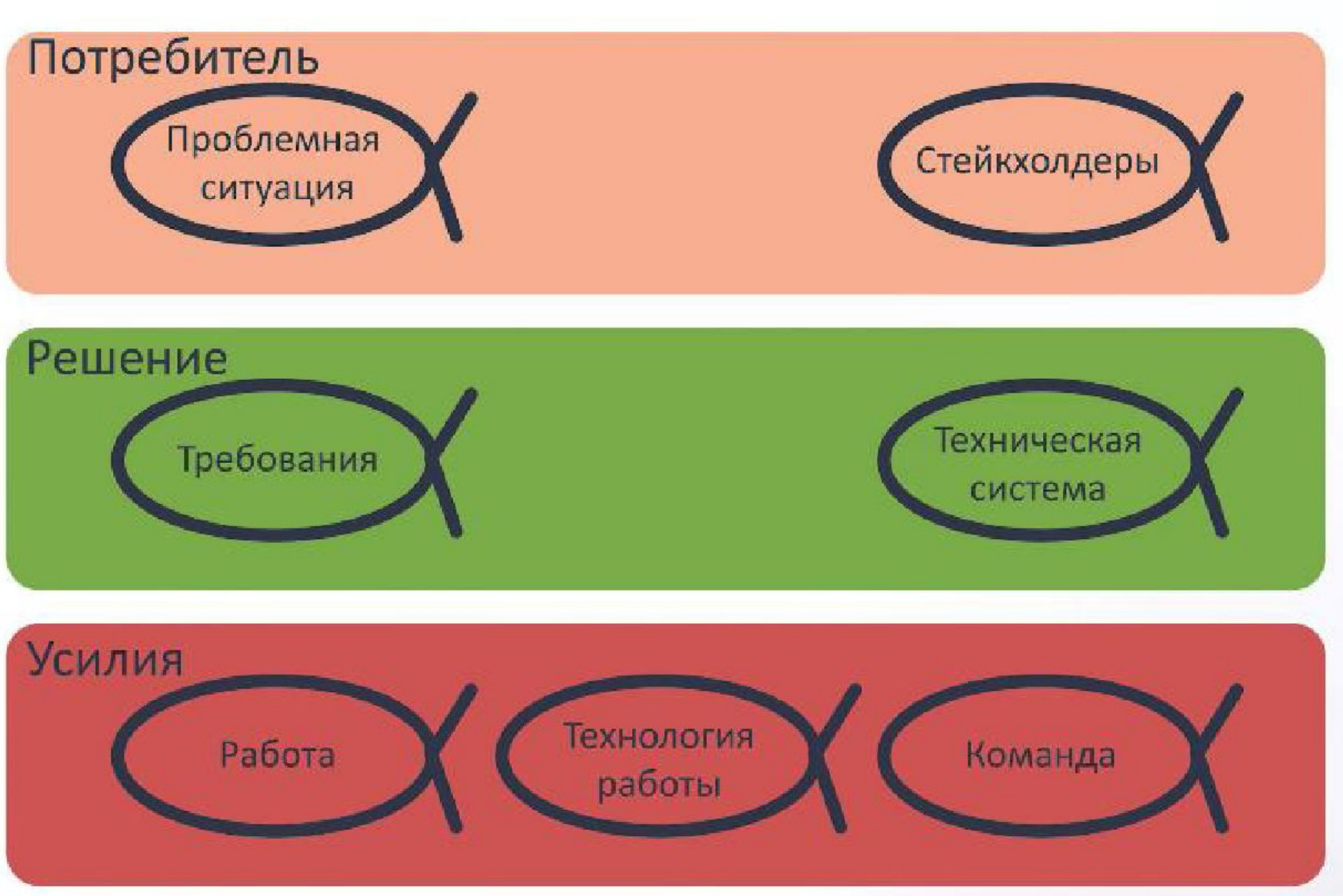
Семь альф проекта

Метамодель Software Engineering Method and Theory (SEMAT)1

- В схеме есть три главных области интересов: потребитель (customer), решение (solution) и усилия (endeavour). Внутри этих областей прописаны те аспекты проекта, к которым надо проявлять особое внимание, чтобы он шел успешно. В метамодели они называются называются «альфами» (ALPHA), и это слово является аббревиатурой фразы Abstract Level Progress and Health Attribute. Она раскрывает смысл этого понятия, «альф» — это показатели прогресса и здоровья вашего проекта, выраженные на максимально обобщенном уровне.
- A — Abstract
- L — Level
- P — Progress
- H — Health
- A — Attribute

SEMAT (семь альф проекта)



Всякая техническая система или продукт, о которых мы сейчас будем говорить, представляет собой единство двух составляющих:

- 1. Функция
- 2. Конструкция

Функция — это тот полезный эффект, который система предоставляет тем, кто ей пользуется.

Конструкция — это то, за счет чего система становится способна предоставлять полезный эффект.

- Задача команды, которая воплощает проект: найти, какое из предложенных решений будет оптимально соответствовать потребностям всех вовлеченных сторон.
- **Идеальный конечный результат** — это когда функция выполняется, а конструкция, которая ее обеспечивает, каким-то образом растворена в среде.

- **Стейкхолдер** (от англ. stake — ставка, интерес) — это всякие лицо, вовлеченное в проблемную ситуацию. Это может быть юридическое лицо (организация), физическое лицо (отдельный человек). Бывает, что проблемная ситуация влияет на этого человека, а бывает, что ситуация подвергается влиянию от этого человека/организации.
- Важно понимать, что стейкхолдер — **это роль**, в которой могут быть человек или организация, состоящая в наличии у этой роли интереса к системе, в том числе к ее функционированию или конструкции, назначению, продукту, обладанию системой какими-либо характеристиками.

У стейкхолдеров есть присущие им цели, интересы, ограничения:

- 1. Что такое цель? Это то, чего они хотят достичь в рамках своей деятельности, тот конечный эффект/результат, который ему нужен. Выражается в терминах из предметной области стейкхолдера, наша система тут не присутствует.

- 2. Ограничения — это законы, регламенты, стандарты, сроки, ресурсы, бюджет, здоровье. Это рамка, за которую стейкхолдер выходить не может и которую вынужден соблюдать.

- 3. Интерес не является прямой целью, но является дополнительным желательным требованием к процессу. Это может быть какой-то фактор удобства, какой-то отрицательный фактор, которого не хотят получить.

Методы анализа стейкхолдеров:

Луковичная диаграмма

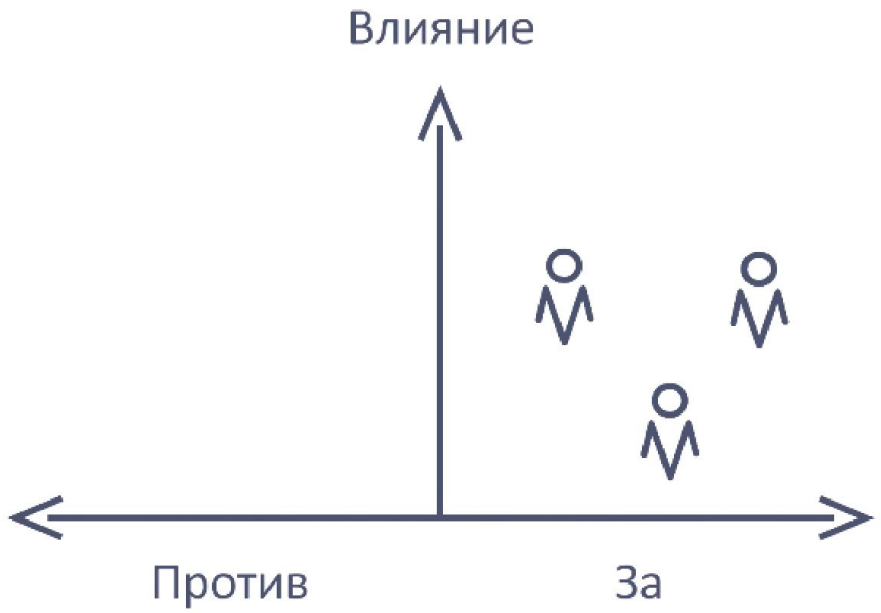
(разбивает всех стейкхолдеров по уровню вовлеченности).

Чек-листы (контрольные списки) типичных стейкхолдеров

- те, кто взаимодействует с системой в ходе эксплуатации (пользователи, техподдержка, функциональные бенефициары — выгодополучатели)
- те, кто держит ресурсы, которые необходимы для того, чтобы система появилась и работала (инвесторы, заказчики)
- различные регуляторы и другие люди, которые будут накладывать разные ограничения (регуляторы — ими могут быть гос-е органы; держатели места внедрения)
- «антистейкхолдеры» (все те, кто не хочет, чтобы ваша система существовала и функционировала)

Карта влияния

- В ней есть два измерения: уровень влияния по вертикали и по горизонтали (либо человек «за», либо «против»). Карта строится для какой-то ситуации, когда происходит выбор. Мы либо пытаемся переманить на нашу сторону тех, кто против, либо стараемся снизить их уровень влияния.

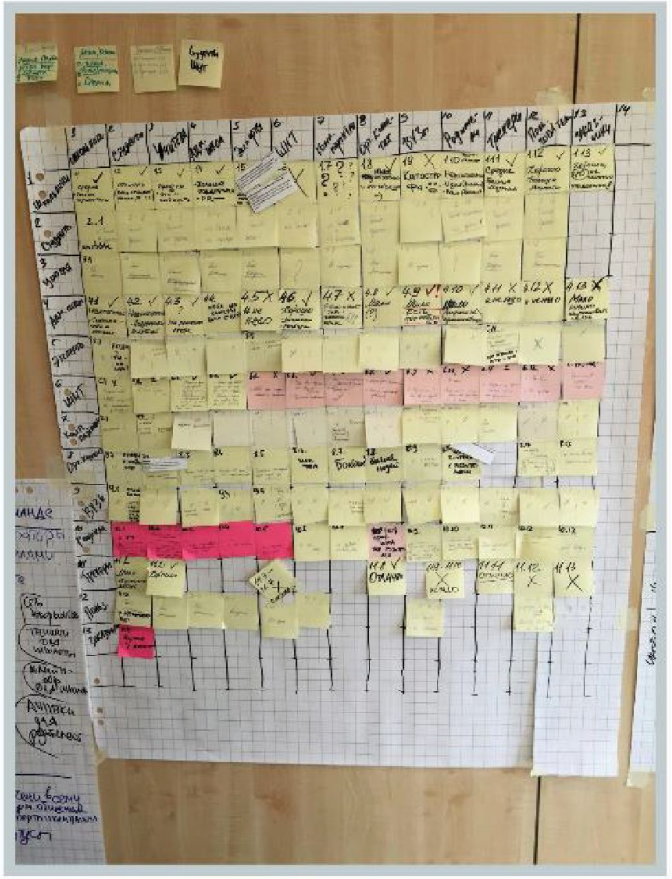


Матрица стейкхолдеров

Таблица, в которой в столбцах и строках одни и те же стейкхолдеры, а на пересечении в ячейках находится

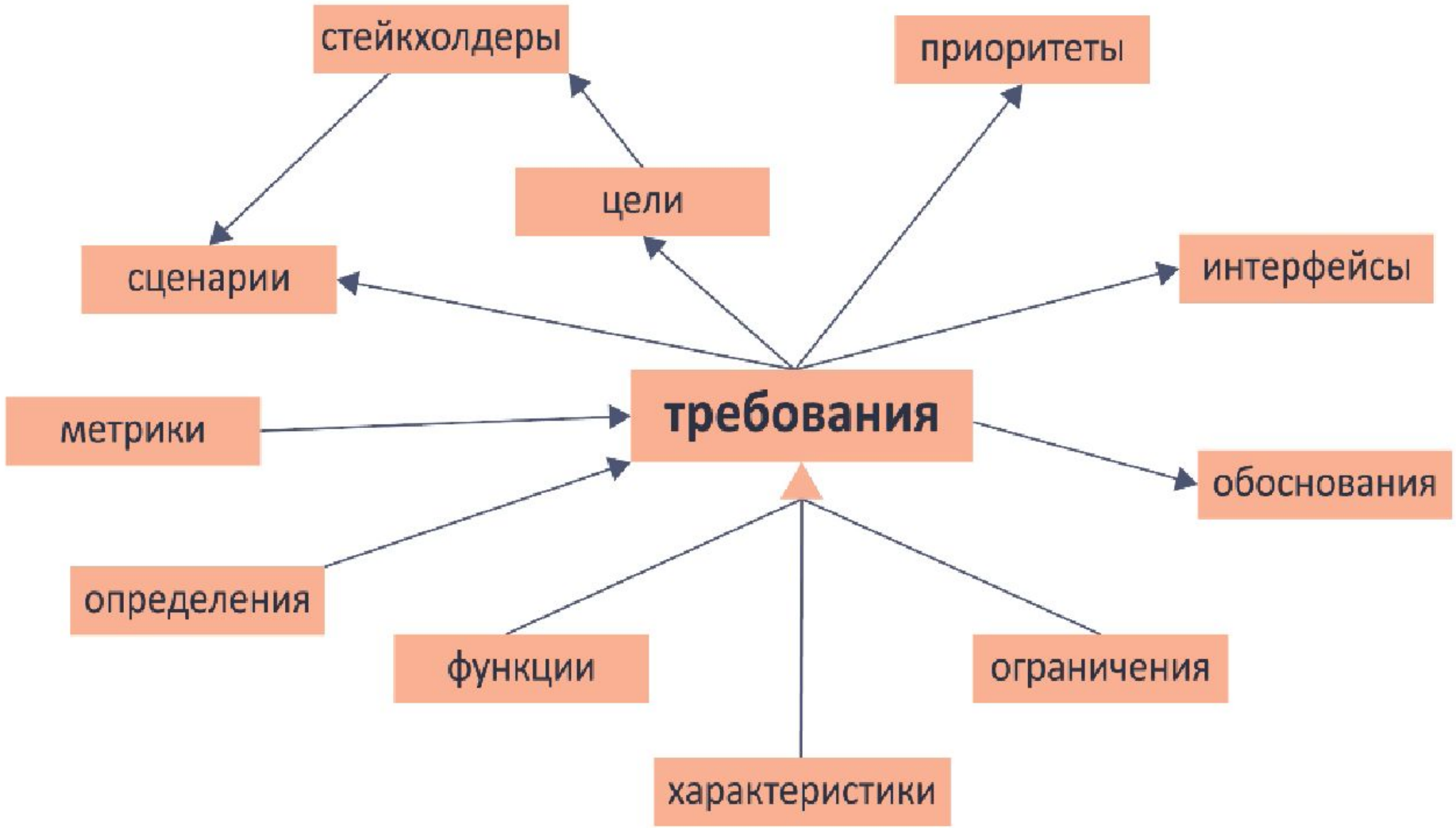
информация об их отношениях:

- как есть
- как хотелось бы
- что нужно поменять

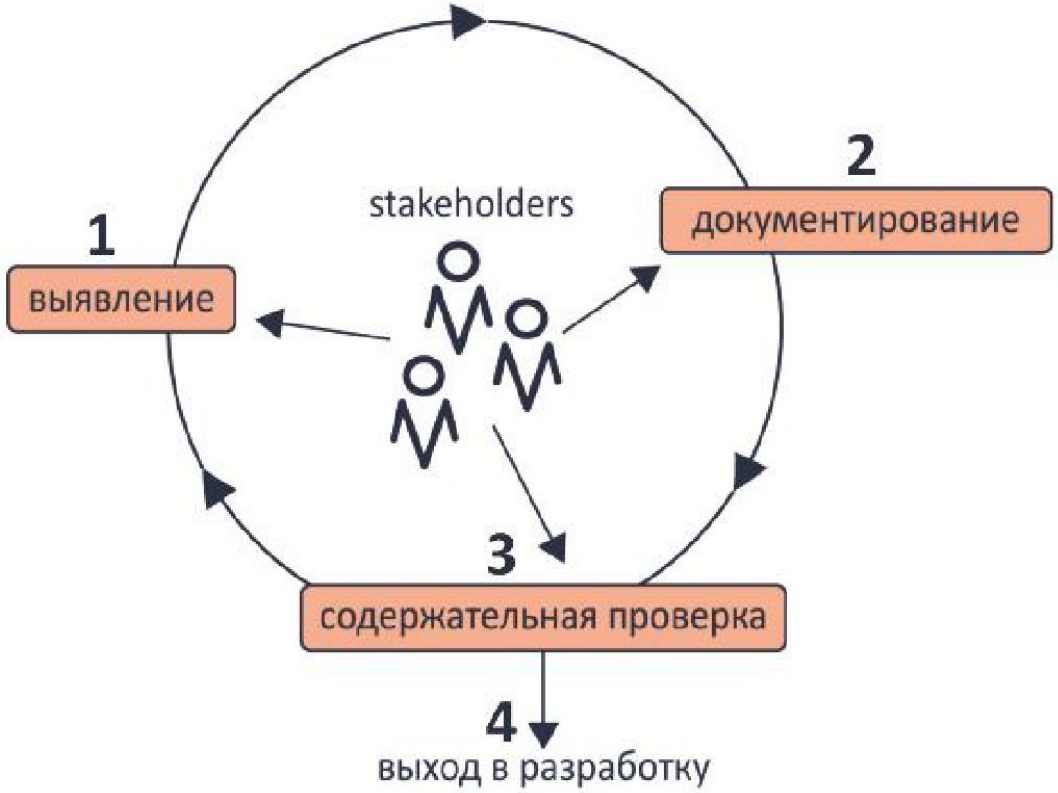


Теперь надо расписать требования к системе, таким образом спланировав продуктовый результат

- Что такое требование? Кто-то называет эту дисциплину «инженерия требований», а кто-то — «управлением требованиями». Требование — это не просто техническое задание (ТЗ), не только «user story» по шаблону и т.д. Так какие должны быть требования к требованиям?
- Ясны и конкретны: понятно, что делать
- Прослеживаемы: понятно, откуда взялись, можно проследить логику принятия решения
- Измеримы: можно оценить степень соответствия продукта требованиям
- Ясны взаимосвязи: что на что влияет, от чего зависит, частью чего является


стейкхолдеры
приоритеты
цели
сценарии
интерфейсы
метрики
требования
обоснования
определения
функции
ограничения
характеристики

Важно понимать, что у разных видов систем будут разные стадии. Есть типичные жизненные циклы, которые используют как эталон, на который можно опираться. Одна из таких моделей называется V-модель:



Существует цикл работы с требованиями, который состоит из нескольких шагов:
Обнаруживаем требования → документируем → проверяем, все ли мы поняли (валидация)→ отдаем требования в разработку

Источники требований:

1. **Индивиды:** ● Интервью; ● Наблюдение (включенное и не включенное); ● «Метод подмастерья» (нас учат, как делать ту или иную задачу)

2. **Группы людей:** ● Воркшопы со стейкхолдерами (получаем список конфликтов, желаемые образы будущего)

● «Очная ставка» — разрешение конфликтов на месте

3. **Археология** (анализируем документы за какой-то период: какие были проблемы, как решались, почему больше так не решаются и т. п.)

4. Законы, стандарты и другие нормативные документы

5. «Вещи»: ● Реверс-инжиниринг (берем хорошее инженерное решение, смотрим, что внутри и перенимаем эту лучшую практику)

● Прототипирование (может ли человек с помощью нашего прототипа чего-то достичь?)

6. Повторное использование требований

Список требований с пояснениями:

1. **Стейкхолдеры и их цели** (чтобы можно было проследить, откуда взялись требования)

2. **Контекст, интерфейсы, границы системы** (нам надо разрабатывать только то, что входит в рамки нашей системы)

3. **Сценарии как еще одна из форм представления требований**. Но у системы в процессе прохождения по сценарию могут быть еще какие-то необходимые характеристики, например, по производительности, по удобству использования. Это тоже надо где-то фиксировать. Также у нас могут быть какие-то неявные допущения, поэтому хорошо просмотреть какой-то наш набор утверждений и проверить, нет ли среди них каких-то неявных смыслов.

4. **Важны и обоснования:** у нас не должно быть таких утверждений о системе, происхождение которых мы не можем проследить. Поэтому мы прописываем: это является сценарием по обеспечению такой-то цели или такого-то стейкхолдера.

5. **Требования должны быть ясны и понятны.** Для этого нам нужно, чтобы в нашем документе были раскрыты все определения и аббревиатуры.

6. **Метрики и приоритеты.** Важно суметь дать ответ про каждое требование: выполнено оно или нет, важно ли это требование (должно реализовываться в первую очередь или может подождать).

После того как мы задокументировали требования, хорошо бы проверить, соответствуют ли они реальным, фактическим потребностям наших стейкхолдеров. Для этого существует несколько методов:

- **Решенческое интервью.** Самый дешевый метод. Мы идем к человеку и спрашиваем, подходит ли ему это, может ли он с помощью этого достичь своей цели?
- **«Театрально-инженерная постановка».** Сценки по сценариям. Например: «Ты будешь нашей системой, а ты ее пользователем, давайте пройдемся по этим сценариям». Возможна еще такая роль как «человек-вдруг», который спрашивает: «А что, если? А вдруг…?». В ходе такого процесса выявляется масса разных пробелов.
- **Пользовательское тестирование прототипов.** Мы делаем подобие того, что у нас будет, выдаем тому, кто является нашим потенциальным пользователем и спрашиваем: «Можешь ли ты достичь той цели, которая заявлена?». На этой стадии приходится доделывать огромную часть работы.

Валидация требований

Рассмотрим пример одного из способов валидации требований. На тренинге мы дали ребятам задание сыграть две сценки.

Сценка 1: как сейчас живет ваш пользователь, когда у него существует проблема?

Сценка 2: как в будущем будет жить ваш пользователь, когда проблемы не станет?

Также мы ввели персонажа «вдруг». Модератор мог в любой момент войти в кадр и сделать что-то, чтобы ситуация изменилась.