МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

имени В.Н. КАРАЗИНА

Реферат по Управление разработкой программных проектов на тему:

**«Модели качества процессов ЖЦ ПО SPICE»**

Выполнили студенты

факультета математики и информатики

группы МФ-51

Минаев А.Ю.

Радченко А. Д.

Харьков 2017

**Содержание**

1. ISO 15504 (SPICE) – происхождение и структура 3

2. SDLC standard ISO 12207 6

2.1 Верификация и валидация ПО 11

2.2 Тестирование 13

3. Стандарты качества программного обеспечения 20

3.1. Управление качеством ПО на стадиях жизненного цикла 25

3.2. Современные модели качества программного обеспечения 34

4. Содержание концепции всеобщего управления качеством (TQM) 42

4.1. Преимущества и недостатки TQM 45

5. Понятие качества. Основные показатели качества деятельности предприятия 46

1. ISO 15504 (SPICE) – происхождение и структура

Аббревиатура SPICE раскрывается как Software Process Improvement and Capability dEtermination, что можно перевести, как: «Оценка возможностей и улучшения процесса разработки программного обеспечения».

Основные цели SPICE:

* удовлетворение растущих потребностей в оценке возможностей процессов производства ПО в подразделениях;
* гармонизация методов и моделей, используемых для оценки процессов.

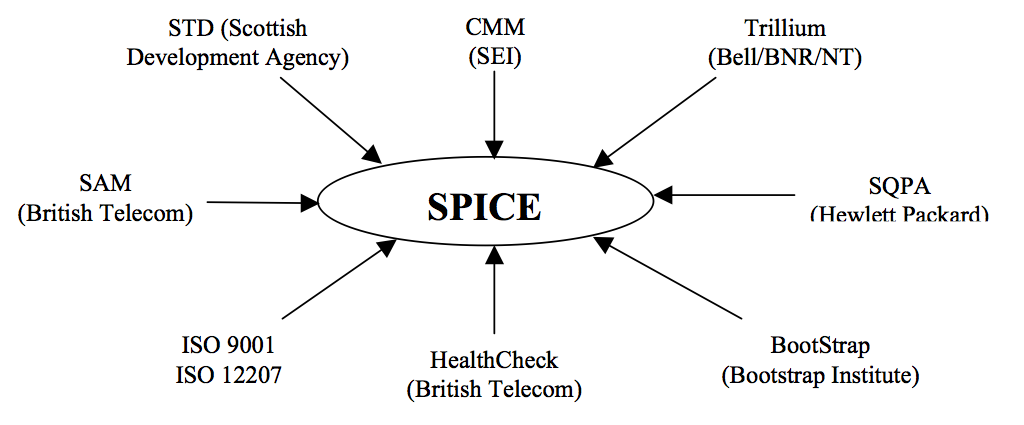
Проект SPICE был начат ISO в июле 1991 года и к настоящему времени объединил лучшие из существующих в мире практик. Архитектура SPICE двумерная и состоит из так называемых "уровней возможностей", их насчитывается 6 (плюс 9 атрибутов процессов и 32 правила менеджмента); категорий процессов (5) и типовых процессов (29), а также типовых практик (200).

Рис.9. Источники для составления стандарта SPICE

Аттестационные возможности SPICE:

* оценивая множество характеристик процессов и документов, дает достаточно объективное представление о процессах;
* дает повторяемые результаты, поэтому на их основе можно сравнивать организации;
* принимает во внимание контекст, в котором выполняются аттестуемые процессы;
* подходит ко всем сферам приложений и для организаций любого размера.

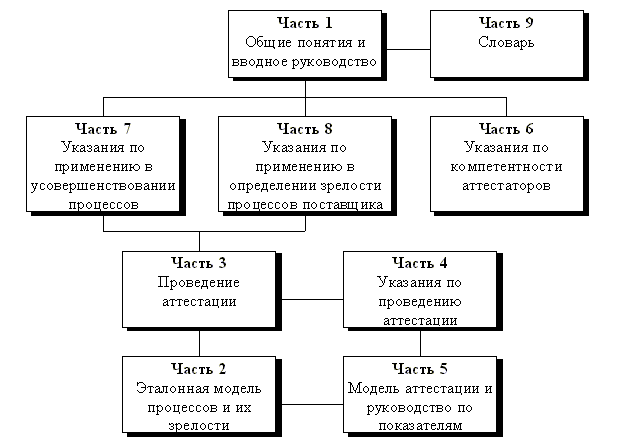


Рис.10. Состав ISO 15504

SPICE предлагает достаточно законченную и подробную модель, предоставляющую пользователям достаточную свободу в выборе путей к улучшению работы. Модель улучшения процессов в SPICE двумерная, где по одной оси откладывается эффективность работы (удовлетворенность заказчиков, качество продукции и продуктивность), по другой – возможности персонала и процесса. Таким образом, можно выбирать траекторию улучшения процесса в трехмерном пространстве, где улучшения по каждой из осей идут параллельно с улучшениями по другой. Собственно, параллельность не является требованием, это, скорее, рекомендация, позволяющая избежать серьезных перекосов в процессе производства.

1. **Оценка процесса** происходит путем сравнения процесса разработки ПО, существующего в данной организации, с описанной в стандарте моделью. Анализ результатов, полученных на этом этапе, помогает определить сильные и слабые стороны процесса, а также внутренние риски, присущие данному процессу, помогает оценить эффективность процессов, определить причины ухудшения качества и связанные с этим издержки во времени или стоимости.
2. **Определение возможностей процесса** позволяет оценить возможности улучшения данного процесса. Очень часто определение возможностей процесса производится компанией-поставщиком, чтобы убедить существующих или потенциальных заказчиков в своей способности достичь заданных показателей.
3. В результате предыдущих шагов, в организации может появиться понимание необходимости **улучшения** того или иного **процесса.** К этому моменту цели совершенствования процесса уже четко сформулированы и остается только техническая реализация поставленных задач. После этого весь цикл работ начинается сначала.

Хотя, как уже говорилось, SPICE вобрал в себя все самое лучшее из целого ряда популярных стандартов, он не стал простым их объединением. Для того чтобы показать, чем же SPICE отличается от своих предшественников стоит провести сопоставление SPICE и наиболее известных стандартов из мира ПО.

В таблице 1 приведен список уровней возможностей модели SPICE и характерные для них процедуры управления (на данный момент не существует русского перевода стандарта SPICE, поэтому использованные термины не являются общепринятыми или официально зарегистрированными).

Таблица 1. Уровни возможностей процесса в стандарте SPICE

|  |  |
| --- | --- |
| **Уровни** | **Название** |
| **Уровень 0** | **Процесс не выполняется** |
| **Уровень 1** | **Выполняемый процесс** |
| 1.1 | Измерение производительности процесса |
| **Уровень 2** | **Управляемый процесс** |
| 2.1 | Управление производительностью |
| 2.2 | Управление созданием продуктов |
| **Уровень 3** | **Установленный процесс** |
| 3.1 | Документирование процесса |
| 3.2 | Отслеживание ресурсов процесса |
| **Уровень 4** | **Предсказуемый процесс** |
| 4.1 | Измерение процесса |
| 4.2 | Управление процессом |
| **Уровень 5** | **Оптимизирующий процесс** |
| 5.1 | Изменение процесса |
| 5.2 | Постоянное совершенствование |

**Сравнение SPICE и ISO 12207**

Итак, ISO 12207 изначально создавался как стандарт, который:

* ориентирован на программную индустрию;
* используется в специфическом контексте разработки ПО;
* реализует процессный подход;
* предоставляет более детальную модель процессов (во многом);
* полностью совместим со SPICE.

2. SDLC standard ISO 12207

Стандарт ISO 12207 является удачной попыткой применения процессного подхода для компаний-разработчиков ПО.

Первая редакция ISO 12207 была подготовлена в 1995 году объединенным техническим комитетом ISO/IEC JTC1 "Информационные технологии, подкомитет SC7, проектирование программного обеспечения”.

Настоящий стандарт устанавливает, используя четко определенную терминологию, общую структуру процессов [жизненного](http://click.begun.ru/kick.jsp?url=4vrJyDwKGbW0TlsJvBaWfpF2Pon6bC8DAOemdA81YoFulTbNptYd2KAWAMgJ7MaSwZxT7FYJ8A7GRQUDFm4Zouds3tDqGo3bBu1iXSI8e_C3m3IinK2ZfVHKXOhPpFvrA1Sk_G91QppOGC5c4Kyd-H2b_r8dk3PQ1dArUGNYN76ib9ERvGJu61l50YjjnexYf6UNZyaM1FYcqq-idM5Tiah0_bsvF_paw46uMqhpFGSc55vNWy9FmmD63Q4Jvvgcx4qPocB04-4ybFwMzbBSBFj9Nni4VG_41leH4TRsXh2eNSQj6d27fujW015ApQlyUqRqtsI_hk4hF5OVEC4IbDBQNXMPrk5BOzqBrE8V4M9BZzdibVYVozlH2VecC6XpQQ8R5kk-zEpGahqGNh5BbbT5Jz8t7WuzSE-dXqaylAvmez4KlhwN2g67jbdWgV0QWT-lVGEDTZGlcX4TiTG6qw) цикла программных средств, на которую можно ориентироваться в программной индустрии. Далее он определяет процессы, работы и задачи, которые используются: при приобретении системы, содержащей программные средства, или отдельно поставляемого программного продукта; при оказании программной услуги, а также при поставке, разработке, эксплуатации и сопровождении программных продуктов. Понятие программных средств также охватывает программный компонент программно-аппаратных средств. ISO 12207 определяет также процесс, который может быть использован при определении, контроле и модернизации процессов жизненного цикла программных средств и набор процессов, работ и задач, предназначенных для адаптации к условиям конкретных программных проектов. Процесс адаптации (настройки на проект) заключается в исключении неприменяемых в условиях конкретного проекта процессов, работ и задач.

По определению, ISO 12207 – базовый стандарт процессов ЖЦ ПО, ориентированный на различные виды ПО и типы проектов разработки информационных систем. Стандарт определяет стратегию и общий порядок в создании и эксплуатации ПО, он охватывает ЖЦ ПО от концептуализации идей до завершения ЖЦ программного продукта.

Нужно отметить, что процессы, используемые во время ЖЦ ПО, должны быть совместимы с процессами, используемыми во время ЖЦ информационной системы. Стандарт ISO 12207 равносильно ориентирован на организацию действий каждой из двух сторон: поставщик (разработчик) и покупатель (пользователь); может быть в равной степени применен, когда обе стороны – из одной организации.

**Общая структура стандарта.**

Процессы ЖЦ. По сравнению с известными стандартами ISO состоит из гораздо более крупных обобщенных процессов: "приобретение", "поставка", "разработка" и т. п.

Каждый процесс разделен на набор действий, каждое действие – на набор задач. Очень важное отличие от ISO 9001: каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем нет заранее определенных последовательностей (естественно, при сохранении логики связей по исходным сведениям задач и т. п.).

**Структура процессов**

В стандарте ISO 12207 дана четкая классификация процессов ЖЦ ПО: 5 основных процессов, 8 вспомогательных и 4 организационных.

**5 основных** процессов ЖЦ ПО:

1. Процесс заказа. Определяет работы заказчика, то есть организации, которая приобретает систему, программный продукт или программную услугу.
2. Процесс поставки. Определяет действия предприятия-поставщика, которое снабжает покупателя системой, программным продуктом или сервисом ПО.
3. Процесс разработки. Определяет работы разработчика, то есть организации, которая проектирует и разрабатывает программный продукт.
4. Процесс эксплуатации. Определяет работы оператора, то есть организации, которая обеспечивает эксплуатационное обслуживание вычислительной системы в заданных условиях в интересах пользователей.
5. Процесс сопровождения. Определяет работы персонала сопровождения, то есть организации, которая предоставляет услуги по сопровождению программного продукта, состоящие в контролируемом изменении программного продукта с целью сохранения его исходного состояния и функциональных возможностей. Данный процесс охватывает перенос и снятие с эксплуатации программного продукта.

**8 вспомогательных** процессов, которые поддерживают реализацию другого процесса, будучи неотъемлемой частью всего ЖЦ программного изделия, и обеспечивают должное качество проекта ПО:

1. процесс документирования;
2. процесс управления конфигурацией;
3. процесс обеспечения качества;
4. процесс верификации;
5. процесс аттестации;
6. процесс совместного анализа;
7. процесс аудита;
8. процесс решения проблем.

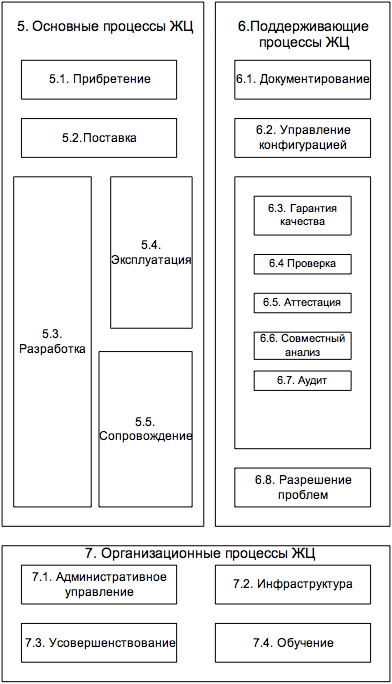


Рис. 8. Процессная область стандарта ISO 12207

Приводится по: Международному стандарту ISO 12207

**4 организационных процесса**.

Они применяются в какой-либо организации для создания и реализации основной структуры, охватывающей взаимосвязанные процессы жизненного цикла и соответствующий персонал, а также для постоянного совершенствования данной структуры и процессов. Эти процессы, как правило, являются типовыми, независимо от области реализации конкретных проектов и договоров; однако уроки, извлеченные из таких проектов и договоров, способствуют совершенствованию организационных вопросов.

1. Процесс управления.
2. Процесс создания инфраструктуры.
3. Процесс усовершенствования. Определяет основные работы, которые организация (заказчика, поставщика, разработчика, оператора, персонала сопровождения или администратора другого процесса) выполняет при создании, оценке, контроле и усовершенствовании выбранных процессов жизненного цикла.
4. Процесс обучения.

К ним примыкает особый Процесс адаптации, который определяет основные действия, необходимые для адаптации стандарта ISO 12207 к условиям конкретного проекта.

Каких-либо этапов, фаз, стадий не предусмотрено, что дает описываемую ниже степень адаптивности.

**Особенности**

"Динамический" характер стандарта определяется способом определения последовательности выполнения процессов и задач, при котором один процесс при необходимости вызывает другой или его часть.

Примеры:

* выполнение процесса приобретения в части анализа и фиксации требований к системе или ПО может вызывать исполнение соответствующих задач процесса разработки;
* в процессе поставки поставщик должен управлять субподрядчиками согласно процессу приобретения и выполнять верификацию и аттестацию по соответствующим процессам и т.д.

Такой характер позволяет реализовывать любую модель ЖЦ. Таким образом, для всех этапов жизненного цикла программных продуктов разработчик должен самостоятельно разрабатывать комплексы показателей качества, которые совокупности образуют систему показателей. Он также самостоятельно должен выявлять факторы, влияющие на качество. Только структурированный индивидуальный подход к выбору и обоснованию показателей и факторов обеспечивает эффективный контроль и управление качеством.

2.1 Верификация и валидация ПО

Верификация и валидация являются видами деятельности, направленными на контроль качества программного обеспечения и обнаружение ошибок в нем. Имея общую цель, они отличаются источниками проверяемых в их ходе свойств, правил и ограничений, нарушение которых считается ошибкой. Верификация проверяет соответствие одних создаваемых в ходе разработки и сопровождения ПО артефактов другим, ранее созданным или используемым в качестве исходных данных, а также соответствие этих артефактов и процессов их разработки правилам и стандартам. В частности, верификация проверяет соответствие между нормами стандартов, описанием требований (техническим заданием) к ПО, проектными решениями, исходным кодом, пользовательской документацией и функционированием самого ПО. Кроме того, проверяется, что требования, проектные решения, документация и код оформлены в соответствии с нормами и стандартами, принятыми в данной стране, отрасли и организации при разработке ПО, а также — что при их создании выполнялись все указанные в стандартах операции, в нужной последовательности. Обнаруживаемые при верификации ошибки и дефекты являются расхождениями или противоречиями между несколькими из перечисленных документов, между документами и реальной работой программы, между нормами стандартов и реальным процессами разработки и сопровождения ПО. При этом принятие решения о том, какой именно документ подлежит исправлению (может быть, и оба) является отдельной задачей. Валидация проверяет соответствие любых создаваемых или используемых в ходе разработки и сопровождения ПО артефактов нуждам и потребностям пользователей и заказчиков этого ПО, с учетом законов предметной области и ограничений контекста использования ПО. Эти нужды и потребности чаще всего не зафиксированы документально — при фиксации они превращаются в описание требований, один из артефактов процесса разработки ПО. Поэтому валидация является менее формализованной деятельностью, чем верификация. Она всегда проводится с участием представителей заказчиков, пользователей, бизнес-аналитиков или экспертов в предметной области — тех, чье мнение можно считать достаточно хорошим выражением реальных нужд и потребностей пользователей, заказчиков и других заинтересованных лиц. Методы ее выполнения часто используют специфические техники выявления знаний и действительных потребностей участников.

Различие между верификацией и валидацией проиллюстрировано на Рис. 1.



Приведенные определения получены некоторым расширением определений из стандарта IEEE 1012 на процессы верификации и валидации. В стандартном словаре терминов программной инженерии IEEE 610.12 1990 года определение верификации по смыслу примерно то же, а определение валидации несколько другое — там говорится, что валидация должна проверять соответствие полученного в результате разработки ПО исходным требованиям к нему. В этом случае валидация являлась бы частным случаем верификации, что нигде в литературе по программной инженерии не отмечается, поэтому, а также потому, что оно поправлено в IEEE 1012 2004 года, это Нужды и потребности пользователей Проектные решения Исходный код Работающее ПО Стандарты, нормы, правила Требования Верификация Валидация Процессы разработки 9 определение следует считать неточным. Частое использование фразы B. Boehm’а: «Верификация отвечает на вопрос "Делаем ли мы продукт правильно?", а валидация — на вопрос "Делаем ли мы правильный продукт?"» также добавляет путаницы, поскольку афористичность этого высказывания, к сожалению, сочетается с двусмысленностью. Однако многочисленные труды его автора позволяют считать, что он подразумевал под верификацией и валидацией примерно те же понятия, которые определены выше. Указанные разночтения можно проследить и в содержании стандартов программной инженерии. Так, стандарт ISO 12207 считает тестирование разновидностью валидации, но не верификации, что, по-видимому, является следствием использования неточного определения из стандартного словаря. В среде исследователей, занимающихся теоретической информатикой (computer science), широко распространено более узкое понимание термина «верификация» — только как формальной верификации. В данном обзоре мы будем пользоваться более широким, инженерным понятием, хотя различные методы формальной верификации тоже будут рассмотрены.

2.2 Тестирование

Тестирование (testing) является методом верификации, в рамках которого результаты работы тестируемой системы или компонента в ситуациях из выделенного конечного набора проверяются на соответствие проектным решениям, требованиям, общим задачам проекта, в рамках которого эта система разрабатывается или сопровождается. Ситуации, в которых выполняется тестирование, называют тестовыми ситуациями (test situations, test purposes), а процедуры, описывающие процесс создания этих ситуаций и проверки, которые необходимо выполнить над полученными результатами, — тестами.

Часто говорят, что тестирование должно проверять соответствие работы ПО требованиям к ней, однако «требования» здесь нужно понимать очень широко, в частности, включать в них детали проектных решений, определяемых на достаточно поздних этапах проекта. Довольно часто проводят тестирование, в котором проверяют только то, что система не «падает», не создает исключительных ситуаций или «странных», совершенно неожиданных для разработчиков результатов. Все эти ограничения обычно можно считать несформулированной явно, но подразумеваемой частью требований (стоит помнить, однако, что системы некоторых типов в определенных ситуациях должны «падать» или создавать исключительные ситуации). Кроме того, чем меньше при тестировании задействованы достаточно точные требования к тому, что система должна делать, чем более оно ориентировано лишь на поиск сбоев или других видов инцидентов, тем меньше пользы оно приносит проекту. Поэтому такое тестирование оправдано только как прелюдия к серьезному тестированию, для проверки того, что система в целом работоспособна, или в тех ситуациях, где описание требований вообще отсутствует, или же они сформулированы крайне нечетко и неполно.

Тестирование может использоваться и для валидации, в этом случае оно проверяет соответствие поведения ПО ожиданиям и потребностям пользователей и заказчиков, а для оценки этого соответствия должны привлекаться сами пользователи, их представители, бизнес-аналитики или эксперты в предметной области.

Тестирование, как и верификация вообще, служит для поиска ошибок или дефектов и для оценки качества ПО. Эффективность решения обеих этих задач во многом определяется тем, какой именно набор тестовых ситуаций выбран для проведения тестирования. Чтобы иметь некоторые гарантии аккуратности полученных в ходе тестирования оценок качества, необходимо выбирать тестовые ситуации систематическим образом, в соответствии с основными задачами и рисками проекта. Правила, определяющие набор необходимых тестовых ситуаций, называют критериями полноты (или адекватности) тестирования (test adequacy criteria). Обычно такой критерий использует разбиение всех возможных при работе проверяемого ПО ситуаций на некоторые классы эквивалентности, такие, что ситуации из одного класса достаточно похожи друг на друга и работа ПО в них не должна отличаться сколь-нибудь значительным образом. Полноту тестирования при этом можно определять по проценту задействованных, «покрытых» в нем классов ситуаций. Такой критерий полноты называется критерием тестового покрытия (test coverage criterion), а процент покрытых в результате работы тестов классов ситуаций — достигнутым тестовым покрытием.

Заметим, что различных критериев полноты довольно много (см. ниже), а тестирование, покрывшее 100% выделенных по одному из таких критериев классов ситуаций, и поэтому часто называемое полным, в действительности не является полным или исчерпывающим в каком-либо смысле, связанным со строгими гарантиями отсутствия ошибок и соответствия ПО требованиям. Такого рода исчерпывающее тестирование невозможно для практически значимых систем.

Подготовка и проведение тестирования в проекте создания или сопровождения ПО проходят примерно по следующему плану.

* Определение целей тестирования (test objectives, test goals) на основе задач и рисков проекта. Такие цели очерчивают проверяемые характеристики и свойства ПО, тщательность тестирования отдельных компонентов и подсистем. Они также определяют используемые в проекте виды тестирования и методы построения тестов.
* Определение требований, свойств и ограничений, которые должны проверяться в ходе тестирования.
* Определение критерия полноты тестирования, который будет использоваться в данном проекте. Критерий полноты должен быть согласован с целями тестирования, он управляет выбором тестовых ситуаций для тестирования, а также определяет, когда можно прекратить тестирование.
* Построение набора тестов, нацеленных на достижение выбранного критерия полноты и проверяющих определенные ранее требования и ограничения.
* Отладка, выполнение тестов и получение итоговых результатов тестирования в виде сообщений о выполненных действиях и нарушениях проверяемых ограничений. Отладка тестов включает пробные прогоны, устранение обнаруживаемых ошибок в самих тестах, а также описаниях проектных решений и требований, на основе которых получены тесты. По результатам отладки может потребоваться пополнение набора тестов для достижения максимального возможной полноты тестирования в рамках выделенных ресурсов.
* Анализ результатов тестирования, составление отчетов о найденных дефектах и о полученных оценках качества ПО.

Виды тестирования

Классификация видов тестирования достаточно сложна, потому что может проводиться по нескольким разным аспектам.

* По уровню или масштабу проверяемых элементов системы тестирование делится на следующие виды.
  + Модульное или компонентное (unit testing, component testing) — проверка корректности работы отдельных компонентов системы, выполнения ими своих функций и предполагаемых проектом характеристик.
  + Интеграционное (integration testing) — проверка корректности взаимодействий внутри отдельных групп компонентов.
  + Системное (system testing) — проверка работы системы в целом, выполнения ею своих основных функций, с использованием определенных ресурсов, в окружении с заданными характеристиками.
* По проверяемым характеристикам качества тестирование может быть тестирование функциональности, производительности (и по времени, и по другим ресурсам), надежности, переносимости или удобства использования. Более специфические виды тестирования, нацеленные на оценку отдельных атрибутов, — тестирование защищенности, совместимости или восстановления при сбоях. Специфическим видом тестирования, нацеленным на минимизацию риска того, что в результате доработки или внесения ошибок качество системы изменилось в худшую сторону, является регрессионное тестирование (regression testing). При его проведении используется уже применявшийся ранее набор тестов, и оно должно выявить различия между результатами, полученными на этих тестах ранее, и наблюдаемыми после внесения изменений.
* По источникам данных, используемых для построения тестов, тестирование относится к одному из следующих видов.
  + Тестирование черного ящика (black-box testing, часто также называется тестированием соответствия, conformance testing, или функциональным тестированием, functional testing) — нацелено на проверку соблюдения требований. Использует критерии полноты, основанные на требованиях, и техники построения тестов, использующие только информацию, заданную в требованиях к проверяемой системе. Частными случаями этого вида тестирования являются тестирование на соответствие стандартам и квалификационное или сертификационное тестирование, нацеленное на получение некоторого сертификата соответствия определенным требованиям или стандартам.
  + Тестирование белого ящика (white-box testing, glass-box testing, также структурное тестирование, structural testing)— нацелено на проверку корректности работы кода. Использует критерии полноты и техники построения тестов, основанные на структуре проверяемой системы, ее исходного кода.
  + Тестирование серого ящика (grey-box testing) использует для построения тестов как информацию о требованиях, так и коде. На практике оно встречается чаще, чем предыдущие крайние случаи.
  + Тестирование, нацеленное на ошибки — использует для построения тестов гипотезы о возможных или типичных ошибках в ПО такого же типа, как проверяемое. К этому типу относятся, например, следующие виды тестирования.
    - Тестирование работоспособности (sanity testing, smoke testing), нацеленное на проверку того, в систему включены все ее компоненты и операции, и система не дает сбоев при выполнении своих основных функций в простейших сценариях использования.
    - Тестирование на отказ, пытающееся найти ошибки в ПО, связанные с контролем корректности входных данных.
    - Нагрузочное тестирование (load testing), проверяющее работоспособность ПО при больших нагрузках — больших объемах входных, выходных или промежуточных данных, большой сложности решаемых задач, большом количестве пользователей, работающих с ПО и пр.
    - Тестирование в предельных режимах (stress testing), проверяющее работоспособность ПО на границах его возможностей и на границах той области, где оно должно использоваться.
* По роли команды, выполняющей тестирование, оно может относиться к следующим видам.
  + Внутреннее тестирование выполняется в рамках проекта по разработке системы силами организации-разработчика ПО.
  + Независимое тестирование выполняется третьими лицами (не разработчиками, не заказчиками и не пользователями) для получения объективных и аккуратных оценок качества системы.
  + Аттестационное тестирование (приемочные испытания) выполняется представителями заказчика непосредственно перед приемкой системы в эксплуатацию для проверки того, что основные функции системы реализованы. Обычно аттестационные тесты являются очень простыми тестами функциональности и производительности.
  + Пользовательское тестирование осуществляется силами пользователей системы. У него есть два часто упоминаемых частных случая.
    - Альфа-тестирование (alpha-testing) выполняется самими разработчиками, но в среде, максимально приближенной к рабочему окружению системы и на наиболее вероятных сценариях ее реального использования.
    - Бета-тестирование (beta-testing) выполняется пользователями, желающими познакомиться с возможностями системы до ее официального выпуска и передачи в эксплуатацию.

3. Стандарты качества программного обеспечения

В настоящее время существует несколько определений качества, которые в целом совместимы друг с другом. Приведем наиболее распространенные:

Определение ISO: Качество - это полнота свойств и характеристик продукта, процесса или услуги, которые обеспечивают способность удовлетворять заявленным или подразумеваемым потребностям.

Определение IEEE: Качество программного обеспечения - это степень, в которой оно обладает требуемой комбинацией свойств.

Основным стандартом качества в области инженерии программного обеспечения в настоящее время является стандарт ISO/IEC 9126:1-4:2002. В дополнение к нему выпущен набор стандартов ISO/IEC 14598, регламентирующий способы оценки характеристик качества. В совокупности они образуют модель качества, известную под названием SQuaRE (Software Quality Requirements and Evaluation).

В соответствии со стандартом ISO 9126 общее представление о качестве программного средства (ПС) рекомендуется описывать тремя взаимодействующими и взаимозависимыми метриками характеристик качества, отражающими:

* внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся в характеристиках конечного продукта;
* внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапах жизненного цикла ПС;
* качество при использовании в процессе нормальной эксплуатации и результативность достижения потребностей пользователей с учетом затрат ресурсов.

Внешние и внутренние характеристики качества касаются свойств самой программной системы и отражают взгляд заказчика и разработчика на нее. Однако конечный пользователь ждет достижения максимального совокупного эффекта от применения ПС - повышения продуктивности работы и общей удовлетворенности программным продуктом. Такой взгляд на качество программной системы обозначается термином «качество при использовании» или «эксплуатационное качество» программного средства.

Атрибуты программной системы, характеризующие ее качество, измеряются с использованием метрик качества. Метрика - это комбинация конкретного метода измерения (способа получения значений), атрибута сущности и шкалы измерения (средства, используемого для структурирования получаемых значений). Метрика определяет меру атрибута - переменную, которой присваивается значение в результате измерения.

Определение требований к качеству, обычно начинается с перечисления внешних характеристик качества, отражающих требования к функционирующему программному продукту. Далее, для того чтобы количественно определить критерии качества, по которым будет осуществляться проверка и подтверждение соответствия программной системы предъявляемым к ней требованиям, специфицируются подходящие внешние измеримые свойства (внешние атрибуты) ПС и связанные с ними метрики, представляющие собой модели оценивания атрибутов, а также приемлемые диапазоны изменения значений (мер) соответствующих атрибутов.

Метрики, определение и применение которых возможно только для работающего на компьютере программного обеспечения, находящегося на стадии тестирования или функционирования в составе системы, называются внешними метриками. Внешние метрики обеспечивают заказчикам, пользователям и разработчикам возможность прослеживать и анализировать качество программного средства в ходе испытаний или опытной эксплуатации.

После определения требований к внешним метрикам специфицируются внутренние характеристики качества и внутренние атрибуты ПС. Они используются для планирования достижения требуемых внешних характеристик качества конечного программного продукта и их встраивания в промежуточные (рабочие) продукты ПС в ходе разработки. Далее определяются внутренние метрики качества. Понятия внутренних характеристик качества, атрибутов и метрик связывается с не работающими на компьютере рабочими продуктами ПС (документами, текстами кода, тестами и др.), получаемыми на стадиях разработки, предшествующих тестированию (определение требований, проектирование, кодирование).

Внутренние метрики дают возможность разработчикам, испытателям и заказчикам прогнозировать качество жизненного цикла программ и заниматься вопросами технологического обеспечения качества до того, как программное средство становится готовым к использованию продуктом. Внутренние метрики могут применяться в ходе проектирования и программирования к компонентам программной системы, таким, как спецификации, исходный программный код или документация. Основная цель применения внутренних метрик — обеспечивать получение требуемого внешнего качества.

Метрики качества в использовании отражают, в какой степени продукт удовлетворяет потребностям конкретных пользователей в достижении заданных целей. Эти метрики не отражены в числе шести базовых характеристик, регламентируемых стандартом ISO 9126-1 вследствие их общности, однако рекомендуются для интегральной оценки результатов функционирования и применения комплексов программ в стандарте ISO 9126-4.

Общий подход к моделированию качества программного обеспечения заключается в том, чтобы сначала идентифицировать небольшой набор атрибутов (характеристик) качества самого высокого уровня абстракции и затем в направлении «сверху вниз» разбить эти атрибуты на наборы подчиненных атрибутов. Стандарт ISO/IEC 9126 является типичным примером такого подхода. Для оценки качества программного средства используется шесть групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками. Характеристики и субхарактеристики в стандарте определены кратко, без комментариев и подробных рекомендаций по их применению к конкретным системам и проектам. Изложение имеет концептуальный характер и не содержит рекомендаций по выбору и упорядочению приоритетов, а также необходимого минимума критериев в зависимости от особенностей объекта, среды разработки, сопровождения и применения. Первая часть стандарта — ISO 9126-1 — распределяет атрибуты качества программных средств по шести характеристикам, используемым в остальных частях стандарта. Исходя из принципиальных возможностей их измерения, все характеристики могут быть объединены в три группы, к которым применимы разные категории метрик:

* категорийные, или описательные (номинальные) метрики используются для оценки функциональных возможностей программных средств;
* количественные метрики применимы для измерения надежности и эффективности сложных комплексов программ;
* качественные метрики в наибольшей степени соответствуют практичности, сопровождаемости и мобильности программных средств.

Вторая и третья части стандарта — ISO 9126-2 и ISO 9126-3 — посвящены формализации соответственно внешних и внутренних метрик характеристик качества сложных программных средств. Все таблицы содержат унифицированную рубрикацию, где отражены имя и назначение метрики; метод ее применения; способ измерения, тип шкалы метрики; тип измеряемой величины; исходные данные для измерения и сравнения; а также этапы жизненного цикла программного средства (по ISO 12207), к которым применима метрика. Четвертая часть стандарта — ISO 9126-4 — предназначена для покупателей, поставщиков, разработчиков, службы сопровождения ПО, пользователей и менеджеров качества. В ней обосновываются и комментируются выделенные показатели сферы использования программных средств и группы выбранных метрик для пользователей. Характеристики, используемые для оценки качества программного обеспечения, и уточняющие их субхарактеристики сведены в таблицу 1.

Табл. 1. Характеристики качества программного обеспечения

|  |  |
| --- | --- |
| Функциональные возможности (Functionality) | Способность программного обеспечения реализовать установленные или предполагаемые потребности пользователей |
| Пригодность для применения по назначению (Suitability) | Наличие и соответствие набора функций конкретным задачам |
| Правильность/корректность реализации требований (Accuracy) | Способность программного обеспечения обеспечивать правильность (или соответствие) результатов |
| Способность к взаимодействию с компонентами и средой (Interoperability) | Способность ПО взаимодействовать с конкретными системами |
| Согласованность (Compliance) | Способность программного обеспечения придерживаться соответствующих стандартов или соглашений, или подробных рекомендаций |
| Защищенность/безопасность функционирования (Security) | Способность ПО предотвращать несанкционированный доступ (случайный или преднамеренный) к программам и данным |
| Надежность (Reliability) | Способность программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени |
| Стабильность/уровень завершенности (Maturity) | Характеризуется частотой отказов, вызванных наличием ошибок в программном обеспечении |
| Устойчивость к ошибке (Fault tolerance) | Способность программного обеспечения поддерживать определенный уровень качества функционирования в случаях программных ошибок или нарушения определенного интерфейса |
| Восстанавливаемость после проявления дефектов (Recoverability) | Способность программного обеспечения восстанавливать уровень качества функционирования и данные, непосредственно поврежденные в случае отказа. Характеризуется необходимыми для этого затратами усилий и времени |
| Практичность (Usability) | Характеризуется объемом работ, требуемых для использования программного обеспечения определенным или предполагаемым кругом пользователей |
| Понятность функций и документации (Understandability) | Характеризует усилия пользователя по пониманию общей логической концепции ПО и его применимости |
| Изучаемость процессов функционирования и применения (Learnability) | Характеризует усилия пользователя по обучению применению программного обеспечения (например, оперативному управлению, вводу, выводу) |
| Простота использования (Operability) | Характеризует усилия пользователя по эксплуатации и оперативному управлению ПО |
| Эффективность (Efficiences) | Определяется соотношением между уровнем качества функционирования программного обеспечения и объемом используемых ресурсов при установленных условиях |
| Временная эффективность реализации комплекса программ (Time behavior) | Характеризуется временем отклика и скоростью выполнения функций |
| Используемость вычислительных ресурсов (Resource behavior) | Характеризуется объемом используемых ресурсов и продолжительностью использования ПО при выполнении функции |
| Сопровождаемость (Maintainability) | Характеризует объем работ, требуемых для проведения конкретных изменений (модификаций) |
| Анализируемость (Analysability) | Характеризует усилия, необходимые для диагностики недостатков или случаев отказов или определения составных частей для модернизации |
| Изменяемость компонентов и комплекса программ (Changeability) | Характеризует усилия, необходимые для модификации, устранения отказа или для изменения условий эксплуатации |
| Устойчивость (Stability) | Характеризует риск от непредвиденных эффектов модификации |
| Тестируемость изменений при сопровождении (Testability) | Характеризует усилия, необходимые для проверки модифицированного программного обеспечения |
| Мобильность (Portability) | Способность программного обеспечения быть перенесенным из одного окружения в другое |
| Адаптируемость к изменениям среды (Adaptability) | Характеризует удобство адаптации ПО к различным конкретным условиям эксплуатации, без применения других действий или способов, кроме тех, что предназначены для этого в рассматриваемом программном обеспечении |
| Простота установки/внедрения/инсталляции после переноса (Installability) | Характеризует усилия, необходимые для внедрения программного обеспечения в конкретное окружение |
| Соответствие (Confortncnce) | Способность программного обеспечения подчиняться стандартам или соглашениям, относящимся к мобильности |
| Взаимозаменяемость компонентов при корректировке комплекса программ (Replaceability) | Характеризует простоту и трудоемкость применения данного ПО вместо другого конкретного программного средства в среде этого средства |

3.1. Управление качеством ПО на стадиях жизненного цикла

Для того чтобы должным образом управлять качеством на каждой стадии жизненного цикла программного средства, необходимо рассматривать разнообразные аспекты качества и учитывать изменение представлений о качестве в ходе процесса разработки. Стандарт ISO/IEC 9126 предлагает варьировать взгляды на качество продукта по стадиям ЖЦ следующим образом:

* целевое качество – необходимое и достаточное качество, которое отражает реальные потребности пользователя. Поскольку потребности, заявленные заказчиком, не всегда отражают реальные нужды пользователей относительно качества программного продукта, и эти нужды могут изменяться после того, как были зафиксированы в документации проекта, целевое качество рассматривается разработчиками как концептуальная сущность, которая не может быть полностью определена в начале проекта и должна восприниматься как ориентир. Требования к целевому качеству должны по возможности включаться в спецификацию требований к качеству ПС и оцениваться путем измерения эксплуатационного качества по завершении разработки программного продукта.
* затребованное (установленное) качество продукта – это тот уровень значений характеристик внешнего качества, который фактически заявлен в спецификации требований к качеству и должен использоваться как цель для его проверки. Требования ко всем или некоторым характеристикам качества указываются в спецификации требований к ПС. Наряду с оптимальными значениями характеристик должны также указываться минимальные значения, что поможет заказчикам и разработчикам избежать ненужного превышения стоимости и сроков разработки.
* качество программного проекта – внутреннее качество программного средства, представленное в описании основных частей или всего проекта в целом, например, архитектуры программного обеспечения, структуры программ, стратегии проектирования интерфейса пользователя и т.п. Качество проекта – основа качества программного продукта, которое может быть лишь незначительно улучшено в ходе кодирования и тестирования.
* оцененное (или прогнозируемое) качество продукта – качество, которое оценивается или предсказывается как качество конечного программного продукта на каждой стадии разработки на основании характеристик качества программного проекта.
* качество поставляемого продукта – это качество готового к поставке продукта, как правило, протестированного в моделируемой среде на моделируемых данных.
* эксплуатационное качество – качество программной системы, измеряемое в терминах результатов ее использования, а не свойств. Пользователь оценивает только те атрибуты качества ПС, которые видны ему в ходе фактического использования, поэтому качество ПО в пользовательской среде может отличаться от качества в среде разработки из-за недоучета особенностей среды и сценариев применения программного средства и, как следствие, неадекватного тестирования.

Кроме того, необходимо учитывать, что каждая из характеристик качества имеет тот или иной вес (значимость) в определенной прикладной сфере и для соответствующего сообщества пользователей, отражая их точку зрения на качество.

Представления о качестве программного обеспечения различных участников программного проекта

Говоря о качестве программного обеспечения, следует иметь в виду, что взгляды различных участников программного проекта на важность достижения тех или иных характеристик качества существенно различаются.

Пользователь – это лицо, непосредственно взаимодействующее с программным обеспечением с целью выполнения определенных задач. С точки зрения пользователя задача программного обеспечения – повысить эффективность его работы, сделать ее более удобной.

Пользователи оценивают программное обеспечение без изучения его внутренних аспектов или того, как программное обеспечение создавалось. Для пользователя наибольшую важность имеют такие атрибуты качества ПО как функциональность и практичность. Для этого при разработке ПО следует уделить внимание двум основным моментам: обеспечению необходимой функциональности и проектированию пользовательского интерфейса.

Должная функциональность является решающим фактором удовлетворения пользователей. Однако покупатели и пользователи иногда не могут точно сформулировать свои требования к программной системе или на начальном этапе могут просто их не знать. Поэтому одна из наиболее важных проблем, которые должна решить компания по разработке ПО, – это понимание того, как покупатель или пользователь планирует использовать данный продукт.

Второй важной проблемой является разработка удобного и профессионального пользовательского интерфейса. Приступая к проектированию интерфейса следует помнить, что возможностей программы, которые не нашли отражения в ее интерфейсе просто не существует! Ничто так не раздражает пользователя как непонятный интерфейс. Не важно, каким образом обеспечивается функциональность системы – все это бесполезно, если пользователь не имеет к ней доступа. Система с хорошим пользовательским интерфейсом всегда ведет себя именно так, как ожидает пользователь.

При создании ПО коллектив разработчиков должен сделать некоторые предположения не только о функциональности, но и об условиях эксплуатации программы (например, сколько пользователей могут одновременно просматривать web – страницу, каково максимальное количество записей в базе данных или минимальное время отклика системы на выполнение определенных задач и т.п.). Иногда эти условия бывают явными, иногда – скрытыми. Поэтому для пользователя очень важен такой атрибут качества как надежность. Надежный (отказоустойчивый) продукт создан таким образом, что работает даже при условиях, выходящих за пределы предположений, принятых при его разработке.

Важным фактором, характеризующим надежность программного обеспечения, является среднее время наработки на отказ. Под отказом можно понимать зависание программы, порчу данных или что-либо другое, препятствующее выполнению программой ее функций. Среднее время наработки на отказ определяет какова вероятность того, что в программе будут выявлены остаточные дефекты. Если его значение велико, то расходы на поиск остаточных дефектов не будут оправданными.

Еще одним фактором, доказывающим надежность программного обеспечения, является работоспособность – процентное соотношение времени, в течение которого программное обеспечение доступно пользователям.

Для того чтобы система была работоспособной она должна иметь большое значение времени наработки на отказ. Достижение высоких показателей работоспособности требует длительного тестирования и значительных ресурсов, поэтому задача руководителя проекта – найти компромиссное решение между стоимостью разработки ПО, сроками поставки системы и показателями ее работоспособности.

Таблица 2: Время простоя ПО при различных значениях показателя работоспособности

|  |  |  |  |
| --- | --- | --- | --- |
| Работоспособность | Минуты | Часы | Дни |
| 99,99999% | 0,05256 | 0,000876 | 3,65Е-05 |
| 99,9999% | 0,5256 | 0,00876 | 0,000365 |
| 99,999% | 5,256 | 0,0876 | 0,00365 |
| 99,99% | 52,56 | 876 | 0,0365 |
| 99,9% | 525,6 | 8,76 | 365 |
| 99% | 5256 | 87,6 | 3,65 |
| 9,9% | 52560 | 876 | 36,5 |

Покупатель – это физическое лицо или организация, приобретающие программное обеспечение и являющиеся владельцем лицензии на его использование. Часто покупатели программного обеспечения не являются его пользователями. Основная проблема покупателя – это удовлетворение нужд пользователя, а также оптимизация расходов, связанных с приобретением и внедрением программного обеспечения.

Фактическая стоимость внедрения не входит в первоначальную стоимость ПО. Это затраты на инсталляцию и настройку ПО, на выполнение текущего технического обслуживания, установку дополнений и обновлений и т.п. Поэтому покупатель прежде всего обращает внимание на такие параметры качества ПО как функциональность, надежность, сопровождаемость и мобильность.

Легкость обслуживания на месте эксплуатации, простота установки дополнений и обновлений (если для установки новой версии программы система должна быть отключена на целую неделю, это создает значительные неудобства для пользователей) – это существенный плюс для покупателя при выборе ПО. Кроме этого, должна существовать возможность адаптации программного обеспечения для выполнения конкретных задач. Покупатель может не захотеть приспосабливать свою аппаратную инфраструктуру к приобретаемому программному приложению; он предпочтет сконфигурировать ПО так, чтобы оно хорошо работало в имеющемся окружении.

Ну и, конечно же, покупателя волнует стоимость ПО. В стоимость входит не только стоимость покупки программного продукта. Сюда может входить ежегодная лицензионная плата, стоимость установки, настройки и внутренней поддержки, стоимость дополнений и обновлений, стоимость дополнительного аппаратного обеспечения, необходимого для работы программы. Для многих (особенно плохо разработанных) систем стоимость владения может существенно превышать первоначальную стоимость покупки.

Инвесторы – это физические лица или организации, оплачивающие разработку. Инвестор ожидает, что его вложения принесут доход. Этот доход может быть получен непосредственно от продажи программного продукта или заключаться в повышении эффективности работы организации, для которой этот продукт предназначался. Если уделяется большое внимание качеству ПО, инвестор обычно оплачивает и эксплуатационную стоимость, включающую расходы на персонал, который оказывает техническую поддержку по телефону и выезжает на места, собирает отчеты о проблемах, возникающих при использовании программы, ищет и устраняет дефекты, создает и поставляет дополнения и обновления к программам. Для заказных программ и программ, предназначенных для длительного использования, эксплуатационная стоимость чаще всего превышает стоимость разработки. Из этого следует, что более важно создание продукта с ограниченной эксплуатационной стоимостью, чем с низкой стоимостью разработки. Одним из основных факторов, влияющих на эксплуатационную стоимость ПО, является его надежность, другим – сопровождаемость и стоимость выявления и устранения дефектов в программном коде и других неисправностей на местах.

Инвесторы программных проектов обычно надеются, что новая разработка станет основой поддержания конкурентоспособности на рынке. В последние годы при разработке ПО (особенно заказного) учитывается возможность повторного использования созданного программного кода для снижения себестоимости следующей версии системы. Кроме того, разработка программного продукта обычно приводит к появлению некоторой интеллектуальной собственности, которую также можно продать.

Итак, инвесторов, в основном, интересуют три аспекта:

* практичность программного продукта для пользователя и покупателя, что обеспечивает спрос на него и, следовательно, повышает доходы.
* эксплуатационная стоимость, которая увеличивает расходы, что может привести к снижению прибыли.
* стоимость интеллектуальной собственности.

Значимость основных характеристик качества для каждого из участников программного проекта схематично отражена в таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут качества | Пользователь | Покупатель | Инвестор |
| Функциональность | + | + | + |
| Надежность | + | + | + |
| Практичность | + |  | + |
| Эффективность | + |  |  |
| Сопровождаемость |  | + | + |
| Мобильность |  | + |  |

Разработчик – это физическое лицо или организация, непосредственно создающие программное обеспечение. Так как разработчики отвечают за то, что созданный программный продукт действительно удовлетворяет заданным требованиям, они заинтересованы в качестве промежуточных артефактов не меньше, чем и в качестве конечного продукта. Причем, при оценке результатов каждой промежуточной фазы цикла разработки, разработчики должны использовать различные метрики для одних и тех же характеристик качества. Так, например, пользователь понимает эффективность в терминах времени реакции, тогда как разработчик использует в проектной спецификации термины длины маршрута и времени ожидания и доступа. В то же время разработчики и заказчики и/или пользователи должны использовать одни и те же характеристики качества при приемке конечного продукта.

Руководитель программного проекта может быть более заинтересован в общем качестве, чем в конкретной характеристике качества, и по этой причине будет нуждаться в определении важности значений, отражающих коммерческие требования для индивидуальных характеристик.

Руководителю может также потребоваться сопоставление уровня повышения качества с критериями управляемости, такими как плановая задержка или перерасход стоимости, потому что он желает оптимизировать качество в пределах ограниченной стоимости, имеющихся трудовых ресурсов и установленного времени.

Время от времени пользователи находят пути нестандартного использования программного обеспечения. Возникающие при этом ошибки найти и устранить тяжелее. На практике в хорошем программном обеспечении дефекты легко обнаружить на ранней стадии тестирования. При продолжении тестирования надежность системы увеличивается и каждый последующий дефект обнаружить все труднее. При этом проведенные исследования показывают, что стоимость поиска дефекта возрастает в геометрической прогрессии. В итоге руководитель должен решить, когда нужно остановиться, т.е. в какой момент стоимость тестирования и потеря рынка придут в равновесие с ожидаемой эксплуатационной стоимостью. Именно в этот момент продукт считается готовым к выходу на рынок.

Руководителя программного проекта должно интересовать, какова будет стоимость разработки качественного ПО, и может ли компания позволить себе такие расходы. Какой уровень качества будет «по карману» для той или иной разработки?

Чтобы ответить на этот вопрос, прежде всего необходимо определить, какие из атрибутов качества приоритетны для каждой из сторон, заинтересованных в данной разработке, и найти компромиссное решение. Надо иметь в виду, что недостаточное качество приведет не просто к увеличению затрат (например на эксплуатационное и гарантийное обслуживание), но прежде всего к потере рынка, поэтому увеличение затрат на проектно-конструкторские и научно–исследовательские работы даже до 50% в целях достижения необходимого качества будет более чем оправдано в коммерческом смысле. Наиболее распространенным заблуждением является предположение, что качественное ПО – это то, которое полностью удовлетворяет требованиям и не имеет дефектов. В целом невозможно найти последний дефект и выпустить бездефектную программу. Для достижения этой цели потребуется такое тестирование, которое не может позволить себе ни одна организация. Кроме того, тестирование как средство достижения качества обладает ограниченными возможностями. В литературе приводятся примеры, когда в системах с особыми требованиями к безопасности после интенсивного тестирования и тысяч часов использования были обнаружены остаточные ошибки. И хотя обнаружение и устранение дефектов является важной задачей, программные средства часто бывают непригодными из-за дефектов разработки. Программа может работать так, как планировали программисты, она может пройти все тесты, но быть непригодной из-за того, что разработчики не понимали, что нужно пользователям. Чтобы принять решение о том, достаточно ли хорошим является разработанное ПО, надо сравнить стоимость непоставки программного продукта с эксплуатационной стоимостью в случае его поставки.

3.2. Современные модели качества программного обеспечения

Качество программного обеспечения должно оцениваться исходя из определенной модели качества. На выбор модели качества влияет множество различных факторов.

В зависимости от цели моделирования, модель качества может в разной степени отражать ожидания различных категорий участников программного проекта – менеджеров, разработчиков, персонала сопровождения, пользователей, – и содержать упорядоченное множество характеристик, обеспечивающих непротиворечивость их интересов. Учет приоритетов этих интересов (по значимости, по времени удовлетворения) кладется в основу разработки стратегии обеспечения качества. Как было сказано выше в большинстве случаев приоритетным, с точки зрения участников программного проекта, является взгляд менеджера проекта и его интересы в минимизации возможных рисков.

За исключением очень простых проектов, различные компоненты ПС обычно имеют совершенно разные требования к характеристикам качества. Например, компоненты, обеспечивающие интерфейс с пользователем, характеризуются высокими требованиями к удобству использования, а базы данных – повышенными требованиями к целостности. Концептуальное разделение единого программного продукта на компоненты, характеризующиеся однотипными требованиями к качеству, помогает снизить конфликты между различными требованиями.

В пределах одной организации процессы разработки могут иметь различное наполнение и варьироваться в зависимости от применяемых в проектах методологий и интегрированных инструментов. Эти различия ограничивают возможность использования одного и того же множества метрик по различным проектам. Например, не имеет смысла сравнивать число строк кода, разработанного с применением структурных методов, с числом строк, разработанных объектно-ориентированным методом.

Если организация – разработчик специализируется на выпуске однотипных программных продуктов, в ней постепенно складываются определенные подходы к обучению сотрудников, стратегии проектирования и программирования, приемы сбора и регистрации всевозможных данных, методологии выполнения измерений и т.д. Это дает возможность использовать собственный опыт разработки программных систем со схожими характеристиками, накапливать полезную историческую информацию по проектам, относящуюся к качеству. Как следствие таким организациям удается более точно строить модель качества, выбирать подходящие метрики, устанавливать количественные меры для всех характеристик качества и прогнозировать их достижение уже на ранних стадиях разработки.

Таким образом, модель качества представляет собой множество взаимосвязанных характеристик, образующих базис для спецификации требований к качеству и оценивания качества программного средства.

Существуют десятки различных подходов к обеспечению качества ПО, и у каждого из них есть как свои преимущества, так и недостатки. Одной из первых моделей качества стал стандарт ISO серии 9000, первая версия которого была выпущена в 1987 году. С тех пор сертификаты ISO серии 9000 сохраняют неизменную популярность и признаются во всем мире.

Однако время не стоит на месте, и методики, положенные в основу стандарта ISO 9001, постепенно устаревают. Наиболее существенными его недостатками считаются:

* недостаточная подробность стандарта, возможность самых различных его толкований в зависимости от представлений аудитора;
* неточность оценки качества процессов, задействованных при создании и внедрении программного обеспечения;
* отсутствие в стандарте механизмов, способствующих улучшению существующих процессов.

Перечисленные проблемы заставили профессиональное сообщество программистов разрабатывать более совершенные решения в области обеспечения качества, что привело к созданию в начале 90-х годов целого ряда новых стандартов и методологий. Примерами наиболее удачных и содержательных стандартов могут служить Capability Maturity Model (CMM) и ISO/IEC 15504 (SPICE).

Capability Maturity Model

Официальная история CMM начинается в 1991 году, когда американский институт программной инженерии SEI, существующий при университете Карнеги-Меллон, опубликовал первую версию этого стандарта.

Изначальной целью разработки стандарта было создание методики, позволяющей крупным правительственным организациям США выбирать наилучших поставщиков ПО. Для этого предполагалось создать исчерпывающее описание способов оценки процессов разработки ПО и методики их дальнейшего усовершенствования. В результате, авторам удалось добиться такой степени подробности и детализации, что стандарт оказался пригодным и для обычных компаний-разработчиков, желающих улучшить существующие процессы.

Базовым понятием модели СММ считается зрелость компании-разработчика. Незрелой считается организация, в которой процесс разработки программного обеспечения зависит только от конкретных исполнителей и менеджеров, и решения зачастую просто импровизируются «на ходу». Как следствие, здесь высока вероятность превышения бюджета или срыва сроков окончания проекта.

Напротив, в зрелой компании работают ясные процедуры управления проектами и построения программных продуктов. По мере необходимости эти процедуры уточняются и развиваются. Оценки длительности и затрат разработки точны, основываются на накопленном опыте. В компании существуют стандарты на процессы разработки, тестирования и внедрения ПО, а также на процессы взаимодействия с заказчиком. В ней четко определены правила оформления конечного программного кода, компонентов, интерфейсов и т.д. Все это создает среду, обеспечивающую качественную разработку программного обеспечения.

Можно сказать, что стандарт в целом состоит из критериев оценки зрелости организации и рецептов улучшения существующих процессов. В этом наблюдается принципиальное различие с моделью, принятой в ISO 9001, так как в ISO 9001 сформулированы только необходимые условия для достижения некоторого минимального уровня организованности процесса, и не дается никаких рекомендаций по его дальнейшему совершенствованию.

В модели CMM определено пять уровней зрелости организаций. В результате аттестации компании присваивается определенный уровень, который в дальнейшем может повышаться или (теоретически) понижаться. Начальный уровень (initial level) описан в стандарте в качестве основы для сравнения со следующими уровнями. Если компания-разработчик находится на начальном уровне организации, в ней фактически не существует стабильных условий для создания качественного программного обеспечения. Результат любого проекта целиком и полностью зависит от личных качеств менеджера и опыта программистов, причем успех в одном проекте может быть повторен только в случае назначения тех же менеджеров и программистов на следующий проект. Более того, если такие менеджеры или программисты уходят из компании, то с их уходом резко падает качество производимых программных продуктов. В стрессовых ситуациях процесс разработки сводится к написанию кода и его минимальному тестированию.

Для достижения повторяемого уровня (repeatable level) на предприятии должны быть внедрены технологии управления проектами. При этом планирование и управление проектами основывается на накопленном опыте, существуют стандарты на разрабатываемое программное обеспечение (причем обеспечивается следование этим стандартам!) и существует специальная группа обеспечения качества. Применяемые средства планирования и управления дают возможность повторения ранее достигнутых успехов. В случае необходимости организация может взаимодействовать с субподрядчиками. Однако в критических условиях процесс имеет тенденцию скатываться на начальный уровень.

Далее следует определенный уровень (defined level), который характеризуется тем, что стандартный процесс создания и сопровождения программного обеспечения задокументирован (включая и разработку ПО, и управление проектами). Подразумевается, что в процессе стандартизации происходит переход на наиболее эффективные практики и технологии. Для создания и поддержания подобного стандарта в организации должна быть создана специальная группа. Наконец, обязательным условием для достижения данного уровня является наличие на предприятии программы постоянного повышения квалификации и обучения сотрудников. Начиная с этого уровня, организация перестает зависеть от личностных и профессиональных качеств конкретных разработчиков, и не имеет тенденции скатываться на уровень ниже в стрессовых ситуациях.

На управляемом уровне (managed level) в организации устанавливаются количественные показатели качества – как на программные продукты, так и на процесс в целом. Это обеспечивает более точное планирование проекта и контроль качества его результатов. Основное отличие от предыдущего уровня состоит в более объективной, количественной оценке продукта и процесса. Таким образом, более совершенное управление проектами достигается за счет уменьшения отклонений различных показателей проекта. При этом осмысленные вариации в производительности процесса можно отличить от случайных вариаций (шума), особенно в хорошо освоенных областях.

Наконец, оптимизирующий уровень (optimizing level) характеризуется тем, что мероприятия по улучшению применяются не только к существующим процессам, но и для оценки эффективности ввода новых технологий. Основной задачей всей организации на этом уровне является постоянное улучшение существующих процессов. При этом улучшение процессов в идеале должно помогать предупреждать возможные ошибки или дефекты. Кроме того, должны вестись работы по уменьшению стоимости разработки программного обеспечения, например, с помощью создания и повторного использования компонентов.

Каждый уровень СММ характеризуется областью ключевых процессов (ОКП), причем считается, что каждый последующий уровень включает в себя все характеристики предыдущих уровней. Иначе говоря, для 3-го уровня зрелости рассматриваются ОКП 3-го уровня, ОКП 2-го уровня и ОКП 1-го уровня. Область ключевых процессов образуют процессы, которые при совместном выполнении приводят к достижению определенного набора целей. Если все цели ОКП достигнуты, компании присваивается сертификат данного уровня зрелости. Если хотя бы одна цель не достигнута, то компания не может соответствовать данному уровню СММ.

При сертификации проводится оценка соответствия всех ключевых областей по 10-балльной шкале. Для успешной квалификации данной ключевой области необходимо набрать не менее 6 баллов. Оценка ключевой области производится по следующим показателям:

1. Заинтересованность руководства в данной области (планируется ли практическое внедрение данной ключевой области, существует ли понимание у руководства необходимости данной области и т.д.).
2. Насколько широко данная область применяется в организации (например, оценке в 4 балла соответствует фрагментарное применение).
3. Успешность использования данной области на практике (например, оценке в 0 баллов соответствует полное отсутствие какого-либо эффекта, а оценка в 8 баллов выставляется при наличии систематического и измеримого положительного результата практически во всей организации).

В принципе, можно сертифицировать только один процесс или подразделение организации, например, подразделение разработки программного обеспечения компании IBM сертифицировано на пятый уровень. Кстати, в мире существует совсем немного компаний, которые могут похвастаться наличием у них пятого уровня CMM хотя бы в одном из подразделений – таких всего около 50. С другой стороны, насчитывается несколько тысяч компаний, сертифицированных по 3 или 4 уровню, то есть существует колоссальный разрыв между оптимизированным уровнем зрелости и предыдущими уровнями. Однако еще больший разрыв наблюдается между количеством организаций начального уровня и числом их более продвинутых собратьев – по некоторым оценкам, свыше 70% всех компаний-разработчиков находится на первом уровне CMM.

Интересен вопрос о соотношении уровней CMM со стандартом ISO 9001: на каком уровне CMM должна находиться организация, чтобы получить сертификат соответствия ISO 9001? На первый взгляд, для этого необходимо иметь как минимум 3 или 4 уровень CMM. Тем не менее, существуют примеры организаций 1-го уровня, имеющих сертификат ISO 9001. Одной из причин для возникновения подобных несуразиц является высокий уровень абстракции ISO 9001 и связанная с этим свобода его интерпретации аудитором, который иногда просто не знаком с реальной практикой разработки программ.

Большую роль в создании качественного ПО играют вопросы отбора, повышения квалификации и сохранения компетентных сотрудников в организации. Особенно важен подбор сотрудников для организаций первого уровня, так как квалификация персонала является для них единственной гарантией качества. Но и на более высоких уровнях зрелости «человеческий фактор» сохраняет свою значимость. Поэтому в 1995 году был опубликован стандарт People CMM, являющийся дополнением к Software CMM и имеющий, в целом, похожую структуру. Внедрение этого стандарта параллельно с обычным CMM обеспечивает организацию целым набором процедур по оценке и развитию всей системы найма, обучения и сохранения квалифицированных сотрудников.

К сожалению, использование CMM затрудняют следующие проблемы:

* стандарт CMM является собственностью Software Engineering Institute и не является общедоступным (в частности, дальнейшая разработка стандарта ведется самим институтом, без заметного влияния остальной части программистского сообщества);
* оценка качества процессов организаций может проводиться только специалистами, прошедшими специальное обучение и аккредитованными SEI;
* стандарт ориентирован на применение в относительно крупных компаниях.

С некоторыми свободно распространяемыми материалами по CMM можно познакомиться на сайте Software Engineering Institute по адресу: http://www.sei.cmu.edu/cmm.

ISO/IEC 15504 (SPICE)

В 1991 году Международная организация по стандартизации инициировала работу по созданию единого стандарта оценки программных процессов. Стандарт получил имя SPICE (сокращение от Software Process Improvement and Capability dEtermination, определение возможностей и улучшение процесса создания программного обеспечения). Официально стандарт называется ISO/IEC 15504: Information Technology - Software Process Assessment и на данный момент существует в качестве рабочей версии, последний выпуск которой состоялся в мае 1998 года. В этом стандарте была предпринята попытка учесть весь накопленный опыт в области разработки ПО. Он унаследовал многие черты более ранних стандартов, в том числе и уже упоминавшихся ISO 9001 и CMM. Для этого пришлось прибегнуть к повышению уровня детализации. Следствием такого основательного подхода является большой объем стандарта: документация к нему содержит около 500 страниц! Больше всего стандарт SPICE напоминает CMM. Точно так же, как и в CMM, основной задачей организации является постоянное улучшение процесса разработки ПО. Кроме того, в SPICE тоже используется схема с различными уровнями возможностей (в SPICE определено 6 различных уровней), но эти уровни применяются не только к организации в целом, но и к отдельно взятым процессам.

Оценка процесса происходит путем сравнения процесса разработки ПО, существующего в данной организации, с описанной в стандарте моделью. Анализ результатов, полученных на этом этапе, помогает определить сильные и слабые стороны процесса, а также присущие ему внутренние риски. Это помогает оценить эффективность процессов, определить причины ухудшения качества и связанные с этим издержки во времени или стоимости.

1. Определение возможностей процесса позволяет оценить возможности улучшения данного процесса. Очень часто определение возможностей процесса производится компанией-поставщиком, чтобы убедить существующих или потенциальных заказчиков в своей способности достичь заданных показателей.
2. В результате предыдущих шагов, в организации может появиться понимание необходимости улучшения того или иного процесса. К этому моменту цели совершенствования процесса уже четко сформулированы и остается только техническая реализация поставленных задач. После этого весь цикл работ начинается сначала.

Набор документов по стандарту SPICE состоит из 9 частей. Первая часть «Введение и основные концепции» и девятая часть «Словарь» носят чисто информативный характер. Самым важным элементом SPICE является оценка процессов, поэтому ей посвящена наибольшая часть документов, а именно части со второй по шестую. Остальные части стандарта – седьмая и восьмая – посвящены соответственно улучшению процесса, и определению возможностей процесса.

4. Содержание концепции всеобщего управления качеством (TQM)

Всеобщее управление качеством (Total Quality Management) - концепция, предусматривающая реализацию скоординированного, комплексного и целенаправленного внедрения и применения систем и методов управления качеством во всех сферах деятельности от стадии проектирования до этапа послепродажного обслуживания при рациональном использовании технического потенциала и активном участии работников, служащих и руководителей всех подразделений и звеньев организации.

Следует подчеркнуть, что всеобщее управление качеством представляет собой технологию руководства (управления) процессом повышения качества. Она состоит из системы применяемых методов и средств, системы технических средств и системы развития самих принципов, содержания данной технологии.

Методы и средства, применяемые для проведения исследования и анализа, - это центральная, ключевая система. Данные методы базируются на применении математико-статистических методов контроля и относятся к разряду всеобщих.

Приемы и программы для обучения персонала этим средствам и методам и их целенаправленное использование образуют систему технического обеспечения, которая характеризуется сильно выраженной национальной спецификой, создается для конкретной организации, конкретного производства.

Необходимым условием функционирования системы саморазвития является непрерывное совершенствование содержания и принципов самой концепции Всеобщего управления качеством. Она еще более подвержена воздействию национальных законодательств, мер регулирования экономической деятельности отдельных стран.

Более подробный перечень элементов концепции всеобщего управления качеством насчитывает 20 пунктов.

Цель всеобщего управления качеством - достижение более высокого уровня качества продукции и услуг

Согласно философии всеобщей концепции в понятие качества входят осязаемые и неосязаемые ощущения покупателя, которые связаны с качественными характеристиками продукции и услуг, их информационным обеспечением и другими обстоятельствами.

Сегодня качество в передовых странах столь важно, что компании, не внедрившие программы TQM, уже в ближайшем будущем не смогут эффективно продолжить конкурентную борьбу. Данный факт все чаще учитывается их потенциальными покупателями при выборе продавца. Процесс внедрения TQM включает все уровни управления организацией, предприятием, охватывает также контроль за поставщиком и системой распределения. Насчитывается семь основных этапов применения системы всеобщего контроля качества:

Первое - определить потребность рынка для разработки и производства товара определенного уровня качества. Лидирующее место - отделу маркетинга. Желательно привлечение клиента к разработке и, особенно, к процессу дальнейшего совершенствования продукции. Обязательное наличие обратной связи с рынком для наблюдения за тем, насколько хорошо принимается ваш товар. Без всеобъемлющей информации о достоинствах и недостатках вашей продукции вы будете лишены перспективы, обречены на провал.

Второе - организовать процесс разработки и проверки продукции или услуги. Уровень качества является определяющим при выборе вариантов. Необходимо тщательно проверить продукцию по завершении разработки. Желательно вовлечение в этот процесс потребителя.

Третье - нацелить организационно-технологический процесс на производство бездефектной продукции, для сведения до минимума затрат на дальнейший ремонт и переработку.

Четвертое - установить непрерывную связь с рынком. Вести документацию результатов контроля, продолжать процесс оценки товара. Организовать управление процессом поведения и оценки продукции потребителем.

Пятое - формулирование четких требований, предъявляемых вами к качеству поставляемого сырья, оборудования и материалов. Их соответствие необходимым стандартам. Выдвижение требований вашим посредникам-потребителям о гарантии сохранения качества продукции вашего предприятия в процессе транспортировки, складирования, хранения, упаковки, переупаковки и конечной доставке покупателю, заказчику.

Шестое - введите различные формы поощрений и мотиваций. Если работники больше времени и сил будут уделять вопросами повышения самоконтроля, это в конечном результате сократит затраты, уменьшит объем работы контролеров.

Седьмое - организуйте работу групп качества.

4.1. Преимущества и недостатки TQM

Концепция всеобщего управления качеством коренится в идее предоставления всех инструментов, обучения и опыта, необходимых для полного контроля качества организации. Она также может предоставить необходимое общение, которое позволит избежать ведомственных проблем, которые могли появиться в процессе в результате дорогостоящих задержек или ошибок. Для того чтобы понять концепцию всеобщего управления качеством, или TQM, мы написали эту статью. Для краткости, мы обозначили целый ряд преимуществ и недостатков в использовании TQM для управления предприятием, которые необходимо учитывать.

Преимущества: Низкая стоимость

TQM снижает затраты для всей деловой инфраструктуры в организации. Это происходит, потому что TQM помогает различным ведомствам сообщать друг другу о нуждах, проблемах и желаниях, так чтобы находить совместные эффективные решении. Эти решения помогут организации сократить издержки по всей цепочке поставок, распределения, доставки, получения, учета и управления отделами без потери производительности или способности действовать быстро в условиях изменений.

Недостатки: сопротивление переменам

Работники могут чувствовать, что их рабочие места или отдельные обязанности в рамках компании находятся в опасности при внедрении TQM, и, как следствие, они могут быть медленными в работе по новым стандартам. Кроме того, можно потерять квалифицированную рабочую силу. Общая атмосфера недовольства может негативно сказаться на распространении идей, возможен даже «бунт на корабле». Если сотрудники не смогут выполнять функции согласно новым стандартам, будут сопротивляться и подговаривать коллег, это приведет к увеличению расходов и торможению всего процесса. На курсах обучения руководителей, как правило, изучается вопрос работы с сопротивлением персонала

Улучшение репутации

TQM программы имеют преимущество в совершенствовании корпоративного имиджа, а также в установлении положительной деловой репутации продуктов на рынке, потому что ошибки и дефекты продукции обнаруживаются быстрее, чем в не TQM системах. Все дефекты товара устраняются еще до его попадания на рынок.

Недостатки: высокая стоимость времени

Высокая стоимость внедрения программы TQM, и тот факт, что от того момента как она будет полностью реализована до первых результатов и выгод пройдет несколько лет — огромный недостаток TQM, особенно в неопределенных экономических условиях. TQM следует рассматривать как долгосрочную инвестицию. Как правило, решение о внедрении той или иной системы качества принимается на стратегических сессиях, мозговых штурмах или на основе методик ТРИЗ или ТОС.

5. Понятие качества. Основные показатели качества деятельности предприятия

Ключевым фактором обеспечения эффективного применения программных продуктов является тщательное оценивание и достижение высоких значений показателей качества. Постоянный рост сложности функций, реализованных программами в информационных системах, непосредственно приводит к увеличению их объема и трудоемкости их создания. Согласно изменениям сложности программ возрастает количество найденных и оставленных в них дефектов и ошибок, что отражается на их качестве. Программа объемом в миллионы строк кода в принципе не может быть безошибочной. Проблема нахождения и устранения ошибок обостряется по мере увеличения сложности задач и программ, которые их решают, и угрожает катастрофами в областях, которые используют программное обеспечение (ПО) [1-3].

Проблема заключается в том, чтобы обеспечить нужное качество функционирования ПО с учетом того, что некоторое неизвестное количество ошибок и дефектов всегда остается в сложных комплексах программ, и должно быть блокировано или сокращено их негативное воздействие до допустимого уровня. В связи с этим стратегическая задача в жизненном цикле современного ПО - обеспечение качества программных продуктов [4].

Качество ПО определяется качеством методов и инструментальных средств, которые применялись для обеспечения всего их жизненного цикла. На практике важно оценивать качество программ не только в завершенном виде, но и в процессе их проектирования и разработки.

Сегодня ПО является определяющей составляющей многих систем, среди которых системы критического применения, встроенные и специализированные системы различного назначения. Для указанных систем наличие ошибок и низкое качество ПО грозят катастрофами, которые приводят к человеческим жертвам, экологическим катаклизмам, экономическим потерям. Развитие современных технологий разработки ПО требует динамичного развития средств оценки качества ПО, причем уже на этапе проектирования (с точки зрения экономической и временной целесообразности).

Современная индустрия ПО характеризуется высокой конкуренцией. Для успешной работы на этом рынке софтверная компания должна разрабатывать, внедрять и сопровождать ПО быстро, укладываясь в сроки, и с удовлетворительным качеством. Поэтому многие софтверные компании вкладывают средства в модернизацию технологий разработки ПО, помня о том, что такое вложение средств обязательно окупается.

На этапе проектирования важно заложить целый ряд требований по качеству: требования к структуре программы, требования к навигации по ПО, требования к дизайну интерфейсов пользователя, требования к мультимедиа-компонентам ПО, требования к удобству применения (usability), технические требования.

На этапе проектирования формируется ответ на вопрос "Каким образом программное обеспечение будет реализовывать выдвинутые к нему требования?".

Информационные потоки этапа проектирования ПО [5]: требования к ПО, представленные информационной, функциональной и поведенческой моделями анализа. Информационная модель описывает информацию, котрую должно обрабатывать ПО по мнению заказчика. Функциональная модель определяет перечень функций обработки информации и перечень модулей ПО. Поведенческая модель фиксирует желаемую динамику системы (режимы ее работы). На выходе этапа проектирования - разработка данных, разработка архитектуры и процедурная разработка ПО.

Для получения оценки значений показателей качества по стандарту [6] используются следующие методы:

1. измерительный - базируется на использовании инструментальных, измерительных и специальных программных средств для получения информации о свойствах и характеристиках ПО (объем ПО, количество строк кода, количество операторов, число ветвей в программе, количество точек входа / выхода и др.);
2. регистрационный - основан на получении информации во время испытаний или функционирования ПО, когда регистрируются или подсчитываются определенные события (время и количество сбоев и отказов, время передачи управления от одного модуля к другому, время начала и завершения работы ПО);
3. органолептический - основан на использовании информации, полученной в результате анализа восприятий органов чувств, и применяется для определения таких показателей как удобство применения, эффективность;
4. расчетный - основан на использовании теоретических и эмпирических зависимостей (на ранних этапах разработки), статистических данных, собранных при проведении испытаний, эксплуатации и сопровождении ПО. Расчетными методами оцениваются показатели надежности, точности, устойчивости, время реакции, необходимые ресурсы и проч.;
5. экспертный - осуществляется группой экспертов. Их оценка базируется на опыте и интуиции, а не на непосредственных результатах расчетов или экспериментов. Этот метод реализуется путем просмотра программ, кодов, сопроводительных документов, описаний требований к ПО группой экспертов. Для этого устанавливаются контролируемые признаки, коррелированные с одним или несколькими показателями качества и включенные в карты опроса экспертов. Метод применяется при оценке таких показателей, как наглядность, полнота и доступность программной документации, легкость усвоения, возможность анализа, документированность, структурированность ПО и др.

Из описания методов измерения показателей (метрик) качества понятно, что на этапе проектирования ПО невозможно измерить ни одной характеристики еще не разработанного ПО, невозможно регистрировать моменты процесса выполнения еще не существующего ПО и невозможно воспринять органами чувств информацию о не разработанном ПО. Так, на этапе проектирования является возможность определять качество ПО только с применением расчетных и экспертных методов.

Согласно [7], метрика определяется как мера степени владения свойством, которая имеет числовое значение. Вообще, метрика ПО – это мера, которая позволяет получить числовое значение некоторого свойства программного обеспечения или его спецификаций. Итак, если речь идет о метрики сложности программ, то это меры, которые дают числовое значение сложности ПО, а метрики качества программ дают числовое значение качества ПО.

Показатели оценки сложности и качества проекта очень важны для получения объективных оценок по проекту. Как правило, данные показатели не могут быть вычислены на ранних стадиях работы над проектом, поскольку требуют, как минимум, детального проектирования. Однако эти показатели важны для получения прогнозных оценок длительности и стоимости проекта, поскольку непосредственно определяют его трудоемкость. Несмотря на многочисленные исследования программных метрик, в этой области остается ряд нерешенных вопросов:

1. технология измерения качества еще не достигла зрелости - только 1.5% софтверных организаций пытаются оценить качество процессов и готового продукта количественно, с помощью метрик, и только 0.5% софтверных организаций пытаются улучшить работу, руководствуясь количественными критериями качества для выпуска бездефектных продуктов;
2. отсутствуют единые стандарты на метрики - создано более тысячи метрик, каждый поставщик "измерительной" системы предлагает свои способы оценки качества и соответственно метрики;
3. существует проблема сложности интерпретации величин метрик - значения метрик, полученные с помощью "измерительных" систем, неинформативны или мало информативны для пользователя, заказчика, а часто и для программиста;
4. метрики рассчитываются только для готового ПО – все "измерительные" системы ориентированы на расчет метрик программного кода, но часто существует необходимость в расчете метрик уже на этапе проектирования - метрик сложности и качества с точными значениями для проекта ПО и метрик сложности и качества с прогнозируемыми значениями для разрабатываемого по проекту программного обеспечения.

Именно из-за нерешенности этих вопросов пока невозможно создать бездефектное высококачественное ПО. Сложность обоснования выбора и интерпретации метрик в процедурах принятия производственных решений, а также игнорирование этапов жизненного цикла ПО не позволяют полноценно использовать метрики для повышения качества ПО и уменьшения его сложности.