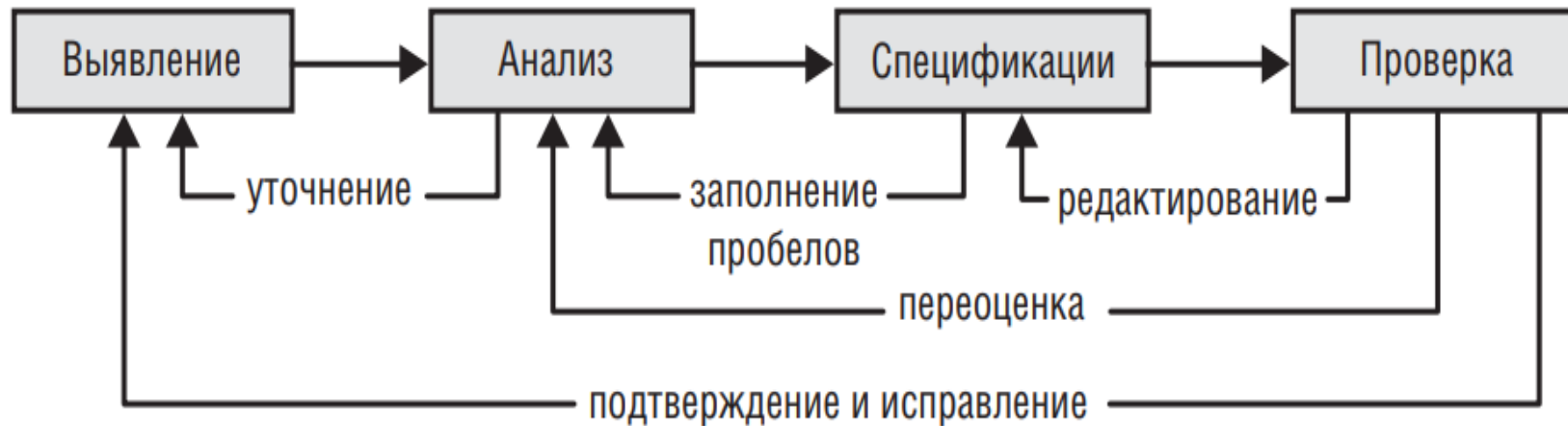


Лекция 2. Выявление требований

- Каркас процесса создания требований
- Распределение работ с требованиями на протяжении жизненного цикла проекта
- Этап выявления требований
- Билль о правах клиента ПО при формировании требований
- Use Case
- User Story. INVEST критерии
- User Story Mapping
- Customer Journey Map
- Методы приоритизации требований

Каркас процесса создания требований

Разработка требований состоит из выявления, анализа, документирования и проверки. На практике эти действия выполняются попеременно, поэтапно и повторяются.



Будучи аналитиком, вы будете задавать клиентам вопросы, слушать их ответы и наблюдать, что они делают (этап выявления требований). Вы обработаете эту информацию и разберетесь в ней, классифицируете ее на разные категории и свяжете потребности клиента с возможными программными требованиями (этап анализа). В результате анализа может оказаться, что нужно уточнить некоторые требования, поэтому вы возвращаетесь назад и дополняете набор требований. После этого вы структурируете информацию от пользователей и выведенные требования в виде письменных требований утверждений и диаграмм (спецификация). При письменной фиксации может потребоваться вернуться назад и выполнить дополнительный анализ, чтобы закрыть пробелы в знаниях. Затем вы просите ряд заинтересованных лиц подтвердить, что представленное вами точно и полно отражает потребности, и исправить ошибки (проверка). Это выполняется по отношению к набору требований, которые важнее и своевременнее всего для начала разработки ПО. В процессе проверки может потребоваться переписать некоторые неясные требования, повторить некоторые действия анализа или даже вернуться и выполнить дополнительное выявление требований. После этого вы переходите к следующему этапу проекта и весь цикл повторяется. Такой итеративный процесс продолжается на всем протяжении разработки требований и возможно — как в проектах гибкой разработки — на протяжении всего времени проекта.

Распределение работ с требованиями на протяжении жизненного цикла проекта в разных моделях разработки



В чистом **водопадном цикле** планируется только один основной выпуск, поэтому основной объем работы с требованиями приходится на начало проекта. Но даже если спланировать этап «выявления требований» в начале проекта, после чего требования используются для проектирования, нужно рассчитывать на выполнение небольшого объема работ с требованиями на протяжении всего проекта.

В проектах, в которых используется **итеративный процесс разработки**, работа с требованиями ведется в каждой итерации процесса разработки, причем на первую итерацию приходится больший объем работы.

В **проектах гибкой разработки (agile)** планируется выпускать функциональность каждые несколько недель. В таких проектах работа над требованиями выполняется часто, но небольшими объемам. Работа начинается с выявления пользовательских требований в форме пользовательских историй. Приоритизация пользовательских требований позволяет определить, какие из них назначить на те или иные этапы разработки, которые называются итерациями или спринтами. Более подробно выделенные на ту или иную итерацию требования изучаются, когда разработка подойдет к этой итерации.

Выявление требований

Источники требований

- ✓ Заинтересованные стороны
- ✓ Существующие системы
- ✓ Существующие документы
- ✓ Конкуренты и др. подобные системы
- ✓ Интерфейсы с системами
- ✓ Законы и стандарты
- ✓ Политика компании

Кто такие заинтересованные стороны?

- **"Клиент"** – люди, которые платят за разработку системы.
- **Пользователи** - их потребности — это то, чему организация должна уделять первостепенное внимание.
- **Эксперты домена** – Это специалисты, которые знают среду, в которой будет использоваться продукт.
- **Инспекторы** – Они являются экспертами в области государственных норм и правил и безопасности, требуемой проектом.
- **Адвокаты** – Они являются экспертами, когда речь идет о законах, а также о стандартах, которые следует учитывать при разработке продукта.

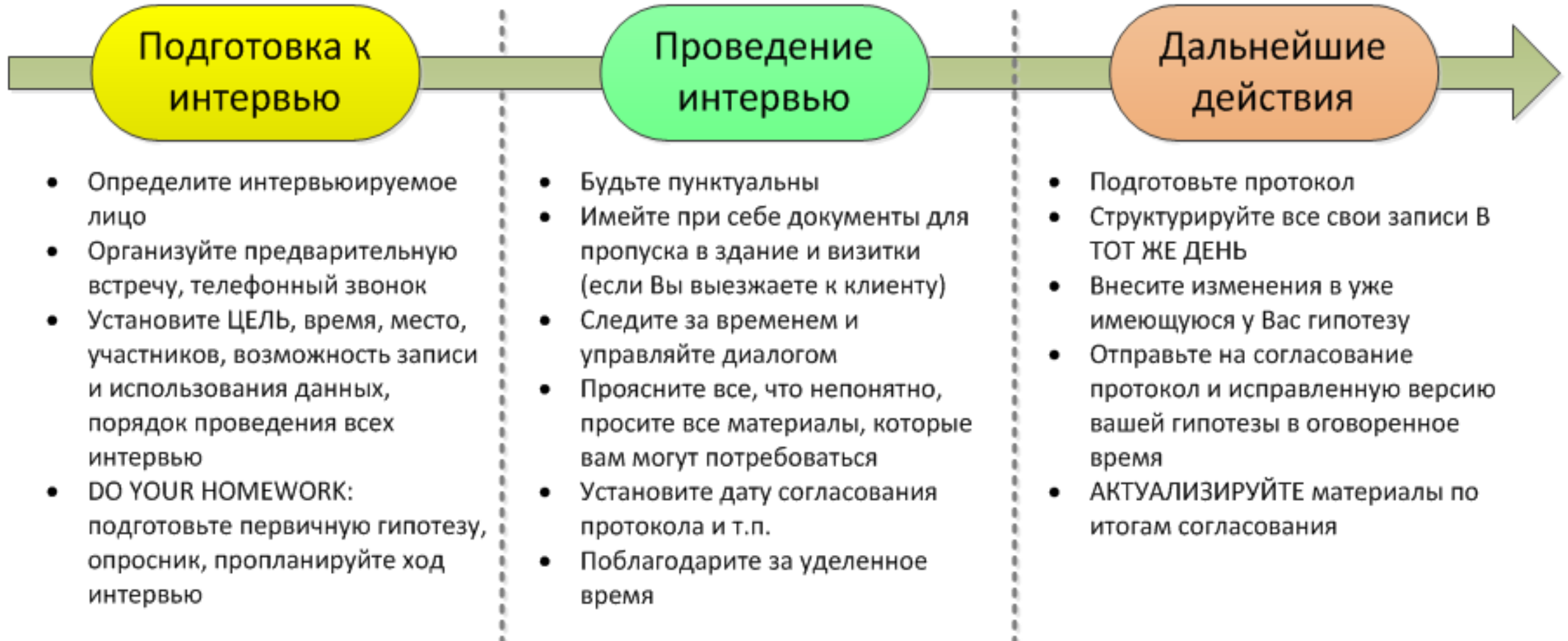
...

Методы выявления требований

- ✓ Интервью, опросы, анкетирование
- ✓ Мозговой штурм, семинар
- ✓ Наблюдение за производств. деятельностью, «фотографирование» рабочего дня
- ✓ Анализ нормативной документации
- ✓ Анализ моделей деятельности
- ✓ Анализ конкурентных продуктов
- ✓ Анализ статистики использования предыдущих версий продукта

Заинтересованное лицо (**stakeholder**) — это человек, группа или организация, которая активно задействована в проекте, подвержена влиянию процесса или результата или может влиять на процесс или результат.

Примерный чек-лист для специалиста по сбору требований



? Как думаете, что лучше выбрать: интервью или анкетирование ?

Сотрудничество клиентов и разработчиков

Отличные программные продукты — результат правильно выполненного проектирования, основанного на требованиях, полученных в результате тесного взаимодействия разработчиков и клиентов. Совместная работа возможна только тогда, когда все участники процесса разработки знают, что именно необходимо им для успеха, и когда они понимают и уважают стремление их соратников к успеху.

«Билль о правах клиента ПО» содержит 10 положений, на выполнении которых клиенты могут на вполне законных основаниях настаивать при общении с аналитиками и разработчиками на этапе формулирования требований к проекту.

Билль о правах клиента ПО при формировании требований

У вас есть право

1. Иметь дело с аналитиком, который разговаривает на вашем языке
2. Иметь дело с аналитиком, хорошо изучившим ваш бизнес и цели, для которых создается система
3. Потребовать, чтобы аналитик зафиксировал требования в надлежащей форме
4. Получить подробный отчет о будущих процедурах и результатах процесса формулирования требований
5. На изменение ваших требований
6. На взаимное уважение
7. Знать о вариантах и альтернативах требований и их реализации
8. Описать характеристики, упрощающие работу с продуктом
9. Узнать о способах корректировки требований для ускорения разработки за счет повторного использования
10. Получить систему, функциональность и качество которой соответствует вашим ожиданиям.

Так как обратной стороной прав являются обязанности, «Билль об обязанностях клиента ПО», напротив, содержит 10 положений, определяющих ответственность клиента перед аналитиком и разработчиком на этапе формулирования требований. Возможно, его стоит назвать «Билль о правах разработчика».

Билль об обязанностях клиента ПО при формировании требований

Клиент обязан

1. Ознакомить аналитиков и разработчиков с особенностями вашего бизнеса
2. Потратить столько времени, сколько необходимо на уточнение требований
3. Точно и конкретно описать требования к системе
4. Принимать своевременные решения относительно требований
5. Уважать определенную разработчиком оценку стоимости и возможности реализации ваших требований
6. Определять реалистичные приоритеты требований совместно с разработчиками
7. Проверять требования и оценивать прототипы
8. Определить критерии приемки
9. Своевременно сообщать об изменениях требований
10. Уважительно относиться к процессам создания требований

Use Case (вариант использования, прецедент использования)

- Use Case описывает сценарий взаимодействия участников (как правило — пользователя и системы).
- Участников может быть 2 и больше.
- Пользователем может выступать как человек, так и другая система.

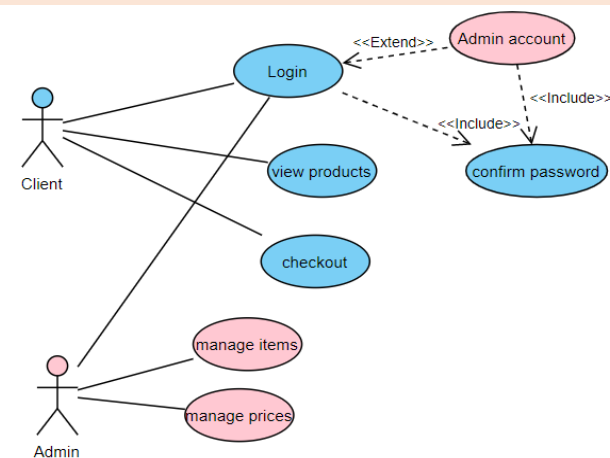
Текстовое описание

Пример 1. Разблокировать учетную запись пользователя (простой короткий пример, без альтернативного потока событий):

Действующие лица	Администратор, Система
Цель	Изменить статус учетной записи пользователя на «активный».
Предусловие	Учетная запись пользователя не активна.
Успешный сценарий:	<ol style="list-style-type: none">Администратор выбирает пользователя и активирует «Разблокировать».Система переключает учетную запись пользователя в статус «активный», и посылает сообщение (тут можно сослаться на текст сообщения из списка сообщений, см. примечание ниже) пользователю на email (если «User Account → email» не пусто).
Результат	Учетная запись пользователя была переведена в статус «активный».

или

UML диаграмма вариантов использования



Полезная статья о том КАК !!!
<https://habr.com/ru/post/566218/>

Словесное описание Use Case

Пример 1. Разблокировать учетную запись пользователя (простой пример):

Действующие лица	Администратор, Система
Цель	Изменить статус учетной записи пользователя на «активный».
Предусловие	Учетная запись пользователя не активна.

Успешный сценарий:

1. Администратор выбирает пользователя и активирует «Разблокировать».
2. Система переключает учетную запись пользователя в статус «активный», и посылает сообщение пользователю на email (если «User Account → email» не пусто).

Результат	Учетная запись пользователя была переведена в статус «активный».
------------------	--

Словесное описание Use Case

Пример 2. Авторизация пользователя:

Действующие лица Пользователь, Система

Цели Пользователь: авторизоваться в системе и начать работать; Система: идентифицировать пользователя и его права.

Успешный сценарий:

1. Пользователь запускает систему. Система открывает сессию пользователя, предлагает ввести логин и пароль.

2. Пользователь вводит логин и пароль.

3. Система проверяет логин и пароль.

4. Система создает запись в истории авторизаций (IP адрес пользователя, логин, дата, рабочая станция).

5. Система выдает пользователю сообщение по поводу успешной авторизации (ссылка на сообщение).

Результат Пользователь успешно авторизирован и может работать с системой.

Расширения:

*a	Нет доступа к БД. Система выдает сообщение (ссылка на сообщение). Результат: пользователь не может войти.
1a	В настройках безопасности для данного IP адреса существует запрет на вход в систему. Результат: форма логина не предоставляется, система выдает сообщение пользователю (ссылка на сообщение).
2a	Пользователь выбирает: «Напомнить пароль». Вызывается сценарий «Напомнить пароль».
3a	Пользователь с введенными логином и паролем не найден. Результат: отказ в авторизации. Система выдает сообщение (ссылка на сообщение). Переход на шаг 2.
3б	Количество неудачных попыток авторизоваться достигло максимального, установленного в настройках. Результат: пользователь не может войти. Выдается сообщение: (ссылка на сообщение). Вход с IP адреса Пользователя заблокирован на время, установленное в настройках.

До появления пользовательских историй использовался формат сценариев использования (Use Cases). К сожалению такие сценарии, лишены эмпатии к человеку, для которого создаётся программа. В сценариях использования его обычно называли презрительным и обезличенным «юзером».

Чтобы исправить эту ситуацию апологеты гуманности в парадигме **User Centered Design** сместили акцент на роль человеческого фактора. Появился метод персон, помогающий создать модели групп будущих пользователей и хоть как-то передать команде разработки понимание целей, задач и чаяний людей, для которых создаётся продукт.

Метод персон — инструмент, который позволяет исследовать целевую аудиторию, сегментировать ее и составить обобщенные портреты клиента — персоны. Помогает персонализировать маркетинговые предложения, выстроить коммуникацию, разработать интерфейс и пользовательский сценарий в продукте.

Это целая наука :о, если интересно смотри здесь

<https://www.unisender.com/ru/glossary/cto-takoe-metod-person/>



Пользовательские истории (User Story)

Пользовательская история - простое описание функциональности на языке пользователя, который говорит какую ценность или выгоду получит этот пользователь.

Бизнесу это тоже выгодно потому что можно создать эту ценность и продать
Пользовательские истории лучше Use Case-ов потому что показывают ценность.

Как **<роль или тип пользователя>**,
я хочу/могу **<выполнить действие или получить результат>**,
чтобы **<получить ценность>**

Как "пациент стоматолога",
я хочу "смотреть фильм в VR-очках во время сеанса лечения",
чтобы "прием прошел приятно и время пролетело незаметно"

Для чего применяется User Story?

- Для описания элементов бэклога
- Для лучшего понимания пользователей
- Для описания требований к продукту на понятном для всех языке
- Для вовлечения в процесс разработки пользователей и заинтересованных лиц
- Для построения User Story Mapping, которые помогают помогают разбить проект на релизы



User Story предложил **Кент Бек**: отец экстремального программирования, паттернов проектирования, JUnit и TDD
Интересная статья о нем на хабре по ссылке
<https://habr.com/ru/company/ju-gru/blog/580976/>

INVEST критерии

Без критериев приемки историю пользователя можно интерпретировать как угодно, особенно если нет близкого контакта с пользователем на протяжении всего проекта

Стандартная формулировка: *Я, как <роль>, хочу <функция>, чтобы <ценность>* не содержит в себе всех критериев, важных для качественной разработки, поэтому в идеальном мире истории записывают, учитывая INVEST критерии.

INVEST - критерий правильности User story.

I.N.V.E.S.T. – это аббревиатура, объединяющая шесть характеристик, **которыми должен обладать Элемент Бэклога Продукта, чтобы соответствовать Критериям Готовности к Разработке.**

I – Independent (независимая от других историй, то есть истории могут быть реализованы в любом порядке)

N – Negotiable (обсуждаемая, отражает суть, а не детали; не содержит конкретных шагов реализации)

V – Valuable (ценная для клиентов, бизнеса и стейкхолдеров)

E – Estimable (оцениваемая по сложности и трудозатратам)

S – Small (компактная, может быть сделана командой за одну итерацию/спринт)

T – Testable (тестируемая, имеет критерии приемки).

Карты пользовательских историй (User Story Mapping)

- ✓ помогают Agile-командам определять приоритетные элементы для разработки;
- ✓ визуально отображают, как отдельные составляющие продукта сочетаются друг с другом;
- ✓ направляют итеративный процесс разработки продукта;
- ✓ помогают учитывать интересы пользователей при обсуждении идей;
- ✓ расставлять характеристики продукта в приоритетном порядке.

Для чего применяется USM?

- для проектирования пользовательского опыта в продукте
- для определения границ MVP (минимальной работоспособной версии продукта) и планирования релизов на базе пользовательского сценария
- для формирования единого понимания пользователя у команды разработки и заинтересованных лиц

Как построить User Story Mapping?

вам потребуется:

- Инструмент визуализации: стикеры или электронный инструмент
- Владелец/пользователи продукта и команда разработки
- Представление о пользовательском сценарии (можно построенный CJM)
- 1-2 часа



Карта пользовательской истории отображает 3 типа действий разной степени детализации: **действия** (в общем смысле), **шаги и детали** (конкретные действия). Действия и шаги пользователя отображаются горизонтально в верхней части карты, а детали — вертикально, под соответствующими шагами в приоритетном порядке.

1. **Действия** представляют собой наиболее общие задачи, которые пользователь стремится выполнить в цифровом продукте, например, “Проверить баланс счета” или “Депонировать чек”. Количество таких действий будет различаться в зависимости от типа приложения или веб-сайта, который вы разрабатываете. При условии существования нескольких путей для различных типов пользователей они могут отражаться последовательно или параллельно. Поисковые исследования главных задач пользователей должны являться основой для создания этого уровня карты.
2. **Шаги** — это конкретные подзадачи, выполняемые пользователями для завершения действия, указанного выше. Они являются следующим уровнем после действий и также отображаются последовательно. Например, для действия “Депонировать чек” можно выделить следующие шаги: “Ввести данные мобильного депозита”, “Подписать чек”, “Сфотографировать чек”, “Внести депозит” и “Подтвердить депозит”.
3. **Детали** представляют собой третий уровень карты истории и описывают конкретные действия пользователей с наибольшей степенью детализации. Например, шаг “Войти в аккаунт” включает две отдельные детали: “Ввести имя пользователя” и “Ввести пароль”.

1. Действия

Общие задачи пользователей в цифровом продукте

Проверить баланс счета

Депонировать чек

Декомпозиция историй выполняется до тех пор, пока каждая из историй не будет отвечать критериям **INVEST** (независимость, обсуждаемость, ценность, возможность оценки сроков реализации, компактность, тестируемость).

2. Шаги

Шаги, которые предпринимают пользователи для выполнения действия выше.

Войти в аккаунт

Зайти в раздел "Счета"

Ввести данные мобильного депозита

Подписать чек

Сфотографировать чек

Внести депозит

Подтвердить депозит

Ввести имя пользователя или email

Проверить баланс счетов

Выбрать счет

Прочитать советы о том, как сфотографировать чек

Разрешить доступ к камере

Подтвердить депозит

Увидеть сообщение о подтверждении

3. Детали

Конкретные взаимодействия, которые необходимы для выполнения шагов.

Ввести пароль

Проверить незавершенные операции

Ввести сумму депозита

Повернуть телефон горизонтально

Понять, какая сумма доступна

Получить email о подтверждении

Нажать кнопку "Войти"

Открыть новый счет

Проверить лимиты транзакции

Сфотографировать обе стороны чека

Отменить внесение депозита

Нажать "Забыл пароль"

Прочитать положения закона

Отправить чек в банк с помощью дрона

Включить авто-заполнение номеров

Сразу получить доступ ко всем деньгам

Найти депозит в разделе "Последние депозиты"

Нажать "Запомнить меня"

Получить консультацию о том, куда вложить сбережения

Просмотреть последние депозиты

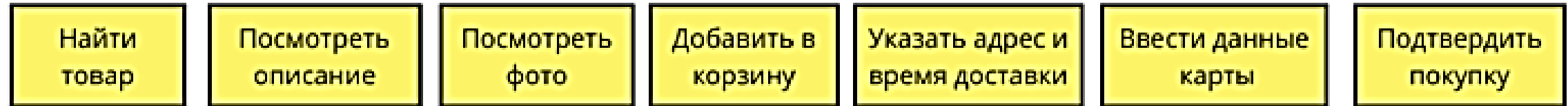
Просмотреть сообщения об ошибках

Получить текстовое сообщение

Рассмотрим USM на примере

Магазин цветов решил запустить сайт. Визуализируем опыт клиентов с помощью техники USM.

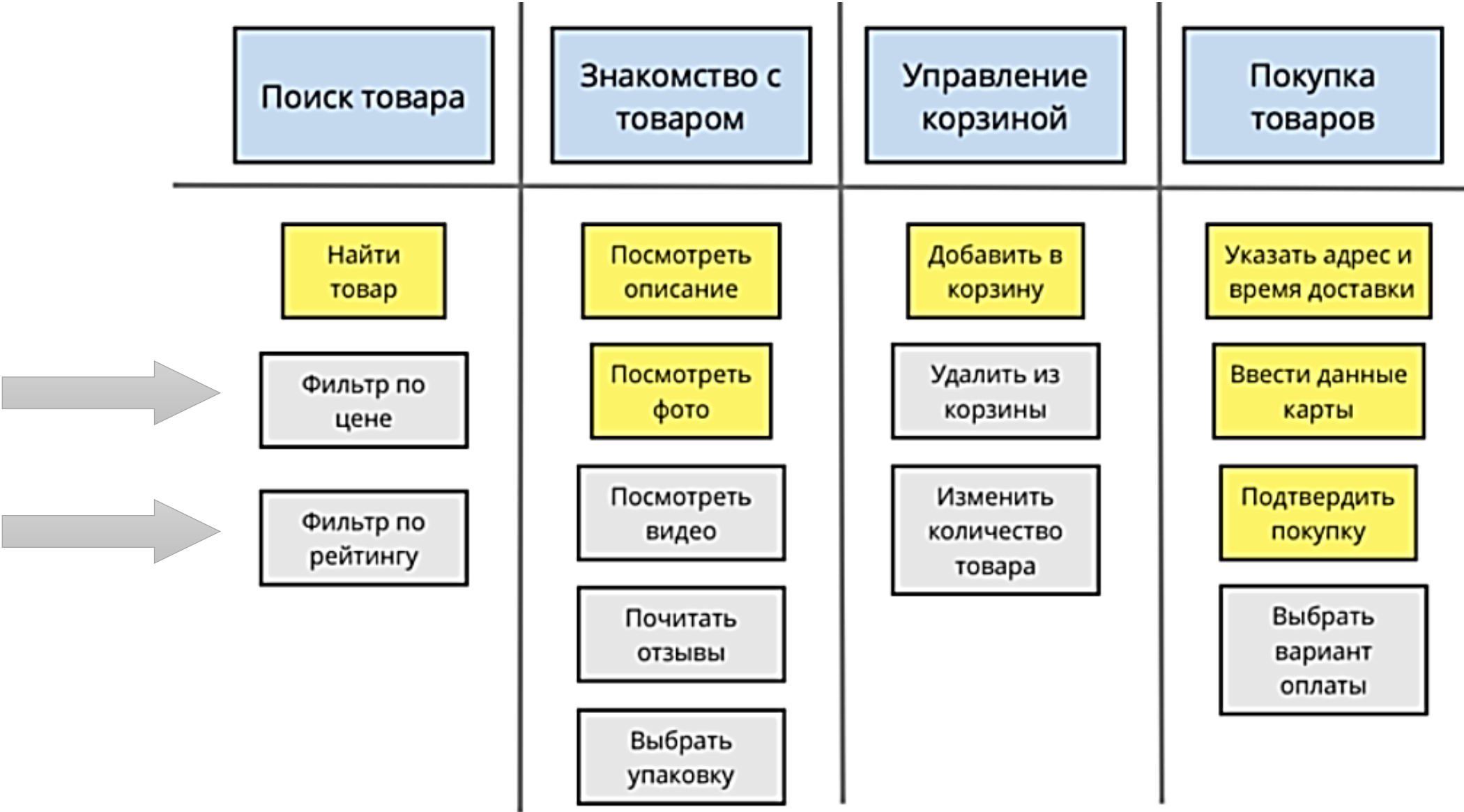
1. Расскажите историю клиента по шагам



2. Сгруппируйте действия клиента в этапы



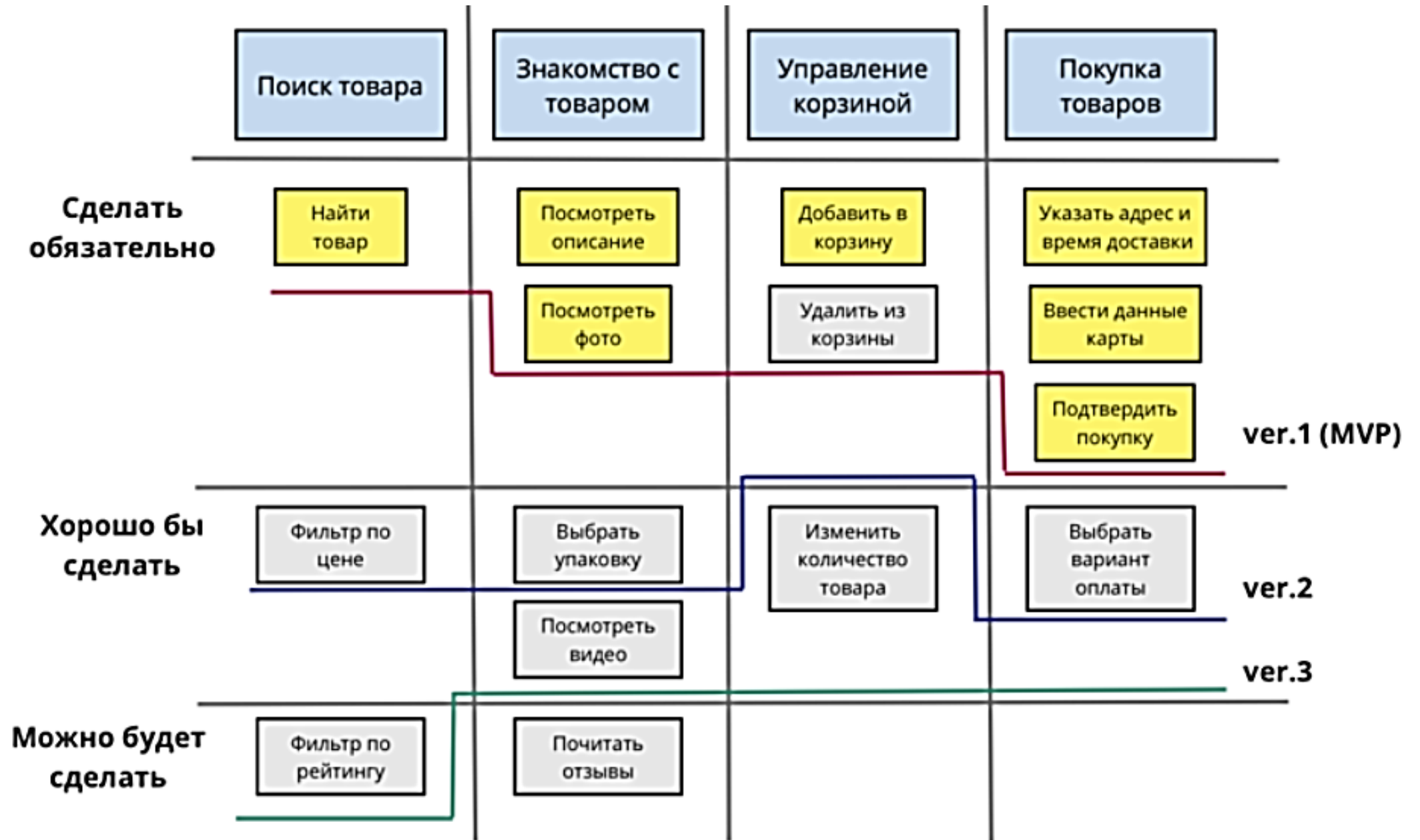
3. Заполнение пробелов в User Story



4. Приоритезируйте истории внутри каждого этапа пути

	Поиск товара	Знакомство с товаром	Управление корзиной	Покупка товаров
Сделать обязательно	Найти товар	Посмотреть описание Посмотреть фото	Добавить в корзину Удалить из корзины	Указать адрес и время доставки Ввести данные карты Подтвердить покупку
Хорошо бы сделать	Фильтр по цене	Выбрать упаковку Посмотреть видео	Изменить количество товара	Выбрать вариант оплаты
Можно будет сделать	Фильтр по рейтингу	Почитать отзывы		

5. Выделите релизы

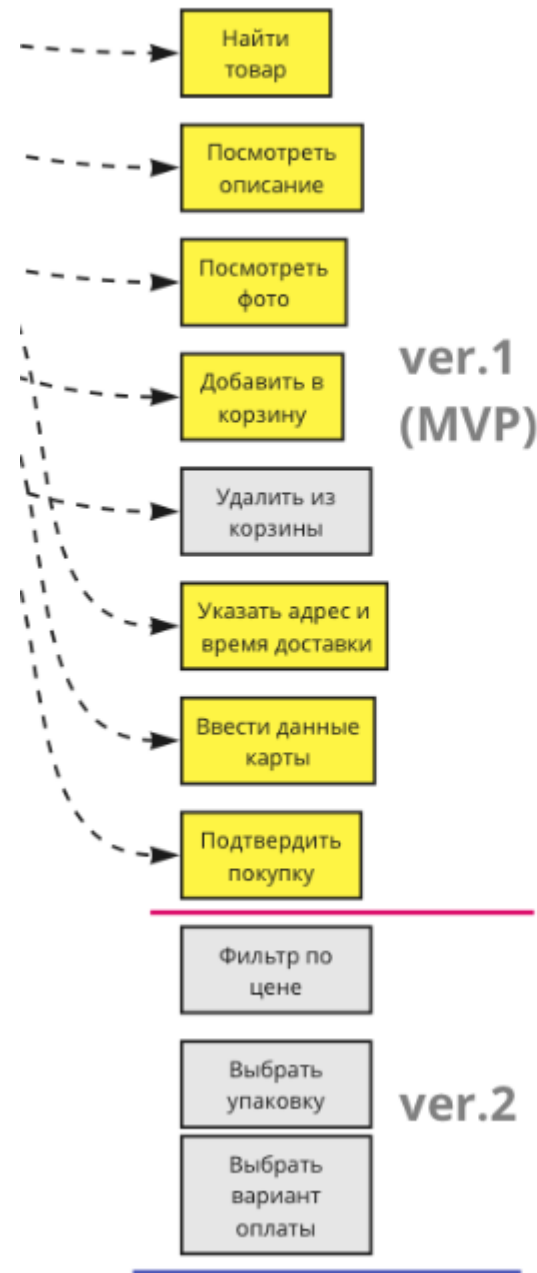


6. Итог- приоритизированный бэклог



User Story Mapping – мощный инструмент, позволяющий команде разработки за пару часов взглянуть на бэклог продукта глазами пользователя.

Бэклог продукта



Во втором примере в качестве иллюстрации для составления USM будем использовать небольшую браузерную игру с программой лояльности какого-нибудь банка. В ней есть регистрация, игровой персонаж, его убежище, сражения и баллы лояльности.

<https://sherer.pro/blog/user-story-mapping-i-funkcionalnaya-arxitektura/>

Шаг 1. Выявляем активности

Сначала нужно собрать список основных **активностей** пользователей. В нашем примере игры их пять:

- Регистрация и логин;
- Работа над аватаром;
- Работа над убежищем;
- Участие в сражении;
- Работа с баллами лояльности.

АКТИВНОСТИ

Регистрация
и логин

Работа над
аватаром

Работа над
убежищем

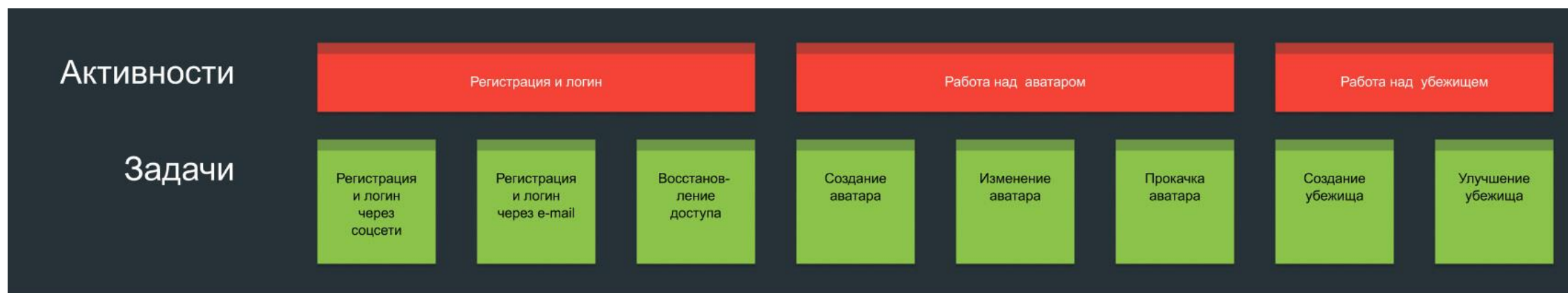
Участие в
сражении

Работа с
баллами
лояльности

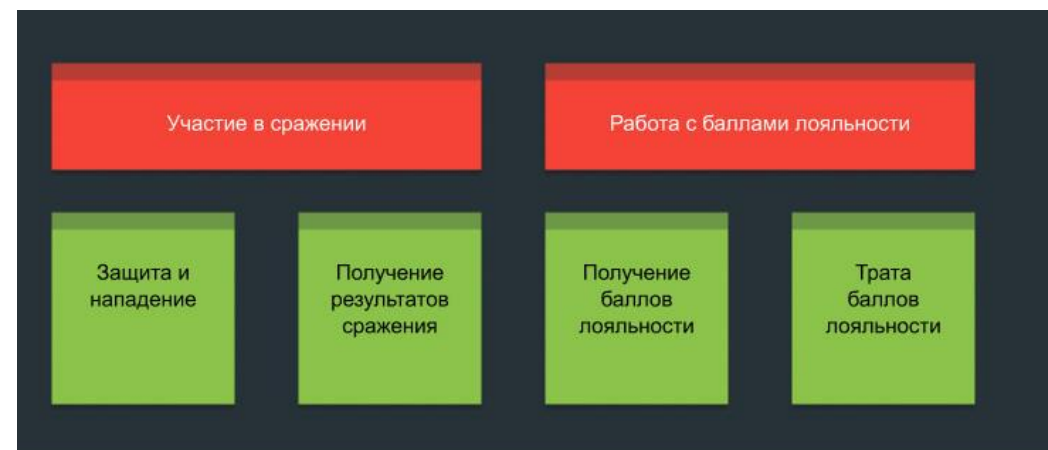
Шаг 2. Разворачиваем активности в задачи

Раскладываем активности на **задачи**. Например, *активность* «Регистрация и логин» превращается в три *задачи*:

- Регистрация и логин через соцсети;
- Регистрация и логин через e-mail;
- Восстановление доступа.



Здесь нет отдельно «логина» или «регистрации». Это более глубокий уровень, задачи же — это такие «категории» пользовательских историй.



Шаг 3. Задачи детализируем до конкретных историй

В нашем примера, *задача* «Регистрация и логин через e-mail» разложилась на 2 *истории*:

- Регистрация с помощью e-mail;
- Логин через e-mail;

Истории (жёлтые карточки) — они расположены в порядке приоритета — сверху критичные, внизу наименее важные. Истории — это *функции* будущей системы. У них есть взаимосвязи и порядок.



Шаг 4. Делаем из приоритетов схему релизов

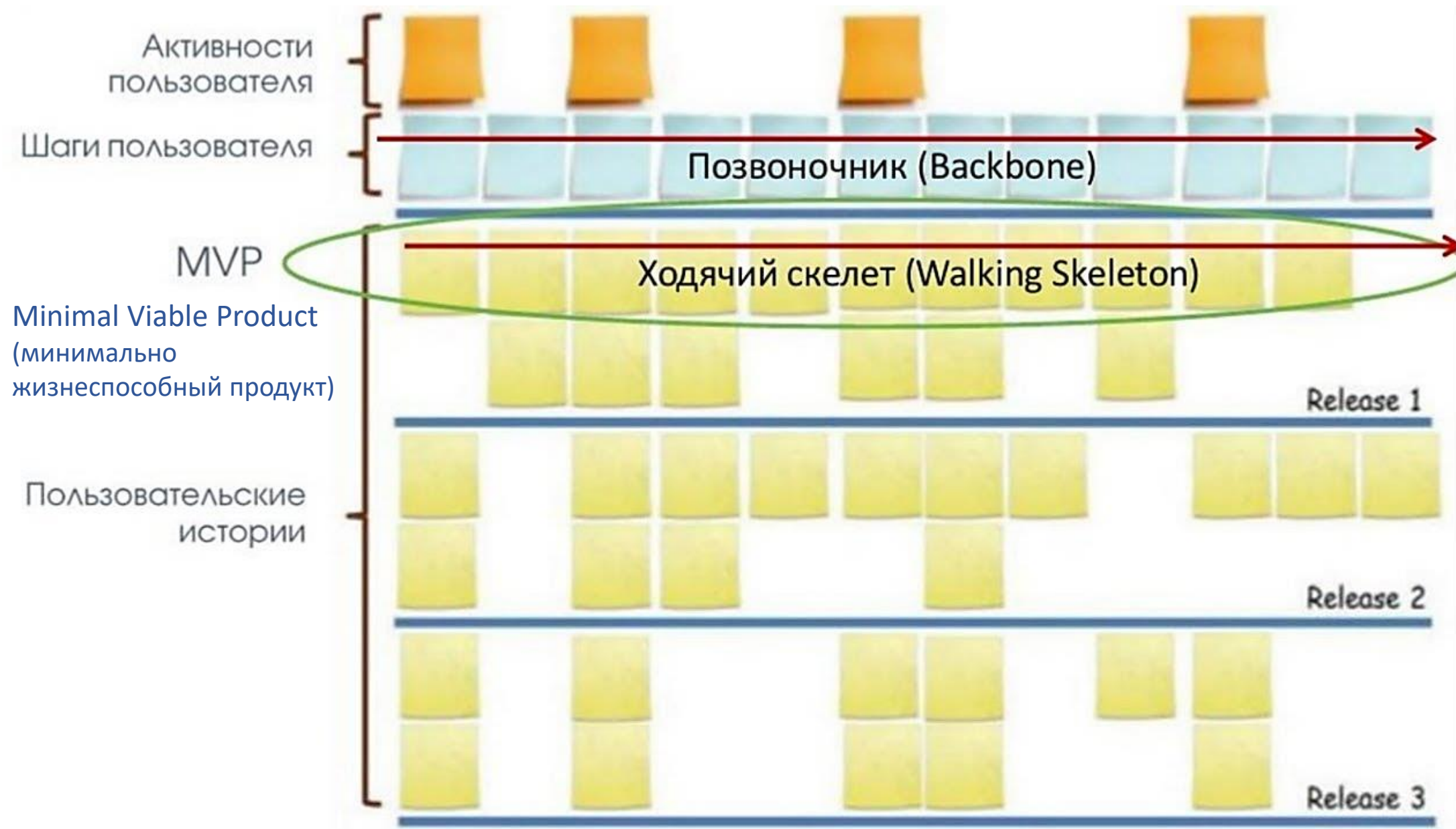
Переносим те истории, которые не критичны или требуют на этом этапе слишком много ресурсов, в следующий релиз. Например, нам проще сделать логин и регистрацию через соцсети — и регистрацию через e-mail мы отодвигаем на второй релиз.



Дополнительно: цветовая индикация и структура

Например, цветом стикеров можно пометать отдельные категории событий/действий, дополнительную важность (например, для бизнеса), типы пользователей и так далее.





Плюсы USM:

Улучшение коммуникаций внутри команды. У каждого из нас уникальное видение чего-либо. В обычной жизни это хорошо, но для людей, работающих над одним продуктом, картина должна быть общей. User Story Mapping — простой способ показать команде, над чем она работает и к каким результатам должна прийти совместными усилиями.

Фокус на пользователях. В глазах потребителей ценный продукт = закрытые потребности. Если это игнорировать и работать только над функционалом, то ценность продукта будет сведена к нулю. Техника User Story Mapping фокусирует внимание команды на потребностях конечных пользователей, а не только на функционале.

Максимальная визуализация. Карты пользовательских историй раскрывают User Flow — визуальное представление того, какие действия потребитель совершает, чтобы достичь цели. Это позволяет разработчикам не упустить важных шагов, а пользователям — получить продукт, который закроет их потребности.

Минусы у User Story Mapping тоже есть. Как минимум — сложность организации. Если продукт объемный, составить бэклог за стандартные 2–3 часа невозможно. Проект придется разбить на несколько отдельных задач и выстраивать USM по каждой в отдельности.

Что такое Customer Journey Map

Бывает, что компания не знает, откуда к ней приходят клиенты и как они принимают решение купить. Если на пути к покупке у клиента возникают сложности, о которых компания не догадывается, тот пытается разобраться во всём самостоятельно. Если у него не получается, он не покупает, а бизнес не зарабатывает.

Чтобы предугадать, где клиенту трудно принять решение о покупке, создают визуальную схему — карту клиента или Customer Journey Map. На ней подробно расписывают, где клиенту нужно помочь, а где он справится сам.



















Карта путешествий учитывает, какие у клиента:

- ✓ цели купить товара;
- ✓ мотивы покупки;
- ✓ эмоции во время покупки;
- ✓ предыдущий покупательский опыт.

Благодаря CJM компания воспринимает цикл продаж не как отчёты, графики и KPI, а наглядный сценарий, которым можно управлять.

Оформляют карту в виде инфографики, таблицы или блок-схемы — кому как удобно.

Карта путешествия клиента интернет-магазина

Стадии путешествия	Мотивация	Поиск в интернете			Поиск на сайте	Оценка товаров		Оплата	
Активности	Хочет купить подарок другу на новый год	Ищет товары в интернете по ключевым словам	Переходит по первому объявлению в результатах поиска	Еще раз сравнивает с другими сайтами в поисковой выдаче	Изучает текущие скидки и предложения	Открывает страницу товара, чтобы изучить детали	Открывает страницы других товаров, чтобы сравнивать информацию	Переходит к оплате выбранного товара	Связывается со службой поддержки для помощи
Чувства									
Счастлив									
В целом доволен									
Несчастлив									
Опыт	Приятно сделать подарок другу	Нравится большой выбор, но не знает как выбрать	Не нравится много бесполезной информации	Доволен обилием информации. Не уверена как найти лучшую цену	Удивлен обилием скидок	Разочарован отзывами других покупателей	Нравится дизайн и функциональность. Расстраивает, что многих товаров нет в наличии	Расстроен, что есть только один способ оплаты	Расстроен, что есть только один способ оплаты
Ожидания	Получать легкий доступ к информации о скидках	Релевантные результаты поисковой выдачи	Возможность скрывать ненужные объявления	Понятный и современный дизайн сайта	Больше праздничных скидок	Высокое качество товаров	Большой ассортимент и быстрая доставка	Больше способов оплаты	Высокая скорость работы сайта

ТОЧКИ КОНТАКТА



Составляя CJM, важно обозначить точки касания, где и как клиент взаимодействует с компанией и продуктом. Это поможет наметить стратегию для коммуникации с ним

Чем отличается карта пользовательских историй от карты пути клиента

User Story Mapping и **Customer Journey Map** — разные инструменты.

USM дает понимание, что нужно минимально сделать для запуска продукта, над какими функциями работать, что выполнить сейчас, а что — позднее. То есть расставить приоритеты.

CJM используют для поиска ниши на рынке, анализа пользовательского опыта.

Еще одно отличие — в сроках. Карта пути клиента применяется еще на этапе планирования, когда определяется ниша, а карта пользовательской истории — позднее, когда команда уже приступила к разработке самого продукта.

Если к моменту создания User Story Mapping у вас будет готовый Customer Journey Map (CJM), то берите его за основу — это добавит определенности

Наиболее распространенные методы приоритизации требований:

- ✓ **RICE** (Reach — охват, Impact — влияние, Confidence — достоверность, Effort — усилия)
- ✓ **ICE** (Impact — влияние, Confidence — уверенность, Ease — легкость реализации)
- ✓ **Value vs Effort** («ценность против усилия»)
Модель Кано
- ✓ **Story mapping** («карта историй»)
- ✓ **MoSCoW**
- ✓ **Opportunity scoring** («оценка возможностей»)
- ✓ **Product tree** («дерево продукта»)
- ✓ **Cost of delay** («стоимость задержки»)
- ✓ **Buy a feature** («купи функцию»)
- ✓ **Weighted scoring** («взвешенная оценка»)
- ✓ **WSJF** (Weighted Shortest Job First, «сначала — более важная и короткая работа»)

Подробнее о методах читаем здесь:
<https://vc.ru/marketing/274778-12-metodov-prioritizacii-produktovyh-celey-rice-wsjf-kano-i-prochie>

Техника приоритизации Agile Backlog: MoSCoW

MoSCoW — это метод расстановки приоритетов, обычно используемый в Scrum. Определяя свои приоритеты, постарайтесь сосредоточиться на том, что важнее, а не на деятельности с меньшей ценностью.

Mo

MUST HAVE

The most vital things you can't live without

S

SHOULD HAVE

Things you consider as important, but not vital

Co

COULD HAVE

Things that are nice to have

W

WON'T HAVE

Things that provide little to no value you can give up on

Все задачи или требования делятся на 4 категории: must, should, could, would.

Must — то, что необходимо сделать в любом случае. Без выполнения этих задач продукт не будет работать в принципе.

Should — не самые важные требования, но они тоже должны быть выполнены. Естественно, после реализации «must».

Could — желательные требования, которые можно сделать, если останется время и будут ресурсы.

Would — требования, которые хотелось бы сделать, но их можно проигнорировать или перенести на следующие релизы без вреда для продукта.