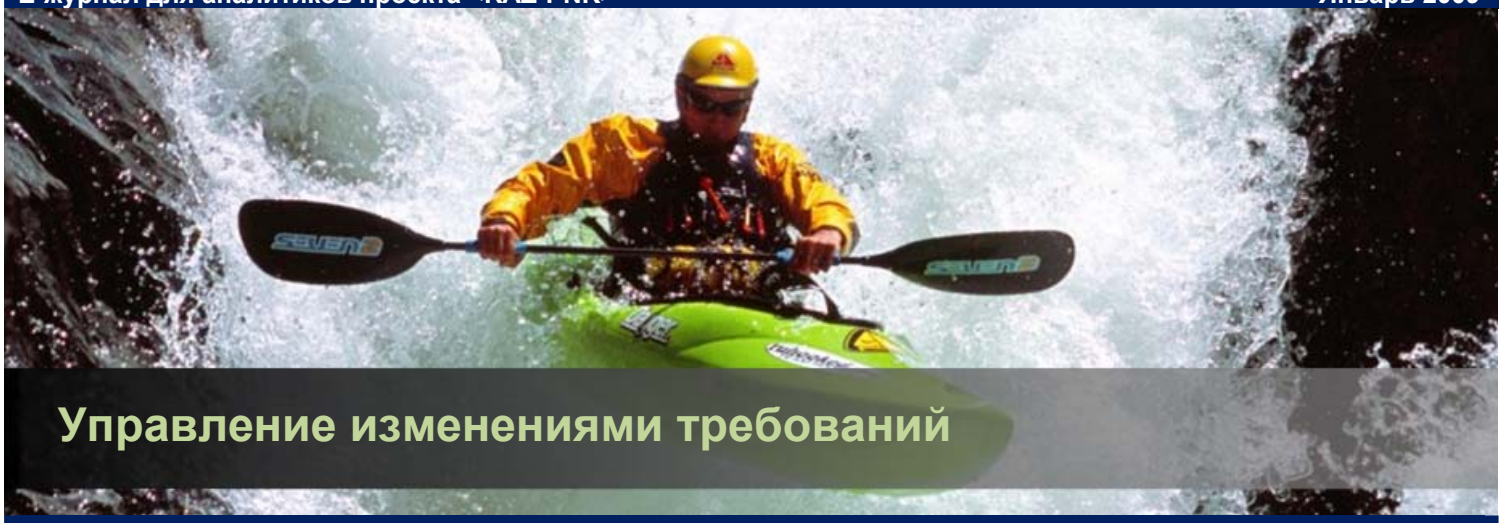
<epam> Delivering Excellence in Software Engineering

Analyze IT #1

E-журнал для аналитиков проекта <KAZ-PNK> **Январь 2009**



Управление изменениями требований

Created by Vitaliy Grigorash, Anastasia Martynenko

Содержание

Вводное слово

"- Бить будете? - Нет. - А что? - Вести разъяснительную работу..."
из кинофильма «Операция Ы и другие приключения Шурика»

Друзья, мы рады представить вашему вниманию первый выпуск электронного журнала для бизнес-аналитиков проекта KAZ-PNK. Мы взяли на себя смелость начать внепроектную деятельность по созданию этого журнала. Погружаясь на целый рабочий день (а часто и больше) в задачи проекта, уделяя свободное время родным и близким, трудно найти время и силы на самообразование и развитие своих профессиональных навыков. В области же постоянно развивающихся информационных технологий необходимо всегда держать «руку на пульсе». Именно поэтому мы решили создать журнал, в котором будут публиковаться как основы бизнес-анализа и разработки требований от «зубров» этих областей, так и самые свежие новости, статьи и модные тенденции в нашей профессиональной деятельности. Вы можете обучаться, восполнять пробелы в теоретических знаниях и быть в курсе новостей в мире бизнес-анализа, не затрачивая много времени на поиск и изучение материала. Наш журнал может стать достойным сопровождением вашей утренней (обеденной, вечерней) чашечки кофе ☺.

Предвидя закономерные вопросы, спешим сообщить, что мы работали над журналом дома без ущерба, а мы надеемся что и с пользой, для проектной деятельности☺. Пока мы не рискуем самостоятельно писать интересные статьи по аналитике (но обещаем, что дойдем и до этого), поэтому предлагаем вашему вниманию интересные вырезки из книг, переводы зарубежных статей и другие материалы, отобранные нами. Если Вам понравится наша работа, и Вы посчитаете ее полезной, то мы продолжим начатую деятельность и будем готовить новые выпуски журнала на интересующие вас темы. Кроме того, количество желающих принять участие в подготовке журнала не ограничено и мы всегда будем рады приветствовать вас в наших рядах.

Итак, тема сегодняшнего выпуска – Управление изменениями требований. Тема выбрана неслучайно – именно этим сейчас приходится заниматься команде аналитиков, и многим из нас не понятно для чего это нужно. Как и зачем управлять изменениями? Почему мы внедряем новые процессы для управления изменениями? Зачем мы тратим кучу времени на составление никому казалось бы не нужных «ченж реквестов»? Две вырезки из бестселлеров по требованиям, представленные в первом выпуске, надеемся, дадут ответы на все эти вопросы и расширят наши познания в данной области. Читайте, познавайте, комментируйте…

С уважением,
В. Григораш и А. Мартыненко

Управление изменениями

Авторы: Д.Лиффингуэлл, Д. Уидриг

Основные положения

- *Процесс управления требованиями может быть полезным только в том случае, если он позволяет распознавать и решать проблему изменений.*
- *Внутренними факторами, приводящими к возникновению изменений, являются неспособность задать нужные вопросы нужным людям в нужное время и отсутствие практического процесса, призванного справиться с изменениями требований.*
- *Чтобы повысить шансы на успех, необходимо предотвратить "просачивание" требований или, по крайней мере, существенно уменьшить его.*

Почему изменяются требования

Если бы можно было раз и навсегда определить множество требований к системе, жизнь была бы намного проще и данная статья была бы не нужна. Мы бы просто создали совершенный документ-концепцию, совершенную спецификацию программных требований и модель вариантов использования, заморозили бы их на время разработки, а затем объявили, что за все, что произошло после этого, отвечает команда сопровождения. **Увы, так не бывает; так никогда не было и не будет даже при самом систематическом подходе к управлению требованиями**.
Есть несколько причин неизбежности изменений требований. Среди них внутренние факторы, которые мы можем контролировать, и внешние, которые не подвластны контролю со стороны разработчиков и пользователей.

Внешние факторы

Внешними факторами являются те источники изменений, которые команда проекта не может контролировать. Вне зависимости от того, как мы справимся с ними, необходимо подготовиться технически, эмоционально и методологически, чтобы воспринять эти изменения как часть "нормального развития событий при разработке программного обеспечения". Изменения возникают по следующим причинам.

- Произошли ***изменения проблемы***, которую мы пытались решить с помощью новой системы. Возможно, возникли изменения в экономике, в правительственных инструкциях, на рынке или же изменились предпочтения потребителей.
- Пользователи ***изменили свое мнение*** *о* том, чего они хотят от системы, или свои предпочтения. Это, в свою очередь, может произойти вследствие ряда причин: не только из-за непостоянства пользователей (которое особенно ярко проявляется при спецификации деталей пользовательского интерфейса), но и из-за того, что их предпочтения зависят от ситуации на рынке, экономики, различных инструкций и т.д.
- Изменилась ***внешняя среда***, что привело к появлению новых ограничений и/или новых возможностей. Одним из наиболее очевидных примеров изменения среды является постоянно происходящее совершенствование систем аппаратного и программного обеспечения: если компьютеры будут работать в два раза быстрее, станут на 50% дешевле и компактнее и будут способны выполнять более сложные приложения, они, скорее всего, вызовут изменения в требованиях к системе.
- Вошла в эксплуатацию ***новая система***. Одним из самых неожиданных внешних факторов возникновения изменений (и главной причиной возникновения синдрома "да, но...") является то, что ***само появление новой системы приводит к тому, что меняются требования к ней***. Благодаря новой системе меняется поведение организации, старые способы выполнения действий больше не применяются; возникает потребность в новых типах информации и неизбежно разрабатываются новые требования к системе. Таким образом, сам факт ввода в строй новой системы выявляет новые требования к ней.

Процесс управления требованиями может быть полезен только в том случае, если он позволяет выявлять и решать проблему изменений. Невозможно предотвратить изменения, но можно научиться ими управлять.

Внутренние факторы

Помимо внешних факторов, существует ряд внутренних, которые также приводят к возникновению изменений.

- При первоначальном выявлении требований нам не удалось задать правильные вопросы нужным людям и в нужное время. Если процесс коснулся не всех заинтересованных лиц или им не были ***заданы правильные вопросы,*** это усугубляет проблему изменений, так как нет понимания истинных требований к системе, и в ходе разработки приходится производить значительные изменения, которых можно было бы избежать, если бы на более раннем этапе удалось добиться более полного понимания.
- Нам не удалось создать практический процесс, позволяющий ***справиться с изменениями*** требований, которые являются нормой при пошаговой разработке. Возможно, мы пытались "заморозить" требования, и "латентные" необходимые изменения накапливались до тех пор, пока не "взорвались" перед лицом разработчиков и пользователей, вызвав переделки и стресс. Или *процесс* внесения изменений отсутствовал вовсе; все могли изменять все, что угодно и когда угодно. В таком случае на некотором этапе практически все оказывается измененным и невозможно понять, что к чему.

Наш враг — мы сами

Вайнберг (Weinberg, 1995) заметил, что изменения могут быть скрытыми. После неудачного завершения одного проекта он сравнил известные требования к системе на момент окончания с требованиями, сформулированными вначале. При этом он обнаружил множество источников изменений требований. Некоторые были "официальными". Они представляли собой запросы пользователей, сделанные по соответствующим каналам сообщения. Но многие оказались на удивление "неофициальными". Данное явление Вайнберг назвал "просачиванием" требований. Среди них можно указать следующие.

- Упомянутые дистрибьюторами улучшения, о которых программисты случайно услышали на переговорах.
- Прямые запросы клиентов, обращенные к программистам.
- Ошибки, оставшиеся в поставленной версии продукта, которые теперь нуждаются в сопровождении.
- Аппаратные функции, которые в итоге не вошли в продукт или не работают.
- Изменение масштаба в ответ на действия конкурентов.
- Функции, включенные программистом из "лучших побуждений" (в расчете, что это понравится клиенту).
- "Сюрпризы" программистов.

Подобных изменений может быть сравнительно немного, но в целом требования, полученные из неофициальных источников, могут составлять до половины масштаба всего проекта Другими словами, половина всех функций рабочего продукта системы будет создаваться для выполнения просочившихся требований, т.е. тех, которые вошли в систему незаметно для членов команды, отвечающих за график, бюджет и критерии качества.

Как менеджер проекта может учесть эти изменения и при этом не выйти из графика и удовлетворить критерии качества? Это сделать невозможно! Чтобы иметь шансы на успех, "просачивание" требований должно быть остановлено или, по крайней мере, сведено к минимуму.

Процесс управления изменениями

Поскольку изменения являются неотъемлемой частью процесса, а источники их поступления могут быть как внешними, так и внутренними, **необходим процесс управления изменениями требований**. Такой процесс дисциплинирует команду, позволяет ей выявлять изменения, производить анализ их воздействия и систематическим способом включать те из них, которые наиболее необходимы и приемлемы для системы. Согласно рекомендациям Вайнберга (Weinberg), процесс обработки изменений должен включать в себя следующие шаги.

1. Осознать, что изменения неизбежны, и разработать *план управления изменениями*.
2. Сформировать *базовый уровень требований (requirements baseline)*.
3. Установить единый *канал контроля изменений*.
4. Использовать *систему контроля изменений* для их фиксации.
5. Обрабатывать изменения по иерархическому принципу.

Рассмотрим более подробно каждый из этих шагов.

Шаг 1. Осознать, что изменения неизбежны, и разработать план управления изменениями

Команда должна признать, что изменения требований к системе неизбежны и даже необходимы. Изменения будут возникать, и команда, зная об этом, должна разработать соответствующий план управления изменениями, который разрешит внесение определенных изменений в исходный базовый уровень.

Что касается легитимности изменений, все они (за исключением сюрпризов) могут считаться таковыми, так как исходят от заинтересованных лиц, которые имеют как реальную потребность, так и возможность внести реальный вклад в приложение.

Например, запросы изменений со стороны команды разработчиков являются правомерными, так как она знает о системе больше, чем кто-либо другой. Естественно, разработчики будут вносить множество предложений о том, что должна делать система. Некоторые требования поступают от тех, кто непосредственно занимается реализацией системы; только они в полной мере осознают, что в действительности она может делать. Следует прислушиваться к их мнению; в результате получится более удачная система для наших пользователей.

Шаг 2. Формирование базового уровня требований

Ближе к концу фазы исследования жизненного цикла разработки команда должна скомпоновать все известные требования к системе. Процесс формирования базового уровня может заключаться просто в наложении контроля исправлений на соответствующие артефакты (документ-концепцию, программные требования, модели вариантов использования) и публикации базового уровня для команды разработчиков. Собранные в этих документах отдельные требования создают базовый уровень информации о требованиях и предполагаемых вариантах использования системы.

Этот простой шаг позволяет команде различать известные ("старые") требования и новые (те, которые были добавлены, удалены или модифицированы). После задания базового уровня гораздо легче выявлять и обрабатывать новые требования. Запрос на новое требование можно сравнить с существующей базой и определить, где оно будет размещаться и не будет ли оно конфликтовать с другими требованиями. К сожалению, этот шаг пользователи зачастую пропускают, стараясь побыстрее отреагировать на изменения в своей среде. Если изменение принимается, можно проследить его эволюцию от концепции к программным требованиям, от программных требований к соответствующим документам и моделям технического проектирования, а затем к коду и тестовым процедурам (трассировка).

Упорядоченное и ответственное внесение изменений позволяет наладить сотрудничество с сообществом пользователей. В прошлом пользователи зачастую чувствовали сопротивление со стороны разработчиков, когда просили об изменениях. Причина заключалась в том, что команда имела в своем распоряжении лишь хаотический бессистемный процесс внесения изменений или не могла описать природу этого процесса пользователям.

Упорядочение процесса внесения изменений не означает, что можно вносить огромное количество всевозможных изменений.

Однако упорядочение процесса внесения изменений не означает, что можно вносить их сколько угодно. С точки зрения, как пользователей, так и разработчиков жизнь будет гораздо проще, если удастся создать набор стабильных корректных требований. Даже при достаточно хорошо организованном процессе управления изменениями существуют ограничения на количество изменений, которые разработчик сможет учесть, особенно на стадиях проектирования и реализации. **Как правило, коэффициент изменения требований во время разработки составляет 1-4% в месяц. Но если его величина превышает 2% в месяц, проект подвергается высокому риску "перемешивания" требований**.

Шаг 3. Задание единого канала контроля изменений

Изменения могут быть коварными. Хотя очевидно, что появление новой функции может оказать существенное влияние на требования к программному обеспечению, системную архитектуру, планы тестов и т.д., все мы оказывались в ситуации, когда "простое изменение" кода вызывало непредвиденные последствия, иногда даже катастрофические. Кроме того, предлагаемая новая функция может устранять важную будущую функцию системы или затруднять ее реализацию (т.е. воздействие распространяется не только на текущую версию). Есть также трудности, связанные с графиком и бюджетом проекта, за которые отвечает руководство. **Пожелания заказчиков о внесении изменений не предполагают официального изменения графика и бюджета, и следует инициировать процесс переговоров до того, как изменение будет принято**.

Таким образом, очень важно, чтобы все изменения поступали по одному каналу, чтобы определить их воздействие на систему и принять официальное решение, стоит ли вносить это изменение в систему вообще. В небольшом проекте этим официальным каналом может быть лидер проекта, менеджер или кто-нибудь другой, кто "владеет" документом-концепцией и другими артефактами требований, а также имеет полное представление о требованиях к системе и ее проекте.

В более крупных системах или в системах, где затрагиваются интересы множества заинтересованных лиц, такой официальный канал может состоять из нескольких человек (образующих Совет по контролю за изменениями — Change Control Board, CCB), которые совместно несут ответственность за принимаемые решения и обладают знаниями и полномочиями, необходимыми для официального принятия запроса об изменении.

В любом случае изменение *системы нельзя инициировать до тех пор, пока механизм контроля за изменениями не признает его "официальным"*.

Шаг 4. Использование системы контроля изменений для их фиксации

Проще всего дело обстоит с внешними изменениями, которые производятся по запросу клиента. Их легко выявлять, и

они будут естественным образом включены в проект руководством или органом, осуществляющим контроль за изменениями. Но во время разработки возникает огромное множество иных изменений системы.
Многие предлагаемые изменения, возникающие во время проектирования, кодирования и тестирования системы, могут казаться не связанными с требованиями (например, исправление ошибок кода или проектирования). Тем не менее необходимо оценить их воздействие. А если подходит срок сдачи, следует даже принять сознательное решение о том, какие ошибки оставить в системе (из-за того, что их исправление может дестабилизировать систему в целом и тем самым поставить под угрозу дату сдачи), а какие — устранить. Помимо этого, многие ошибки могут влиять на требования, вызывать необходимость их согласования или устранения неоднозначности отдельного известного требования.
Команде следует разработать некую систему для фиксации всех запросов на изменения.
В любом случае необходимо проанализировать ситуацию, а также принять решение о том, где изменение будет реализовано в иерархии документов. Следовательно, команде необходимо разработать формальный метод фиксации всех запрашиваемых изменений системы. Это может осуществляться с помощью системы отслеживания изменений и неполадок (bug tracking), которая обеспечивает создание централизованного архива запросов.
Эту систему следует использовать, чтобы фиксировать все предложения и передавать их руководству Совета по контролю за изменениями (CCB) для принятия решения. Совет играет ключевую роль в успехе проекта. Он должен состоять из трех-пяти человек, представляющих интересы основных участников проекта: заказчиков, представителей маркетинга и руководства проекта. Когда решается вопрос, принять ли запрос об изменении, CCB должен учитывать следующие факторы.

- Влияние изменения на *стоимость и функциональные возможности* системы.
- Воздействие изменения на *заказчиков и внешних участников*.
- Возможность того, что *изменение дестабилизирует систему*.

При принятии решения CCB также несет ответственность за то, чтобы отметить все, на что повлияет изменение, даже если оно не принимается.
После принятия изменения необходимо решить, куда его вставить. (Например, нужно определить, предлагается ли изменить требование или тест.) Последующие изменения будут распространяться по иерархии так, как показано на рисунке ниже.

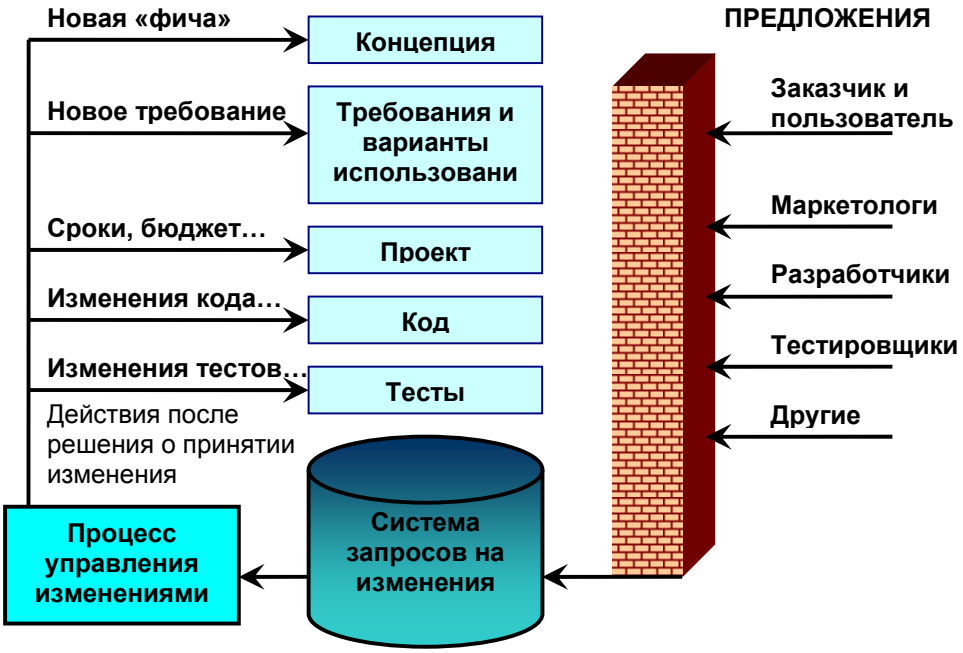


Рисунок 1 – Источники изменений и процесс внесения изменений в проект

Шаг 5. Иерархическое управление изменениями

То, что все вокруг заинтересованы во внесении изменений, не так уж плохо. Можно даже предположить, что все эти изменения, за исключением сюрпризов, полезны. **Проблемой является то, что изменения могут не документироваться и не анализироваться.** Если их тщательным образом не обрабатывать, это может привести к неприятным последствиям. Изменение одного требования может оказать возмущающее воздействие на связанные с ним требования, проектирование или другие подсистемы. К сожалению, это не учитывают представители маркетинга, которые часто просят программиста "быстро и просто" изменить систему.

Проблема осложняется тем, что при отсутствии явно заданного процесса изменения обычно производятся "восходящим" образом. Это означает, что если изменение появляется во время написания кода новой системы, оно, как правило, вносится непосредственно в программный код. Если разработчики предельно дисциплинированы, они могут затем задуматься: "Не вызовут ли изменения, которые мы вносим в код, изменений в проекте? А повлияют ли изменения на уровне проектирования на требования? Окажут ли изменения требований к программе воздействие на документ-концепцию?" (Тем не менее никто не вспомнит о том, что надо хоть что-нибудь сказать об этом изменении группе тестирования, члены которой полагают, что им нужно создавать планы тестирования, ориентируясь на исходные программные требования!)
Чтобы ослабить возмущающий эффект изменений требований, необходимо выполнять их в иерархии нисходящим образом от документа концепции к требованиям, коду и далее к тестам. Так как изменение документа-концепции может заключаться в удалении функций, нам может понадобиться создать совершенно новый исходный набор требований к программному обеспечению, а это может привести к соответствующим изменениям проектирования, кода и планов тестирования.

Управление конфигурацией требований

Процесс рассмотрения и принятия изменений в некоторых организациях носит название "контроль изменений", "контроль версий", или управление конфигурацией (configuration management, CM). Интересно, что большинство организаций имеет достаточно строгий процесс управления конфигурацией исходного кода, производимого на стадии реализации жизненного цикла проекта, но не имеет соответствующего процесса для требований к проекту. Даже если в организации имеется формальный процесс создания документа-концепции и требований к программному обеспечению, он зачастую игнорирует многие затрагивающие требования изменения, которые закрадываются в проект на стадии кодирования.
При наличии современных инструментальных сред достаточно несложно контролировать все элементы иерархии требований посредством управления конфигурацией
Преимущества основанного на CM процесса управления требованиями очевидны, но мы все же кратко перечислим их.

- Предотвращаются недозволенные и потенциально деструктивные изменения требований.
- Предотвращаются исправления документов требований.
- Упрощается получение и/или восстановление предыдущих версий документов.
- Осуществляется поддержка реализационной стратегии, основанной на задании базового уровня и инкрементных улучшениях и обновлениях системы.
- Предотвращается одновременное обновление документов (т.е. некоординированное обновление различных документов в одно и то же время).

Управление изменениями при поддержке программных средств

Итак, мы предлагаем практический подход к управлению изменениями, предполагая, что у вас имеется набор автоматических средств поддержки данных действий. Если вы захотите использовать собственные методы, реализуемые вручную, часть данного раздела вам не пригодится, но общие идеи все равно заслуживают внимания.
Здесь будут рассмотрены следующие важные вопросы.

- Если предлагается изменить отдельную «фичу» (feature) продукта, какими будут последствия данного изменения? Другими словами, управление изменениями поможет определить, какое количество переделок потребуется. Это может оказать существенное влияние на планирование ресурсов проекта и распределение нагрузки.
- Если предлагается изменить некий элемент, какие другие элементы системы может затронуть данное изменение? Это имеет первостепенную важность как для планирования разработки, так и для клиента.
- В ходе разработки неизбежны неверные "повороты". В такой ситуации нужно иметь возможность совершить "откат" требования и исследовать предыдущую его версию. Кроме того, полезно помнить, как и почему было изменено требование. Другими словами, полезно иметь контрольный журнал для каждого требования (это может даже предписываться регулирующими органами как часть процесса проектирования).

Элементы, на которые воздействует изменение

После задания отношений трассировки между требованиями связи можно использовать для управления изменениями. Предположим, что в системе необходимо изменить формулировку «фичи» FEA5, чтобы отразить пересмотренное определение свойства продукта (Рисунок 2). В Rational RequisitePro для отслеживания влияния изменений на связанные требования используются "подозрительные связи", предназначены для того, чтобы предупредить о том, что изменение данного требования может повлиять на SR1 и SR3, и поэтому следует произвести их ревизию.
По мере развития проекта неизбежно будут предлагаться изменения различных его элементов: от высокоуровневого документа-концепции до спецификации, реализации и тестов. Повсюду при возникновении изменения следует использовать "подозрительные связи", чтобы отметить отношения, на которые могло повлиять данное изменение.

Действия по управлению изменениями обычно заключаются в одном из следующих двух шагов.

1. Если изменение функции не влияет на требование, необходимо только очистить "подозрительную связь". Заметим, что если позднее данная функция вновь будет меняться, связь снова будет отмечена как подозрительная.
2. Если функция действительно влияет на требование, может понадобиться переделать подвергшийся воздействию элемент. Например, предлагаемое изменение функции может потребовать изменения спецификации требования.

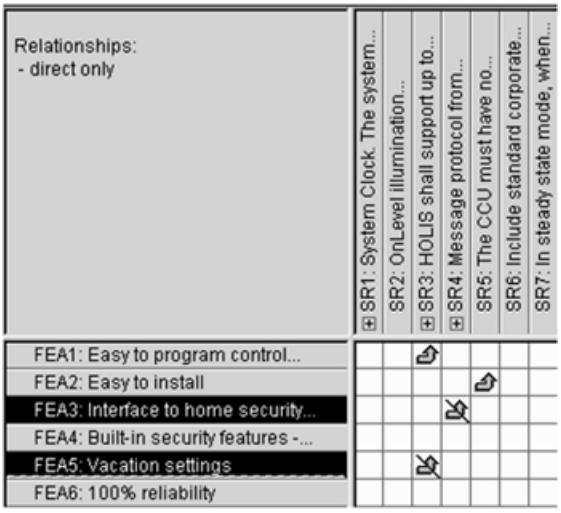


Рисунок 2 – Матрица трассировки с «подозрительными» связями

Возможность управления изменениями должна существовать на различных уровнях отношений трассировки. Так, изменение функции документа-концепции может затронуть несколько требований к программному обеспечению в SRS и/или некоторые варианты использования, которые могут, в свою очередь, воздействовать на несколько компонентов реализации, а те — на один или несколько планов тестирования. Необходимо отслеживать трассировочные связи в двух направлениях. Например, изменение спецификации плана тестирования может заставить рассматривать возможное влияние на компоненты реализации. В свою очередь, изменение компонента реализации может потребовать повторной проверки затронутых программных требований и даже функций верхнего уровня, которые связаны с этим компонентом посредством трассировочных отношений.

Контрольный журнал изменений

Полезно вести контрольный журнал изменений, где в частности, фиксируются изменения отдельных требований. При поддержке вспомогательной программы это позволит работать с каждым требованием отдельно, независимо от того, частью какого документа или модели оно является. Все изменения требований автоматически фиксируются (образуя историю изменения одного требования), и их можно впоследствии проверять и пересматривать. В истории изменений указывается текущая формулировка требования, в том числе текущие значения всех его атрибутов. Кроме того, в истории изменений в хронологическом порядке представлена последовательность всех предшествующих изменений данного требования и его атрибутов. Также записываются автор, дата и время изменения. Программа также должна предоставлять возможность вводить описание изменения. Как правило, вводится одно - два предложения, чтобы объяснить, почему было сделано изменение, сослаться на другие документы и т.д.

Заключение

Безусловно, по мере развития проекта требования будут меняться, но эти изменения не обязательно дестабилизируют процесс разработки. Если создан всеобъемлющий процесс контроля изменений, а артефакты требований подконтрольны используемой командой разработчиков системе управления конфигурацией, команда будет хорошо подготовлена к решению основных задач управления изменениями. Важно понимать, что управление изменениями в крупном проекте обычно является слишком объемной задачей, чтобы ее можно было решать вручную. Необходим процесс, чтобы контролировать, каким образом изменения попадают в проект. Также необходимы автоматические средства для того, чтобы понять разветвление изменений, т.е найти все затронутые им элементы проекта.

Изменения случаются

Автор: Карл И. Вигерс

Как-то, оценивая процесс разработки ПО, я спросил специалистов, работающих над проектом, как они включают изменения в требования к продукту. После неловкой паузы, один из них сказал: «Каждый раз, когда специалист отдела маркетинга хочет внести изменение, он обращается к Брюсу или Сэнди, потому что они всегда соглашаются, а мы нет». Эта процедура внесения изменений не показалось мне хорошей.

Процесс контроля изменений

Процесс контроля изменений позволяет руководителю проекта принимать информированное бизнес-решение, которое удовлетворяет клиента и имеет коммерческую ценность, при этом контролируются затраты на жизненный цикл продукта. Вы можете отслеживать статус всех предложенных изменений и убедиться, что предложенные требования не были потеряны или упущены. После утверждения базовой версии набора требований придерживайтесь этого порядка для внесения всех предлагаемых изменений в нее.

Клиенты и другие заинтересованные лица часто уклоняются от нового процесса, однако процесс контроля изменений — не препятствие для модификации. Это механизм суммирования и фильтрации, дающий уверенность, что в проект включены наиболее ценные изменения. Если предложенное изменение не настолько важно, чтобы заинтересованное в проекте лицо потратило пару минут на передачу его через стандартные, простые каналы, то стоит задуматься о его ценности вообще. Процесс управления должен быть тщательно задокументирован, максимально прост и, что важнее всего, эффективен.

Если вы предложите заинтересованным лицам новый процесс контроля изменений неэффективный, громоздкий или слишком сложный, они постараются найти способ избежать этого и, возможно, будут правы.

Политика контроля изменений

Руководство должно ясно довести до сведения сотрудников политику, в которой описываются ожидания того, как участники проекта должны обрабатывать предложенные изменения требований. Политики имеют смысл, только если они реалистичны, добавляют ценность и проводятся в жизнь. Могут быть полезными следующие положения политики контроля изменений:

- все изменения требований должны вноситься в соответствии с процессом. Если запрос на изменение подается каким-то иным способом, он не рассматривается;
- нельзя заниматься дизайном или реализовать неутвержденные требования, кроме проверки их осуществимости;
- сам по себе запрос на изменение не гарантирует выполнение этого изменения.

Описание процесса контроля изменений.

На рис. 3 показан шаблон описания процесса контроля изменений для обработки модификаций требований и других изменений проекта.

Введение

Во введении описывается назначение процесса и определяются организационные границы, в которых он применяется. Если этот процесс затрагивает только изменения в определенных продуктах, их следует определить здесь. Также укажите, существуют ли специальные виды изменений, которые не подлежат контролю, например изменения в промежуточных или служебных продуктах, созданных в ходе проекта.

1. Введение
    - 1.1. Назначение
    - 1.2. Границы
    - 1.3. Определения
2. Роли и ответственности
3. Состояние запроса на изменение
4. Входные критерии
5. Задачи
    - 2.1. Оценка запроса
    - 2.2. Принятие решения
    - 2.3. Внесение изменений
    - 2.4. Уведомление всех затронутых сторон
6. Проверка
    - 6.1. Проверка изменения
    - 6.2. Установка продукта
7. Критерии выхода
8. Отчет контроля изменений о статусе запроса

Приложение. Элементы данных, сохраненные для каждого запроса

Рисунок 3 - Шаблон описания процесса контроля изменений

Роли и ответственности

Перечислите членов команды — по ролям, которые участвуют в процессе контроля изменений, и опишите их обязанности. В таблице приведены некоторые типичные роли; адаптируйте их в соответствии с конкретной ситуацией. Каждый человек может выполнять несколько ролей. Например, председатель совета по управлению изменениями также может получать подаваемые запросы на изменения. В небольших проектах несколько, а возможно, и все роли выполняются одним человеком.

Таблица 1 - Возможные роли участников проекта при управлении изменениями

Роль	Описание
Председатель совета по управлению изменениями	Возглавляет совет по управлению изменениями; как правило, обладает правом принятия окончательного решения, если совет не приходит к согласию; выбирает тех, кто оценивает и вносит изменения для каждого запроса на изменение
Совет по управлению изменениями	Группа, принимающее решение утвердить или отклонить каждое изменение для определенного проекта
Тот, кто оценивает изменение	Человек, которого председатель совета просит проанализировать влияние предложенного изменения; это может быть сотрудник технического отдела, клиент, маркетолог или сотрудник, занимающийся всем этим понемногу
Тот, кто вносит изменение	Отвечает за внесение изменений в продукт, когда запрос на изменение утвержден
Тот, кто инициирует запрос	Некто, подающий новый запрос на изменение
Получатель запроса	Лицо, которому передаются новые запросы на изменение
Тот, кто проверяет изменение	Человек, определяющий, корректно ли изменение

Состояние запроса на изменение

Запрос на изменение проходит через определенные стадии на протяжении жизненного цикла, причем на каждой стадии он имеет различные статусы. Вы можете представить эти изменения состояния, используя диаграмму перехода состояний, как показано на рис. 4. Обновляйте состояние запроса, только когда удовлетворяются определенные критерии.

Входной критерий

Основным входным критерием для процесса контроля изменений является действительный запрос на изменение, полученных по утвержденным каналам. Все потенциальные инициаторы запросов должны знать, как подавать запросы на изменение, независимо от того, отправляют они его, заполняя бумажную форму или Web-форму, отправляя сообщение по электронной почте или используя средство контроля изменений.

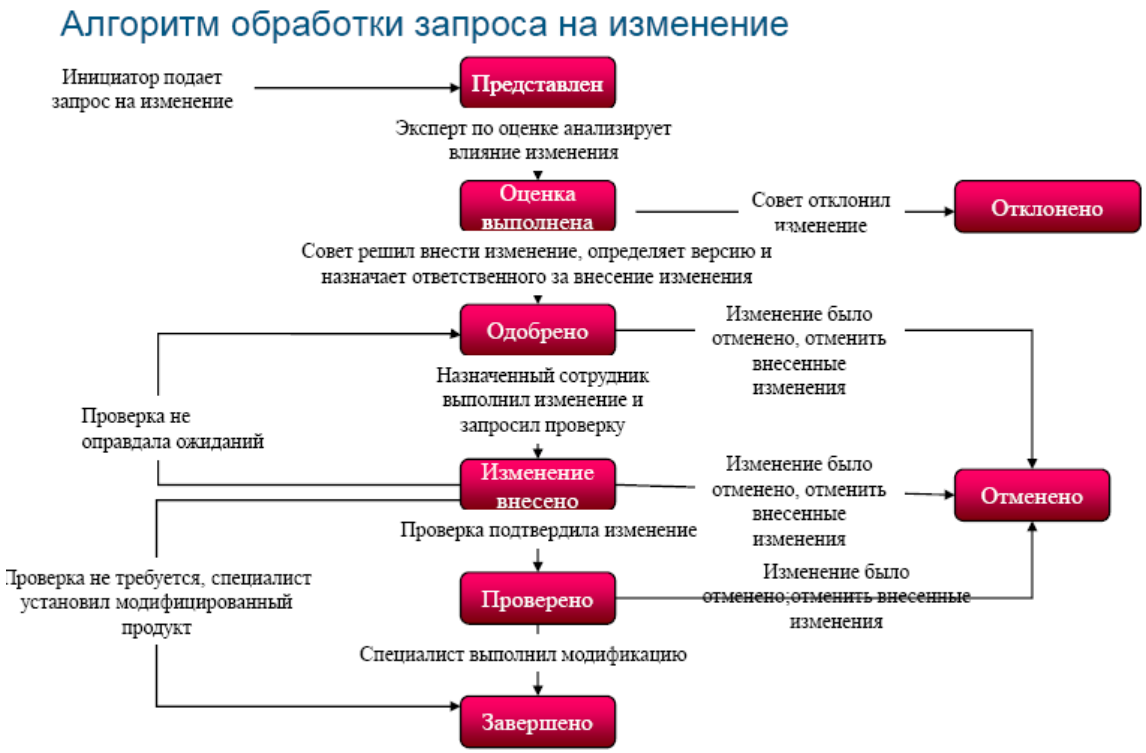


Рисунок 4 - Алгоритм обработки запроса на изменение

Задачи

Следующий шаг — это оценка запроса на предмет технической выполнимости, затрат и соответствия бизнес-требованиям проекта и ограничениям ресурсов. Председатель совета по управлению изменениями может назначить сотрудника для оценки влияния изменения, анализа риска, анализа опасности и др. Анализ позволяет убедиться, что вероятные последствия изменения ясны. Тот, кому поручена оценка, и члены совета по управлению изменениями должны также рассмотреть коммерческие и технические последствия отклонения изменения.
Затем совет по управлению изменениями решает, утвердить или отклонить запрошенное измененииеи присваивает каждому одобренному изменению уровень приоритета, или назначает дату реализации, или назначает это изменение определенной сборке или номеру версии. Далее совет по управлению изменениями оповещает о решении, обновляя состояние запроса или уведомляя всех членов команды, которые занимаются модификацией.

Проверка

Изменения требования, как правило, проверяют с помощью экспертной оценки, чтобы убедиться, что измененные требования, варианты использования и модели соответствующим образом отражают все аспекты изменения. Информация о трассируемости поможет вам найти все части системы, которые это изменение затрагивает и которые, как следствие, необходимо проверить.

Выходной критерий

Все перечисленные далее выходные критерии должны быть творены, чтобы должным образом завершить процесс управлений изменениями:

- состояние запроса: Отклонено, Завершено или Отменено
- все изменения записаны в соответствующих разделах;
- инициатор запроса, председатель совета по управлению изменениями, менеджер проекта и другие значимые участники проекта оповещены о деталях изменения и текущем состоянии запроса на изменение;
- матрица трассировки требований обновлена.

Отчет о состоянии управления изменения

Определите графики и отчеты, которые вы будет использовать при обобщении содержимого базы данных контроля изменений. Обычно этих графиках показано количество запросов на изменение в каждой категории состояния как

функция времени.

Измерение активности изменений

Измерение ПО позволяет проникнуть в суть проекта, продуктов и процессов более точно, чем если опираться на субъективные впечатления или неясные воспоминания о прошлом опыте. Измерение активности изменений — это способ оценки стабильности требований. Он позволяет также обнаружить возможности для улучшения процесса, чтобы снизить количество изменений в будущем. Можно контролировать следующие показатели изменений:

- количество полученных запросов на изменение, открытых и закрытых в настоящее время;
- общее число пришедших запросов, включая добавленные, удаленные и измененные требования;
- количество полученных запросов на изменение, исходящих из каждого источника;
- количество изменений, предложенных и внесенных в каждое требование после создания базовой версии;
- общие усилия на обработку и реализацию запросов на изменение;
- количество циклов процесса изменений, необходимых для должной реализации каждого утвержденного изменения (иногда изменения реализуются неправильно или являются причиной появления других ошибок, которые необходимо исправить).

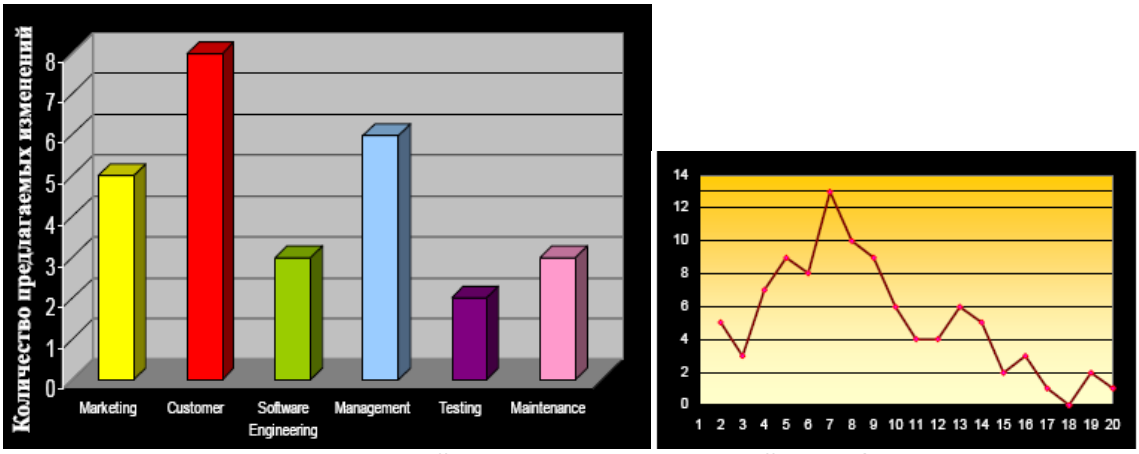


Рисунок 5 – Источники изменений и количество изменений в требованиях по неделям

Изменение не бесплатно: анализ влияния

Анализ влияния — это ключевой аспект ответственного управления требованиями (Arnold и Bohner, 1996). Он обеспечивает точное понимание подтекста предложенного изменения, что помогает команде принимать информированные бизнес-решения о том, какое изменение одобрить. Анализ позволяет выявить компоненты, которые может понадобиться создать, изменить или отклонить, и оценить затраты, связанные с реализацией изменения. До того, как разработчик ответит: «Конечно, без проблем», он или она должны потратить время на анализ результата изменения.

Процедура анализа влияния изменения

Председатель совета по управлению изменениями обычно просит опытного разработчика выполнить анализ результата определенного изменения. Анализ результатов изменений затрагивает три аспекта:

1. Определите возможные последствия изменения. Часто они вызывают значительный волновой эффект. Включение множества функций в продукт может снизить его производительность до неприемлемого уровня, например, когда системе, запускаемой ежедневно, потребуется 24 часа для завершения одного запуска.
2. Определите все файлы, модели и документы, которые, придется изменить, если команда включит все запрошенные изменения,
3. Определите задачи, необходимые для реализации изменения, и оцените усилия, необходимые для выполнения этих задач.

Заключение

Изменение требований происходит на всех проектах, но хорошо организованные приемы контроля изменений уменьшают число проблем, возникающих из-за этого. Улучшенные приемы сбора информации по требованиям могут снизить количество изменений требований, а эффективное управление требованиями позволит вам более точно выполнять проектные обязательства.