Программная инженерия

Качество программного обеспечения (Software Quality)

Глава базируется на IEEE Guide to the Software Engineering Body of Knowledge[(1)] - SWEBOK®, 2004. Содержит перевод описания области знаний SWEBOK® “Software Quality”, с комментариями и замечаниями[(2)].

Сергей Орлик.

Программная инженерия

Качество программного обеспечения (Software Quality)

Что такое качество и почему оно должно быть столь глубоко представлено (в виде самостоятельной области знаний, *прим. автора*) в SWEBOK? На протяжении многих лет отдельные авторы и целые организации определяли термин “качество” по-разному. Фил Кросби (Phil Crosby) в 1979 году дал определение качеству как “соответствие пользовательским требованиям”. Уотс Хемпфри (Watts Hamphrey, оригинальный автор концепции модели оценки зрелости CMM, а также PSP и TSP – People Software Process и Team Software Process, *прим. автора*) описывает качество как “достижение отличного уровня пригодности к использованию”. Компания IBM, в свою очередь, ввела в оборот фразу “качество, управляемое рыночными потребностями” (“market-driven quality”). Критерий Бэлдриджа (Baldrige) для организационного качества (см. NIST - National Institute of Standards and Technology, “Baldrige National Quality Program”, http://www.quality.nist.gov) использует похожую фразу - “качество, задаваемое потребителем” (“customer-driven quality”), рассматривая удовлетворение потребителя в качестве главного соображения в отношении качества. Чаще, понятие качества используется в соответствии с определением системы менеджмента качества ISO 9001 как “степень соответствия присущих характеристик требованиям” (именно так это сформулировано в официальном переводе ИСО 9000-2000 "Системы менеджмента качества. Основные положения и словарь”, *прим. автора*).

Данные взгляды перекликаются и с введенным автором “приемлемым качеством”, определяемым не только уровнем запросов конечных потребителей в отношении параметров создаваемого продукта, но и заданным контекстом/ограничениями проекта. Это не значит, что “приемлемое качество” противопоставляется “качеству, диктуемому заказчиком”. Конечно, не стоит и проводить параллель “приемлемого качества” с “продуктом второй свежести”. Введение категории “приемлемости” в отношении качества является лишь прагматичным взглядом на желаемую степень совершенства создаваемого продукта (услуги), способную удовлетворить пользователей и достижимую в рамках заданных проектных ограничений. Интересно, что и сама “степень приемлемости” также выступает в роли ограничения проекта, а в приложении к индустрии программного обеспечения представлена практически во всех областях проектной деятельности – от управления требованиями (“атрибуты качества” как категория нефункциональных требований), до тестирования (т.н. наработка на отказ, такие метрики как MTTF - Mean Time To Failure, то есть среднее время между обнаруженными сбоями системы, и т.п.). В какой-то степени, “приемлемое качество” можно сравнивать с уровнем обслуживания в рамках заданного SLA – Service Level Agreement, давно уже принятого на вооружение в телекоммуникационной индустрии. Таким образом, приемлемое качество может рассматриваться как <количественно выраженный> *компромисс между заказчиком и исполнителем в отношении характеристик продукта*, создаваемого исполнителем в интересах <решения задач> заказчика *с учетом других ограничений проекта* (в частности, стоимостью, что часто именуется как “cost of quality” –

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

“стоимость качества”). Можно сказать, что такой взгляд может в какой-то степени рассматриваться как расширение определения ISO 9001 с учетом достигнутого компромисса между заказчиком и исполнителем (поставщиком) в отношении характеристик качества.

Данная глава (область знаний) рассматривает вопросы качества программного обеспечения, выходя за рамки <отдельных> процессов жизненного цикла. Качество программного обеспечения является постоянным объектом заботы программной инженерии и обсуждается во многих областях знаний (что вполне обосновано, если учесть поистине катастрофический уровень проваленных проектов и неудовлетворенность пользователей программных продуктов, ставшая притчей во языцех для программной индустрии, *прим. автора*). В общем случае, SWEBOK описывает ряд путей достижения качества программного обеспечения. В частности, эта область знаний касается *статических техник*, не требующих выполнения оцениваемых программных систем, в отличие от *динамических техник*, рассмотренных в области знаний SWEBOK “Тестирование”.

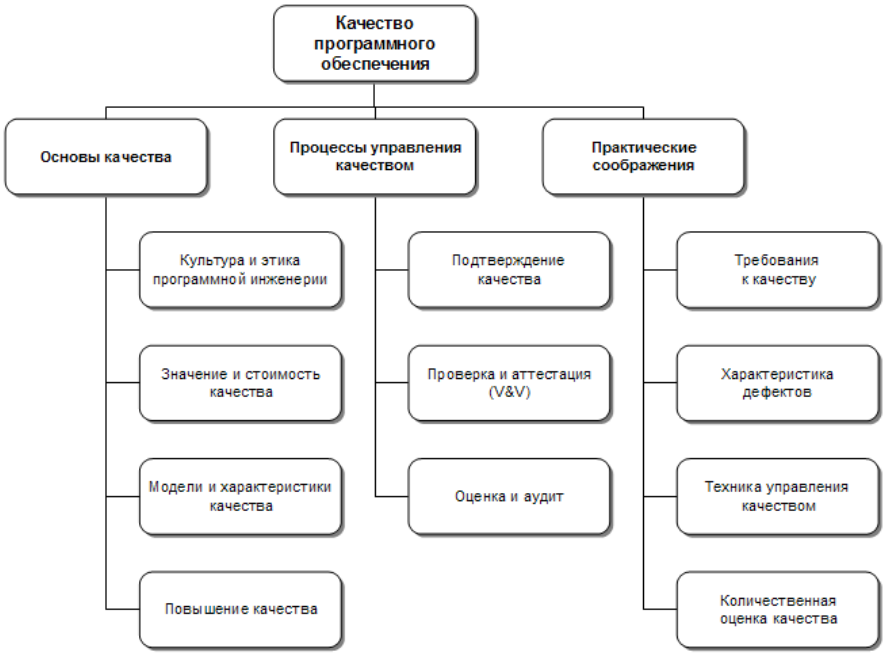

Рисунок 1. Область знаний “Качество программного обеспечения” [SWEBOK, 2004, с.10-2, рис. 1]

1. Основы качества программного обеспечения (Software Quality Fundamentals)

Согласие, достигнутое по требованиями к качеству (в оригинале - quality requirements), наравне с четким доведением до инженеров того, что составляет качество <получаемого продукта>, требуют обсуждения и формального определения многих аспектов качества.

Инженеры должны понимать смысл, вкладываемый в концепцию качества, характеристики и значение качества в отношении разрабатываемого или сопровождаемого программного обеспечения.

Важной идеей является то, что программные требования определяют требуемые характеристики качества программного обеспечения, а также влияют на методы количественной оценки и сформулированные для оценки этих характеристик <соответствующие> критерии приемки.

1.1 Культура и этика программной инженерии (Software Engineering Culture and Ethics)

Ожидается, что инженеры по программному обеспечению воспринимают вопросы качества программного обеспечения как часть своей <профессиональной> культуры. SWEBOK дает ссылки на источники, описывающие здоровую культуру программной инженерии.

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

Этические аспекты могут играть значительную роль в обеспечении качества программного обеспечения, культуре и отношении инженеров <к своей работе>. IEEE Computer Society и ACM разработали кодекс этики (“моральный кодекс” – code of ethics) и профессиональной практики, основанный на восьми принципах, помогающих инженерам укрепить их отношение к качеству и независимость <в решении вопросов обеспечения достойного качества создаваемых программных продуктов> в их повседневной работе (автор планирует рассмотреть этот кодекс более подробно за рамками перевода и комментариев к SWEBOK, *прим. автора*).

1.2 Значение и стоимость качества (Value and Costs of Quality)

Понятие “качество”, на самом деле, не столь очевидно и просто, как это может показаться на первый взгляд. Для любого инженерного продукта существует множество <интерпретаций> качества, в зависимости от конкретной “системы координат” (в оригинале – “перспективы”, прим. автора). Множество этих точек зрения необходимо обсудить и определить на этапе выработки требований к программному продукту. Характеристики качества могут требоваться в той или иной степени, могут отсутствовать или могут задавать определенные требования, все это может быть результатом определенного компромисса (что вполне перекликается с пониманием “приемлемого качества”, как менее жесткой точки зрения на обеспечение качества, как достижение совершенства, *прим. автора*).

Стоимость качества (cost of quality) может быть дифференцирована на стоимость предупреждения <дефектов> (prevention cost), стоимость оценки (appraisal cost), стоимость внутренних (internal failure cost), а также внешних сбоев (external failure cost).

Движущей силой программных проектов является желание создать программное обеспечение, обладающее определенной ценностью (значимое для решения определенных задач или достижения целей, *прим. автора*). Ценность программного обеспечения в может выражаться в форме стоимости, а может и нет. Заказчик, обычно, имеет свое представление о максимальных стоимостных вложениях, возврат которых ожидается в случае достижения основных целей создания программного обеспечения. Заказчик может, также, иметь определенные ожидания в отношении качества ПО. Иногда, заказчики не задумываются о вопросах качества и связанной с ними стоимостью. Является ли характеристики качества чисто декоративными (умозрительной, *прим. автора*) или, все же, это неотъемлемая часть программного обеспечения? Ответ, вероятно, находится где-то посередине, как почти всегда бывает в таких случаях, и является предметом обсуждения степени вовлечения заказчика в процесс принятия решений и полного понимания заказчиком стоимости и выгоды, связанной с достижением того или иного уровня качества. В идеальном случае, большинство такого рода решений должно приниматься процессе работы с требованиями (см. область знаний SWEBOK “Программные требования”), однако эти вопросы могут (и должны, *прим. автора*) подниматься на протяжении всего жизненного цикла программного обеспечения. Не существует каких-то <“стандартных”> правил того, как именно необходимо принимать такие решения. Однако, инженеры должны быть способны представить различные альтернативы (в достижении различного уровня качества, *прим. автора*) и их стоимость. Здесь SWEBOK приводит некоторые источники, в которых более подробно обсуждаются вопросы значимости качества и соответствующих характеристик стоимости.

1.3 Модели и характеристики качества (Models and Quality Characteristics)

В различных источниках (таксономиях и моделях) терминология характеристик качества программного обеспечения отличается. Каждая модель включает различное число уровней иерархии и общее число <”распознанных”> характеристик качества. Различные авторы создали разные модели качества со своим набором характеристик и атрибутов (в частности, Барри Боэм, автор спиральной модели жизненного цикла разработки программного обеспечения, которая рассматривается автором за рамками перевода и комментирования SWEBOK, *прим. автора*). Эти модели могут быть полезны для обсуждения, планирования, (адаптации, *прим. автора*) и оценки качества программных продуктов. ISO/IEC определяет три связанных модели качества программного обеспечения (ISO 9126-01 Software Engineering - Product Quality, Part 1: Quality Model) – внутреннее качество, внешнее качество и качество в процессе эксплуатации, а также набор соответствующих работ по оценке качества программного обеспечения (ISO14598-98 Software Product Evaluation).

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

1.3.1 Качество процессов программного обеспечения (Software engineering process quality)

Управление качеством (software quality management) и качество процессов программной инженерии (software engineering process quality) имеют непосредственное отношение к качеству создаваемого программного продукта.

Модели и критерии оценки возможностей организаций, занимающихся разработкой программного обеспечения, прежде всего касаются рассмотрения организации проектных работ и аспектов управления. Соответственно, они рассматриваются в областях знаний SWEBOK “Управление программной инженерией” и “Процесс программной инженерии”.

Конечно, невозможно полностью отделить качество процесса от качества продукта.

Качество процесса, обсуждаемое в области знаний “Процесс программной инженерии”, влияют на характеристики качества продукта, которые, в свою очередь, отражаются в восприятии качества продукта в процессе эксплуатации со стороны заказчика.

Существует два важнейших стандарта в области качества программного обеспечения. TickIT - касается рассмотрения общей системы менеджмента качества ISO 9001-00 в приложении к программным проектам (и, в частности, сочетания такого взгляда с положениями стандарта жизненного цикла ISO 12207, *прим. автора*) и представленный, также, в виде специальных рекомендаций ISO 90003-04 “Software and Systems Engineering - Guidelines for the Application of ISO9001:2000 to Computer Software”.

Другой важный стандарт – CMMI, обсуждаемый в области знаний “Процесс программной инженерии”, предоставляет рекомендации по совершенствованию процесса. (здесь нельзя не упомянуть и ISO 15504 “Information Technology - Software Process Assessment”, известный как SPICE - Software Process Improvement and Capability dEtermination, который также рассматривается в упомянутой области знаний, *прим. автора*). Непосредственно с управлением качеством связаны процессные области (области компетенции) CMMI: обеспечение качества процесса и продукта (process and product quality assurance, категория процессов CMMI “Support”), проверка (verification, категория “Engineering”) и аттестация (validation, категория “Engineering”). При этом, CMMI классифицирует обзор (review) и аудит (audit) в качестве методов верификации, но не как самостоятельные процессы, в отличие, например, от стандарта 12207.

Дебаты в отношении того, какой именно стандарт стоит использовать инженерам для обеспечения качества программного обеспечения – CMMI или ISO 9001, продолжаются с самого создания этих стандартов. Сегодня можно сказать о том, что данные стандарты все же рассматривают как взаимодополняющие и, что сертификация по ISO 9001 помогает в достижении старших уровней зрелости по CMMI.

1.3.2 Качество программного продукта (Software product quality)

Прежде всего, инженеры должны определить цели создания программного обеспечения. В этом контексте, особо важно помнить, что требования заказчика - первичны и содержат требования в отношении качества, а не только функциональности (функциональные требования). Таким образом, инженеры ответственны за извлечение требований к качеству, которые не всегда представлены явно, а также обсуждение их важности и степени сложности их достижения. Все процессы, ассоциированные с качеством (например, сборка, проверка и повышение качества), должны проектироваться с учетом этих требований и несут на себе тяжесть дополнительных расходов (как важную составную часть стоимости программного обеспечения, *прим. автора*).

Стандарт ISO 9126-01 (Software Engineering - Product Quality, Part 1: Quality Model) определяет для двух из трех описанных в нем моделей, связанные характеристики и "суб-характеристики" качества, а также метрики, полезные для оценки качества программных продуктов.

Понимание термина “продукт” расширено включением всех артефактов, создаваемых на выходе всех процессов, используемых для создания конечного программного продукта. Примерами продукта являются (но не ограничиваются этим): полная спецификация системных требований (system requirements specification), спецификация программных требований для программных компонент системы (software requirements specification, SRS), модели, код, тестовая документация,

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

отчеты, создаваемые в результате работ по анализу качества. Хотя, чаще всего термин качество используется в отношении конечного продукта и поведения системы в процессе эксплуатации, хорошей инженерной практикой является требование к тому, чтобы соответствие заданным характеристикам качества оценивалось и для промежуточных результатов/продуктов жизненного цикла в рамках всех процессов программной инженерии.

1.4 Повышение качества (Quality Improvement)

Качество программного обеспечения может повышаться за счет итеративного процесса постоянного улучшения. Это требует контроля, координации и обратной связи в процессе управления многими одновременно выполняемыми процессами: (1) процессами жизненного цикла, (2) процессом обнаружения, устранения и предотвращения сбоев/дефектов и (3) процессов улучшения качества.

К программной инженерии применимы теории и концепции, лежащие в основе совершенствования качества. Например, предотвращение и ранняя диагностика ошибок, постоянное совершенствование (continuous improvement) и внимание к требованиям заказчика (customer focus), составляющие принцип “building in quality”. Эти концепции основываются на работах экспертов по качеству, пришедших к мнению, что качество продукта напрямую связано с качеством используемых для его создания процессов.

Такие подходы, как TQM (Total Quality Management – всеобщее управление качеством) PDCA (Plan, Do, Check, Act – Планирование, Действие, Проверка, Реакция/Корректировка), являются инструментами достижения задач, связанных с качеством. Поддержка менеджмента помогает в выполнении процессов, оценке продуктов и получению всех необходимых данных. Кроме этого, разрабатываемая программа совершенствования (improvement program, обычно является целевой и охватывает работу подразделения или организации, в целом, *прим. автора*) детально идентифицирует все действия и проекты по улучшению <отдельных аспектов деятельности> в рамках определенного периода времени, за который такие проекты можно осуществить с успешным решением соответствующих задач. При этом, поддержка менеджмента означает, что все проекты по улучшению обладают достаточными ресурсами для достижением поставленных целей. Поддержка менеджмента тесно связана с реализацией активного взаимодействия в коллективе, и должна предупреждать возникновение потенциальных проблем (и пассивного или даже активного противодействия реализации программы совершенствования или отдельных ее проектов, *прим. автора*). Формирование рабочих групп, поддержка менеджеров среднего звена и выделенные ресурсы на уровне проекта – эти вопросы обсуждаются в области знаний “Процесс программной инженерии”.

2. Процессы управления качеством программного обеспечения (Software Quality Processes)

Управление качеством программного обеспечения (SQM, Software Quality Management) применяется ко всем аспектам процессов, продуктов и ресурсов. SQM определяет процессы, владельцев процессов, а также требования к процессам, измерения процессов и их результатов, плюс – каналы обратной связи.

Процессы управления качеством содержат много действий. Некоторые из них позволяют напрямую находить дефекты, в то время, как другие помогают определить где именно может быть важно провести более детальные исследования, после чего, опять-таки, проводятся работы по непосредственному обнаружению ошибок. Многие действия также могут вестись с целью достижения и тех и других целей.

Планирование качества программного обеспечения включает:

(1) Определение требуемого продукта в терминах характеристик качества (см., например, область знаний “Управление программной инженерией”).

(2) Планирование процессов для получения требуемого продукта (см., например, области знаний “Проектирование” и “Конструирование”).

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

Эти процессы отличаются от процессов SQM, как таковых, которые, в свою очередь, направлены на оценку планируемых характеристик качества, а не на реальную реализацию этих планов. *Процессы управления качеством должны адресоваться вопросам, насколько хорошо продукт будет удовлетворять потребностям заказчика и требованиям заинтересованных лиц, обладать ценностью для заказчика и заинтересованных лиц и качеством, необходимым для соответствия сформулированным требованиям к программному обеспечению.*

SQM может использоваться для оценки и конечных и промежуточных продуктов.

Некоторые из специализированных процессов SQM определены в стандарте 12207:

- Процесс обеспечения качества (quality assurance process)
- Процесс верификации (verification process)
- Процесс аттестации (validation process)
- Процесс совместного анализа (joint review process)
- Процесс аудита (audit process)

Все эти процессы поддерживают стремление к достижению качества и, кроме того, помогают в поиске возможных ошибок. Однако, они отличаются в том, на чем концентрируют внимание.

Процессы SQM помогают в обеспечении лучшего качества программного обеспечения в данном проекте. Они предоставляют менеджерам основную информацию по каждому продукту и, кроме того, включают параметры качества всего процесса программной инженерии. Области знаний SWEBOK “Процесс программной инженерии” и “Управление программной инженерией” обсуждают программы качества для организаций, занимающихся разработкой программного обеспечения. SQM может предоставить соответствующую обратную связь для этих областей.

Процессы SQM состоят из задач и техник, предназначенных для оценки того, как начинают реализовываться планы по созданию программного обеспечения и насколько хорошо промежуточные и конечные продукты соответствуют заданным требованиям. Результаты выполнения этих задач представляются в виде отчетов для менеджеров перед тем, как будут предприняты соответствующие корректирующие действия. Управление SQM-процессом ведется исходя из уверенности, что данные отчетов точны.
Как описано в данной области знаний, процессы SQM тесно связаны между собой. Они могут перекрываться, а иногда даже и совмещаться. Они кажутся *реактивными* по своей природе, в силу того, что они рассматривают процессы в контексте полученной практики и уже произведенные продукты. Однако, они играют главную роль на стадии планирования, являясь *проактивными* как процессы и процедуры, необходимые для достижения характеристик и уровня качества, востребованных заинтересованными лицами <проекта> программного обеспечения.

Управление рисками также может играть значительную роль для выпуска качественного программного обеспечения. Включение “регулярного” (как постоянно действующего, а не периодического; в оригинале – disciplined, *прим. автора*) анализа рисков и <соответствующих> техник управления <рисками> в процессы жизненного цикла программного обеспечения может увеличить потенциал для производства качественного продукта. Более подробную информацию по управлению рисками можно найти в области знаний “Управление программной инженерией”.

2.1 Подтверждение качества программного обеспечения (Software Quality Assurance, SQA)

Процессы SQA обеспечивают подтверждение того, что программные продукты и процессы жизненного цикла проекта соответствуют заданным требованиям. Такое подтверждение проводится на основе планирования (planning), постановки <работ> (enacting) и исполнения (performing) набора действий, направленных на то, чтобы качество стало неотъемлемой частью программного обеспечения (см. выше определения качества). Такой взгляд подразумевает ясное и точное формулирование проблемы, а также то, что определены и четко выражены (полны и однозначно интерпретируемы, *прим. автора*) требования к соответствующему <программному> решению. SQA добивается обеспечения качества в процессе разработки и сопровождения за счет выполнения различных действий на всех этапах <жизненного цикла>, что позволяет идентифицировать проблемы еще на ранних стадиях, которые практически неизбежны в любой сложной деятельности.

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

По мнению автора, такая идентификация возможна во многих случаях (если даже не в большинстве ситуаций), когда проблема еще является риском и возможно ее предотвращение. Это – задача *управления рисками*, которое вполне можно было бы вынести в качестве самостоятельной области знаний SWEBOK, в силу уже достаточно большого совокупного опыта не только индустрии ИТ или дисциплины управления проектами. Так или иначе, можно сказать, что уже было подчеркнуто при обсуждении SQM, управление рисками (Risk Management) является серьезным дополнительным инструментом для обеспечения качества программного обеспечения. Однако, ограничиваться упоминанием управления рисками только в контексте SQM было бы неправильно, так как сегодняшнее понимание Risk Management включает в себя не только вопросы предупреждения рисков, но и управление процессом разрешения проблем.

SQA, как это сформулировано SWEBOK, концентрируется на процессах. Роль SQA состоит в том, чтобы обеспечить соответствующее планирование процессов, дальнейшее исполнение процессов на основе заданного плана и проведение необходимых измерений процессов с передачей результатов измерений заинтересованным сторонам (организационными структурам и лицам).

SQA-план определяет средства, которые будут использоваться для обеспечения соответствия разрабатываемого продукта заданным пользовательским требованиям с максимальным уровнем качества, возможным при заданных ограничениях проекта (т.е., в терминологии автора – приемлемым уровнем качества, *прим. автора*). Для того, чтобы этого добиться, в первую очередь необходимо, чтобы цели качества были четко определены и понимаемы (а также, однозначно интерпретируемы, что является обязательным условием любых целей и соответствующих требований, *прим. автора*). Это, в обязательном порядке, должно быть отражено в соответствующих планах управления <проектом>, разработки и сопровождения. Подробности можно найти в стандарте IEEE 730-02 “IEEE Standard for Software Quality Assurance Plans”.

Конкретные работы и задачи по обеспечению качества структурируются с детализацией требований по их стоимости и ассоциированным ресурсам, целям с точки зрения управления и соответствующим расписанием в контексте целей, заданных планами управления, разработки и сопровождения. SQA-план должен согласовываться с планом конфигурационного управления (см. область знаний “Software Configuration Management”). План SQA идентифицирует документы, стандарты, практики и соглашения, применяемые при контроле проекта, а также то, как эти аспекты будут проверяться и отслеживаться для обеспечения достаточности и соответствия заданному плану. Также, SQA-план идентифицирует метрики, статистические техники, процедуры формирования сообщений о проблемах и проведения корректирующих действий, такие средства (в оригинале SWEBOK используется термин resources, *прим. автора*), как инструменты, техники и методологии, вопросы безопасности физических носителей (это, скорее, вопрос базовой инфраструктуры проектов, а не SQA-плана, *прим. автора*), тренинги, а также формирование отчетности и документации, относящиеся к вопросам SQA. Кроме того, SQA-план касается и вопросов работ по обеспечению качества, относящихся к другим типам деятельности, описанным в <различных> планах по созданию программного обеспечения, к которым также относятся поставка, установка, обслуживание (поддержка и сопровождение, *прим. автора*) заказных и/или тиражируемых/готовых программных решений (commercial off-the-shelf, COTS), необходимых для данного проекта программного обеспечения. Наконец, SQA-план может содержать необходимые для обеспечения качества критерии приемки программного обеспечения и действия по формированию отчетности и управлению <и контролю над> работами.

2.2 Проверка (верификация) и аттестация (Verification and Validation, V&V)

С целью краткости <изложения> (что, не мешало их детализировать достаточным образом, в виде соответствующих "подтем" в рамках формата SWEBOK, так как, будучи тесно связанными и взаимодополняющими, проверка и аттестация все же обладают и самостоятельным содержанием, *прим. автора*), *проверка (верификация) и аттестация – Validation and Verification (V&V)* – рассмотрены в SWEBOK в рамках единой темы. В свою очередь, они являются самостоятельными темами, например, в стандарте жизненного цикла программного обеспечения 12207. Стандарт IEEE 1059-93 “IEEE Guide for Software Verification and Validation Plans” дает такое определение V&V*: “Проверка и аттестация программного обеспечения – упорядоченный подход в оценке программных продуктов, применяемый на протяжении всего жизненного цикла. Усилия, прилагаемые в рамках работ по проверке и аттестации, направлены на обеспечение качества как неотъемлемой характеристики программного обеспечения и удовлетворение пользовательских требований” (здесь и далее, как и при обсуждении SQA, пользовательские требования, скорее, не

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

user requirements в понимании управления требованиями, а потребности пользователей, ради удовлетворения которых создается программное обеспечение, п*рим. автора*).

* здесь и далее в переводе намеренно используется обозначение V&V, как общепринятое в индустрии программного обеспечения, *прим. автора*

V&V напрямую адресуется вопросам качества программного обеспечения и использует соответствующие техники тестирования для обнаружения тех или иных дефектов. V&V может применяться для промежуточных продуктов, однако, в том объеме, который соответствует промежуточным “шагам” <соответствующих> процессов жизненного цикла.

Процесс V&V определяет в какой степени продукт (результат) тех или иных работ по разработке и сопровождению соответствует требованиям, сформулированным в рамках этих работ, а конечный продукт удовлетворяет заданным целям и пользовательским требованиям (корректнее было бы говорить не только и, может быть, не столько о “требованиях”, то есть потребностях, сколько об ожиданиях, *прим. автора*). *Верификация* – попытка обеспечить *правильную разработку продукта* (продукт построен правильным образом; обычно, для промежуточных, иногда, для конечного продукта, *прим. автора*), в том значении, что получаемый в рамках соответствующей деятельности продукт соответствует спецификациям, заданным в процессе предыдущей деятельности. *Аттестация* – попытка обеспечить *создание правильного продукта* (построен правильный продукт; обычно, в контексте конечного продукта, *прим. автора*), с точки зрения достижения поставленной цели. Оба процесса – верификация и аттестация – начинаются на ранних стадиях разработки и сопровождения. Они обеспечивают исследованию (экспертизу) ключевых возможностей продукта как в контексте непосредственно предшествующих результатов (промежуточных продуктов), так и с точки зрения удовлетворения соответствующих спецификаций.

Целью планирования V&V является обеспечение процессов верификации и аттестации необходимыми ресурсами, четкое назначение ролей и обязанностей. Получаемый план V&V документирует и <детально> описывает различные ресурсы, роли и действия, а также используемые техники и инструменты. Понимание различных целей каждого действия в рамках V&V может помочь в точном планировании техник и ресурсов (включая, финансовые, *прим. автора*), необходимых для достижения заданных целей. Стандарты IEEE 1012-98 “Software Verification and Validation” и 1059-93 “IEEE Guide for Software Verification and Validation Plans” (Appendix A) определяют типичное содержание плана проверки и аттестации.

План также касается аспектов управления, коммуникаций (взаимодействия), политик и процедур в отношении действий по верификации и аттестации и их взаимодействия. Кроме того, в нем могут быть отражены вопросы формирования отчетности по дефектам и документирования требований (конечно, с точки зрения выполнения задач по проверке и аттестации продуктов, *прим. автора*).

2.3 Оценка (обзор) и аудит (Review and Audits)

В целях краткости изложения, оценка (review) и аудит объеденены в SWEBOK в одну тему (в данном случае, по мнению автора, такое объединение выглядит более обоснованным, чем в случае V&V, где частью процесса аттестации могут быть, например, приемо-сдаточные испытания, наверняка отсутствующие при верификации; оценка же и аудит, на практике, часто отличаются лишь степенью детализации, акцентам в отношении исследуемых аспектов, лицом/органом, выполняющим соответствующие работы, а также степенью формализации процесса, *прим. автора*). В стандарте жизненного цикла 12207 эти работы разделены на самостоятельные темы. Более детально они описаны в стандарте IEEE 1028-97 “IEEE Standard for Software Reviews”, в котором представлено пять типов оценок и аудитов (обратите внимание, что классификация рассматривает аудит лишь как один из типов оценки, что, в частности, согласуется с мнением автора, *прим. автора*):

- Управленческие оценки (management reviews)
- Технические оценки (technical reviews)
- Инспекции (inspections)
- “Прогонки” (walk-throughs)
- Аудиты (audtis)

2.3.1 Управленческие оценки (Management Reviews)

“Назначение управленческих оценок состоит в отслеживании развития <проекта/продукта>, определения статуса планов и расписаний, утверждения требования и распределения ресурсов, или оценки эффективности управленческих подходов, используемых для достижения поставленных целей.” - IEEE 1028-97 “IEEE Standard for Software Reviews”. Управленческие оценки поддерживают принятие решений о внесении изменений и выполнении корректирующих действий, необходимых в процессе выполнения программного проекта. Управленческие оценки определяют адекватность планов, расписаний и требований, в то же время, контролируя их прогресс или несоответствие. Эти оценки могут выполняться в отношении продукта, будучи фиксируемы в форме отчетов аудита, отчетов о состоянии (развитии), V&V-отчетов, а также различных типов планов – управления рисками, проекта/проектного управления, конфигурационного управления, безопасности <использования> программного обеспечения (safety), оценки рисков и т.п. Информация, связанная с данными вопросами, также представлена в областях знаний “Управление программной инженерией” и “Конфигурационное управление”.

2.3.2 Технические оценки (Technical Reviews)

“Назначением технических оценок является исследование программного продукта для определения его пригодности для использования в надлежащих целях. Цель состоит в идентификации расхождений с утвержденными спецификациями и стандартами.” - IEEE 1028-97 “IEEE Standard for Software Reviews”.

Для обеспечения технических оценок необходимо распределение следующих ролей: лицо, принимающее решения (decision-maker); лидер оценки (review leader); регистратор (recorder); а также технический персонал, поддерживающий (непосредственно исполняющий, *прим. автора*) действия по оценке. Техническая оценка требует, в обязательном порядке, наличия следующих входных данных:

- Формулировки целей
- Конкретного программного продукта (подвергаемого оценке)
- Заданного плана проекта (плана управления проектом)
- Списка проблем (вопросов), ассоциированных с продуктом
- Процедуры технической оценки

Команда <технической оценки> следует заданной процедуре оценки. Квалифицированные (с технической точки зрения) лица представляют обзор продукта (представляя команду разработки, *прим. автора*). Исследование <продукта> проводится в течение одной и более встреч (между теми, кто представляет продукт и теми, кто провидит оценку, *прим. автора*). Техническая оценка завершается после того, как выполнены все предписанные действия по исследованию продукта.

2.3.3 Инспекции (Inspections)

“Назначение инспекций состоит в обнаружении и идентификации аномалий в программном продукте.” - IEEE 1028-97 “IEEE Standard for Software Reviews”. Существует два серьезных отличия инспекций от оценок (управленческой и технической):

1. Лица, занимающие управленческие позиции (менеджеры) в отношении к любым членам команды инспектирования, не должны участвовать в инспекциях.
2. Инспекция должна вестись под руководством непредвзятого (независимого от проекта и его целей) лидера, обученного техникам инспектирования.

Инспектирование программного обеспечения всегда вовлекает авторов промежуточного или конечного продукта, в отличие от оценок, которые не требуют этого в обязательном порядке. Инспекции (как временные организационные единицы – группы, команды, *прим. автора*) включают лидера, регистратора, рецензента и нескольких (от 2 до 5) инспекторов. Члены команды инспектирования могут специализироваться в различных областями экспертизы (обладать различными областями компетенции), например, предметной области, методах проектирования, языке и т.п. В заданный момент (промежуток) времени инспекции проводятся в отношении отдельного небольшого фрагмента продукта (в большинстве случаев, фокусируясь на отдельных функциональных или других характеристиках; часто, отталкиваясь от отдельных бизнес-правил,

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

функциональных требований или атрибутов качества, *прим. автора*). Каждый член команды должен исследовать программный продукт и другие входные данные до проведения инспекционной встречи, применяя, возможно, те или иные аналитические техники (описанные ниже в подтеме 3.3.3) в небольшим фрагментам продукта или к продукту, в целом, рассматривая в последнем случае только один его аспект, например, интерфейсы. Любая найденная аномалия должна документироваться, а информация передаваться лидеру инспекции. В процессе инспекции лидер руководит сессией <инспекции> и проверяет, что все <члены команды> подготовились к инспектированию. Общим инструментом, используемым при инспектировании, является проверочный лист (checklist), содержащий аномалии и вопросы, связанные с аспектами <программного продукта>, вызывающими интерес. Результирующий лист часто классифицирует аномалии (см. стандарт IEEE 1044-93 “IEEE Standard for the Classification of Software Anomalies”) и оценивается командой с точки зрения его завершенности и точности. Решение о завершении инспекции принимается в соответствии с одним (любым) из трех критериев:

1. Принятие <продукта> с отсутствием либо малой необходимостью переработки
2. Принятие <продукта> с проверкой переработанных фрагментов
3. Необходимость повторной инспекции

Инспекционные встречи занимают, обычно, несколько часов, в отличие от технической оценки и аудита, предполагающих, в большинстве случаев, больший объем работ и, соответственно, длящиеся дольше.

2.3.4 Прогонки (Walk-throughs)

“Назначение прогонки состоит в оценке программного продукта. Прогонка может проводиться с целью ознакомления (обучения) аудитории с программным продуктом.” - IEEE 1028-97 “IEEE Standard for Software Reviews”. Главные цели прогонки состоят (по IEEE 1028) в:

- Поиске аномалий
- Улучшении продукта
- Обсуждении альтернативных путей реализации
- Оценке соответствия стандартам и спецификациям

Прогонка похожа на инспекцию, однако, обычно проводится менее формальным образом. В основном, прогонка организуется инженерами для других членов команды с целью получения отклика от них на свою работу, как одного из элементов (техник) обеспечения качества.

2.3.5 Аудиты (Audits)

“Назначением аудита программного обеспечения является независимая оценка программных продуктов и процессов на предмет их соответствия применимым регулирующим документам, стандартам, руководящим указаниям, планам и процедурам.” - IEEE 1028-97 “IEEE Standard for Software Reviews”. Аудит является формально организованной деятельностью, участники которой выполняют определенные роли, такие как главный аудитор (lead auditor), второй аудитор (another auditor), регистратор (recorder) и инициатор (initiator). В аудите принимает участие представитель оцениваемой организации/организационной единицы. В результате аудита идентифицируются случаи несоответствия и формируется отчет, необходимый команде <разработки> для принятия корректирующих действий.

При том, что существуют различные формальные названия (и классификации, *прим. автора*) оценок и аудита (например, как мы видели в стандарте IEEE 1028-97), важно отметить, что такого рода действия могут проводиться почти для любого продукта на любой стадии процесса разработки или сопровождения.

3. Практические соображения (Practical Considerations)

3.1 Требования к качеству программного обеспечения (Software Quality Requirements)

3.1.1 Факторы влияния (Influence factors)

На планирование, управление и выбор SQM-действий и техник оказывают влияние различные факторы, среди которых:

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

- Область применения системы, в которой будет работать программное обеспечение (критичное для безопасности <людей>), критичное для бизнеса и т.п.)
- Системные и программные требования
- Какие компоненты используются в системе – коммерческие (внешние) или стандартные (внутренние)
- Какие стандарты программной инженерии применимы в заданном контексте
- Каковы методы и программные инструменты, применяемые для разработки и сопровождения, а также для обеспечения качества и совершенствования (продукта и процессов, *прим. автора*)
- Бюджет, персонал, организация проектной деятельности, планы и расписания для всех процессов
- Кто целевые пользователи и каково назначение системы
- Уровень целостности системы

Информация об этих факторах влияет на то, как именно будут организованы и документированы процессы SQM, какие SQM-работы будут отобраны (стандартизированы в рамках проекта, команды, организационной единицы, организации, *прим. автора*), какие необходимы ресурсы и каковы ограничения, накладываемые в отношении усилий, направляемых на обеспечение качества.

3.1.2 Гарантоспособность (Dependability)

(Гарантоспособость – гарантия <высокой> надежности, защищенности от сбоев, *прим. автора)*

В случаях, когда сбой системы может привести к крайне тяжелым последствиям (такие системы иногда называют в англоязычных источниках “high confidence” или “high integrity system”, в русском языке к ним иногда применяют название “системы повышенной надежности”, “высокой доступности” и т.п.), общая (совокупная) гарантоспособность системы (как сочетания аппаратной части, программного обеспечения и человека) является главным и приоритетным требованием качества, по отношению к основной функциональности <системы>. Гарантоспособность (dependability) программного обеспечения включает такие характеристики, как защищенность от сбоев (fault-tolerance), безопасность использования (safety – безопасность в контексте приемлемого риска для здоровья людей, бизнеса, имущества и т.п.), информационная безопасность или защищенность (security – защита информации от несанкционированных операций, включая доступ на чтение, а также гарантия доступности информации авторизованным пользователям, в объеме заданных для них прав), а также удобство и простота использования (usability). Надежность (reliability) также является критерием, который может быть определен в терминах гарантоспособности (см. стандарт ISO/IEC 9126-1:2001 “Software Engineering - Product Quality, Part 1: Quality Model”).

В обсуждении данного вопроса существенную роль играет обширная литература по системам повышенной надежности. При этом, применяется терминология, пришедшая из области традиционных механических и электрических систем (в т.ч. не включающих программное обеспечение) и описывающая концепции опасности, рисков, целостности систем и т.п. SWEBOK приводит ряд источников, где подробно обсуждаются эти вопросы.

3.1.3 Уровни целостности программного обеспечения (Integrity levels of software)

Уровень целостности программного обеспечения определяется на основании возможных последствий сбоя программного обеспечения и вероятности возникновения такого сбоя. Когда важны различные аспекты безопасности (применения и информационной безопасности), при разработке планов работ в области идентификации возможных очагов аварий могут использоваться техники анализа опасностей (в контексте безопасности использования, safety) и анализа угроз (в информационной безопасности, security). История сбоев аналогичных систем может также помочь в идентификации наиболее полезных техник, направленных на обнаружение сбоев и <всесторонней> оценки качества программного обеспечения. Уровни целостности (например, градации целостности) предлагаются, в частности, стандартом IEEE 1012-98 “IEEE Standard for Software Reviews”.

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

По мнению автора, при более детальном рассмотрении целостности программного обеспечения в контексте конкретных проектов, необходимо уделять специальное внимание (выделяя соответствующие ресурсы и проводя необходимые работы) не только SQM-процессам (особенно, формальным, включая аудит и аттестацию), но и аспектам управления требованиями (в части критериев целостности), управления рисками (включая планирование рисков как на этапе разработки, так и на этапе эксплуатации и сопровождения системы), проектирования (которое, для повышения гарантоспособности, в обязательном порядке предполагает глубокий анализ и проверку планируемых к применению архитектурных и технологических решений, часто, посредством создания пилотных проектов, демонстрационных стендов и т.п.) и тестирования (для обеспечения всестороннего исследования поведенческих характеристик системы, в том числе с эмуляцией рабочего окружения/конфигурации, в которых система должна использоваться в процессе эксплуатации).

3.2 Характеристика дефектов (Defect Characterization)

SQM-процессы обеспечивают нахождение дефектов. Описание характеристик дефектов играет основную роль в понимании продукта, облегчает корректировку процесса или продукта, а также информирует менеджеров проектов и заказчиков о статусе (состоянии) процесса или продукта. Существуют множество таксономий (классификации и методов структурирования) дефектов (сбоев). На сегодняшний день нет четкого консенсуса по этому вопросу и SWEBOK приводит некоторые источники, освещающие его более подробно, упоминая, в частности, стандарт IEEE 1044-93 “IEEE Standard for the Classification of Software Anomalies”. Характеристика дефектов (аномалий) также используется в аудите и оценках, когда лидер оценки часто представляет для обсуждения на соответствующих встречах список аномалий, сформированный членами оценочной команды.

На фоне эволюции (и появления новых) методов проектирования и языков, наравне с новыми программными технологиями, появляются и новые классы дефектов. Это требует огромных усилий по интерпретации (и корректировке) ранее определенных классов дефектов (сбоев). При отслеживании дефектов инженер интересуется не только их количеством, но и типом. По-мнению автора, данный аспект (а именно, распределение дефектов по типам) особенно важен для определения наиболее слабых элементов системы, с точки зрения используемых технологий и архитектурных решений, что приводит к необходимости их углубленного изучения, создания специализированных пилотных проектов, дополнительной проверки концепции (proof of concept, POC – часто применяемый подход при использовании новых технологий), привлечения сторонних экспертов и т.п. SWEBOK отмечает, что сама по себе информация, без классификации, часто бывает просто бесполезна для обнаружения причин сбоев, так как для определения путей решения проблем необходима их группировка по соответствующим типам. Вопрос состоит в определении такой таксономии дефектов, которая будет значима для инженеров и организации, в целом.

SQM обеспечивает сбор информации на всех стадиях разработки и сопровождения программного обеспечения. Обычно, когда мы говорим “дефект”, мы подразумеваем “сбой”, в соответствии с определением, представленным ниже. Однако, различные культуры и стандарты могут предполагать различное смысловое наполнение этих терминов. Частичные определения понятий такого рода (из стандарта IEEE 610.12-90 “ IEEE Standard Glossary of Software Engineering Terminology ”) выглядят следующим образом:

- *Ошибка (error)*: “Отличие … между корректным результатом и вычисленным результатом < полученным с использованием программного обеспечения>”
- *Недостаток (fault)*: “Некорректный шаг, процесс или определение данных в компьютерной программе”
- *Сбой (failure)*: “<Некорректный> результат, полученный в результате недостатка”
- *Человеческая/пользовательская ошибка (mistake)*: “Действие человека, приведшее к некорректному результату”

Данные понятия достаточно детально рассматриваются в области знаний SWEBOK “Тестирование программного обеспечения”.

При обсуждении данной темы, под *дефектом (defect)* понимается результат сбоя программного обеспечения. Модели надежности строятся на основании данных о сбоях, собранных в процессе

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

тестирования программного обеспечения или его использования. Такие модели могут быть использованы для предсказания будущих сбоев и помогают в принятии решения о прекращении тестирования.

По результатам SQM-работ, направленных на обнаружение дефектов, выполняются действия по удалению дефектов из исследуемого продукта. Однако, этим дело не ограничивается. Есть и другие возможные действия, позволяющие получить полную отдачу от результатов выполнения соответствующих SQM-работ. Среди них – анализ и подведение итогов (резюмирование) <по обнаруженным несоответствиям/дефектам>, использование техник количественной оценки (получение метрик) для улучшения продукта и процесса, отслеживание дефектов и удаления их из системы (с управленческим и техническим контролем проведения необходимых корректирующих действий, *прим. автора*). Улучшение процесса рассматривается более детально в области знаний SWEBOK “Процесс программной инженерии”. В свою очередь источником информации для улучшения процесса, в частности, является SQM-процесс.

Данные о несоответствиях и дефектах, найденных в процессе реализации соответствующих техник SQM, должны фиксироваться для предотвращения их потери. Для некоторых техник (например, технической оценки, аудита, инспекций), присутствие регистратора (recorder) – обязательно, именно для фиксирования такой информации, наравне с вопросами (в том числе, требующими дополнительного рассмотрения, *прим. автора*) и принятыми решениями. В тех случаях, когда используются соответствующие средства автоматизации, они могут обеспечить и получение необходимой выходной информации о дефектах (например, сводную статистику по статусам дефектов, ответственным исполнителям и т.п., *прим. автора*). Данные о дефектах могут собираться и записываться в форме запросов на изменения (SCR, software change request) и могут, впоследствии, заноситься в определенные типы баз данных (например, в целях отслеживания кросс-проектной/исторической статистики для дальнейшего анализа и совершенствования процессов, *прим. автора*), как вручную, так и в автоматическом режиме из соответствующих средств анализа (ряд современных средств проектирования и специализированных инструментов позволяют анализировать код и модели с применением соответствующих метрик, значимых для обеспечения качества продуктов и процессов, *прим. автора*). Отчеты о дефектах направляются управленческому звену организации/организационной единицы или структуры (для принятия соответствующих решений в отношении проекта, продукта, процесса, персонала, бюджета и т.п., *прим. автора*).

3.3 Техники управления качеством программного обеспечения (Software Quality Management Techniques)

Техники SQM могут быть распределены по нескольким категориям:

- статические
- техники, требующие интенсивного использования человеческих ресурсов
- аналитические
- динамические

3.3.1 Статические техники (Static techniques)

Статические техники предполагают <детальное> исследование (examination) проектной документации, программного обеспечения и другой информации о программном продукте без его исполнения. Эти техники могут включать другие, рассматриваемые ниже, действия по “коллективной” оценке (см. 3.3.2) или “индивидуальному” анализу (см. 3.3.3), вне зависимости от степени использования средств автоматизации.

3.3.2 Техники коллективной оценки (People-intensive techniques)

Действительно, SWEBOK использует термин “people-intensive”, точный перевод содержания которого, по мнению автора, достаточно пространен: “Техники, требующие интенсивного использования человеческих ресурсов”. По-сути, их можно было бы назвать и техниками “очных оценок”, так как их идея заключается именно в форме прямого - “очного” взаимодействия специалистов. Однако, такое краткое название не подчеркивало бы фактора вовлеченности множества специалистов, который имеет важное значение для принятия решения о выборе и применении таких техник в полном объеме. Именно поэтому, данные техники в переводе названы

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

автором “техниками коллективной оценки”. Все же посмотрим, как именно SWEBOK описывает данные техники.

Форма такого рода техник, включая оценку и аудит, может варьироваться от формальных собраний до неформальных встреч или обсуждения продукта даже без обращения к его коду. Обычно, такого рода техники предполагают очного взаимодействия минимум двух, а в большинстве случаев, и более специалистов. При этом, такие встречи могут требовать предварительной подготовки (практически всегда касающейся определения содержания встреч, то есть перечня выносимых на обсуждение вопросов, *прим. автора*). К ресурсам, используемым в таких техниках, наравне с исследуемыми артефактами (продуктом, документацией, моделями и т.п., *прим. автора*) могут относится различного рода листы проверки (checklists) и результаты аналитических техник (рассматриваются ниже) и работ по тестированию. Данные техники рассматриваются, например, в стандарте 12207 при обсуждении оценки (<joint> review) и аудита (audit). SWEBOK приводит и другие полезные источники, в которых можно найти дополнительную информацию по обсуждаемому вопросу.

3.3.3 Аналитические техники (Analytical techniques)

Инженеры, занимающиеся программным обеспечением, как правило, применяют аналитические техники.

Если в данном случае создатели SWEBOK предполагали смысловую нагрузку “generally” в отношении применения аналитических техник именно подразумевая “как правило”, а не “достаточно широко”, то, по мнению автора, такого рода суждение является крайне консервативным и ограниченным. Особенно это заметно в контексте широкого (и достаточно успешного) применения Agile-методик и подходов, в которых *individuals and interactions* (см.первое положение The Agile Manifesto) предполагает <непосредственное> общение и постоянное взаимодействие членов команды (включая представителей заказчика – см. третье положение Agile Manifesto - *customer collaboration*). В частности, Agile-взгляд на SQM, вероятно, требует расширения вариантов форм оценки дополнительными категориями.

Иногда, несколько инженеров используют одну и ту же технику, но в отношении разных частей продукта. Некоторые техники базируются на специфике применяемых инструментальных средств, другие – предполагают “ручную” работу. Многие могут помогать находить дефекты напрямую, но чаще всего они используются для поддержки других техник (например, статической, *прим. автора*). Ряд техник также включает различного рода экспертизу (assessment) как составной элемент общего анализа качества. Примеры таких техник - анализ сложности (complexity analysis), анализ управляющей логики (или анализ контроля потоков управления - control flow analysis) и алгоритмический анализ (algorithmic analysis).

Каждый тип анализа обладает конкретным назначением и не все типы применимы к любому проекту. Примером техники поддержки является анализ сложности, который полезен для определения фрагментов дизайна системы, обладающих слишком высокой сложностью для корректной реализации, тестирования или сопровождения. Результат анализа сложности может также применяться для разработки тестовых сценариев (test cases). Такие техники поиска дефектов, как анализ управляющей логики, может также использоваться и в других случаях. Для программного обеспечения с обширной алгоритмической логикой крайне важно применять алгоритмические техники, особенно в тех случаях, когда некорректный алгоритм (не его реализация, а именно логика, *прим. автора*) может привести к катастрофическим результатам (например, программное обеспечение авионики, для которой вопросы безопасности использования – safety играют решающую роль, *прим. автора*). Существует обширный спектр аналитических техник, поэтому приведение их списка здесь выглядит нецелесообразным. SWEBOK указывает ряд источников, касающийся детального обсуждения выбора и самого списка аналитических техник.

Другие, более формальные типы аналитических техник известны как формальные методы. Они применяются для проверки требований и дизайна (надо признать, лишь иногда, в реальной сегодняшней практике промышленной разработки программного обеспечения; см. обсуждение формальных методов в области знаний SWEBOK “Инструменты и методы программной инженерии”, *прим. автора*). Проверка корректности применяется к критическим фрагментам программного обеспечения (что, вообще говоря, мало связано с формальными методами – это

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

естественный путь достижения приемлемого качества при минимизации затрат, *прим. автора*). Чаще всего они используются для верификации особо важных частей критически-важных систем, например, конкретных требований <информационной> безопасности и надежности.

3.3.4 Динамические техники (Dynamic techniques)

В процессе разработки и сопровождения программного обеспечения приходится обращаться к различным видам динамических техник. В основном, это техники тестирования. Однако, в качестве динамических техник могут рассматриваться техники симуляции, проверки моделей и “символического” исполнения (symbolic execution, часто предполагает использование модулей-“пустышек” с точки зрения выполняемой логики, с эмулируемым входом и выходом при рассмотрении общего сценария поведения многомодульных систем; иногда под этим термином понимаются и другие техники, в зависимости, от выбранного первоисточника, *прим. автора*). Просмотр (чтение) кода обычно рассматривается как статическая техника, но опытный инженер может исполнять код непосредственно “в процессе” его чтения (например, используя диалоговые средства пошаговой отладки для ознакомления или оценки чужого кода, *прим. автора*). Таким образом, данная техника вполне может обсуждаться и как динамическая. Такие расхождения в классификации техник ясно показывают, что в зависимости от роли человека в организации, он может принимать и применять одни и те же техники по-разному.

В зависимости от организации <ведения> проекта, определенные работы по тестированию могут выполняться при разработке программных систем в SQA и V&V процессах. В силу того, что план SQM адресуется вопросам тестирования, данная тема включает некоторые комментарии по тестированию. В свою очередь, область знаний SWEBOK “Тестирование” детально обсуждает и дает ссылки (за исключением стандартов, представленных в переводе, полный список ссылок присутствует только в оригинальном издании SWEBOK на английском языке, как и для других областей знаний, *прим. автора*) по теории, техникам и вопросам автоматизации работ по тестированию.

3.3.5 Тестирование (Testing)

Процессы подтверждения <качества>, описанные в SQA и V&V <планах>, исследуют и оценивают любой выходной продукт (включая промежуточный и конечный, *прим. автора*), связанный со спецификацией требований к программному обеспечению, на предмет трассируемости (traceability), согласованности (consistency), полноты/завершенности (completeness), корректности (correctness) и непосредственно выполнения <требований> (performance). Такое подтверждение также охватывает любые выходные артефакты процессов разработки и сопровождения, сбора, анализа и количественной оценки результатов. SQA-деятельность обеспечивает гарантию того, что соответствующие (необходимые в заданном контексте проекта, *прим. автора*) типы тестов спланированы, разработаны и реализованы, а V&V – разработку планов тестов, стратегий, сценариев и процедур <тестирования>.

Вопросы тестирования детально обсуждаются в области знаний “Тестирование”. Два типа тестирования следуют задачам, задаваемым SQA и V&V, потому как на них ложится ответственность за качество данных, используемых в проекте:

- Оценка и тестирование инструментов, используемых в проекте (IEEE 1462-98, ISO/IEC 14102 “Information Technology - Guideline for the Evaluation and Selection of CASE Tools.”)

- Тестирование на соответствие (или оценка тестов на соответствие) компонент и COTS-продуктов (COTS - commercial of-the-shelf, готовый к использованию продукт) для использования в создаваемом продукте; на это существует соответствующий стандарт (IEEE Std 1465-1998//ISO/IEC12119:1994, IEEE Standard Adoption of International Standard IDO/IEC12119:1994(E), Information Technology – Software Packages - Quality Requirements and Testing)

Иногда, независимые V&V-организации могут требовать возможности мониторинга процесса тестирования и, в определенных случаях, заверять (или, чаще, документировать/фиксировать, *прим. автора*) реальное выполнение <тестов> на предмет их проведения в соответствии с заданными процедурами. С другой стороны, может быть сделано обращение к V&V может быть

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

направлено на оценку и самого тестирования: достаточности планов и процедур, соответствия и точности результатов.

Другой тип тестирования, которое проводится под началом V&V-организации – тестирование третьей стороной (third-party testing). Такая третья сторона сама не является разработчиком продукта и ни в какой форме не связана с разработчиком продукта. Более того, третья сторона является независимым источником оценки, обычно аккредитованным на предмет обладания соответствующими полномочиями (например, организацией-разработчиком того или иного стандарта, соответствие которому оценивается независимым экспертом и чьи действия подтверждены создателем стандарта, *прим. автора*). Назначение такого рода тестирования состоит в проверке продукта на соответствие определенному набору требований (например, по информационной безопасности, *прим. автора*).

3.4 Количественная оценка качества программного обеспечения (Software Quality Measurement)

Модели качества программных продуктов часто включают метрики для определения уровня каждой характеристики качества, присущей продукту.

Если характеристики качества выбраны правильно, такие измерения могут поддержать качество (уровень качества) многими способами. Метрики могут помочь в управлении процессом принятия решений. Метрики могут способствовать поиску проблемных аспектов и узких мест в процессах. Метрики являются инструментом оценки качества своей работы самими инженерами – как в целях, определенных SQA, так и с точки зрения более долгосрочного процесса совершенствования <достигаемого> качества.

С увеличением внутренней сложности, изощренности программного обеспечения, вопросы качества выходят далеко за рамки констатации факта – работает или на работает программное обеспечение. Вопрос ставится – насколько хорошо достигаются количественно оцениваемые цели качества.

Существует еще несколько тем, предметом обсуждения которых являются метрики, напрямую поддерживающие SQM. Они включают содействие в принятии решения о моменте прекращения тестирования. В этом контексте представляются полезными модели надежности и сравнение с образцами (эталонами, принятыми в качестве примеров определенного качества – benchmarks, *прим. автора*).

Стоимость процесса SQM является одним из <проблемных> вопросов, который всегда всплывает в процессе принятия решения о том, как будет организован проект (проектные работы, *прим. автора*). Часто, используются общие (generic) модели стоимости, основанные на определении того, когда именно дефект обнаружен и как много усилий необходимо затратить на его исправление по сравнению с ситуацией, если бы дефект был найден на более ранних этапах жизненного цикла. Проектные данные могут помочь в получении более четкой картины стоимости. SWEBOK приводит источники, в которых эта тема обсуждается более подробно. Связанная информация по этим вопросам может быть найдена в областях знаний “Процесс программной инженерии” и “Управление программной инженерией”.

Наконец, сама по себе SQM-отчетность обладает полезной информацией не только о самих процессах (подразумевая их текущее состояние, *прим. автора*), но и о том, как можно улучшить все процессы жизненного цикла. Обсуждение этой темы, в частности, представлено в стандарте IEEE 1012-98 “Software Verification and Validation”.

Хотя, как количественные оценки (в данном случае речь идет о результатах оценок, а не о процессе измерений, *прим. автора*) характеристик качества могут полезны сами по себе (например, число неудовлетворенных требований и пропорция таких требований), могут <эффективно> применяться математические и графические техники, облегчающие интерпретацию значений метрик. Такие техники вполне естественно классифицируются, например, следующим образом:

- Основанные на статистических методах (например, анализ Pareto, нормальное распределение и т.п.)
- Статистические тесты

mailto:sorlik@borland.ru
http://sorlik.blogspot.com

- Анализ тенденций
- Предсказание (например, модели надежности)

Техники, основанные на статистических методах и статистические тесты часто предоставляют “снимок” наиболее проблемных областей исследуемого программного продукта (и, кстати, то же часто верно и в отношении процессов, *прим. автора*). Результирующие графики и диаграммы визуально помогают лицам, принимающим решения, в определении аспектов, на которых необходимо сфокусировать ресурсы <проекта>. Результаты анализа тенденций могут демонстрировать, что нарушается расписание, например, при тестировании; или что сбои определенных классов становятся все более частыми до тех пор, пока не предпринимаются корректирующие действия в процессе разработки. Техники предсказания помогают в планировании времени тестов и в предсказании сбоев. Более детальное обсуждение вопросов, касающихся количественных оценок, можно найти в областях знаний SWEBOK “Процесс программной инженерии” и “Управление программной инженерией”. Более специализированная информация по метрикам, используемым при тестировании, представлена в области знаний “Тестирование программного обеспечения”.

SWEBOK предоставляет ссылки на источники, в которых более подробно рассматриваются аспекты анализа дефектов (defect analysis), количественной оценки возникновения дефектов и последующего применения статистических методов для формирования понимания типов наиболее часто встречающихся типов дефектов и отвечая на вопрос соответствующей оценки плотности дефектов <различных типов>. Они могут, также, помочь в понимании тенденций и оценке того, насколько хорошо работают техники обнаружения дефектов и насколько успешно развиваются (как в плане выполнения, так и в контексте совершенствования, *прим. автора*) процессы разработки и сопровождения. Оценка покрытия тестами (test coverage) облегчает формирование ожиданий в отношении оставшегося объема тестирования и предсказании возможного количества дефектов, которые будут еще обнаружены <до окончания процесса тестирования>. На основе этих методов количественной оценки могут быть сформированы, так называемые профили дефектов (defect profiles) для конкретных прикладных областей (application domains). В дальнейшем, для будущих программных систем в данной организации, такие профили могут направлять процессы SQM, увеличивая усилия, направленные на наиболее вероятные источники проблем в создаваемых продуктах. Аналогично этому, результаты эталонных сравнений (benchmarks) или типовое для данной прикладной области количество дефектов могут служить в качестве вспомогательных средств для определения момента, когда продукт готов для передачи в эксплуатацию (помните обсуждение концепции “приемлемого качества”?, *прим. автора*).

Обсуждение вопросов использования данных, полученных в результате SQM-деятельности, в целях улучшения процессов разработки и сопровождения, представлено в областях знаний SWEBOK “Управление программной инженерией” и “Процесс программной инженерии”.

mailto:sorlik@borland.ru
http://sorlik.blogspot.com