затратах в рамках этого проекта или организации.

Хотя данные измерения качества программного обеспечения могут быть полезны сами по себе (например, количество дефектных требований или доля дефектных требований), математические и графические методы могут быть применены для облегчения интерпретации показателей (см. Основы инженерии КА). Эти методы включают

- на основе описательной статистики (например, анализ Парето, бегущие диаграммы, диаграммы рассеяния, нормальное распределение)
- статистические тесты (например, биномиальный тест, критерий хи-квадрат)
- анализ тенденций (например, контрольные диаграммы; см . *«Панель инструментов качества* » в списке дополнительной литературы)
- предсказание (например, модели надежности).

Методы и тесты, основанные на описательной статистике, часто дают представление о наиболее проблемных областях исследуемого программного продукта. Полученные в результате диаграммы и графики являются вспомогательными средствами визуализации, которые лица, принимающие решения, могут использовать для сосредоточения ресурсов и проведения улучшений процессов там, где они кажутся наиболее необходимыми. Результаты анализа тенденций могут указывать на то, что график соблюдается, например, при тестировании, или на то, что определенные классы отказов могут стать более вероятными, если не будут предприняты какие-либо корректирующие действия в процессе разработки. Методы прогнозирования помогают оценить усилия и график тестирования, а также предсказать сбои. Более подробное обсуждение измерений в целом можно найти в КА «Процесс разработки программного обеспечения» и «Управление разработкой программного обеспечения». Более подробная информация об измерении тестирования представлена в КА по тестированию программного обеспечения.

Измерение качества программного обеспечения включает измерение возникновения дефектов и применение статистических методов для понимания типов дефектов, которые возникают чаще всего. Эта информация может быть использована при совершенствовании процессов программного обеспечения для определения методов предотвращения, уменьшения или устранения их повторения. Они также помогают понять тенденции, насколько хорошо работают методы обнаружения и сдерживания, а также насколько хорошо продвигаются процессы разработки и обслуживания.

На основе этих методов измерения можно разработать профили дефектов для конкретной области применения. Затем для следующего программного проекта в этой организации профили можно использовать для управления процессами SQM, то есть для приложения усилий там, где вероятность возникновения проблем наиболее высока. Точно так же тесты или счетчики дефектов, типичные для этой области, могут помочь в определении того, когда продукт готов к поставке. Обсуждение использования данных из SQM для улучшения процессов разработки и обслуживания появляется в документах «Управление разработкой программного обеспечения» и «Процесс разработки программного обеспечения».

4 инструмента качества программного обеспечения

Инструменты качества программного обеспечения включают инструменты статического и динамического анализа. Инструменты статического анализа вводят исходный код, выполняют синтаксический и семантический анализ без выполнения кода и представляют результаты пользователям. Существует большое разнообразие инструментов статического анализа по глубине, тщательности и объему, которые можно применять к артефактам, включая модели, в дополнение к исходному коду. (Описание инструментов динамического анализа см. в КА «Создание программного обеспечения», «Тестирование программного обеспечения» и «Сопровождение программного обеспечения».)

Категории инструментов статического анализа включают следующее:

- Инструменты, упрощающие и частично автоматизирующие обзоры и проверки документов и кода. Эти инструменты могут направлять работу разным участникам, чтобы частично автоматизировать и контролировать процесс проверки. Они позволяют пользователям вводить дефекты, обнаруженные во время проверок и обзоров, для последующего устранения.
- Некоторые инструменты помогают организациям выполнять анализ угроз безопасности программного обеспечения. Эти инструменты обеспечивают, например, автоматизированную поддержку анализа видов и последствий отказов (FMEA) и анализа дерева отказов (FTA).

- Инструменты, поддерживающие отслеживание проблем с программным обеспечением, обеспечивают ввод аномалий, обнаруженных во время тестирования программного обеспечения, и последующий анализ, устранение и устранение. Некоторые инструменты включают поддержку рабочего процесса и отслеживания состояния решения проблемы.
- Инструменты, которые анализируют данные, полученные из сред разработки программного обеспечения и сред тестирования программного обеспечения, и создают визуальное отображение количественных данных в виде графиков, диаграмм и таблиц. Эти инструменты иногда включают в себя функции для статистического анализа наборов данных (с целью выявления тенденций и составления прогнозов). Некоторые из этих инструментов обеспечивают скорость ввода дефектов и удаления; плотности дефектов; урожайность; распределение внедрения и удаления дефектов для каждой из фаз жизненного цикла.

ДАЛЬНЕЙШИЕ ЧТЕНИЯ

Н. Левесон, *Safeware: системная безопасность и компьютеры* [20]

В этой книге рассказывается о важности методов обеспечения безопасности программного обеспечения и о том, как эти методы могут быть включены в проекты разработки программного обеспечения.

Гилб Т. *Принципы управления программной инженерией* [21].

Это одна из первых книг по методам итеративной и инкрементной разработки. Метод Эво определяет количественные цели, частые ограниченные по времени итерации, измерения прогресса в достижении целей и адаптацию планов на основе фактических результатов.

Т. Гилб и Д. Грэм, *Проверка программного обеспечения* [22].

Эта книга знакомит с измерением и статистической выборкой отзывов и дефектов. В нем представлены методы, которые дают количественные результаты для уменьшения дефектов, повышения производительности, отслеживания проектов и создания документации.

К.Е. Вигерс, *Рецензирование программного обеспечения: Практическое руководство* [23].

В этой книге даются четкие и краткие объяснения различных методов рецензирования, отличающихся уровнем формальности и эффективности. Предоставляется практическое руководство по внедрению методов и тому, как выбрать методы, подходящие для данных обстоятельств.

Н. Р. Таг, *Набор инструментов для контроля качества* , 2-е изд., [24].

Предоставляет практическое объяснение всеобъемлющего набора методов, инструментов и приемов для решения проблем повышения качества. Включает в себя семь основных инструментов контроля качества и многие другие.

Стандарт IEEE P730-2013 Проект стандарта для процессов обеспечения качества программного обеспечения *[5].*

Этот проект стандарта расширяет процессы SQA, определенные в IEEE/ISO/IEC 12207-2008. P730 устанавливает стандарты для инициирования, планирования, контроля и выполнения процессов обеспечения качества программного обеспечения в рамках проекта разработки или сопровождения программного обеспечения. Утверждение этого проекта стандарта ожидается в 2014 году.

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

[1] П. Б. Кросби, *Качество бесплатно* , McGraw-Hill, 1979.

[2] У. Хамфри, *Управление программным процессом* , издательство Addison-Wesley, 1989.

[3] С. Х. Кан, *Метрики и модели в разработке программного обеспечения* , 2-е изд., Addison-Wesley, 2002.

[4] ISO/IEC, *25010:2011 Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Systems and Software Quality* Models *, ISO/IEC, 2011.*

[5] IEEE, *Проект стандарта P730™/D8 для процессов обеспечения качества программного обеспечения* , IEEE, 2012 г.

[6] Ф. Ботт и др., *Профессиональные вопросы разработки программного обеспечения* , 3-е изд., Тейлор и Фрэнсис, 2002 г.

[7] Д. Галин, *Обеспечение качества программного обеспечения: от теории к реализации* , Pearson Education Limited, 2004.

[8] С. Найк и П. Трипати, *Тестирование программного обеспечения и обеспечение качества: теория и практика* , Wiley-Spektrum, 2008.

[9] П. Клементс и др., *Документирование архитектуры программного обеспечения: взгляды и не только* , 2-е изд., Pearson Education, 2010.

[10] Г. Воланд, *Проектирование по дизайну* , 2-е изд., Прентис Холл, 2003.

[11] RTCA, *DO-178C, Вопросы программного обеспечения при сертификации бортовых систем и оборудования* , Радиотехническая комиссия по аэронавтике, 2011 г.

[12] IEEE Std., *15026.1-2011. Принятие стандарта пробного использования стандарта ISO/IEC TR 15026-1:2010 «Системная и программная инженерия — Обеспечение систем и программного обеспечения — Часть 1:* концепции и словарь *», IEEE, 2011.*

[13] IEEE Std., *12207-2008 (он же ISO/IEC 12207:2008) Стандарт системной и программной инженерии — процессы жизненного цикла программного обеспечения* , IEEE, 2008.

[14] ISO, *9000:2005 Системы управления качеством — основы и словарь* , ISO, 2005.

[15] Стандарт IEEE, *стандарт IEEE. 1012-2012 Standard for System and Software Verification and Validation* , IEEE, 2012.

[16] IEEE Std., *1028-2008, Обзоры и аудиты программного обеспечения* , IEEE, 2008.

[17] Дж. В. Мур, *Дорожная карта разработки программного обеспечения: руководство* , основанное на стандартах, издательство Wiley-IEEE Computer Society Press, 2006.

[18] К.Е. Вигерс, *Требования к программному обеспечению* , 2-е изд., Microsoft Press, 2003.

[19] ISO/IEC/IEEE, *24765:2010 Системная и программная инженерия — словарь* , ISO/IEC/IEEE, 2010.

[20] Н. Левесон, *Safeware: System Safety and Computers* , Addison-Wesley Professional, 1995.

[21] Т. Гилб, *Принципы управления программной инженерией* , Addison-Wesley Professional, 1988.

[22] Т. Гилб и Д. Грэм, *Проверка программного обеспечения* , Addison-Wesley Professional, 1993.

[23] К. Вигерс, *Рецензирование программного обеспечения: практическое руководство* , Addison-Wesley Professional, 2001.

[24] NR Tague, *The Quality Toolbox* , 2-е изд., ASQ Quality Press, 2010.

- Последнее изменение этой страницы состоялось 29 августа 2015 г., в 15:09.