Атрибуты качества

Качество ПО является относительным понятием, которое имеет смысл только при учете реальных условий его применения, поэтому требования, предъявляемые к качеству, ставятся в соответствии с условиями и конкретной областью их применения.

Качество ПО характеризуется тремя главными аспектами: качество программного продукта, качество процессов ЖЦ и качество сопровождения или внедрения

Модель качества ПО имеет следующие четыре уровня детализации.

Первый уровень *соответствует определению характеристик (показателей) качества для* ПО, каждая из них отражает отдельную точку зрения пользователя на качество.

Согласно стандартов определено шесть характеристик или шесть показателей качества в стандартной модели качества:

1. функциональность (functionality),
2. надежность (realibility),
3. удобство (usability),
4. эффективность (efficiency),
5. сопровождаемость (maitainnability),
6. переносимость (portability).

Модель качества ПО

- ***Второму уровню*** *соответствуют атрибуты качества для каждой характеристики,* которые детализируют разные аспекты конкретной характеристики. Набор атрибутов характеристик качества используется при оценки качества.
- ***Третий уровень*** *предназначен измерения качества с помощью метрик, каждая из них* согласно стандарта определяется как комбинация метода измерения атрибута и шкалы измерения значений атрибутов. Для оценки атрибутов качества на этапах ЖЦ (при просмотре документации, программ и результатов тестирования программ) используются метрики с заданным оценочным весом для нивелирования результатов метрического анализа совокупных атрибутов конкретного показателя и качества в целом. Атрибут качества определяется с помощью одной или нескольких методик оценки на этапах ЖЦ и на завершающем этапе разработки ПО.

Атрибуты качества ПО

Важны преимущественно для пользователей	Важны преимущественно для разработчиков
Доступность Эффективность Гибкость Целостность Способность к взаимодействию Надежность Устойчивость к сбоям Удобство и простота использования	Легкость в эксплуатации Легкость перемещения Возможность повторного использования Тестируемость

Определение атрибутов качества

Доступность. Под доступностью понимается запланированное *время доступности* (uptime), в течение которого система действительно доступна для использования и полностью работоспособна. Формально доступность равна *среднему времени наработки на отказ* системы, деленному на сумму среднего времени наработки на отказ и ожидаемого времени до восстановления системы после сбоя. На доступность также влияют периоды планового технического обслуживания. Некоторые авторы рассматривают доступность как совокупность надежности, легкости в эксплуатации и целостности

Доступность-1. *Система должна быть доступна как минимум на 99,5% по рабочим дням, с 6:00 до полуночи по местному времени и доступна как минимум на 99,95% по рабочим дням, с 16:00 до 18:00 по местному времени.*

Эффективность - показатель того, насколько эффективно система использует производительность процессора, место на диске, память или полосу пропускания соединения .

Эффективность связана с производительностью еще одним классом нефункциональных требований. Если система тратит слишком много доступных ресурсов, пользователи заметят снижение производительности — видимого показателя неэффективности. Недостаточная производительность раздражает пользователей, которые ожидают вывода на экран результата запроса к базе данных. Но проблемы производительности кроме того, ставят под удар безопасность, например, при перегрузке системы контроля процессов реального времени. Определите минимальную конфигурацию оборудования, при которой удается достичь заданных эффективности, пропускной способности и производительности. Чтобы позволить нижний предел в случае непредвиденных условий и определить последующий рост, вы можете воспользоваться такой формулировкой:

Эффективность-1. Как минимум 25% пропускной способности процессора и оперативной памяти, доступной приложению, не должно использоваться в условиях запланированной пиковой нагрузки.

Гибкость. Этот атрибут также называют *расширяемостью, дополняемостью, наращиваемостью* или *растяжимостью*. Гибкость показывает с какой легкостью в продукт удается добавить новые возможности. Если ожидается, что при разработке придется вносить множество улучшений, стоит выбрать такие решения, которые позволят увеличить гибкость ПО. Этот атрибут важен для продуктов, в качестве модели разработки которых выбрано улучшение и повтор успешных выпусков или развитие прототипа. Для проекта, были сформулированы следующие цели, касающиеся реализации гибкости в ПО:

Гибкость-1. *Программист по техническому обслуживанию, не менее шести месяцев работающий с продуктом, должен уметь подключать новое устройство для создания печатных копий, что предусматривает изменение кода и тестирование, не более чем за час рабочего времени.*

- ***Целостность***, которая включает в себя безопасность, связана с блокировкой неавторизированного доступа к системным функциям, предотвращением потери информации, антивирусной защитой ПО и защитой конфиденциальности и безопасности данных, введенных в систему. Целостность очень важна для интернет-приложений. Пользователи систем электронной коммерции хотят обезопасить данные своих кредитных карточек. Посетители Web-сайтов не желают, чтобы приватная информация о них или список посещаемых ими сайтов использовались не по назначению, а поставщики услуг доступа к Интернету хотят защититься от атак типа «отказ в обслуживании» и прочих хакерских атак. В требованиях к целостности нет места ошибкам. Используйте следующие точные термины для формулирования требований к целостности: проверка идентификации пользователя, уровни привилегий пользователя, ограничения доступа или определенные данные, которые должны быть защищены. Вот как можно' сформулировать требования к целостности:

- ***Целостность-1.*** *Только пользователи, обладающие привилегиями уровня Аудитор, должны иметь возможность просматривать транзакции клиентов.*

Способность к взаимодействию показывает, каким образом система обменивается данными или сервисами с другими системами. Чтобы оценить способность к взаимодействию, вам необходимо знать, какие приложения клиенты будут применять совместно с вашим продуктом и обмен каких данных предполагается.

- ***Надежность.*** Надежностью называется вероятность работы ПО без сбоев в течение определенного периода времени. Иногда одной из характеристик надежности считают устойчивость к сбоям. Для измерения надежности ПО используют такие показатели, как процент успешно завершенных операций и средний период времени работы системы до сбоя. Определите количественные требования к надежности, основываясь на том, насколько серьезными окажутся последствия сбоя и оправдана ли цена повышения надежности. Системы, для которых требуется высокая надежность, следует проектировать с высокой степенью возможности тестирования, чтобы облегчить выявление недостатков, отрицательно влияющих на надежность.
- Команда разрабатывала ПО для управления лабораторным оборудованием, предназначенным для опытов с редкими дорогостоящими химикатами, длящихся целый день. Пользователям требовался программный компонент, который обеспечил бы надежность проведения экспериментов. Другие системные функции, такие как периодическая запись в журнал данных о температуре, были не столь важными. Требования к надежности для данной системы звучали так:
- ***Надежность-1.****Не более пяти из тысячи начатых экспериментов могут быть потеряны из-за сбоев ПО.*

Устойчивость к сбоям. Под устойчивостью к сбоям понимают уровень, до которого система продолжает корректно выполнять свои функции, несмотря на неверный ввод данных, недостатки подключенных программных компонентов или компонентов оборудования или неожиданные условия работы. Устойчивое к сбоям ПО легко восстанавливается после различных проблем и «не замечает» ошибок пользователей. Выясняя требования к устойчивости работы ПО, спросите пользователей, какие ошибочные ситуации возможны при работе с системой и как система должна на них реагировать. Вот один из примеров требования к устойчивости к сбоям:

Устойчивость к сбоям-1. ***Если при работе с редактором произошел сбой и пользователь не успел сохранить файл, то редактор должен восстановить все изменения, внесенные раньше, чем за минуту до сбоя, при следующем запуске программы данным пользователем.***

- ***Устойчивость к сбоям-2.*** ***Для всех параметров, описывающих графики, должны быть указаны значения по умолчанию, которые ядро Graphics Engine будет использовать в случае, если данные входного параметра указаны неверно или отсутствуют.***
- Выполнение этого требования позволит избежать сбоя программы, если, например, приложение запрашивает цвет, который плоттеру не удается воспроизвести. Graphics Engine использует значение по умолчанию — черный цвет — и продолжит работу. Тем не менее это можно рассматривать как сбой ПО, поскольку конечный пользователь не получил желаемый цвет. Однако такой подход снизил серьезность последствий сбоя — вместо краха программы получен неправильный цвет, что является примером отказоустойчивости.

Удобство и простота использования. Также называется *легкостью использования и инженерной психологией.* Этот атрибут связан с массой факторов, которые составляют основу того, что пользователи часто описывают *как дружелюбие к пользователю.* Но в лексиконе аналитиков и разработчиков нет термина «дружественное ПО», они говорят о ПО, которое спроектировано для эффективного и необременительного использования. Удобство и простота использования измеряется усилиями, требуемыми для подготовки ввода данных, эксплуатации и вывода конечной информации.

Удобство и простота использования-1. Пользователь, прошедший соответствующую подготовку, должен иметь возможность выбрать требуемый химикат из каталога поставщика в среднем за четыре и максимум за шесть минут.

Узнайте, должна ли новая система соответствовать каким-либо стандартам или соглашениям, касающимся пользовательского интерфейса, и должен ли последний быть совместим с другими часто используемыми системами. Вы можете сформулировать такое требование следующим образом:

- ***Удобство и простота использования-2. Всем функциям меню File должны соответствовать быстрые клавиши, нажимаемые одновременно с Ctrl. Командам меню, которые также присутствуют в меню File пакета Microsoft WordXP, должны соответствовать те же быстрые клавиши, что и в Word.***
- Кроме того, удобство и простота использования определяется и тем, насколько легко новые или непостоянные пользователи научатся работать с продуктом. Простота обучения также поддается исчислению и измерению:
- ***Удобство и простота использования-3. Химик, который прежде никогда не использовал Chemical Tracking System, должен не более чем за 30 минут разобраться, как правильно запросить химикат.***

Атрибуты, важные для разработчиков

Легкость в эксплуатации. Этот атрибут показывает, насколько удобно исправлять ошибки или модифицировать ПО. Легкость в эксплуатации зависит от того, насколько просто разобраться в работе ПО, изменять его и тестировать, и тесно связано с гибкостью и тестируемостью. Этот показатель крайне важен для продуктов, которые подвергаются частым изменениям, и тех, что создаются быстро (и, возможно, с экономией на качестве). Легкость в эксплуатации измеряют, используя такие термины, как среднее время, требуемое для разрешения проблемы, и процент корректных исправлений. Для Chemical Tracking System одно из требований к легкости в эксплуатации сформулировано таким образом:

Легкость в эксплуатации-1 *Программист, занимающийся техническим обслуживанием ПО, должен модифицировать существующие отчеты, чтобы привести их в соответствие с изменениями в положениях федерального правительства в области химии, затратив на разработку не более 20 рабочих часов.*

Атрибуты, важные для разработчиков

Если нам придется часто вносить исправления в ПО, чтобы оно соответствовало потребностям пользователей. Указали примерно такие критерии, чтобы разработчикам удалось создать ПО, более легкое в эксплуатации:

Легкость в эксплуатации-2. *Вложенность вызываемых функций не должна превышать два уровня.*

Легкость в эксплуатации-3. *Для каждого программного модуля непустые комментарии в соотношении к исходному коду должны составлять как минимум 0,5.*

Для аппаратных устройств со встроенным ПО часто имеются требования к легкости в эксплуатации. Одни из них относятся к выбору проектирования ПО, в то время как другие влияют на проектирование оборудования. Например последнее можно сформулировать так:

Легкость в эксплуатацин-4. *Проектирование принтера позволяет сертифицированному ремонтнику заменить кабельный шнур печатающей головки не более чем за 10 минут, датчик ленты не более чем за 5 минут и привод ленты не более чем за 5 минут.*

Легкость перемещения. Мерой ее измерения можно считать усилия, необходимые для перемещения ПО из одной операционной среды в другую. Некоторые практики считают возможность интернационализации и локализации продукта высшей степенью его мобильности. Приемы разработки ПО, которые делают легким его перемещение очень схожи с теми, что применяют, чтобы сделать ПО многократного используемым. Обычно мобильность продукта или не имеет никакого значения, или крайне важна для успеха проекта. Важно определить те части продукта, которые необходимо легко перемещать в другие среды, и описать эти целевые среды. Затем разработчики выберут способы разработки и кодирования, которые увеличат мобильность продукта.

Например, одни компиляторы определяют размер типа данных integer в 16 бит, а другие — 32 бит. Чтобы выполнить требования к мобильности, программисту надо в символической форме определить тип данных *WORD* как 16-битное целое без знака и использовать тип данных *WORD* вместо целочисленного типа данных, принятого в компиляторе по умолчанию. Таким образом гарантируется, что все компиляторы будут одинаково обращаться к элементам данных типа *WORD,* что сделает работу системы предсказуемой в различных операционных средах.

Возможность повторного использования. Постоянная задача разработки ПО — возможность повторного использования — показывает усилия, необходимые для преобразования программных компонентов с целью их дальнейшего применения в других приложениях. Затраты на разработку ПО с возможностью повторного использования значительно выше, чем на создание компонента, который будет работать только в одном приложении. Оно должно быть модульным, хорошо задокументированным, не зависеть от конкретных приложения и операционной среды, а также обладать некоторыми универсальными возможностями. Цели многократного использования сложно количественно измерить. Укажите, какие элементы новой системы необходимо спроектировать таким образом, чтобы упростить их повторное применение, или укажите библиотеки компонентов многократного использования, которые необходимо создать дополнительно к проекту. Например:

Возможность повторного использования-1. Функции ввода химических структур должны быть спроектированы таким образом, чтобы их удавалось повторно использовать на уровне объектного кода в других приложениях, построенных с учетом международных стандартов для представления химических структур.

Тестируемость. Этот атрибут также называют *проверяемостью,* он показывает легкость, с которой программные компоненты или интегрированный продукт можно проверить на предмет дефектов. Такой атрибут крайне важен для продукта, в котором используются сложные алгоритмы и логика или имеются тонкие функциональные взаимосвязи. Тестируемость также важна в том случае, если продукт необходимо часто модифицировать, поскольку предполагается подвергать его частому регрессивному тестированию, чтобы выяснить, не ухудшают ли внесенные изменения существующую функциональность.

Поскольку я и команда разработчиков твердо знали, что нам придется тестировать много раз в период его неоднократных усовершенствований, мы включили в спецификацию следующую директиву, касающуюся тестируемости ПО:

Тестируемость-1. Максимальная цикломатическая сложность модуля не должна превышать 20.

Цикломатическая сложность — это количество логических ответвлений в модуле исходного кода.

Чем больше ответвлений и циклов в модуле, тем тяжелее его тестировать, понимать и поддерживать. Конечно, проект не будет провален, если значение цикломатической сложности одного из модулей достигнет 24, однако наша директива указала разработчикам уровень качества, к которому следует стремиться. Если бы ее (она представлена как требование к качеству) не было, не факт, что разработчики при написании своих программ приняли бы во внимание такую характеристику, как цикломатическая сложность. В результате код программы мог оказаться так запутан, что его тщательное тестирование и дополнение оказалось бы практически невозможным, а отладка вообще стала бы кошмаром.

Требования к производительности определяют, насколько быстро и качественно система должна выполнять определенные функции. Они определяют такие параметры, как скорость (например, время отклика БД), пропускная способность (количество транзакций в секунду), мощность (нагрузка при совместном использовании) и распределение по времени (интенсивные запросы реального времени). Жесткие требования к производительности сильно влияют на стратегию разработки ПО и выбор оборудования, поэтому определите задачи, касающиеся производительности, для соответствующей операционной среды. Все пользователи хотят, чтобы их приложение запускалось моментально, но реальные требования к производительности различаются для функции проверки орфографии в программе подготовки текстов и для радиолокационной системы наведения ракеты. Требования к производительности также должны учитывать снижение производительности при такой перегрузке, как девятый вал звонков в службу спасения 911. Вот несколько простых требований к производительности:

Производительность-1. ***Цикл контроля температуры должен быть полностью выполнен за 80 миллисекунд.***

Производительность-2. ***Интерпретатор должен проводить в минуту разбор как минимум 5000 операторов, не содержащих ошибок.***

Производительность-3. ***Каждая Web-страница должна загружаться не более чем за 15 секунд при модемном соединении со скоростью 50 кбит/с.***

Производительность-4. ***Авторизация запроса на получение денег из банкомате должна длиться более 10 секунд.***

Реализация нефункциональных требований

Проектировщики и программисты должны определить наилучший способ удовлетворения требования для каждого атрибута качества и производительности. Хотя атрибуты качества относятся к нефункциональным требованиям, они помогают сформулировать функциональные требования, определить направления разработки или техническую информацию, которая позволяют реализовать продукт желаемого качества. В табл. перечислены некоторые категории технической информации, которые могут быть выведены с помощью атрибутов качества. Например, в медицинское устройство со строгими требованиями к доступности может входить источник резервного питания (архитектура) и функциональные требования для визуального или звукового оповещения, когда продукт работает на резервном питании. Это преобразование требований к качеству, имеющих значение для пользователей или для разработчиков, в соответствующую техническую информацию является частью процесса разработки требований и проектирования высокого уровня.

Реализация нефункциональных требований

Типы атрибутов качества	Категория технической информации
Целостность, способность к взаимодействию, устойчивость к сбоям, легкость и простота использования, безопасность	Функциональное требование
Доступность, эффективность, гибкость, производительность, надежность	Архитектура системы
Способность к взаимодействию, легкость и простота использования	Ограничения проектирования
Гибкость, легкость в эксплуатации, легкость перемещения, надежность, возможность повторного использования, тестируемость, легкость и простота использования	Руководство по проектированию
Легкость перемещения	Ограничение реализации

Определение требований к качеству

Атрибуты качества и задачи, связанные с производительностью— еще один аспект требований пользователей, который имеет большое значение при выборе пакетного решения. Следует рассмотреть, по крайней мере, следующие атрибуты.

Производительность. Каково максимальное время отклика, приемлемое для определенных операций? Может ли пакет обработать ожидаемую загрузку одновременно подключенных пользователей и пропускную способность транзакций?

Легкость и простота использования. Соответствует ли пакет всем принятым стандартам пользовательского интерфейса? Похож ли интерфейс на интерфейсы других приложений, с которыми знакомы пользователи? Насколько легко ваши клиенты смогут обучить работать с пакетом?

Гибкость. Насколько легко разработчики смогут изменить или расширить пакет, чтобы удовлетворить ваши потребности? Входят ли в пакет соответствующие «ловушки» (точки подключения и расширения) и прикладные программные интерфейсы, чтобы добавлять расширения?

Определение требований к качеству

Способность к взаимодействию. Насколько легко вы сможете интегрировать пакет с другими приложениями, используемыми в компании? Используются ли стандартные форматы для обмена данными?

Целостность. Допускает ли пакет контроль допуска пользователей к системе или применение специальных функций? Предохраняет ли он данные от потери, повреждения или неавторизованного доступа? Можете ли вы определять различные уровни привилегий пользователей?

При приобретении готовых пакетов организация получает меньшую гибкость при работе с требованиями, чем при разработке продукта «под заказ». Вам необходимо выяснить, какие из запрошенных возможностей не могут быть предметом переговоров, а какие вы можете изменять в рамках ограничений, налагаемых пакетом. Единственная возможность выбрать правильный пакет решений — понять задачи, пользователей и коммерческие задачи, для решения которых вы приобретаете пакет.

Спасибо за внимание!