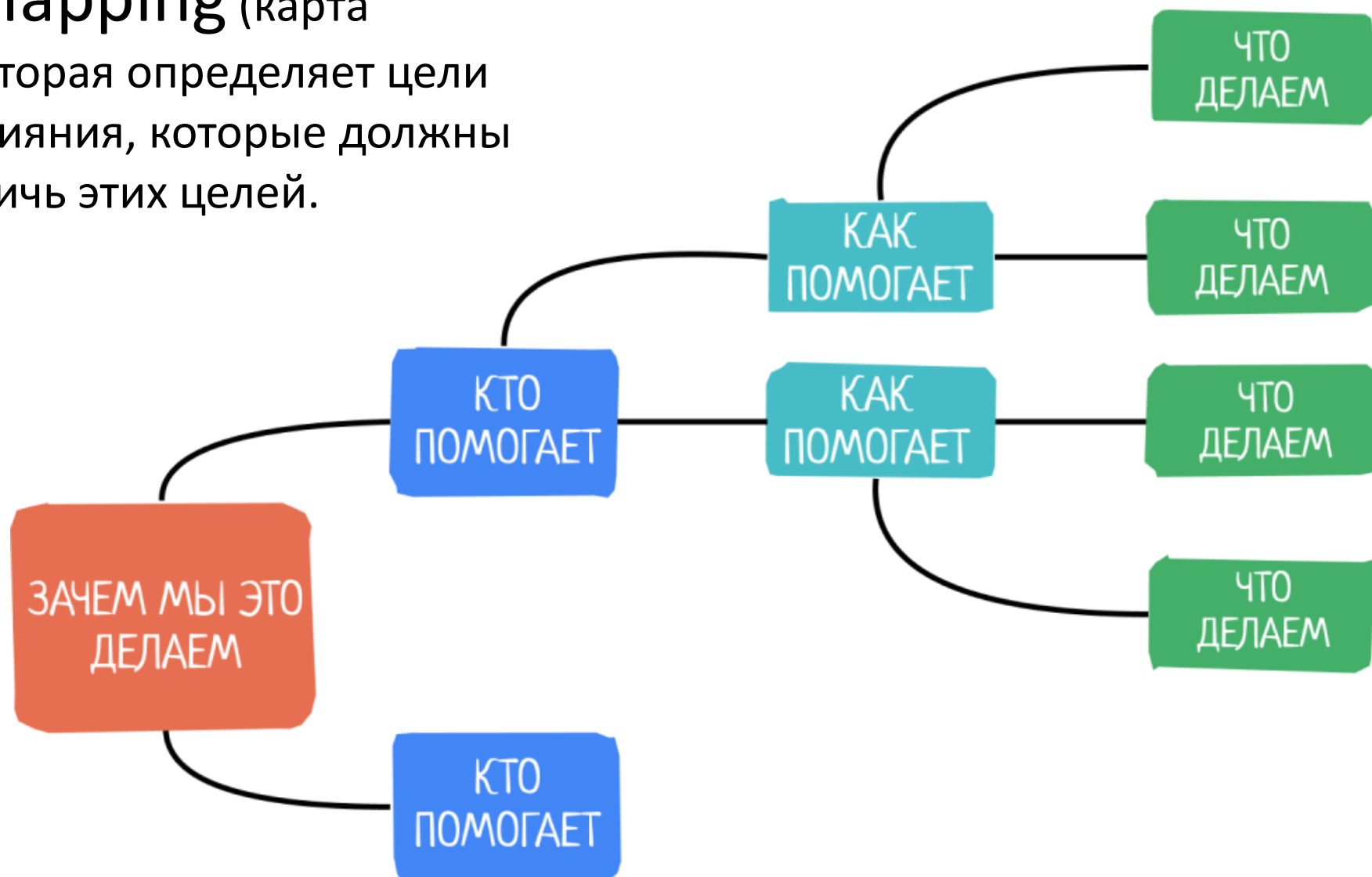




Лекция 5

Критерии готовности и критерии приёмки User Story

Impact Mapping (карта влияний), которая определяет цели проекта и влияния, которые должны помочь достичь этих целей.



Impact Mapping

Пример

Объект измерения	Количество выдаваемых кредитов через сайт банка в месяц
Где считаем	Ежемесячный отчет кредитного отдела
Сейчас	500
Цель	1 000

Увеличить количество выдаваемых кредитов с сайта в 2 раза через 6 месяцев после запуска

Клиенты сайта

Отправлять валидные данные в анкете на кредит

Валидация анкет

Автоматическое заполнение полей анкеты через ГосУслуги

Подавать заявки на кредит через сайт

Оптимизация интерфейса и логики работы анкет на сайте

Отдел кредитования банка

Быстрее обрабатывать заявки на кредит

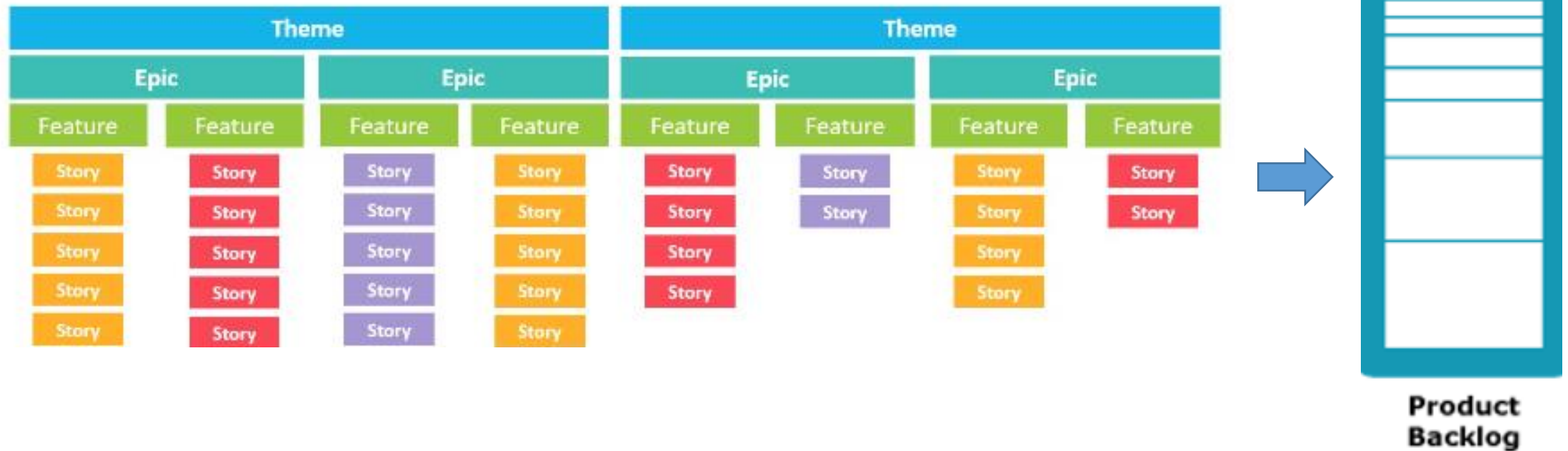
Автоматическая проверка данных анкет на валидность

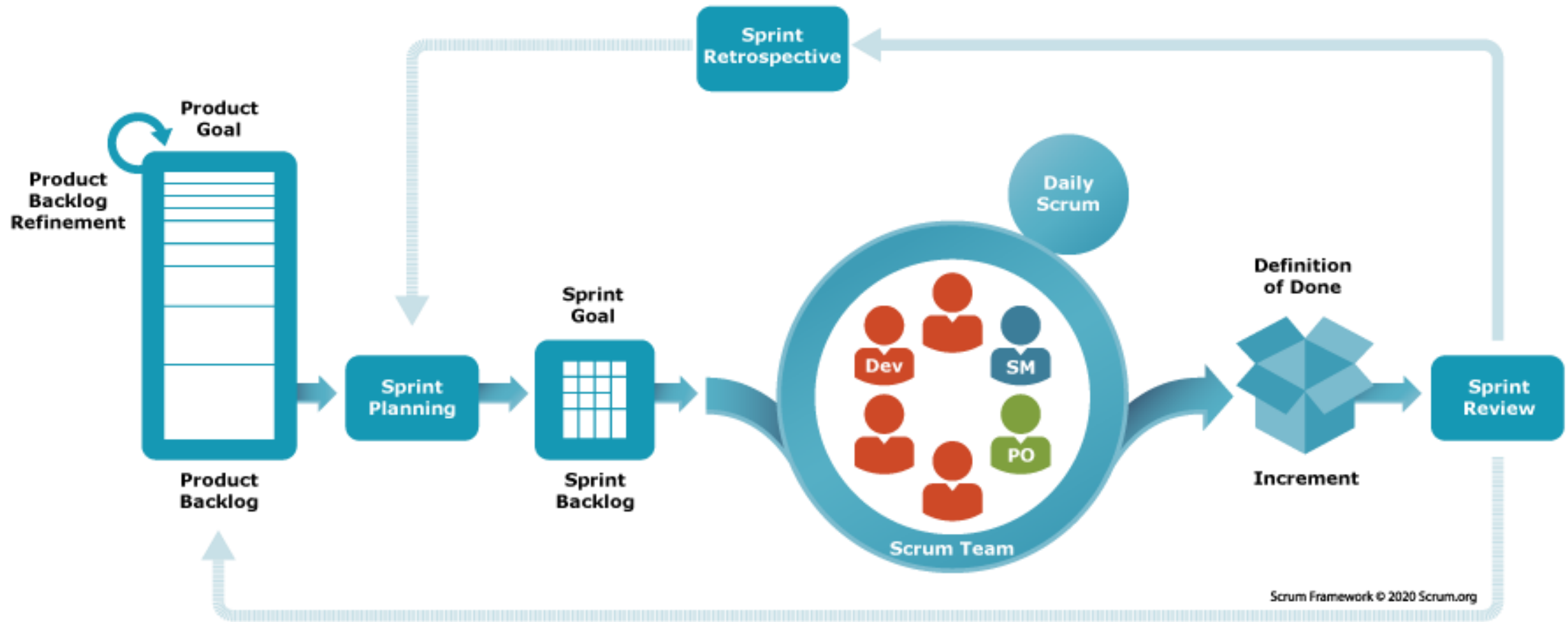
Автоматическая проверка заявителя в БКИ

Интерфейс для работы с заявками — типа CRM

Impact Map помогает определить влияние различных факторов на бизнес-результаты, а **USM** помогает структурировать и организовать работу над продуктом.

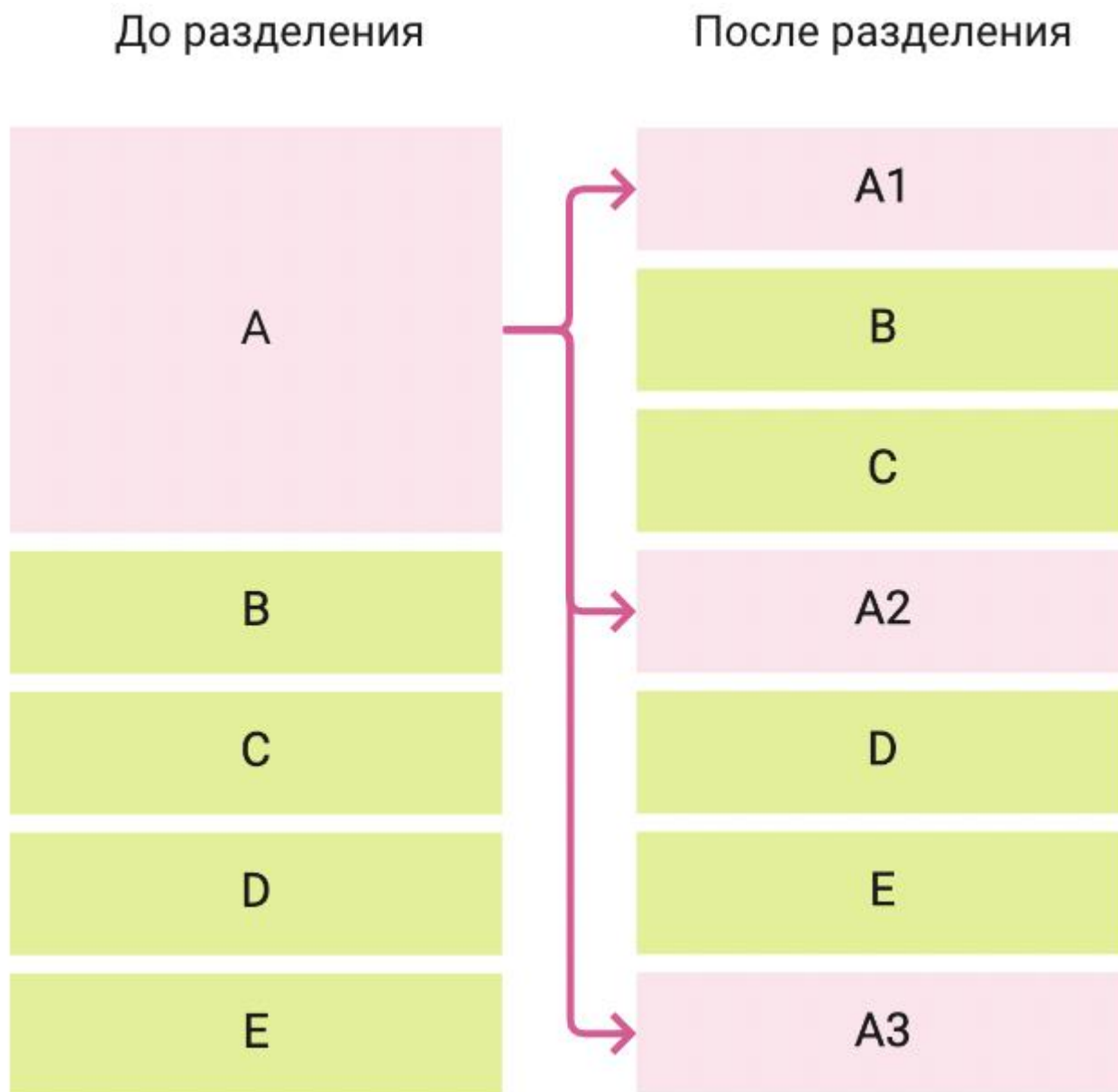
USER STORY MAPPING





- 1) Разделяя крупные истории на истории поменьше, вы автоматически проводите более детальный анализ задач и учитываете большее количество нюансов.
 - 2) И тем самым снижаете риски провала задачи.
 - 3) А это означает, что повышается вероятность успешного выполнения такой задачи.
 - 4) И оценка задачи будет точнее.
 - 5) А еще, меньший размер историй означает, что можно запланировать на итерацию их большее количество.
 - 6) При возникновении какой-то непредвиденной ситуации, заблокированной окажется только одна история из нескольких запланированных на спринт, а не та единственная большая, которую вы еле-еле впихнули бы в итерацию. Аналогично, при необходимости изменений, они затронут конкретные небольшие задачи, а не одну большую целиком, и другие задачи смогут двигаться дальше.
 - 7) Уменьшится время ожидания выполнения работы другими командами по каждой истории.
 - 8) Разработчик хорошо помнит код последних X часов. Чем меньше задача, тем лучше разработчик помнит написанный код, а значит быстрее произведет поиск и исправление дефектов.
 - 9) В конце спринта (по Скраму) история, как и любой другой элемент бэклога, должна соответствовать Definition of Done – определению готовности. Команда учится завершать каждую задачу до состояния DoD, а не копить набор из десятка задач готовых на 99%.
 - 10) Команда сможет показать готовую задачу еще в ходе спринта, не дожидаясь его окончания, получить комментарии и успеть адаптировать результат до Обзора спринта и демонстрации инкремента.
- Это ведет к продукту, более соответствующему запросам клиентов и пользователей.

11. Небольшие задачи позволяют точнее приоритизировать, и тем самым раньше поставлять самое важное и откладывать менее важное.



DoR, DoD, AC: критерии готовности и критерии приёмки User Story

При разработке продуктов в рамках Agile-подхода с использованием **User Story**

Как [РОЛЬ]

Я хочу [ДЕЙСТВИЕ]

Чтобы [ЦЕННОСТЬ]

US — короткая формулировка потребности пользователя, в которой больше внимания уделяется цели пользователя и меньше — деталям реализации.

НО на практике качественная реализация US невозможна, если требования сформулированы только в виде US: каждый участник команды разработки, включая заказчика, может иметь разное понимание US и степень ее готовности.

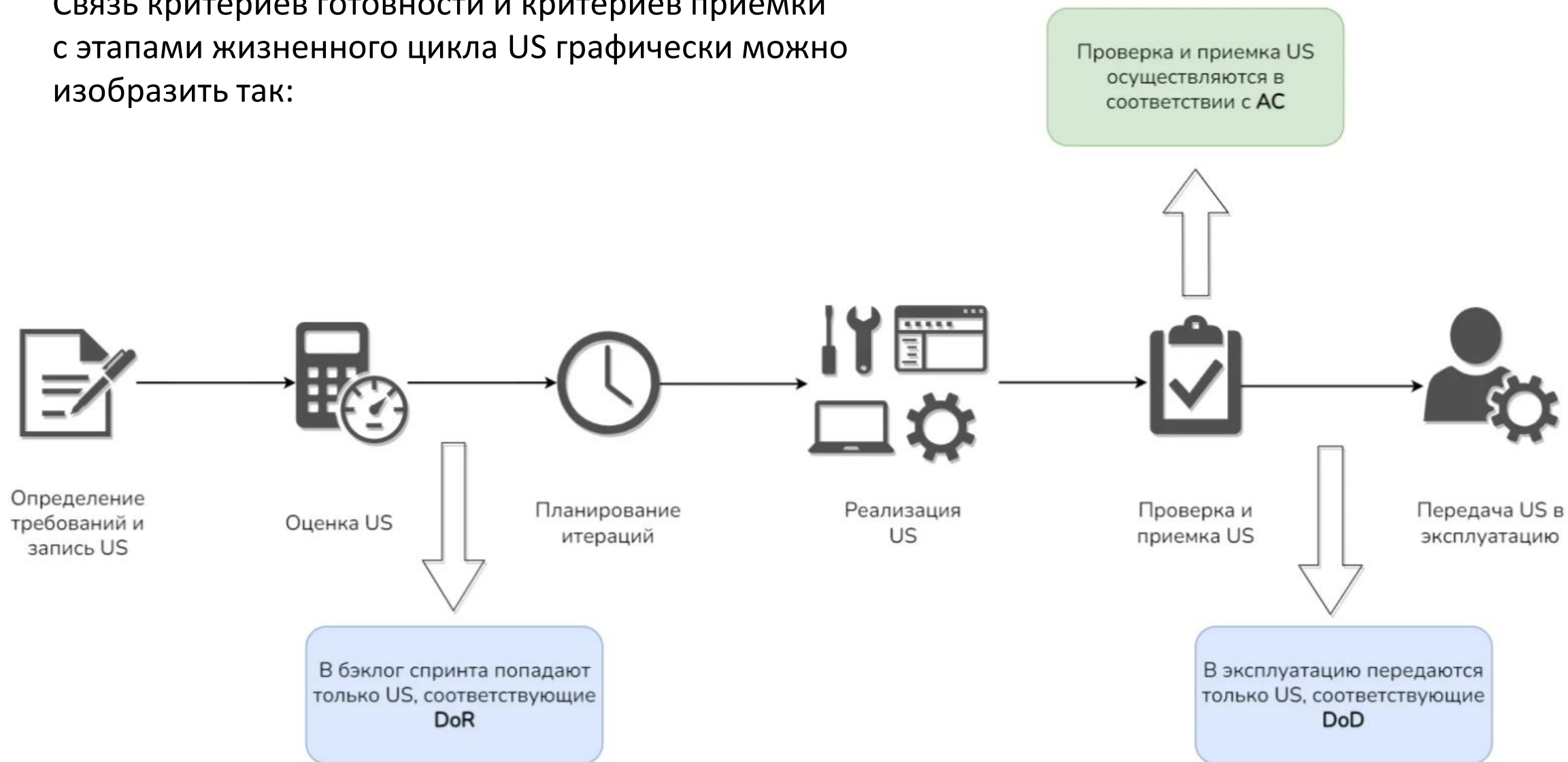
В решении этой проблемы помогает практика совместной командной выработки:

- ✓ **дополнительных артефактов**, описывающих реализацию (*макеты интерфейсов, диаграммы процессов, модели данных, спецификации и т.д.*);
- ✓ **критериев приемки AS** (Acceptance Criteria);
- ✓ артефактов процесса разработки — **критериев готовности**: DoR (Definition of Ready), DoD (Definition of Done).

Этапы жизненного цикла User Story в Agile

Наименование этапа	Описание этапа
Определение требований и запись US	Аналитики определяют потребности пользователей, формулируют конкретные требования к системе и записывают US с использованием определённого формата (название, описание, приоритет, критерии приёмки и т.д.)
Оценка US	Вся команда оценивает сложность и затраты, условия и готовность к реализации каждой US. При этом не требуется оценивать сразу все US в бэклоге — достаточно оценить самые приоритетные. Именно здесь команде нужны DoR — переход US на следующий этап невозможен, если она не соответствует критериям DoR.
Планирование итераций	Вся команда планирует реализацию US в конкретные итерации продукта.
Реализация US	Команда разработки реализует US в соответствии с определёнными критериями приёмки — Acceptance Criteria, уникальными для каждой US.
Проверка и приёмка US	US проверяется на готовность к сдаче и передаче конечному пользователю, а также на соответствие заданным критериям приёмки и, если они успешно проходят проверку, то считаются завершёнными. . Здесь команде нужны AC и DoD. US не может быть объявлена готовой без соответствия AC и не может перейти на следующий этап, если она не соответствует DoD.
Передача US в эксплуатацию	Когда реализованная US соответствует DoD, она включается в состав инкремента продукта и передаётся в эксплуатацию пользователю.

Связь критериев готовности и критериев приёмки с этапами жизненного цикла US графически можно изобразить так:



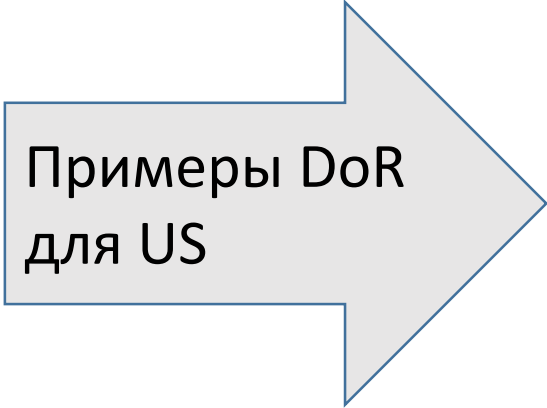
DoR для UserStory

DoR (Definition of Ready) — общие критерии готовности US к передаче в работу команде разработки.

В контексте US DoR представляет собой некий контрольный список условий, которым должна удовлетворять US перед тем, как она может быть передана в разработку.

DoR одинаковы для всех US продукта.

Использование DoR позволяет исключить взятие в спринт недостаточно проработанной US.



Примеры DoR
для US

- ✓ US имеет ясное и краткое описание
- ✓ DoD определены и утверждены командой
- ✓ Все необходимые ресурсы, технические спецификации и прототипы доступны
- ✓ Все зависимости от других US зафиксированы
- ✓ US приоритезирована и оценена командой
- ✓ Команда договорилась об Acceptance Criteria

DoR для спринта

- В контексте спринта DoR — перечень условий, которые должны быть выполнены, чтобы команда могла начать планирование и выполнение спринта.
- DoR одинаковы для всех спринтов в разработке продукта.

Примеры DoR для спринта

- ✓ Все пользовательские истории, выбранные для спринта, оценены и имеют DoR
- ✓ У команды определены цели и ожидаемые результаты для спринта
- ✓ Вся необходимая документация и дизайн доступны
- ✓ Команда имеет доступ к необходимым ресурсам, таким как серверы, тестовые данные и так далее
- ✓ Команда имеет определённый бюджет и ресурсы для выполнения спринта
- ✓ Команда имеет понимание требований продукта и его структуру
- ✓ Команда имеет определённый план спринта и понимает, как он будет реализован
- ✓ Команда имеет определённую методологию разработки, такую как Scrum или Kanban
- ✓ Команда имеет общее понимание того, какие задачи будут выполнены в течение спринта и кто за них ответственен
- ✓ Команда имеет определённую систему отслеживания прогресса и отчётности о выполнении задач в рамках спринта

DoD Definition of Done

Для DoD актуальны те же особенности, что для DoR:

- ✓ Могут применяться как к US, так и к спринтам
- ✓ Одинаковы для всех US (спринтов) в разработке продукта
- ✓ Определяются командой до начала первого спринта
- ✓ Могут видоизменяться в процессе работы команды

DoD для US

DoD (Definition of Done) — **перечень условий**, которым должна соответствовать US для того, **чтобы её можно было передать в эксплуатацию пользователям.**

Примеры DoD для US

- ✓ Реализация US соответствует Acceptance Criteria
- ✓ Код, связанный с историей протестирован и отлажен
- ✓ Документация истории создана или обновлена
- ✓ Все необходимые одобрения истории получены
- ✓ Код, связанный с историей выложен в систему контроля версий
- ✓ Все возможные риски оценены и предусмотрены

DoD для для спринта

В контексте спринта DoD — **список условий, которые должны выполняться для завершения спринта.**

Примеры DoD
для спринта

- ✓ Все пользовательские истории, выбранные для релиза, завершены и протестированы
- ✓ Качество продукта оценено и проверено командой QA
- ✓ Документация обновлена и актуальна
- ✓ Продукт совместим с целевой производственной средой
- ✓ Производственное окружение настроено и готово к использованию
- ✓ Продукт интегрирован с другими необходимыми системами и приложениями
- ✓ Все задачи, связанные с релизом, закрыты
- ✓ Команда имеет понимание процесса релиза и план выпуска
- ✓ Ресурсы, такие как серверы, базы данных и т.д., готовы к использованию
- ✓ Продукт протестирован и утверждён заказчиком

Критерии приёмки (Acceptance Criteria)

AC (Acceptance Criteria) — критерии приёмки результата реализации US.

AC представляет собой **список условий**, которые должны выполняться, **чтобы US удовлетворяла требованиям пользователей**.

Для каждой User Story продукта AC будут уникальными.

Acceptance Criteria позволяет упростить процессы разработки и тестирования через однозначное понимание командой требований, которым должен удовлетворять продукт.

С помощью AC User Story проверяется на соответствие требованиям пользователей и заказчика.

Acceptance Criteria помогают:

- ✓ Определить границы User Story
- ✓ Достигнуть консенсуса между командой и заказчиком о том, что именно делаем
- ✓ Увеличить точность оценки User Story, качественно планируем занятость команды и сроки выпуска продукта
- ✓ Готовить тестовые кейсы и автоматизировать тестирование
- ✓ Выявлять негативные сценарии

При написании АС наиболее часто применяемые подходы:

- ✓ Свод правил
- ✓ Сценарно-ориентированный подход

✓ Критерии приёмки как свод правил

Свод правил включает в себя описание требуемого функционала, ограничения по производительности, безопасности и другие.

В этом случае критерии оформляются в виде обычного списка. Иногда такой перечень называют чек-листом или контрольным списком.

Как пользователь

Я хочу искать товар в пределах установленного диапазона стоимости

Чтобы найти подходящие мне товары в моей ценовой категории

Формат истории позволяет понять задачу, в ней указана роль, функционал и польза от его реализации. Мы обсудили и оценили User Story с командой, определили, что US соответствует INVEST-критериям: она не имеет зависимостей, достаточно небольшая и тестируемая.

Однако представленного описания недостаточно для передачи User Story в разработку. Поэтому мы должны задать себе вопрос: «Какие проверки должна выполнить команда перед передачей функции заказчику?»

Задаем вопросы заказчику

1. Какой диапазон стоимости товаров будет доступен для поиска?
2. Будет ли у пользователя возможность указать точную цену или только минимальную и максимальную цену?
3. Нужна ли функция автодополнения при вводе диапазона цен, чтобы пользователи могли быстро выбрать подходящий диапазон?

Вопросы имеют большое значение для пользовательского опыта и являются ключевыми для понимания ожиданий от функции.

4. Какие ошибки или нежелательное поведение должны быть обработаны, если пользователь вводит неправильный диапазон цен?
5. Нужно ли кэшировать результаты поиска, чтобы ускорить поиск при повторном запросе с тем же диапазоном цен?
6. Как будет осуществляться поиск товаров в заданном диапазоне цен? Будет ли это происходить автоматически при вводе диапазона, или пользователю нужно будет нажать кнопку для начала поиска?

Тоже важны для пользователя, но, если заказчик не готов на них ответить, эти аспекты стоит решить в команде..

7. Каким образом будут отображаться результаты поиска товаров в заданном диапазоне цен?
8. Какие ограничения на количество результатов поиска?
9. Будут ли в поиске учитываться товары, которых нет в наличии?
10. Будут ли в поиске учитываться товары, ожидающие поступления в ближайшее время?

Вопросы относятся к функционалу, который может улучшить опыт пользователя и производительность системы, но не являются обязательными.

Команда



Какой диапазон стоимости товаров будет доступен для поиска?

Заказчик

Нам нужно, чтобы пользователь мог указывать диапазон цен самостоятельно, но мы также хотим предоставить предустановленные фильтры для цен, например, дешевые товары до 50 рублей или дорогие товары свыше 5 000 рублей.



Команда



Будет ли у пользователя возможность указать точную цену или только минимальную и максимальную цену?

Заказчик

Хотелось бы, чтобы пользователь мог указывать как точную цену, так и диапазон цен. Можно использовать бегунок или вводить диапазон цены с клавиатуры





Нужна ли функция автодополнения при вводе диапазона цен, чтобы пользователи могли быстро выбрать подходящий диапазон?

Заказчик

Да, мы бы хотели, чтобы функция поддерживала автодополнение, чтобы пользователи могли быстро выбрать подходящий диапазон. Причем автодополнение должно учитывать категорию товара. Например, для категории "Одежда" диапазон может быть от 100 до 50 000 рублей, а для категории "Электроника" - от 50 до 5 000 000 рублей.



Команда



Какие ошибки или нежелательное поведение должны быть обработаны, если пользователь вводит некорректный диапазон цен, например, указывает минимальную цену больше максимальной?

Заказчик

Если пользователь вводит неправильный диапазон цен, мы должны сообщить ему об этом и запросить правильный диапазон.



Команда



Будем ли мы кэшировать результаты поиска? Это позволит при повторном запросе с тем же диапазоном цен ускорить работу приложения и уменьшить нагрузку на сервер. Но нужно учитывать, что в таком случае возможна устаревшая информация в кэше, если цены на товары изменились с момента последнего поиска.

Заказчик



Мы хотим сохранять результаты поиска в кэше на сервере и использовать их при следующем запросе. Но при смене стоимости товара мы хотим, чтобы результат поиска был актуальным.

Команда



Каким образом будут отображаться результаты поиска товаров в заданном диапазоне цен?

Команда



Какие ограничения на количество результатов поиска?

Заказчик

Мы предпочитаем отображать результаты поиска в виде строк, по 10 штук на странице, товары расположить в порядке возрастания цены с возможностью сортировки по цене или другим параметрам.



Команда



Будут ли в поиске учитываться товары, которых нет в наличии?

Команда



Будут ли в поиске учитываться товары, ожидающие поступления в ближайшее время?

Заказчик

Мы не хотим выводить товары, которых нет в наличии. Но хотим показывать товары, подвоз которых ожидается в ближайшее время.



Оформляем **Acceptance Criteria** в виде свода правил с учётом ответов заказчика:

1. Пользователь может установить свой диапазон цены поиска при помощи бегунка, где изначальный диапазон цен соответствует минимальной и максимальной ценам товаров в выбранной категории или группы категорий.
2. Пользователь может установить диапазон цены с клавиатуры.
3. Для активации поиска пользователь нажимает кнопку «Найти».
4. Если пользователь вводит некорректный диапазон цен, система должна выдавать сообщение об ошибке и не показывать пустой список товаров. В приложении — примеры экранных форм и уведомлений об ошибках.
5. Результаты поиска должны быть точными и соответствовать установленному диапазону цен, по умолчанию отображаются в порядке возрастания цены.
6. Результаты поиска разбиваются по 10 штук на странице, обеспечивается переход на предыдущие или следующие страницы при помощи ссылок.
7. Если результаты поиска были закешированы, то они должны быть конкретными и актуальными, с учётом изменений в ценах на товары.
8. Функция поддерживает автодополнение, которое учитывает категорию товара, чтобы пользователи могли быстро выбрать подходящий диапазон цен.
9. Для каждой категории товаров предусмотреть предустановленные фильтры для цен: «дешёвые товары», «дорогие товары» и прочее. Варианты — в зависимости от категории — в приложении.
10. Результаты поиска исключают товары, которых нет в наличии и включают товары, подвоз которых ожидается.

Дополняем список

Нефункциональные требования также являются важными для успешной разработки и работы ПО. Заказчики могут иметь определённое представление о том, как ПО должно работать, но, как правило, команда разработки лучше знает его возможности и ограничения.

Важно договориться о том, какие требования к надёжности, доступности, производительности будут у разрабатываемого ПО. Например, заказчик может формулировать требования к информационной безопасности таким образом: ПО должно иметь защиту от взлома и несанкционированного доступа.

Обсуждаем полученные от заказчика ответы и список критериев приёмки с владельцем продукта и командой разработки. В рамках обсуждения **добавляем в список следующие ограничения:**

- 11. Время поиска товаров должно быть не более 3 секунд.*
- 12. Функция поиска должна быть защищена от SQL-инъекций и атак на XSS.*
- 13. Проверка входных данных должна быть осуществлена, чтобы предотвратить возможность межсайтовой подделки запросов (CSRF).*

✓ Сценарно-ориентированный подход при описании критериев приёмки

Суть сценарно-ориентированного подхода заключается в описании поведения программы через пользовательские сценарии использования. Он позволяет описать варианты использования без подробного описания того, как это поведение реализовано «под капотом» системы. Для такого описания используются специальные языки.

Подход получил своё развитие при

- эволюции **TDD** (Test Driven Development — разработка, основанная на тестировании)
- и появлении **BDD** (Behavior Driven Development — «разработка через поведение») — метода разработки, когда сначала пишутся приёмочные тесты, а потом код).

Преимущества сценарно-ориентированного подхода

- ✓ Простота в написании и понимании как для разработчиков, так и для других заинтересованных сторон. Это облегчает взаимодействие между командой разработки и заказчиком.
- ✓ Ориентация на требования разрабатываемого продукта гарантирует, что процесс разработки будет основываться на потребностях конечных пользователей.
- ✓ Возможность повторного использования кода благодаря тому, как написаны эти сценарии. Это позволяет экономить ресурсы и использовать части этого кода при создании других тестов.

Использование Gherkin

Gherkin — это сценарно-ориентированный язык, который легко читается бизнесом и используется для описания функциональности программного обеспечения. Использование Gherkin позволяет сократить разрыв между бизнесом и разработчиками,

Как пользователь

Я хочу искать товар в пределах установленного диапазона стоимости

Чтобы найти подходящие мне товары в моей ценовой категории



Feature:

Фильтрация товаров по цене

Scenario:

Поиск товаров по диапазону цен – базовый вариант

Given

Я нахожусь на главной странице магазина

When

Я ввожу "1000" в поле "от"

And

Я ввожу "5000" в поле "до"

And

Я нажимаю кнопку "Найти"

Then

Я вижу список товаров со стоимостью в указанном диапазоне

But

Я не вижу товары, которых нет в наличии

Сценариев
может
быть
несколько

Feature: Фильтрация товаров по цене

Background:

Given Я авторизованный пользователь

And Я нахожусь на главной странице магазина

Scenario: Поиск товаров по диапазону цен - базовый вариант

When Я ввожу "1000" в поле "от"

And Я ввожу "5000" в поле "до"

And Я нажимаю кнопку "Найти"

Then Я вижу список товаров со стоимостью в указанном диапазоне

But Я не вижу товары, которых нет в наличии

Scenario: Поиск товаров по диапазону цен из листа ожидания

When Я ввожу "1000" в поле "от"

And Я ввожу "5000" в поле "до"

And Я выбираю вариант "До двух дней" из выпадающего списка "Могу подождать до"

And Я нажимаю кнопку "Найти"

Then Я вижу список товаров со стоимостью в указанном диапазоне

And товары в статусах "в Наличии" и "Подвоз до 2х дней"

Scenario: Поиск товаров по диапазону цен - ограничение диапазона

When Я ввожу "1000" в поле "от"

And Я ввожу "500000000000" в поле "до"

And Я нажимаю кнопку "Найти"

Then Я вижу список товаров со стоимостью в диапазоне от 1000 до максимальной стоимости товара в указанной категории

And Я вижу сообщение об уменьшении диапазона цены

Acceptance Criteria (AC) решают множество задач, главная из которых — избежать двусмысленности и обеспечить чёткое понимание того, что должно быть достигнуто в результате работы над User Story.

- ✓ **Acceptance Criteria как свод правил** имеет свободный формат и может включать ограничения на функциональность, производительность, безопасность или любые другие требования, которые могут быть сформулированы в виде списка.
- ✓ **Сценарно-ориентированный подход** — описание поведения программы через сценарии использования. Мы разобрали на примере как можно писать сценарии на языке Gherkin, расширяя условия и объединяя их общим бэкграундом.

Сравнительная таблица: DoR, DoD, AC (в контексте US):

Definition of Ready (DoR)	Definition of Done (DoD)	Acceptance Criteria (AC)
Готовность (подготовленность) User Story к передаче в работу	Готовность User Story к передаче в эксплуатацию.	Критерии приёмки результата реализации US
DoR — контрольный список команды, позволяющий убедиться, что у вас есть всё необходимое для начала работы над User Story.	DoD — контрольный список команды, обеспечивающий ясность и однородность в определении всеми членами команды того, что работа над User Story полностью завершена.	AC — список условий, которые должны выполняться для US, чтобы она удовлетворяла потребностям пользователя и соответствовала ожиданиям заказчика.
Какие		
Одинаковые для всех User Stories продукта.		Уникальные для каждой User Story.

Beatifulness...



Hum.... let's see the
acceptance criteria for
this item...

Acceptance criteria #225:

- It must be beautiful
- Not ugly
- Shiny
- Pleasant
- Unicornlike

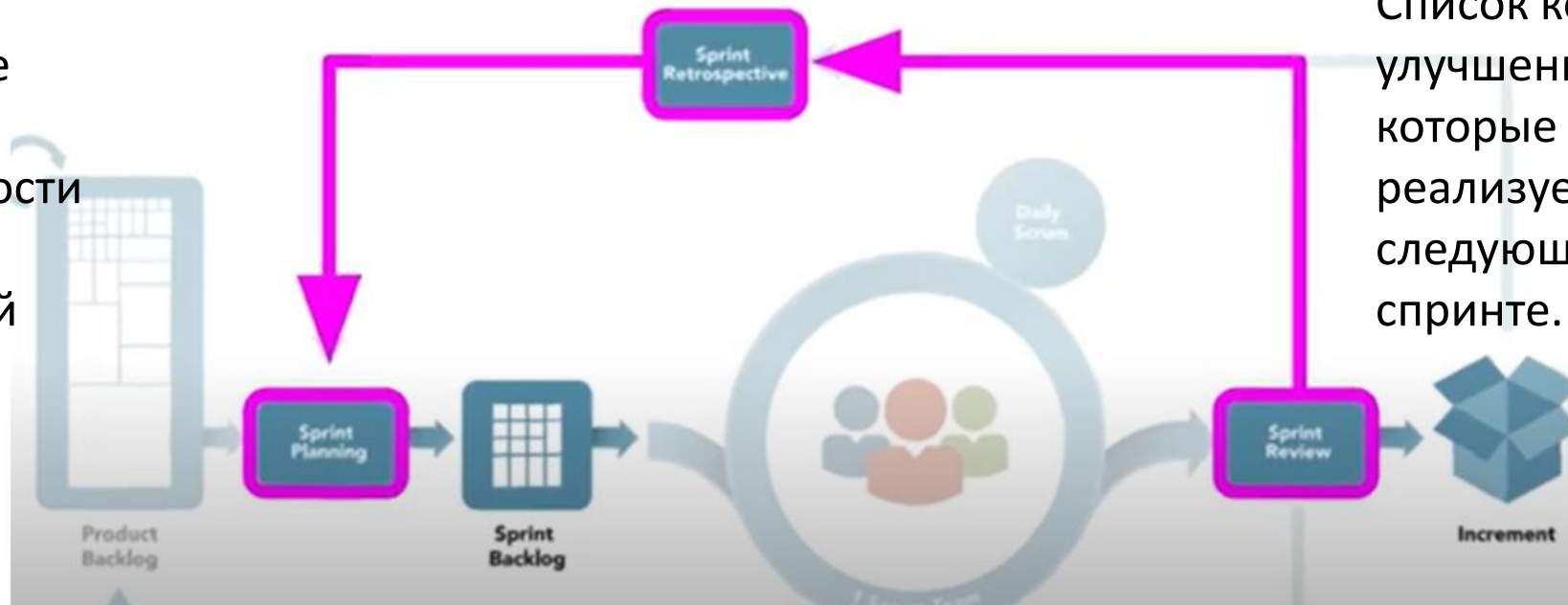
Ретроспектива

Цель Sprint Retrospective — запланировать повышение качества и эффективности. Scrum Team инспектирует то, как прошел последний Sprint в отношении людей, взаимодействий, процессов, инструментов и определения готовности.

Фреймворк Скрам

Что на входе:

- Прошедший спринт
- Обратная связь после Обзора спринта
- Определение готовности DoD
- Задачи с предыдущей Ретроспективы



Что на выходе:

Список конкретных улучшений, которые команда реализует в следующем спринте.