Традиционно на старте проекта создают технические задания, схемы работы системы и, в хорошем варианте, прототипы интерфейса. В этих документах может не хватать понимания динамики развития проекта и приоритетов в работе.

[User Story](http://en.wikipedia.org/wiki/User_story) и [Story Mapping](http://en.wikipedia.org/wiki/User_story#Story_map) добавляют понимание того, как проект будет развиваться, какие сейчас приоритеты, дают возможность продуктивнее общаться с заказчиком. Продукты существуют не в вакууме, нужно видеть более глобальные задачи, которые лежат где-то выше историй использования системы. Игровая практика по постановке целей проекта, из которых потом будут появляться Story Mapping и список User Story.

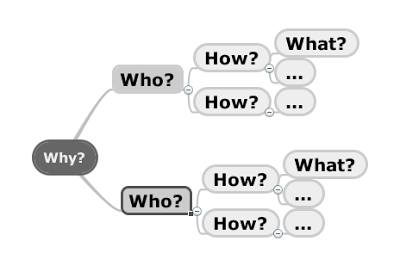
Техника призвана помогать командам [фокусироваться на бизнес-целях, выявлять заинтересованные стороны и их потребности](http://specificationbyexample.com/key_ideas.html).

Effect mapping содержит несколько усовершенствований, таких как: приоритизация целей и воздействий, возможность уходить от технических деталей на уровне What, цикличность в предположениях и экспериментах и т.п. такая техника называется— Impact Mapping.

Необходимо узнать у заказчика реальные цели бизнеса, которые мы хотим достичь. Как сделать так, чтобы команда услышала их, приняла и начала с ними работать?

**Составляем Impact Mapping**

Impact Mapping — это [mind map](http://en.wikipedia.org/wiki/Mind_map) по целям проекта с картой влияний, которые должны подтолкнуть бизнес заказчика к достижению целей.



|  |
| --- |
| Центральный элемент нашей карты, который отвечает на ключевой вопрос: *Зачем мы это делаем?* Это цель, которую бизнес пытается достичь. |
| **Who?** | На первом уровне мы отвечаем на вопросы: *Кто может поможет достичь желаемого результата? Кто может помешать? Кто пользователи нашего продукта?* Сюда войдут все заинтересованные стороны, которые могут повлиять на цели бизнеса. |
| **How?** | На втором уровне мы должны описать воздействия, которые должны оказать заинтересованные стороны, чтобы бизнес достиг целей. Мы ищем ответ на вопросы: *Как они помогут бизнесу достичь целей? Как они могут помешать успеху проекта?* |
| **What?** | После ответа на основные вопросы можно обсудить конкретные задачи. Третий уровень отвечает на вопросы: *Что мы можем сделать как организация или команда разработки, чтобы создать необходимые воздействия?* Здесь будет описан конечный результат нашей работы. |

### Организация процесса

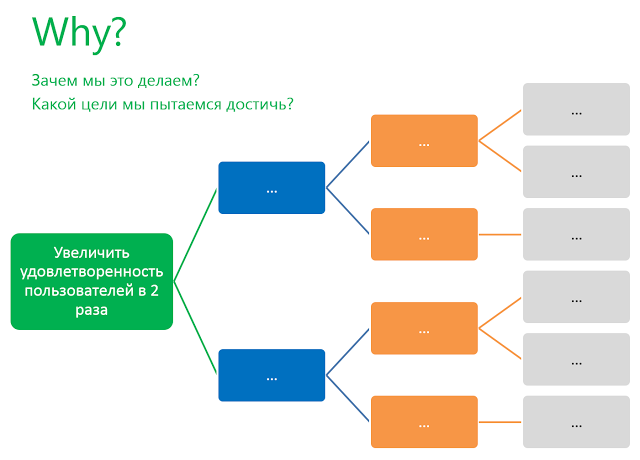
1. Приглашайте не больше 10-15 человек на это мероприятие, иначе будет сложно проводить. Оптимально позвать 3-4 человека со стороны заказчика и столько же со стороны команды.
2. Со стороны заказчика обязательно взять представителей бизнеса, а не только технических специалистов, у которых уже сложилось мнение по поводу конкретных реализаций всех целей.
3. Карту будете строить на доске или стене, подготовьте их заранее. Impact Mapping для задачи длительностью в 6-8 месяцев умещается на доске стандартного размера.  
   Онлайн вариант этой техники я ещё не пробовал, но думаю подойдет любой интерактивный инструмент. Коммуникация будет не та, но вас может спасти, если вы лично знаете заказчика или являетесь отличным фасилитатором/модератором.
4. Составление карты займёт от одного часа до двух дней. Эта цифра сильно зависит от состава участников и ваших навыков проведения.
5. Каждый блок карты можно рисовать маркером или делать стикерами. Я предпочитаю стикеры, потому что они более мобильны, а impact map будет часто сортироваться и меняться по ходу погружения в проект.
6. Перед началом обязательно проговорите правила и цели составления карты. Если есть время, то разошлите всем материал по теме для подготовки
7. Если есть возможность и обстоятельства позволяют, то сделайте несколько [Icebreaker'ов.](http://en.wikipedia.org/wiki/Icebreaker_%28facilitation%29)
8. И самое главное — сам процесс должен проходить легко и весело. Не добавляйте в него бюрократии!

## Пример из практики

Разберём пример, очень приближенный реальному проекту, для которого в начале мы сделали Impact Mapping. Остановимся на ключевых моментах при составлении Impact Mapping и на ошибках, которые могу погубить всю идею.

### Why?

Корневым элементом нашей карты будет список бизнес-целей. Например, это может быть *увеличение удовлетворенности пользователей в 2 раза*. Важно, что *удовлетворенность пользователей* — это индекс, т.е. конкретная цифра, которую можно взять из CRM, а не мнение/ощущение заказчика. Мы же хотим после поставки фич *измерить* достижение цели и понять, в том направлении мы идём или нет. Если бы *удовлетворенность пользователей* была не цифрой, то как бы мы узнали, что достигли цели? Ещё важно, что мы написали именно *в 2 раза*, а не просто *увеличение*. Хорошие цели должны быть [SMART](http://ru.wikipedia.org/wiki/SMART):

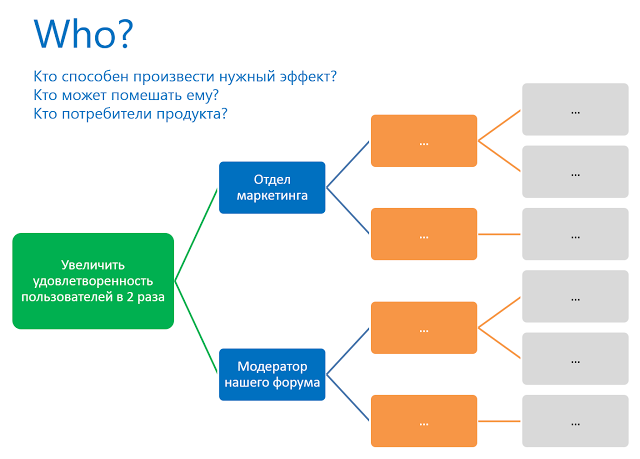
[](http://4.bp.blogspot.com/-M0CIs6VLHe8/VIgCnJTaxnI/AAAAAAAABnU/Dr82g9beVhY/s1600/why.png)

Первый этап довольно сложный, он должен дать импульс для составления остальной карты и создать доверие между участниками. Что я могу посоветовать, исходя из своей практики:

1. Не торопитесь предлагать решения на этом этапе. Выслушайте заказчика, *по-настоящему* выслушайте. В ходе дальнейшего обсуждения вы успеете всё откорректировать и причесать, а пока запишите то, что есть у него в голове.
2. Самая распространенная проблема заключается в навязывании решений (этап What?) до того, как цели стали понятны. Инженерная мысль летит со скорость света — заказчик только открыл рот, только начал говорить о своих целях, а мы уже создали в голове БД со всеми таблицами, придумали архитектуру и накидали куски кода. Зачем слушать дальше, если мы и так всё придумали? Будет ошибкой начать перебивать заказчика и предлагать решения. Запомните анекдот на эту тему и постарайтесь избежать проблемы: «Дима сказал "Привет", а Даша мысленно сыграла свадьбу и родила троих детей».
3. Не переубеждайте заказчизка на этом этапе. В самом начале вы не знаете его бизнес во всех тонкостях. Заказчики могут вам доверять, как профессионалам в IT, и из-за этого быстро соглашаться на ваши корректировки. Вы сами не заметите как на доске окажутся только те цели, которые вы навязали, а не те, с которыми заказчик жил всё это время.
4. Даже если цель трудноизмерима, то постарайтесь придумать критерий её достижения. Мысленно перенеситесь на финал проекта и подумайте, как вы узнаете достигнута цель или нет?
5. Процесс выработки целей итерационный, не обязательно выжамать из заказчика все цели на первом круге.
6. Не надо вытягивать искусственные цели. Бывают проекты, которые просто есть, потому что инвесторам хочется поиграть в создателей ПО. С этим нужно смириться и свернуть работу по составлению Impact Mapping.

### Who?

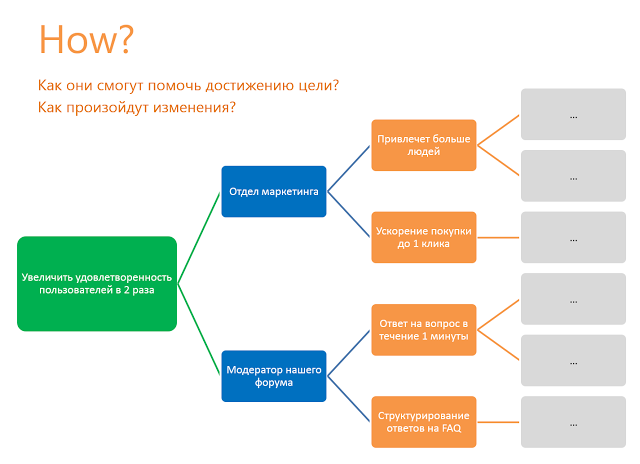
На этом этапе мы должны выявить всех, кто поможет оказать влияние на цель, кто поспособствует её достижению или помешает. В нашем примере это будут *Отдел маркетинга* и *Модератор форума*. По мнению заказчика именно они могут изменить удовлетворенность пользователей:

[](http://1.bp.blogspot.com/-gTssXiIXajA/VIgCrl6mUFI/AAAAAAAABnc/dyqAvRAQwiQ/s1600/who.png)

Здесь мы можем указывать конкретных людей, названия отделов, сегменты рынка и т.д. Выбирайте любой уровень абстракции, лишь бы он был адекватен вашему проекту.

### How?

Теперь нам надо определить те воздействия, которые будут сделаны для достижения цели. Например, модератор форума может попробовать давать ответы на вопросы в течение 1 минуты. Как вы думаете, повысит это удовлетворенность пользователей? У нас есть *предположение*, что повысит, поэтому записываем этот «impact». Тоже самое делаем для остальных ролей:

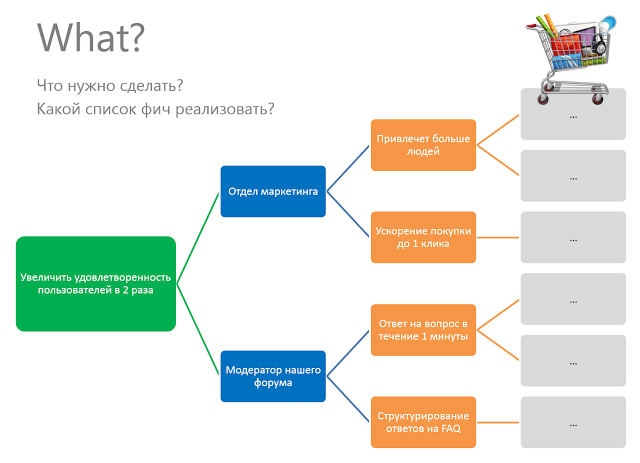
[](http://4.bp.blogspot.com/-_UyJqqqRatY/VJGKipO1EPI/AAAAAAAABoc/zh6r8v4HLiE/s1600/how.png)

Несколько рекомендаций:

1. Необязательно, но желательно, чтобы воздействия тоже были измеримимыми. Мы написали не просто *Отвечать на форуме*, а *Отвечать на форуме в течение 1 минуты*.
2. Не записывайте все возможные воздействия каждой роли. Нам нужны только те активности, которые приводят к достижению цели.

### What?

Мы дошли до самого несущественного в Impact Mapping. В последнем узле нашей карты находится та самая корзина с покупками, с которой обычно начинается работа над проектом. Разница в том, что теперь мы понимаем ценность каждой фичи, почему эта фича здесь и к чему приведет её реализация:

[](http://3.bp.blogspot.com/-K6Leghak_r4/VJGgpykWGsI/AAAAAAAABos/9IocsP7ln00/s1600/what.png)

Несколько замечаний и рекомендаций:

1. В конечных узлах карты можно написать User Story или названия модулей/подсистем.
2. Эту часть карты можно подбробно не расписывать, можно даже вообще не заполнять, а только проговорить её основные моменты. Полный список всех User Story вы успеете создать на Story Mappging'е.
3. Здесь не обязательно описывать IT-задачи. Вместо этого можно написать какие-то организационные преобразования и вообще любые действия для реализации воздействия на цель.
4. Понимание целей даёт нам возможность создавать более дешёвые и быстрые решения по достижению этих целей. За счёт карты мы начинаем использовать не только руки разработчиков, но и голову — каждый член команды может принимать обоснованные решения.

**Результаты создания Impact Mapping**

Вот и готов наш Impact Mapping. Осталось приоритизировать каждую колонку. Не все цели одинаково важны, тоже самое можно сказать про остальные узлы карты. Есть разные способы приоритизировать. Т.к. мы идем по пути простоты и визуализации, то я могу рекомендовать ставить звездочки. Каждому участнику даете по 5 звезд и он может ставить их куда хочет. Таким образом, можно выявить самые приоритетные узлы.

Результат работы нужно повесить у всех на виду. Если команда распределенная, то надо выложить Impact Mapping в общую базу знаний или повесить перед экраном, который видят все участники разработки. Главная цель — обеспечить видимость и достижимость этой информации, ведь мы опираемся на неё при работе над проектом.

Когда я рассказывал про Impact Mapping на [AgileClub](http://scrumtrek.ru/trainings/view/70/125/), коллеги заметили, что есть и другие способы понять стратегические цели. Например, можно использовать [Lean Canvas](http://en.wikipedia.org/wiki/Business_Model_Canvas) или собрать требования в проектной документации с описанием целей и заинтересованных сторон. На самом деле Impact Mapping *не противоречит* другим подходам и может использовать вместе с ними. Лично мне он больше нравится, потому что:

* Это простая техника, которая способствует общению и взаимодействию, в ней нет бюрократии.
* Заказчикам, которые не разбираются в IT и производстве ПО, такой подход очень просто объяснить, хватает пары минут.
* Визуализация в виде mind map

### Фильтр входящих задач

Даже когда все согласились с целями проекта и способами их достижения, заказчик может добавить в проект фичу, которая ему очень нравится — pet feature. Мы можем отфильтровать её через цели, показать что эта фича никаким образом не приведет нас к достижению целей.

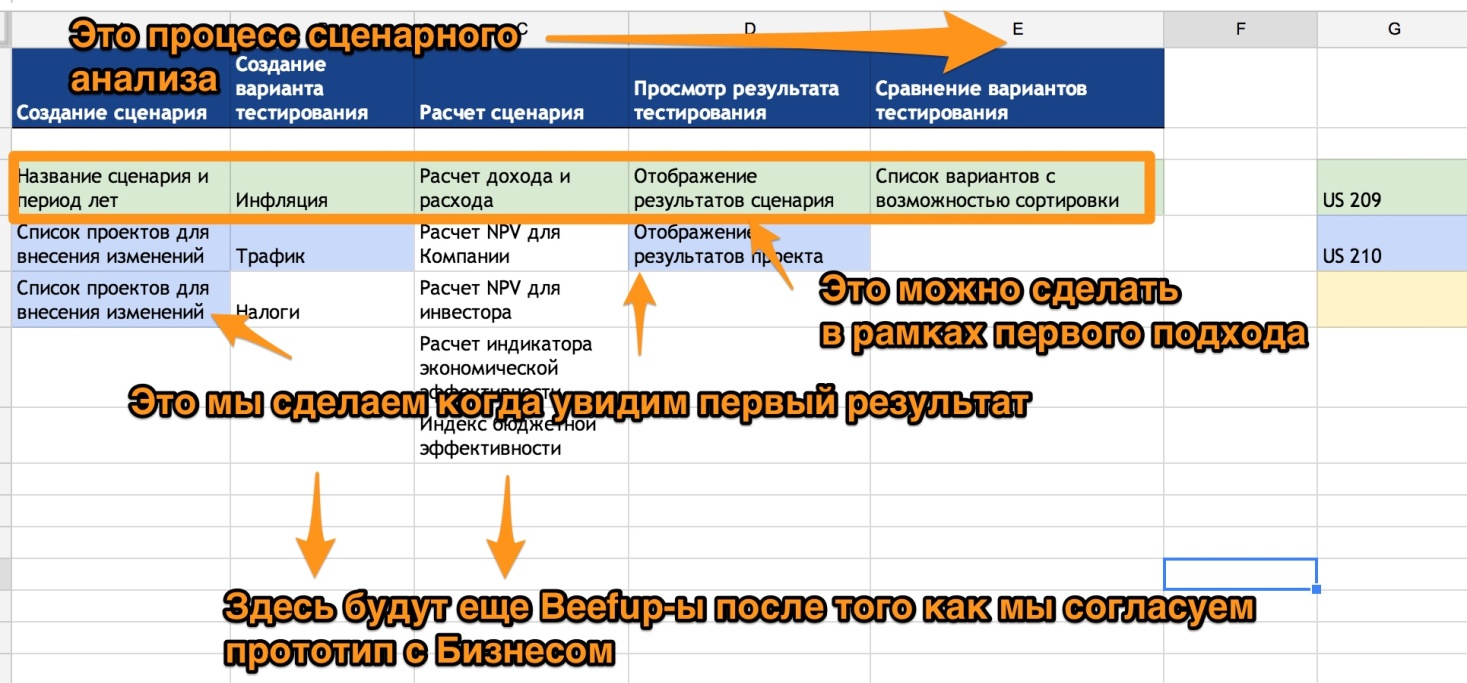
Аналогично мы будем фильтровать идеи по архитектуре и дизайну системы, которые исходят от команды разработки. Ведёт ли переделка архитектуры к более быстром и дешевому достижению цели? Если нет, то зачем нам это делать?

***User Story mapping***

Story map в исходном понимании используется для формирования и ведения карты функционала по продукту. Большая и подробная карта функций/историй. Идея в том, чтобы собрать всех членов команды и за 1-2 дня сформировать 2 сотни историй, которые сформирует исходный беклог продукта, не продуктивна. Уже через неделю мы узнаем о продукте гораздо больше и половину историй нужно будет выкинуть, иначе мы их сделаем и это будет никому не нужно.  
  
Потому для ведения scope по продукту используют Impact mapping, и функционал представляется в виде дерева. В рамках Impact map процесс стратегического планирования более эффективен.  
  
Story map удобен, чтобы разбивать крупные фичи, которые являются атомарными в Impact map. "Стресс-тестирование портфеля проектов", "Оценка эффективности портфеля проектов". Для Impact map это атомарные фичи, Стресс-тестирование - это отдельная фича сценарного анализа, а оценка эффективности - это часть процесса мониторинга. Но в части разработки это будет стоить нам 2-3 недели на каждую из этих задач, потому давайте разбивать.

## Как я разбиваю задачи?

Как инструмент я использую "Google Sheets".

[](http://1.bp.blogspot.com/-Ptcp8lNnxes/UzzmJp5rCfI/AAAAAAAANMk/JElYyePBxhU/s3200/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B8%CC%86+%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7.jpg)

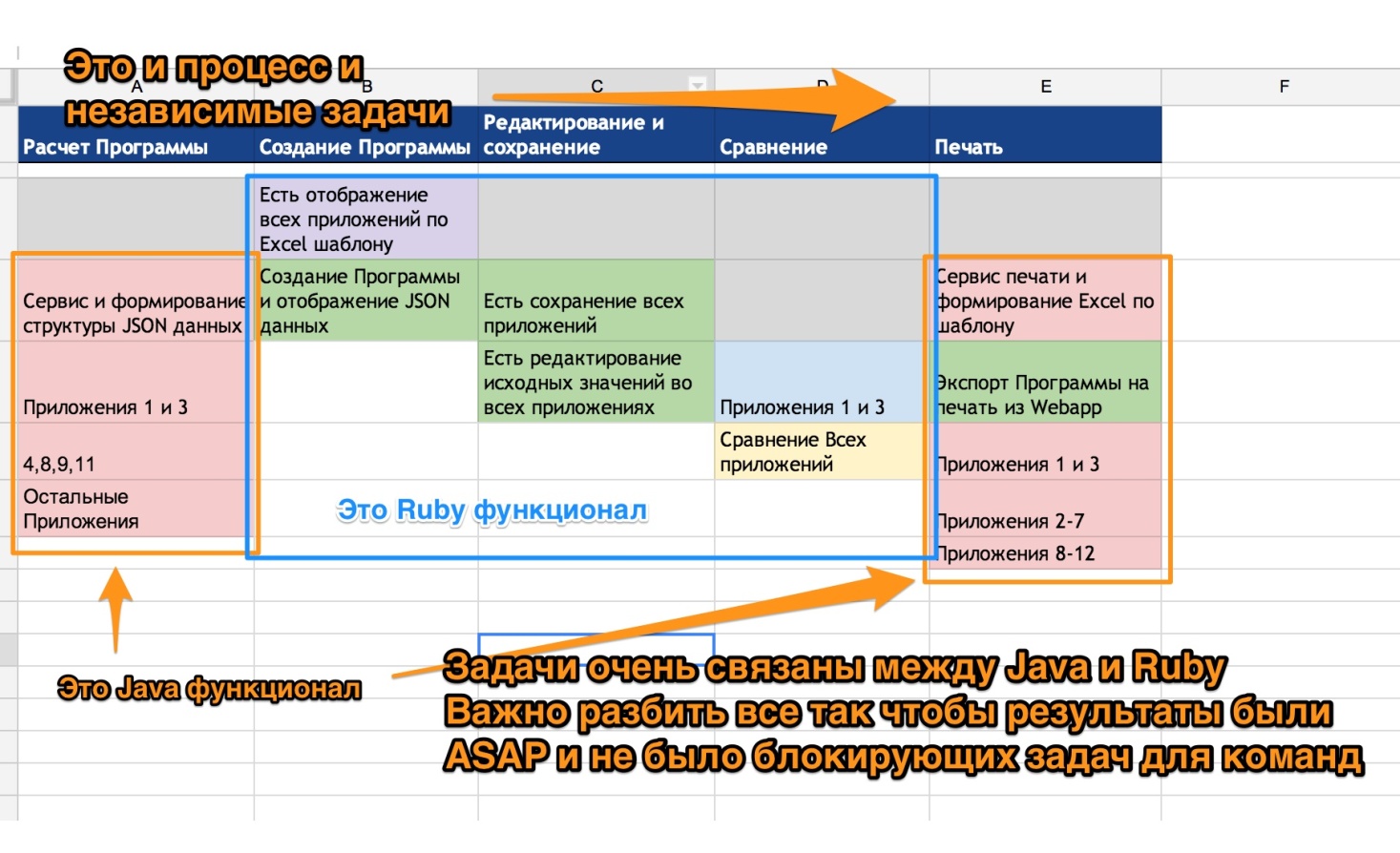
Сверху слева-направо разбивают задачу на этапы/шаги/части в зависимости от типа задачи.  
Сверху-вниз описываю инкремент по задачам, которые можно сделать в рамках этого шага, отсортированные по их ценности/реализуемости. Чаще всего первый шаг - это "мы ничего не делаем".

1. Нужно позволять делать ввод данных - не делаем ввод данных.
2. Нужно делать расчет - не делаем расчет.
3. Нужно визуализировать результаты - показываем статику.

В результате есть матрица из N x M мелких задач, которые можно сделать независимо.  
  
Так как задачи слишком много и они слишком мелкие, то нужно сделать их объединение в рамках одной задачи Вашего беклога. Для этого можно использовать следующую логику:

1. Задачи по самым рисковым шагам/этапам решаем первыми.
2. Группируем задачи так, чтобы были все шаги по горизонтали, и можно было максимально быстро показать результат Бизнесу.
3. Группируем задачи исходя из их сложности. Сколько задач можно объединить чтобы получилась задача на 2-3 дня.

Далее цветами можно сгруппировать задачи в рамках user stories, и определиться с их приоритетом в беклоге.

[](http://3.bp.blogspot.com/-ay0cAmheS3M/UzzmRiawVUI/AAAAAAAANMs/26bS-wdSTRA/s3200/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0.jpg)

Для разбиения Story map по столбцам я использую следующие подходы:

1. У нас есть use case на процесс, который имеет ряд шагов. Столбец == шаг процесса.
2. У нас есть большой функционал, который независим между собой. Столбец == независимая часть задачи.

## Какие есть проблемы/примеры их решения?

Какие могут быть проблемы с построением Story Map или разбиением задач?

### Интеграция 2-х систем

Есть мнение что статус интеграции систем бинарен - он или есть или нет. В результате это занимает 2 месяца и проходит довольно мучительно. А именно:

1. Разрабатываем API для интеграции и согласуем его для обоих систем. Неделя.
2. Реализуем это API для обоих систем. Так главная наша надежда/удача: "мы делаем это параллельно и независимо, потому выйдет дешевле". 1 месяц
3. Начинаем тестировать интеграцию и вылазят проблемы. В результате мы много раз переделаем весь API и переписываем обе реализации. Еще +1 месяц.

Как это можно сделать:

1. Формируем Story Map по методам, которые входят в API.
2. Реализуем 1 метод, который передает 1 параметр, в обеих системах и начинаем тестировать. 1 день.
3. Находим все проблемы с сетевой связностью, со различных стандартами SOAP в Java и С++ и тп. Решаем их. Пишем интеграционные тесты, чтобы можно было автоматически тестировать интеграции 100 раз в день. Пишем моки сервисов для каждой системы, чтобы можно было тестировать интеграцию локально. 1 день
4. Реализуем 1 метод, которые передаем все параметры, в обеих системах и начинаем тестировать. 2 дня.
5. Находим проблемы с различным представлением строк/многомерных массивов/последовательностей в реализациях SOAP для Java и С++. Решаем их и договариваемся что все дальнейшее API будет это учитывать. 1 день.
6. Итеративно наращиваем API с постоянным тестированием интеграции. 1 месяц
7. Остальное время мы съэкономили.

### Модернизация

У нас есть 7 услуг, которые мы оказываем. Нужно внести изменение, которое затронет все оказываемые услуги, причем пока это изменение не реализовано, то все услуги не работают. Это одна большая задача, которая так же имеет бинарный статус - или все 7 услуг изменены или все 7 услуг работают по старому.  
  
Как это можно сделать:

1. "Сделать изменение для 7 услуг" и "сделать изменение для 7 услуг так чтобы все услуги всегда работали" - это 2 разные задачи и вторая задача сильно дороже.
2. Делаем Story map по шагам услуги
3. Делаем изменение для самых рисковых шагов для 1 услуги. Ломаем все остальные услуги.
4. Итеративно дорабатываем 1 услугу и показываем Бизнесу. Получаем OK.
5. Дорабатываем остальные услуг по аналогии.

## Какие результаты вы можете получить?

Какие результаты от разбивания задач на более мелкие, разработка по которым будет стоить 2-3 дня? Не все происходит именно из-за мелких задач, но это сильно влияет:

1. СИЛЬНО сокращает время цикла по задачам для Kanban. Далее идут уже следствия сокрашения Cycle time.
2. Это уменьшает вариативность процесса. То есть более предсказуемое и постоянное кол-во задач, закрываемых за релиз (для Kanban), и более постоянный velocity в Scrum.
3. За счет того, что у вас более предсказуемый и постоянный поток, вы сможете уменьшить WIP и размер очередей в процессе.
4. Это снизит переработки (когда или очень много работы или ее мало) сотрудников и повысит их загрузку.
5. Меньше Cycle time и выше Work time -> увеличение эффективности команды
6. Это повысит качество результатов.
7. Это сильно ускорит обратную связь от Бизнеса, в результате снизит ваши риски и увеличит доверие Бизнеса.