Глава 2

Написание историй

В этой главе рассматривается процесс написания пользовательских историй. Приступая к нему, мы должны сконцентрировать внимание на следующих шести атрибутах, характеризующих качественно написанную историю:

- независимость (**I**ndependent);
- обсуждаемость (**N**egotiable);
- ценность для пользователей или заказчиков (**V**aluable);
- оцениваемость (**E**stimatable);
- компактность (**S**mall);
- тестируемость (**T**estable).

Для перечисленных свойств Билл Уэйк, автор книг *Extreme Programming Explored* и *Refactoring Workbook*, предложил использовать акроним INVEST [54].

Независимость

По мере возможностей старайтесь избегать создания зависимостей между историями. Такие зависимости порождают проблемы при установлении приоритетов историй и планировании. Например, может оказаться так, что в качестве высокоприоритетной заказчик выберет историю, которая зависит от истории с более низким приоритетом. Кроме того, зависимости между историями могут необоснованно усложнить их оценку. Предположим, что мы работаем над проектом веб-сайта BigMoneyJobs и необходимо подготовить истории, описывающие, каким образом компании могут оплачивать публикацию на нашем сайте объявлений об открывающихся вакансиях. Мы могли бы написать следующие истории.

1. Компания может оплатить публикацию вакансии с помощью кредитной карты Visa.
2. Компания может оплатить публикацию вакансии с помощью кредитной карты MasterCard.
3. Компания может оплатить публикацию вакансии с помощью кредитной карты American Express.

Предположим, разработчики решили, что для написания кода поддержки любой карты (независимо от ее типа), выбранной в качестве первой, им потребуется три дня, а для организации поддержки остальных двух типов карт — по одному дню на каждый тип. В случае историй со столь сильной взаимной зависимостью, как приведенные выше, трудно принять решение об их оценке, т.е. определить, разработку какой именно истории следует оценить в три дня.

В ситуациях с подобным типом зависимости можно пойти двумя путями:

- объединить две зависимые истории в одну бо́льшую, но независимую;
- найти другой способ разбиения историй.

Объединение историй, относящихся к кредитным картам различного типа, в одну большую историю ("Компания может оплатить публикацию вакансии с помощью кредитной карты") в данном случае работает хорошо, поскольку длительность разработки объединенной истории составляет всего пять дней. При значительно более длительных сроках разработки объединенной истории лучше подыскать какой-нибудь другой признак, по которому следует осуществлять разбиение историй. Если бы оценки продолжительности этих историй были бо́льшими, то был бы возможен следующий вариант их разбиения.

1. Компания может оплатить публикацию вакансии с помощью кредитных карт одного типа.
2. Компания может оплатить публикацию вакансии с помощью двух дополнительных типов кредитных карт.

Если же вы не хотите объединять истории и вам не удается подыскать подходящий способ их разбиения, в вашем распоряжении всегда имеется простой выход — проставить на карточке истории две оценки: одну для случая, когда данная история разрабатывается до начала разработки остальных историй, и вторую, более низкую, для случая, когда разработка истории начинается после завершения разработки другой истории.

Обсуждаемость

Уже написанные истории впоследствии можно обсуждать и вносить в них изменения. Истории не являются формальными контрактными обязательствами или требованиями, которые программное обеспечение должно реализовать. Карточки историй — это краткие описания функциональности, детали которых должны уточняться в ходе устных обсуждений между командами заказчика и разработчиков. Поскольку карточки историй призваны лишь напоминать нам о том, что еще подлежит дальнейшему обсуждению, и вовсе не обязаны содержать подробное изложение требований, их не следует излишне перегружать деталями. Но если в ходе составления истории вам стали известны какие-то важные подробности, их можно включить в карточку истории в виде примечаний, как показано на примере карточки 2.1. Вся трудность состоит в том, чтобы научиться ограничивать себя лишь минимально необходимым уровнем детализации.

Карточка истории 2.1 является вполне подходящей, поскольку она предоставляет достаточный объем информации для разработчика и заказчика, которые будут обсуждать эту историю. Когда разработчик приступит к написанию исходного кода для этой истории, он увидит напоминание о том, что ранее было принято решение относительно поддержки трех основных видов платежных карт, и это заставит его поинтересоваться у заказчика, принято ли также дополнительное решение относительно карт Discover. Примечания на карточке истории помогают разработчику и заказчику возобновлять незаконченные обсуждения. В идеальном случае это не должно вызывать затруднений, независимо от того, продолжают обсуждение те же самые представители команд разработчика и заказчика или другие люди. Руководствуйтесь этим, когда решаете, включать или не включать описание деталей в историю.

Компания может оплатить публикацию вакансии с помощью кредитной карты.

Примечание: принимаются карты Visa, MasterCard и American Express. Под вопросом карты Discover.

■ Карточка истории 2.1. Карточка истории с примечаниями относительно дополнительных деталей

С другой стороны, рассмотрим историю, содержащую избыточное количество примечаний, как показано на примере карточки истории 2.2. В этой истории содержится слишком много детальных сведений ("Проверять дату, когда истекает срок действия платежной карты"), а кроме того, в нее включено то, что, вероятно, можно было бы выделить в отдельную историю ("Система может определить тип карты по первым двум цифрам ее номера и сохранить номер для будущего использования").

Компания может оплатить публикацию вакансии с помощью кредитной карты.
Примечание: принимаются карты Visa, MasterCard и American Express. Под вопросом карты Discover. При совершении покупок на сумму свыше 100$ запросить подтверждающий код с обратной стороны карты. Система может определить тип карты по первым двум цифрам ее номера и сохранить номер для будущего использования. Проверять дату, когда истекает срок действия платежной карты.

■ Карточка истории 2.2. Карточка истории с избыточным количеством деталей

Работать с подобными историями очень трудно. Большинство людей, которые будут читать эту историю, ошибочно воспримут наличие избыточных деталей как *исчерпывающую* точность описания. Однако во многих случаях преждевременное указание деталей просто задает больше работы. Например, если два разработчика обсуждают и оценивают историю, в которой лишь сказано, что "компания может оплатить публикацию вакансии с помощью кредитной карты", то они не забудут о том, что тема их обсуждения сформулирована довольно абстрактно. Для того чтобы разработчики могли считать тему своего обсуждения полностью определенной, а свои оценки — точными, не хватает слишком многих деталей. Но если в историю включено избыточное количество детальных описаний, как в карточке 2.2, то ее обсуждение может восприниматься как конкретное и максимально исчерпывающее. Это может создать ошибочное впечатление,

будто карточка истории отражает абсолютно все детали и нет никакой необходимости далее обсуждать эту историю с заказчиком.

Если мы относимся к карточке истории как к напоминанию о том, что именно должны обсудить между собой разработчики и заказчик, то полезно исходить из того, что в карточке должна содержаться следующая информация:

- одна или две фразы, служащие напоминанием о том, что именно подлежит обсуждению;
- примечания, касающиеся вопросов, подлежащих последующему обсуждению.

Детали, уже прошедшие обсуждение, становятся тестами. Тесты могут записываться в виде примечаний на оборотной стороне карточки истории, если она бумажная, или сохраняться в любой ее электронной версии, если она используется. Карточки 2.3 и 2.4 показывают, как можно превратить избыточные детали карточки истории 2.2 в тесты, оставив среди записей на лицевой стороне карточки лишь примечания для обсуждения. Благодаря этому лицевая сторона карточки будет содержать историю и примечания, касающиеся вопросов, которые остаются открытыми, а оборотная — детали, касающиеся истории и записанные в виде тестов, которые подтверждают, работает или не работает история так, как ожидается.

Компания может оплатить публикацию вакансии с помощью кредитной карты.

Примечание: будут ли приниматься карты Discover?
Примечание по поводу интерфейса: поле для ввода типа карты не предусматривать (его можно определить по первым двум цифрам номера карты).

Карточка истории 2.3. Исправленный текст на лицевой стороне карточки истории, содержащий лишь саму историю и вопросы, подлежащие обсуждению

Тестировать с картами Visa, MasterCard и American Express (проходит).
Тестировать с картами Diner's Club (не проходит).
Тестировать с правильными, неправильными и отсутствующими подтверждающими кодами.
Тестировать с просроченными платежными картами.
Тестировать с суммами до 100$ и свыше 100$.

Карточка истории 2.4. Детали, определяющие варианты тестирования, отделяются от самой истории. В данном случае они отображаются на оборотной стороне карточки истории

Ценность для покупателей или пользователей

Существует соблазн сказать нечто вроде “Каждая история должна представлять определенную ценность с точки зрения пользователей”. Но такое утверждение было бы неверным. Многие проекты включают истории, не представляющие для пользователей никакого интереса. Помня о различиях между *пользователями* и *покупателями* программного обеспечения, проанализируем случай, когда команда разработчиков создает программное обеспечение, которое будет развертываться на широкой пользовательской базе, например на 5000 компьютеров в пределах одной компании. Покупатель продукта хочет, чтобы на каждом компьютере использовалась одна и та же конфигурация программного обеспечения. Это может привести к истории наподобие “Вся конфигурационная информация считывается из централизованного источника”. Пользователям безразлично, где хранится конфигурационная информация, но для покупателя это может иметь значение.

Аналогичным образом истории наподобие следующей также могли бы представлять ценность для покупателей, намеревающихся приобрести продукт, но не для фактических пользователей.

- На протяжении всего процесса команда разработчиков готовит документацию, удовлетворяющую требованиям стандарта ISO 9001.
- Команда разработчиков создает программное обеспечение в соответствии с моделью CMM Level 3.

Чего следует избегать, так это историй, которые имеют ценность только в глазах разработчиков. Примеры подобных историй приводятся ниже.

- Все подключения к базе данных осуществляются через пул соединений.
- Обработка и протоколирование ошибок осуществляются набором общих классов.

В том виде, как они записаны, эти истории сфокусированы на вопросах технологии и удобства для программистов. Вполне вероятно, что в этих историях содержатся неплохие идеи, но их следует переписать так, чтобы стали очевидными выгоды, предоставляемые ими для заказчиков или пользователей. Благодаря этому заказчик сможет назначить разумные приоритеты историям при составлении плана разработки. В качестве улучшенных вариантов этих историй можно использовать, например, следующие.

- Благодаря пяти клиентским лицензиям на доступ к базе данных с приложением могут работать до пятидесяти пользователей.
- Все ошибки протоколируются, а сообщения о них направляются пользователям унифицированным способом.

Точно так же старайтесь не включать в истории предположения, относящиеся к пользовательскому интерфейсу и вопросам технологии. Например, из приведенных выше переписанных вариантов истории исключены упоминания о предполагаемом использовании пула соединений и наборе классов для обработки ошибок.

Лучший способ добиться того, чтобы каждая история представляла ценность для заказчика или пользователей, — поручить написание историй заказчику. При этом на первых порах заказчики часто чувствуют себя довольно неуютно — возможно, потому, что опыт общения с разработчиками приучил их к мысли, будто все написанное ими впоследствии может быть обращено против них. (“Видите ли, в требованиях ничего не

сказано о том, что...") Однако большинство заказчиков начинают самостоятельно писать истории, как только приходят к пониманию того, что карточки историй — это всего лишь основание для дальнейших обсуждений, а не формальные контрактные обязательства или описание конкретной функциональности.

Оцениваемость

Важно, чтобы разработчики были в состоянии оценить (или, по крайней мере, обоснованно спрогнозировать) размер истории, т.е. количество времени, которое понадобится для того, чтобы превратить историю в работающий код. Существуют три основные причины, по которым трудоемкость истории может не поддаваться количественной оценке.

1. Разработчикам недостает знаний в данной предметной области.
2. Разработчикам недостает технического опыта.
3. История слишком велика по размеру.

Во-первых, разработчикам может не хватать знаний в данной предметной области. Если история в том виде, как она написана, непонятна разработчикам, им следует обсудить ее с заказчиком, который ее написал. Опять-таки, вовсе не обязательно понимать все детали, связанные с историей, но в целом она должна быть понятна разработчикам.

Во-вторых, трудности с оценкой истории могут возникнуть из-за того, что разработчики не владеют необходимой технологией. Допустим, при разработке Java-проекта вас попросили предусмотреть в системе интерфейс CORBA. Ранее никто из членов команды этим не занимался, в связи с чем оценка трудоемкости такой задачи вызовет затруднения. Выход состоит в том, чтобы поручить одному или нескольким разработчикам выполнение кратковременного исследования, целью которого является ознакомление с предметной областью приложения. В методологии экстремального программирования такая исследовательская история называется *спайком* (spike). Полученных в процессе выполнения спайка дополнительных сведений разработчикам должно быть достаточно для того, чтобы суметь оценить трудоемкость предстоящей задачи. Эксперимент всегда ограничивается *фиксированными временными рамками*, что позволяет присвоить ему количественную оценку. С учетом всего вышесказанного не поддающаяся оценке история преобразуется в две: историю, описывающую проведение быстрого исследования для сбора информации, и историю, соответствующую реальной работе.

Наконец, разработчики могут оказаться не в состоянии оценить историю, так как она слишком велика. Таковой, например, была бы история "Соискатель может осуществить поиск вакансий" для веб-сайта BigMoneyJob. Чтобы оценить ее, разработчикам потребуется разбить ее на несколько меньших историй.

Отсутствие знаний о предметной области

В качестве примера проекта, потребовавшего ознакомления с предметной областью, могу привести веб-сайт, который мы разрабатывали для обеспечения долгосрочного медицинского обслуживания хронических больных. Заказчик (высококвалифицированная медсестра) подготовил историю следующего содержания: "Новые пользователи проходят обследование на диабет". Разработчики не были уверены в том, что они все поняли правильно, поскольку такая фраза может означать что угодно — от

создания простой веб-анкеты до выезда с оборудованием на дом к новому пользователю с целью его физического обследования, как это было предусмотрено в программном продукте данной компании, предназначенном для обслуживания астматиков. Разработчики переговорили с заказчиком и выяснили, что в данном случае имелась в виду простая веб-анкета, содержащая ряд вопросов, на которые должен был ответить пациент.

Хотя они и слишком велики, чтобы их можно было надежно оценить, все же иногда полезно использовать эпические истории наподобие "Соискатель может осуществить поиск вакансий", поскольку они выступают в качестве заполнителей, напоминающих о крупных компонентах системы, которые еще предстоит обсудить. Если вы сознательно принимаете решение о том, чтобы отложить на время рассмотрение крупных частей системы, подумайте о том, чтобы написать одну-две масштабные истории, охватывающие эти части. Эпической истории можно присвоить высокую оценку, взяв ее "с потолка".

Компактность

Словно в сюжете сказки о Златовласке, пытавшейся найти кроватку подходящего размера, одни истории могут оказаться слишком большими, другие — слишком маленькими, а третьи — в самый раз. От размера истории зависит очень многое, поскольку слишком большие или слишком маленькие истории сложно оценивать при планировании. С эпическими историями трудно работать, так как они часто содержат в себе несколько историй. Например, в случае системы для турбюро история "Пользователь может составить маршрут путешествия на время отпуска" является эпической. Планирование маршрута представляет собой важную функциональную часть системы турбюро, но подразумевает решение целого ряда задач. Эпические истории следует разбивать на истории меньшего размера. Окончательное решение относительно того, имеет ли история подходящий размер, зависит от команды, ее возможностей и используемых технологий.

Разбиение историй

Обычно эпические истории можно отнести к одной из двух категорий:

- составные истории;
- сложные истории.

Эпическую историю называют *составной*, если она заключает в себе несколько небольших историй. Например, рассмотрим историю "Пользователь может поместить на сайте свое резюме". Для начальной стадии планирования системы эта история вполне подойдет. Но когда разработчики приступят к обсуждению проекта с заказчиком, то выяснится, что "поместить на сайте свое резюме" фактически означает следующее:

- резюме может включать данные об образовании, трудовом стаже, зарплате, публикациях, волонтерской деятельности и желаемой должности;
- пользователи могут помечать резюме как неактивные;
- пользователи могут иметь несколько резюме;

- пользователи могут изменять резюме;
- пользователи могут удалять резюме.

В зависимости от того, сколько времени займет разработка, каждый из этих пунктов можно было бы сделать самостоятельной историей. Но если пойти по такому пути, главное — не увлечься и не превратить эпическую историю в множество слишком мелких историй. Так, в зависимости от используемых технологий, а также численности и квалификации команды, приведенные ниже истории в большинстве случаев можно считать чрезмерно детализированными.

- Соискатель может ввести в резюме даты начала и окончания для каждого из периодов волонтерской деятельности.
- Соискатель может изменить в резюме даты начала и окончания для каждого из периодов волонтерской деятельности.
- Соискатель может ввести в резюме сроки пребывания на каждой из предыдущих должностей.
- Соискатель может изменить в резюме сроки пребывания на каждой из предыдущих должностей, указанных в резюме.

Вообще говоря, рекомендуется группировать небольшие истории, как показано ниже.

- Пользователь может создать резюме, включающее данные об образовании, трудовом стаже, зарплате, публикациях, волонтерской деятельности и желаемой должности.
- Пользователь может изменить резюме.
- Пользователь может удалить резюме.
- Пользователь может иметь несколько резюме.
- Пользователь может активизировать резюме или сделать его неактивным.

Как правило, составную историю можно разбить на ряд историй меньшего размера несколькими способами. В приведенном выше примере разбиение истории на более мелкие было выполнено в соответствии с обычно используемыми операциями создания, изменения и удаления элементов. Такой подход работает хорошо, если история, соответствующая созданию элемента, достаточно невелика, чтобы ее можно было оставить в качестве неделимой истории. Другой возможный подход основывается на разбиении историй в соответствии с используемыми типами данных. Для этого следует представить себе, что каждый элемент резюме добавляется и изменяется по отдельности. Такой способ рассуждений приводит к совершенно иной схеме разбиения.

- Пользователь может добавить или изменить данные об образовании.
- Пользователь может добавить или изменить данные о трудовом стаже.
- Пользователь может добавить или изменить данные о заработной плате.
- Пользователь может добавить или изменить данные о публикациях.
- Пользователь может добавить или изменить данные о волонтерской деятельности.
- Пользователь может добавить или изменить название желаемой должности и т.д.

Сложными, в отличие от составных, называют истории, большой размер которых обусловлен самой их сутью и которые нелегко расчленить на ряд составляющих историй. Если история является сложной в силу каких-либо связанных с ней неопределенных обстоятельств, попытайтесь разбить ее на две истории: исследовательскую и относящуюся непосредственно к разработке новой функциональности. Например, предположим, что разработчикам передали историю "Компания может оплатить публикацию вакансии с помощью кредитной карты", но никто из разработчиков до этого никогда не сталкивался с обработкой платежных карт. Тогда они могут разбить первоначальную историю на следующие две.

- Изучить способы работы с платежными картами в Интернете.
- Пользователь может произвести оплату с помощью кредитной карты.

В данном случае первая история потребует выполнения экспериментальной разработки (спайка) силами одного или нескольких разработчиков. Если используется такой способ разбивки, всегда устанавливайте для исследовательской истории строго фиксированные временны*е* рамки. Даже если эту историю не удается оценить с разумной точностью, всегда есть возможность определить максимальный промежуток времени, которое должно быть отведено для соответствующего изучения.

Сложные истории часто встречаются при разработке новых или расширении известных алгоритмов. В одной биотехнологической компании группе разработчиков пришлось столкнуться с пользовательской историей, связанной с модернизацией стандартного статистического подхода, называемого EM-алгоритмом (expectation maximization algorithm). Он используется для нахождения оценок максимального правдоподобия параметров вероятностных моделей, зависящих от скрытых переменных. Сложная история была переписана в виде двух историй: одной — для проведения исследования и вынесения заключения относительно возможности осуществления запланированного новшества и второй — для добавления соответствующей функциональности в продукт. В ситуациях, подобных этой, трудно оценить, сколько времени уйдет на исследовательскую историю.

Подумайте о включении спайка в другую итерацию

При малейшей возможности старайтесь поместить исследовательскую историю в одну итерацию, а остальные истории — в одну или несколько последующих итераций. Обычно оценить удается только исследовательскую историю. Включение остальных, не поддающихся оценке историй в одну итерацию с исследовательской будет означать, что степень неопределенности того, какой объем работ может быть выполнен за данную итерацию, будет выше, чем обычно.

Ключевое преимущество разбиения историй, оценка которых затруднена, состоит в том, что это позволяет заказчику установить для исследований и создания новой функциональности отдельные приоритеты. Назначая приоритет сложной истории ("Добавить новые расширения в EM-алгоритм") и зная только суммарную оценку ее трудоемкости, заказчик будет неявно предполагать, что новая функциональность заведомо сможет быть реализована в отведенные сроки, хотя это предположение может оказаться ошибочным. Если же вместо этого заказчику предоставят исследовательскую историю — спайк ("изучить возможность расширения EM-алгоритма и сделать соответствующее

заключение") и функциональную историю ("расширить EM-алгоритм"), то он сможет выбирать, как следует поступить — включить ли в данную итерацию исследовательскую историю, которая не добавляет никакой новой функциональности, или, возможно, какую-то другую историю, привносящую новую функциональность.

Объединение историй

Иногда истории оказываются слишком мелкими. Как правило, разработчики не хотят записывать или оценивать такие истории, поскольку на это уйдет больше времени, чем на внесение самих изменений в продукт. Типичными примерами историй, часто имеющих небольшой размер, являются истории, в которых содержатся описания изменений в пользовательском интерфейсе и способов генерации отчетов об ошибках. Обычной практикой команд, использующих методологию экстремального программирования, является объединение совсем небольших историй в более крупные, на которые уйдет от половины рабочего дня до нескольких рабочих дней. Объединенная история получает имя, включается в план и обрабатывается точно так же, как и любая другая история.

Пусть имеется проект, в котором обнаружено пять ошибок и для которого запрошены изменения в цветовой гамме экрана поиска. Разработчики оценивают общую трудоемкость связанных с этим работ, и вся совокупность изменений обрабатывается как одна история. В случае использования бумажных карточек это технически обеспечивается путем прикрепления карточек всех небольших историй к карточке-обложке степлером.

Тестируемость

Истории должны писаться так, чтобы их можно было тестировать. Подтверждением того, что история разработана успешно, служит удачное прохождение ею тестов. Если историю нельзя протестировать, каким образом разработчики смогут узнать, что создание исходного кода завершено?

Нетестируемые истории обычно появляются в случае нефункциональных требований, т.е. таких, которые относятся к самому программному обеспечению, а не непосредственно к его функциональным возможностям. Рассмотрим, например, следующие нефункциональные истории.

- Пользователь должен считать программное обеспечение удобным в использовании.
- Пользователь никогда не должен долго ждать появления какого-либо экрана.

В том виде, в котором они написаны, эти истории не являются тестируемыми. По возможности следует автоматизировать все тесты, т.е. стремиться к 99% автоматизации, а не к 10%. Почти всегда можно автоматизировать больше, чем кажется на первый взгляд. Если продукт разрабатывается инкрементно, то все изменяется очень быстро и код, прекрасно работавший вчера, сегодня может перестать работать. Вам нужны такие тесты, которые смогут обнаружить это на как можно более ранних стадиях.

Существует лишь небольшое подмножество тестов, которые не поддаются реалистичной автоматизации. Например, пользовательская история, в которой записано: "Пользователь-новичок в состоянии выполнять типичные операции без специального обучения", может быть протестирована, но не автоматизирована. Вероятно, для тестирования этой истории потребуется привлечь специалиста по учету человеческого фактора,

чтобы тот спроектировал тест, включающий в себя наблюдение за репрезентативной выборкой пользователей-новичков. Тест такого рода может оказаться дорогостоящим и затратным по времени, но зато история станет тестируемой и вполне подойдет для некоторых продуктов.

История "Пользователь никогда не должен долго ждать появления какого-либо экрана" не относится к тестируемым, поскольку в ней фигурирует обстоятельство "никогда" и не определено, что означает "долго ждать". Продемонстрировать, что нечто *никогда* не случится, невозможно. Гораздо легче и рациональнее стремиться продемонстрировать, что это "нечто" может происходить очень редко. Данная история может быть переписана в виде "В 95% случаев для отображения нового экрана потребуется не более двух секунд". А еще лучше — написать автоматизированный тест, с помощью которого это можно было бы проверить.

Резюме

- В идеальном случае истории не должны зависеть друг от друга. Это не всегда получается, но в той степени, в какой это возможно, истории следует писать так, чтобы их можно было разрабатывать в любой очередности.
- Детали истории отрабатываются в ходе обсуждения между пользователем и разработчиками.
- Истории следует писать таким образом, чтобы их ценность для пользователей или заказчиков была очевидной. Наилучший способ достижения этой цели состоит в том, чтобы поручить написание истории заказчику.
- Истории могут снабжаться примечаниями с описанием деталей, но чрезмерное обилие деталей мешает пониманию сути истории и может создавать ложное впечатление, будто никакого обсуждения деталей между разработчиками и заказчиком уже не потребуется.
- Одним из наилучших способов комментирования историй является написание вариантов тестирования для каждой истории.
- Слишком большие, составные или сложные истории лучше разбивать на несколько небольших историй.
- Если имеется множество мелких историй, имеет смысл объединить их в одну историю большего размера.
- Истории должны быть тестируемыми.

Обязанности разработчика

- Вы обязаны помогать заказчику в написании историй, которые не были бы подробными спецификациями, а отражали намерение провести дальнейшее обсуждение, представляли ценность для пользователей или заказчика, а также были независимыми, тестируемыми и имели адекватный размер.
- Вы обязаны следить за тем, чтобы в историях описывались не особенности использования той или иной технологии или части инфраструктуры, а потребность

в них с точки зрения ценности, которую они представляют для пользователей или заказчика.

Обязанности заказчика

- Вы ответственны за то, чтобы написанные истории были не подробными спецификациями, а служили своего рода протоколом о намерениях провести дальнейшее обсуждение, представляли ценность для пользователей или для вас, а также были независимыми, тестируемыми и имели подходящий размер.

Контрольные вопросы

2.1. Укажите для каждой из приведенных ниже историй, может ли она считаться хорошей. Если нет, то почему?

А. Пользователь может быстро освоить систему.

Б. Пользователь может изменить адрес, указанный в резюме.

В. Пользователь может добавить, изменить и удалить несколько резюме.

Г. Система может аппроксимировать распределения квадратичных форм в нормальных координатах методом седловой точки.

Д. Все ошибки времени выполнения заносятся в журнал унифицированным способом.

2.2. Разбейте эпическую историю “Пользователь может создавать и изменять автоматизированные агенты поиска вакансий” на составляющие истории подходящего размера.