УДК 004.05/.5/.9

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО УПРАВЛЕНИЮ ТРЕБОВАНИЯМИ НА РАЗРАБОТКУ ВЕБ-САЙТОВ

[1]Кийкова Е.В., [1]Лаврушина Е.Г., [2]Еременко А.В.

[1]Владивостокский государственный университет экономики и сервиса, Владивосток, e-mail: elena_kiykova@list.ru;
[2]ООО Фарпост, Владивосток

Основной проблемой применения гибких методологий разработки программного обеспечения (Agile) остается факт отсутствия готовых решений по управлению требованиями, с учетом особенностей конкретных предметных областей, в том числе и области разработки и сопровождения веб-сайтов. Разработанная методика управления требованиями заключается в представлении рекомендаций и шаблонов по представлению требований на разработку всех видов задач и подробных рекомендаций по описанию пользовательских требований, на управление которыми было нацелено исследование. Она наиболее применима к задачам по поддержке и расширению функционала веб-сайтов (информационных или развлекательных порталов, интернет-магазинов и социальных сетей). Применение методики позволяет выделить и разработать ценный для пользователя функционал в кротчайшие сроки, за счет точного определения приоритетов и исключения переделывания функционала, реализованного по неточным требованиям.

Ключевые слова: требования, технология, методология agile, веб-сайт, шаблон, анализ, требования пользователя, методика

GUIDELINES FOR MANAGING REQUIREMENTS FOR SOFTWARE DEVELOPMENT WEBSITES

[1]Kiykova E.V., [1]Lavrushina E.G., [2]Eremenko A.V.

[1]Vladivostok State University Economics and Service, Vladivostok, e-mail: elena_kiykova@list.ru;
[2]LLC Farpost, Vladivostok

The main problem of the agile software development (Agile) usage is the absence of ready-made requirements on solutions management, taking into account the characteristics of specific subject areas, including the website development and maintenance procedures. The developed method of management requirements is to provide guidelines and templates for reporting requirements on the all kinds of tasks development and detailed guidance on the description of user requirements management, which the study was aimed at. The developed method is the most applicable to the tasks to support and expand the functionality of Web sites (information or entertainment portals, e-commerce (online shops)? and social networks). The application of the method allows identifying and developing valuable functionality to the user in the shortest possible time, by accurately identifying priorities and exceptions of remaking functionality implemented by inaccurate requirements.

Keywords: requirements, technology, methodology of an agile, web site, template, analysis, user requirements, methods

Работающий функционал сайта является основным показателем развития процесса работы над проектами веб-сайтов. Это является одним из главных различий, которое привносит методология agile в процесс управления разработкой и оценку реализации проекта, в отличие от водопадного подхода. Все гибкие методологии, представленные в «Agile манифесте», выделяют необходимость предоставления заказчику небольших частей функционала веб-сайта через заданные интервалы.

Каждой проектной команде необходимо решить, что именно подразумевается под словами «работающий функционал», которые часто используются как определение выполненной работы за спринт.

С точки зрения концептуального уровня, часть функциональности считается завершенной только тогда, когда реализованные требования проходят все этапы разработки, тесты и они доступны для использования на рабочем веб-сайте для конечного пользователя, в виде работающего функционала. Как минимум, в процессе работы над выпуском части нового функционала, команда разработчиков должна пройти уровень модульного тестирования и выполнить тестирование на уровне системы. Лучшие команды также включают в свое определение готового фрагмента функциональности тестирование интеграции, тестирование производительности и тестирование приемки заказчиком.

С точки зрения разработчиков, зачастую функционал считается сделанным, как только код написан, протестирован и проверен другим разработчиком. Однако не всегда такой код вовремя поставляется на рабочую версию веб-сайта, поэтому не должен считаться за готовую работу, так как она не несет никакой ценности пользователю, пока не доступна ему.

Условия успешной и качественной реализации проекта: определение приемочных

тестов при описании и тестировании задач; реализация задач в порядке приоритета; запуск приемочных тестов для каждой задачи, как только она реализована; исправление ошибок, определенных как наивысший приоритет как можно скорее.

Команды, производящие подходящие для клиентов и имеющие ценность для бизнеса продукты, должны оперативно реагировать на изменения.

Отраслевые данные показывают, что более половины требований к продуктам или проектам меняются в процессе разработки программного обеспечения. Даже когда традиционные проекты завершаются вовремя, укладываются в бюджет и реализуют все функции из плана, клиенты часто недовольны, так как реализованный функционал не соответствует в точности тому, чего они хотели [4, 7].

Все гибкие методологии содержат встроенные процессы по изменению запланированных требований на основе обратной связи от заказчика или пользователя [3]. Успешные проекты, реализуемые по методологии Agile, стремятся к раннему завершению, в то время как большинство проектов, реализуемых согласно методологии Waterfall, завершают разработку поздно.

Гибкие методологии основаны на знании о том, что для успешного завершения проекта необходимо планировать изменения. Вот почему в рамках agile установлены процессы, такие как анализ и ретроспектива, предназначенные для регулярного сдвига приоритетов на основе обратной связи от заказчика требований и ценности для бизнеса [8].

В процессе разработки требований для веб-сайтов было использовано несколько вариантов представления требований в задачах, однако, большинство из них сбивало с толку разработчиков и не учитывало все необходимые нюансы требований (как функциональных, так и нефункциональных).

Применение неэффективных форм представления требований сказывалось на сроках реализации и снижало производительность команды и качество функционала. Потому задача по разработке методики управления требований на разработку вебсайта стояла наиболее остро в рамках всего проекта.

Разработанная методика управления требованиями заключается в представлении рекомендаций и шаблонов по представлению требований на разработку всех видов задач и подробных рекомендаций по описанию пользовательских требований, на управление которыми было нацелено исследование.

Описание разработанной методики приводится согласно разделению требований и типов задач на разработку веб-сайта.

Класс bug задач, когда реализованный ранее функционал дал сбой или был реализован с ошибкой, которая попала в релиз и видна пользователю веб-сайта, требует определить, где и как она воспроизводится, что именно относится к ошибке и в каком виде она должна быть исправлена. Это позволяет команде разработчиков определить срочность и критичность поставленных требований. Подобное представление описания ошибок (включающих в себя, как функциональные, так и нефункциональные требования) позволяет предоставить всю необходимую информацию о характере и алгоритме воспроизведения задачи, зная которые, разработчик сможет без затруднений приступить к реализации.

Шаблон описания требований начинается с описания проблемы в названии задачи. После чего в описании bug задачи приводится список мест, где и как воспроизводится ошибка, в отдельный блок выделяется описание самой ошибки – «что не так», чтобы любой разработчик смог разобраться в возникающей ошибке, а также приводится описание желаемого решения ошибки в блоке «как должно быть».

При описании требований к классу задач «technical story», «technical task» и «technical debt», когда требуется реализовать какой-то техническое решение или правку, требуется избегать описания технического решения аналитиком или владельцем продукта.

Методика управления требованиями на разработку веб-сайтов предполагает, что для описания технических задач используется более краткое описание требований. Это позволяет не ограничивать разработчиков в выборе решения технической задачи. Описание таких задач предназначено больше для фиксирования работы разработчика, с целью выделения на это времени.

Шаблон описания таких задач состоит из выделения проблемы, которая потребовала технического вмешательства или причины, а также краткого описания решения, которое было принято, или цели, которой необходимо добиться в результате реализации поставленных в проблеме требований.

Такое представление описания технических задач позволяет заинтересованным лицам либо владельцу продукта расставить приоритеты задач и добавлять их в бэклог для дальнейшего планирования.

Класс epic задач отличается от представленных ранее, так как данные задачи не реализуются разработчиками, а служат для описания нового функционала, объединяя пользовательские требования под одной тематической задачей.

Определением и формулировкой epic задач занимается владелец продукта, так как данный класс задач отражает направление развития сайта (какой именно функционал панируется к реализации в ближайшем будущем), за что ответственен именно владелец продукта (или проектный менеджер).

Набор задач класса epic отражает общее видение дальнейшего развития веб-сайта (набор нового функционала, который планируется реализовать в ближайшем будущем).

Очень важно указывать в подобных задачах приоритет (business value), сортировка по которому позволяет ограничить выбор функционала для разработки и описания требований на итерацию. Приоритеты epic задач проставляются в виде числовых обозначений от 0. Выбор шагов для приоритезации задач задается в зависимости от удобства и количества задач в бэклоге.

Шаблон описания epic-задач состоит из выделения целевой пользовательской истории, которую необходимо реализовать, для удовлетворения пользовательской потребности, а формулировка данной истории, в отличие от правил описания обычных пользовательских историй, представляет собой одно обобщенное требование к новому функционалу или сервису веб-сайта, реализовать которое в одной задаче, имея только одно требование, невозможно.

Так как одно из основных правил пользовательских историй как требований – независимость и целостность, второй частью шаблона описания epic-требований является перечень всех пользовательских историй, который охватывает epic задача. Перечень пользовательских сценариев, объединенных единой темой, определяется и задается владельцем продукта на предварительном планировании краткосрочных и долгосрочных планов развития сайта и его подпроектов.

Класс в story задач, включает в себя описание функциональных требований и нефункциональных требований. Реализация данного класса задач приносит наибольшую ценность проекту. В процессе описания требований в story задачах, важно учитывать все особенности пользовательских требований и соблюдать достаточную глубину детализации требований.

Шаблон описания пользовательских историй в story задачах состоит из выделения пользовательской истории по общепринятому шаблону, описания критериев приемки, сценария приемочного функционального теста, а также с приведением прототипов интерфейса (в случае если требование пользовательской истории захватывает интерфейс взаимодействия пользователя с веб-сайтом).

Крупные и громоздкие пользовательские истории необходимо фрагментировать с учетом конкретики задач. Пользовательская история должна представлять собой минимальное самостоятельное требование, реализация которого добавит стоимости проекту.

Оптимальный размер «user story» определяется на основе времени, которое занимает ее реализация, на разработку одной пользовательской истории должно уходить от 0,5 до 4 «идеальных рабочих дней». Если по предварительным оценкам на реализацию пользовательской истории уходит больше четырех дней, то имеет смысл декомпозировать требования на более мелкие, но полноценные и независимые, иначе необходимо объединять несколько пользовательских историй в одну.

Обязательно указывать в описании пользовательской истории критерии приемки, поскольку это позволяет тестировать соответствие готового продукта изначально заявленным требованиям. В большинстве случаев формат описания пользовательской истории должен соответствовать основным рекомендациям, представленным ранее, но, если, к примеру, речь идет о дизайне интерфейса веб-сайта, рекомендуется представлять требования в виде пользовательских историй, подкрепленными свободным форматом прототипов или набросков интерфейса.

Не следует чрезмерно детализировать пользовательские требования (user story). Из-за слишком подробного описания деталей процесс обсуждения задачи проектной командой может быть сведен к минимуму, что не всегда хорошо для поиска оптимального решения поставленной задачи, так как разработанное решение может быть не самым оптимальным или ошибочным.

Описывать пользовательскую историю следует не с целью получения формальных «требований», а с целью выявления и понимания ценности требований для продукта, бизнеса и пользователей. Описание пользовательских требований в формате: «Как <роль пользователя>, я <хочу получить что-то>, чтобы <цель>», позволяет

добиться наиболее простого, но при этом достаточного понимания требований.

В данной формулировке отображается бизнес-ценность продукта для пользователя. Однако главная важность пользовательской истории в том, что она формулирует не только бизнес-ценность, но и требования для разработки и тестирования, чего позволяет добиться добавление в описание пользовательской истории критериев приемки и технических заметок.

Такая формулировка задачи помогает разработчикам и тестировщикам не пытаться сравнивать готовую задачу с интерфейсными требованиями, которые быстро устаревают, а смотреть на основные проблемы, которые должны быть решены в рамках задачи. Дополненная прототипами пользовательская история легко становится задачей, которую можно делать даже без финального дизайна.

Пользовательские истории требуется описывать не только для того, чтобы выразить мнение и желание заказчика, а с целью выразить мнение пользователей, которые будут использовать разрабатываемый функционал веб-сайта. Важно учитывать, что это не только конечные пользователи, но и те, кто оказывает влияние на них [2, 5, 8, 10].

Чтобы написать ценную и реалистичную пользовательскую историю, нужно получить максимум информации о будущих пользователях [5]. К примеру, актуальны будут следующие сведения о пользователях: считают ли они проблему, которую решает продукт, достаточно серьезной; как они решают свои проблемы сейчас; какие заменители или конкуренты есть у продукта, и многие другие аспекты, определяющие ценность разрабатываемого сайта, которую он дает конечному пользователю. Важно также производить проверку качества написанной пользовательской истории [1].

Пользовательская история должна быть достаточно информативна для разработчиков, чтобы они поняли, что нужно реализовывать. Но не настолько подробна, чтобы определять детали реализации. Подробное описание деталей реализации ограничит разработчиков в выработке решений, технологий и подходов к реализации.

В процессе разработки пользовательских требований необходимо заменять формализованные потребности пользователя на проработанные детальные требования к сценариям использования будущей программы.

Далее на основе пользовательских историй и критериев приемки строится интерфейс. На этапе проектирования интерфейсов приходит понимание того, от какого функционала стоит отказаться, например, нужны ли комментарии к фотографиям или нет [6, 9].

Правильный пользовательский сценарий должен быть понятным для всех участников проекта (и конечного пользователя); коротким, чтобы можно было оценить сроки его выполнения, но при этом с достаточно точным описанием; совпадать по смыслу и идее с основным проектом (чтобы очередная итерация «не выпадала» из общей идеи проекта).

Объективно можно оценить полезность пользовательской истории только в том случае, если по ней можно сформировать удобный и понятный конечному потребителю продукта интерфейс.

Инструменты управления требованиями, которые предлагают гибкие методологии, позволяют обеспечить требования гибкость и достаточной детализацией.

Визуальные прототипы (wireframes (mockups)) позволяют визуализировать требования продукта и улучшить восприятие их всеми членами команды. В прототипе отображаются только основные элементы и общие наброски (какой функционал на каком этапе где расположить).

Требования, описание которых подкреплено прототипами, наиболее эффективно работают в случае разработки новых интерфейсов, так как уже на этапе проектирования требований возможно пройти по всем пользовательским сценариям и выявить главную потребность пользователя.

Карта пользовательских историй (atory maps) – используется для визуализации списка пользовательских требований. Группировка пользовательских историй в функциональные категории (epics) с целью их визуализации позволяет выделить наиболее приоритетные и ценные требования (core features). Таким образом, на карте пользовательских историй вверху располагаются наиболее приоритетные истории, а снизу те, которые пока не столь приоритетны, но в будущем должны быть реализованы.

Вышеперечисленных инструментов достаточно для эффективного управления требованиями на разработку и поддержку веб-сайта.

Методические рекомендации по выявлению, анализу и представлению требований пользователя в виде пользовательских историй (user story) позволяют выделять требования в виде минимального функционала, который приносит ценность конечному продукту в кротчайшие сроки.

Список литературы

1. Анализ и управление требованиями [Электронный ресурс] // Software-Testing.Ru. – Режим доступа: http://software-testing.ru/books/requirements-books.

2. Виды требований [Электронный ресурс] // Все для аналитиков. – Режим доступа: http://foranalysts.blogspot.ru/2011/08/blog-post_17.html.

3. Еременко А.В. Специфика описания функциональных требований в гибких методологиях разработки программного обеспечения / А.В. Еременко, Е.В. Кийкова/ Интеллектуальный потенциал XXI века: ступени познания: материалы XXX Молодежной международной научно-практической конференции. / ред. д-р экон. наук С.С. Чернов. – Новосибирск: Изд-во Коллектив авторов, 2015 – С. 67–71.

4. Новобрицкая Е.А., Пауль Л.В., Лаврушина Е.Г. Информационные технологии в создании туристского продукта // Современные научные исследования и инновации. – 2014. – № 9; URL:http://web.snauka.ru/issues/2014/09/38285.

5. Принципы и значение гибкой разработки, Джеф Сазерленд (Jeff Sutherland) [Электронный ресурс] // Microsoft. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/dd997578(v=vs.120).aspx.

6. Разработка: Управление требованиями к IT-проектам [Электронный ресурс] // Habrahabr.ru. – Режим доступа: https://habrahabr.ru/post/114571/.

7. Слугина Н.Л., Кийкова Е.В., Мурадова Я.В. Разработка типового шаблона web-представительства гостиничного комплекса // Современные проблемы науки и образования. – 2014. – № 4; URL: www.science-education.ru/118-13885.

8. Управление требованиями [Электронный ресурс] // INFORMICUS. – Режим доступа: http://www.informicus.ru/Default.aspx?SECTION=6&id=73&subdivisionid=11.

9. Управление требованиями для систем и сложных ИТ-приложений [Электронный ресурс] // IBM. – Режим доступа: http://www-03.ibm.com/software/products/ru/ratidoor.

10. Business Analyst in Agile [Электронный ресурс] // Bainagile. – Режим доступа: http://bainagile.blogspot.ru/2010/12/что-такое-product-backlog.html.