Управление трудоемкостью проекта

Проект - это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов.

Проект обладает следующими характеристиками:

- **сложность (объем работ)**
- ресурсы (это исполнители, оборудование и материалы)
- продолжительность (вычисляется из сложности и ресурсов)
- стоимость (вычисляется из сложности)

Задача (task) - деятельность, осуществляемая в рамках проекта, для достижения определенного результата. Задачи являются основными блоками, из которых строится любой проект, они представляют работу, которую нужно выполнить для достижения поставленной цели. Во всем проекте набор задач характеризуется их логической последовательностью, а каждая задача - длительностью и требованиями к ресурсам.
Ресурсы - исполнители, оборудование и материалы, необходимые для выполнения задачи.

Назначения - связь конкретной задачи с ресурсами, выделенными для ее выполнения.

Проект, как правило, содержит большое количество задач, поэтому весь набор задач необходимо представить в виде укрупненных групп, логически связанных между собой. Так формируются суммарные задачи (фазы)

Суммарная задача (фаза, summary task) - состоит из нескольких задач. Результат фазы обобщает (суммирует) результаты задач, входящих в нее. Суммарная задача может содержать в себе как задачи, так и другие суммарные задачи.

Веха (milestone) - задача, достижение результата которой особенно важно для проекта. Вехой может быть завершающая задача фазы. Как правило, веха используется для обозначения окончания основных этапов проекта.

Трудозатраты (work) или же сложность - для задач: объем работ (в единицах рабочего времени) необходимый ресурсу (исполнителю) для выполнения задачи.

Сложность измеряется в человеко-часах. ЧЕЛОВЕКО-ЧАС - единица измерения рабочего времени, соответствующая часу фактической работы одного человека.

Длительность задачи (duration) - время, которое запланировано для работы над задачей.

Управление трудоемкостью проекта

Процесс определения сложности (estimation) это процесс

- выделения фич проекта
- определения трудоемкости для всех подзадач для каждой фичи проекта
- определения ресурсов
- определения длительности и промежуточных milestones (вех), на каждой из которых определяется результат.
- Определение рисков связанных с проектом
- Определение допущений (assamptions) на случай если некоторые требования не достаточно понятны, ясны, делаются предположения, которые ложатся в основу эстимейта

Результатом определения трудоемкости проекта является документ эстимейт.

Эстимейт — документ, предоставляющий детальное обоснованное описание трудоемкости проекта с указанием всех выделенных элементов.

Пример шаблона эстимейта:

Activity	**Optimistic**	**Most-likely**	**Pessimistic**	**Description (Required field)**
Iteration Zero				
Creative assets development				
High-Level architecture				
Mockup development				
Configuration management				
Release estimation				
SRS development				
Construction				
Feature 1 (Takes into account dev, testing, reqs clarification).				
Feature 2.				
...				
Stabilization				
Regression Testing				
Stabilization Bugfixing				
Management & communacation				

TOTAL:				
Resources:				
Duration				
Risks	Trigger	Phrase		
Risk 1				
Risk 2				
Assumptions				
Assumption 1. (Напр. Creative assets should be prepared in time before Construction phrase)				
Assumption 2.				

Эстимейты классифицируются по типу задач на следующие 3 категории:

1. эстимейт на основе оценки сложности выполнения функциональных требований. В этом случае каждый task соответствует функциональному требованию, которое подвергается критическому анализу. В результате для каждого требования оценивается его сложность выраженная в человеко-часах, то есть фактически вычисляется время которое необходимо выполнить для достижения требования. Сложность проекта в этом случае является суммой сложностей всех оцененных функциональных требований.
2. Эстимейт на основе оценки сложности архитектуры проекта. На основе функциональных требований выполняется рекуррентная декомпозиция проекта на более мелкие компоненты времени выполнения (tasks). После чего происходит оценка сложности для их реализации. Сложность проекта в этом случае является суммой сложностей всех выявленных компонент.
3. Смешанный эстимейт — при котором часть функциональных требований оценивается исходя из первой методики, остальные требования используются второй методикой.

Кроме того с проектов всегда существует ряд задач, не попадающих в действие выше

Методы оценки трудоемкости задач:

1. Метод назначения задачам обычных человеко-часов на основе некоторых эвристик например:
    - интуиция и опыт разработчика
    - декомпозиция на более мелкие задачи, сумма исходной равна сумме составляющих задач
    - Карта технологий
    - Counting. Подсчет объектов, связанных с задачей. Например, кол-во экранов или окошек, которые нужно реализовать для задачи.
2. Planning-Pocker.
3. WAG Метод. Используется для ballpark & prelimenary эситмейтов.

Управление ресурсами и длительностью проекта

При определении сложности проекта важно не только получить общую оценку сложности проекта (как правило выраженную в одной цифрой в человеко-часах), а также количество ресурсов, задействованных в проекте и его длительность, которая не всегда совпадает со сложностью.(длительность совпадает со сложностью только в случае если в проекте

задействован один ресурс).

Управление стоимостью проекта

Стоимость проекта вычисляется на основе сложности проекта и стоимости одного человеко-часа по следующей формуле:

Стоимость проекта=сложность проекта * стоимость 1 человеко-часа

Фактически заработок компании и сотрудников (ресурсов) заложен в стоимость 1 человеко-часа.

Стоимость проекта должна учитывать также стоимость дополнительных затрат, связанных с проектом. Например это может быть оплата лицензий сторонней платной компоненты, задействованной в проекте или же стоимость использования специализированных фреймворков.

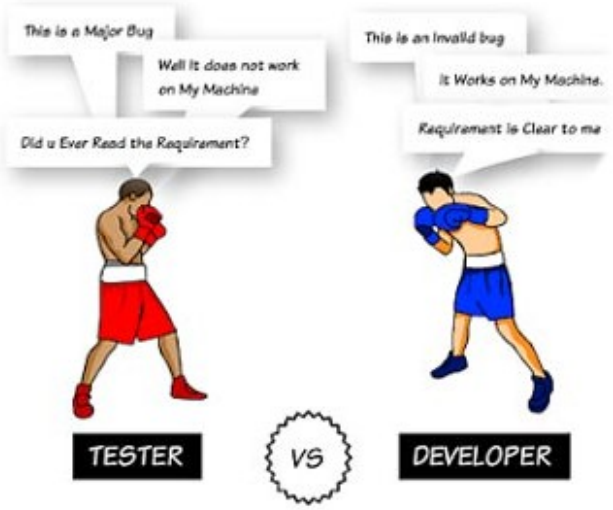
Обеспечение качества - основные понятия и определения

1) Качество программного **обеспечения (Software Quality) -** это степень, в которой программное обеспечение обладает требуемой комбинацией свойств.

2) Качест**во программного обеспечения (Software Quality) -** это совокупность характеристик программного обеспечения, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Обеспечение качества (Quality Assurance - QA) - это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации программного обеспечения (ПО) информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО, для обеспечения требуемого уровня качества выпускаемого продукта.

Контроль качества (Quali**ty Control - QC) -** это совокупность действий, проводимых над продуктом в процессе разработки, для получения информации о его актуальном состоянии в разрезах: "готовность продукта к выпуску", "соответствие зафиксированным требованиям", "соответствие заявленному уровню качества продукта".



Тестирование программного обеспечения. (Software Testing) - проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование - это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

Ручное тестирование (manualtesting) тестировщики вручную выполняют тесты, не используя никаких средств автоматизации. Ручное тестирование – самый низкоуровневый и простой тип тестирования, не требующих большого количества дополнительных знаний.

Тем не менее, перед тем как автоматизировать тестирование любого приложения, необходимо сначала выполнить серию тестов вручную. Мануальное тестирование требует значительных усилий, но без него мы не сможем убедиться в том, возможна ли автоматизация в принципе. Один из фундаментальных принципов тестирования гласит: **100% автоматизация невозможна**. Поэтому, ручное тестирование – необходимость.

Автоматизированное тестирование предполагает использование специального программного обеспечения (помимо тестируемого) для контроля выполнения тестов и сравнения ожидаемого фактического результата работы программы. Этот тип тестирования помогает автоматизировать часто повторяющиеся, но необходимые для максимизации тестового покрытия задачи.

Некоторые задачи тестирования, такие как низкоуровневое регрессионное тестирование, могут быть трудозатратными и требующими много времени если выполнять их вручную. Кроме того, мануальное тестирование может недостаточно эффективно находить некоторые классы ошибок. В таких случаях

автоматизация может помочь сэкономить время и усилия проектной команды. После создания автоматизированных тестов, их можно в любой момент запустить снова, причем запускаются и выполняются они быстро и точно. Таким образом, если есть необходимость частого повторного прогона тестов, значение автоматизации для упрощения сопровождения проекта и снижения его стоимости трудно переоценить. Ведь даже минимальные патчи и изменения кода могут стать причиной появления новых багов.

Баг Репорт (Bug Report) - это документ, описывающий ситуацию или последовательность действий приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

Программное обеспечение для управления проектами (bugtracker)

— определение для комплексного программного обеспечения, включающее в себя приложения для планирования задач, регистрации ошибок (багов), совместной работы, общения, быстрого управления, документирования, которое используются совместно для управления крупными проектами.

Anonymous
account

Main | My View | View Issues | Change Log | Roadmap | Wiki

Issue # Jump

Unassigned [^] (1 - 10 / 63)

0014705	[Demo] Here is some description Website - **2012-10-14 13:35**
0014704	[Demo] Here is some description Website - **2012-10-14 13:32**
0014701	[Demo] looks bad GUI - 2012-10-12 18:56
0014700	[Demo] test æøå Other - 2012-10-12 14:48
0014687	[Demo] Test bug GUI - 2012-10-11 11:20
0014686	[Demo] Child issue Other - 2012-10-11 09:37
0014685	[Demo] Parent item Other - 2012-10-11 08:29
0014647	[Demo] Test A Other - 2012-10-11 04:51
0014684	[Demo] need help Website - 2012-10-11 03:38

Resolved [^] (1 - 7 / 7)

~~0014683~~	[Demo] logo not visible GUI - 2012-10-11 03:22
~~0014671~~	[Demo] test de bug Other - 2012-10-10 03:55
~~0014660~~	[Demo] Parturient pellentesque. Risus nec mauris ac. GUI - 2012-10-09 07:15
~~0014624~~	[Demo] Enhancement: Add support for group permissions Other - 2012-10-05 13:56
~~0014590~~	[Demo] does this thing work? Other - 2012-10-03 11:43
~~0014546~~	[Demo] Tugged GUI - 2012-09-29 02:41
~~0014535~~	[Demo] Demo web test GUI - 2012-09-29 02:41

Серьезность и Приоритет Дефекта

Разные системы баг трекинга предлагают нам разные пути описания серьезности и приоритета баг репорта, неизменным остается лишь смысл, вкладываемый эти поля. Все знают такой баг-трекер, как Atlassian JIRA. В нем, начиная с какой-то версии вместо одновременного использования полей Severity и Priority, оставили только Priority**.** Таким образом, те кто привык работать с JIRA не всегда понимают разницу между этими понятиями, так как не имели опыта их совместного использования. Однако смысл, вкладываемый в них, различный:

Серьезность (**Severity**) - это атрибут, характеризующий влияние дефекта на работоспособность приложения.

Приоритет (**Priority**) - это атрибут, указывающий на очередность выполнения задачи или устранения дефекта. Можно сказать, что это инструмент менеджера по планированию работ. Чем выше приоритет, тем быстрее нужно исправить дефект.

Градация Серьезности дефекта (Severity)

S1Блокирующая (Blocker)

Блокирующая ошибка, приводящая приложение в нерабочее состояние, в результате которого дальнейшая работа с тестируемой системой или ее ключевыми функциями становится невозможна. Решение проблемы необходимо для дальнейшего функционирования системы.

S2 Критическая (Critical)

Критическая ошибка, неправильно работающая ключевая бизнес логика, дыра в

системе безопасности, проблема, приведшая к временному падению сервера или приводящая в нерабочее состояние некоторую часть системы, без возможности решения проблемы, используя другие входные точки. Решение проблемы необходимо для дальнейшей работы с ключевыми функциями тестируемой системой.

S3 Значительная (Major)

Значительная ошибка, часть основной бизнес логики работает некорректно. Ошибка не критична или есть возможность для работы с тестируемой функцией, используя другие входные точки.

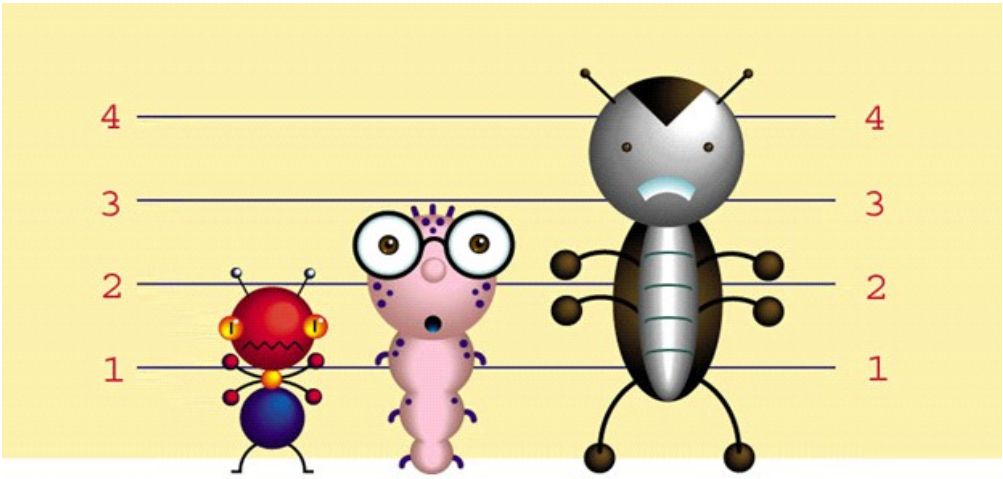
S4 Незначительная (Minor)

Незначительная ошибка, не нарушающая бизнес логику тестируемой части приложения, очевидная проблема пользовательского интерфейса.

S5 Тривиальная (Trivial)

Тривиальная ошибка, не касающаяся бизнес логики приложения, плохо воспроизводимая проблема, малозаметная посредствам пользовательского интерфейса, проблема сторонних библиотек или сервисов, проблема, не оказывающая никакого влияния на общее качество продукта.

Градация Приоритета дефекта (Priority)



P1 Высокий (High)

Ошибка должна быть исправлена как можно быстрее, т.к. ее наличие является критической для проекта.

P2 Средний (Medium)

Ошибка должна быть исправлена, ее наличие не является критичной, но требует обязательного решения.

P3 Низкий (Low)

Ошибка должна быть исправлена, ее наличие не является критичной, и не требует срочного решения.

Порядок исправления ошибок по их приоритетам:

High -> Medium -> Low

- Наличие открытых дефектов P1, P2 и S1, S2, считается неприемлемым для проекта. Все подобные ситуации требуют срочного решения и идут под контроль к менеджерам проекта.
- Наличие строго ограниченного количества открытых ошибок P3 и S3, S4, S5 не является критичным для проекта и допускается в выдаваемом приложении. Количество же открытых ошибок зависит от размера проекта и установленных критериев качества.

Дымовое тестирование или Smoke Testing

Понятие дымовое тестирование пошло из инженерной среды:

"При вводе в эксплуатацию нового оборудования ("железа") считалось, что тестирование прошло удачно, если из установки не пошел дым."

В области же программного обеспечения, *дымовое тестирование рассматривается*

как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции.

Вывод о работоспособности основных функций делается на основании результатов поверхностного тестирования наиболее важных модулей приложения на предмет возможности выполнения требуемых задач и наличия быстронаходимых критических и блокирующих дефектов. В случае отсутствия таковых дефектов дымовое тестирование объявляется пройденным, и приложение передается для проведения полного цикла тестирования, в противном случае, дымовое тестирование объявляется проваленным, и приложение уходит на доработку.

```
                        | Эстимируется
                        |
Идея  -> Проект ->      |  Dev-> QA -> Sales
                        |                 Дата
                        |
```

Расчет сложности проекта Прогноз погоды.

Модель

Weather — 1 h/h
IForecast — 2 h/h
City — 1 h/h
Universum – 2 h/h
Serializer - 6 h/h

Итог 2 дня

Контроллер

NetWorkController (CURL) – 25
UniversumNetWorkController – 2
UniversumController – 2

Итог 5 дней

Общий итог 42 часа — неделя.

Ресурсы

1. Про Тестинг - Тестирование Программного Обеспечения http://www.protesting.ru/

2. Estimation Toolkit www.infoq.com/articles/estimation-toolkit