# **Waterfall**

Waterfall – один из старейших способов разработки, но используется до сих пор. Принцип – разработка идет только в одном направлении, так же, как и водный поток.

**Этапы:**

1. Анализ – общение с заказчиком, понимание того, что он хочет, осознание того, что необходимо сделать. В результате получаются требования – массивный документ, который остается неизменным до конца разработки.
2. Дизайн – начало работы над дизайном приложения: БД, таблицы, минимальный прототип с базовым функционалом.
3. Разработка – само написание кода. Самый затратный по времени и сложный этап.
4. Тестирование – подготовка тестов, отслеживание ошибок, выводы о качестве продукта. Сравниваются требования из стадии анализа и продукт, который получился в итоге в ходе этапа разработки.
5. Релиз и дальнейшая поддержка – передача продукта заказчику и дальнейшая поддержка: фикс багов, добавление фич и т.д.

**Особенности:**

* первая модель в истории
* каждая фаза начинается только после того, как предыдущая идеально полностью завершена
* почти не используется сейчас

**Минусы:**

* высокая цена фикса ошибок из-за того, что они могут быть найдены очень поздно
* требуются годы на первую версию проекта
* сложно вводить фичи
* много документации

**Когда применяется:**

* требования очень хорошо и ясно задокументированы
* высокий уровень контроля, как например в госзаказах

# **Agile**

Один из самых популярных. Не ставит четких рамок; устанавливает, как это делать и что важнее всего. Важна коммуникация между людьми. Остаются только процессы, которые помогают людям делать работу и имеют большой выхлоп. Работа – важнее, чем документация.

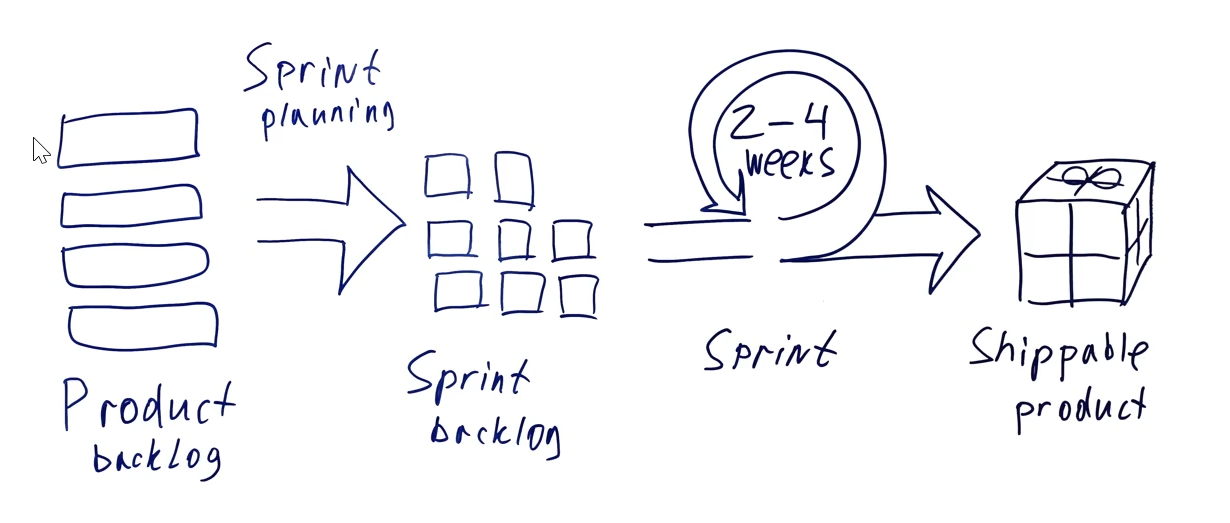
* Важна коммуникация между людьми и процессы, которые упрощают работу
* Работа важнее, чем документация
* Контакт с заказчиком важнее, чем контракт
* Адаптация к обстоятельствам важнее, чем следование плану

**Принципы Agile:**

1. Удовлетворение клиента – самое важное
2. Быстрый отзыв на изменения ТЗ
3. Быстрая разработка – от пары недель до нескольких месяцев
4. Бизнесмены и разработчики должны постоянно работать вместе
5. Давать работу замотивированным сотрудникам
6. Важна коммуникация в жизни, лицом к лицу
7. Отсутствие переработок
8. Простота важна
9. Команды должны быть самоорганизованными
10. *Главный критерий оценки – рабочий продукт*

# **Scrum**

Основан на принципах Agile. Главный плюс – любая фича будет реализована в течение одного спринта (примерно 2 недели). На выходе получается большое количество версий продукта с разным функционалом внутри.



Ивенты:

- Sprint Grooming – обсуждение перед стартом

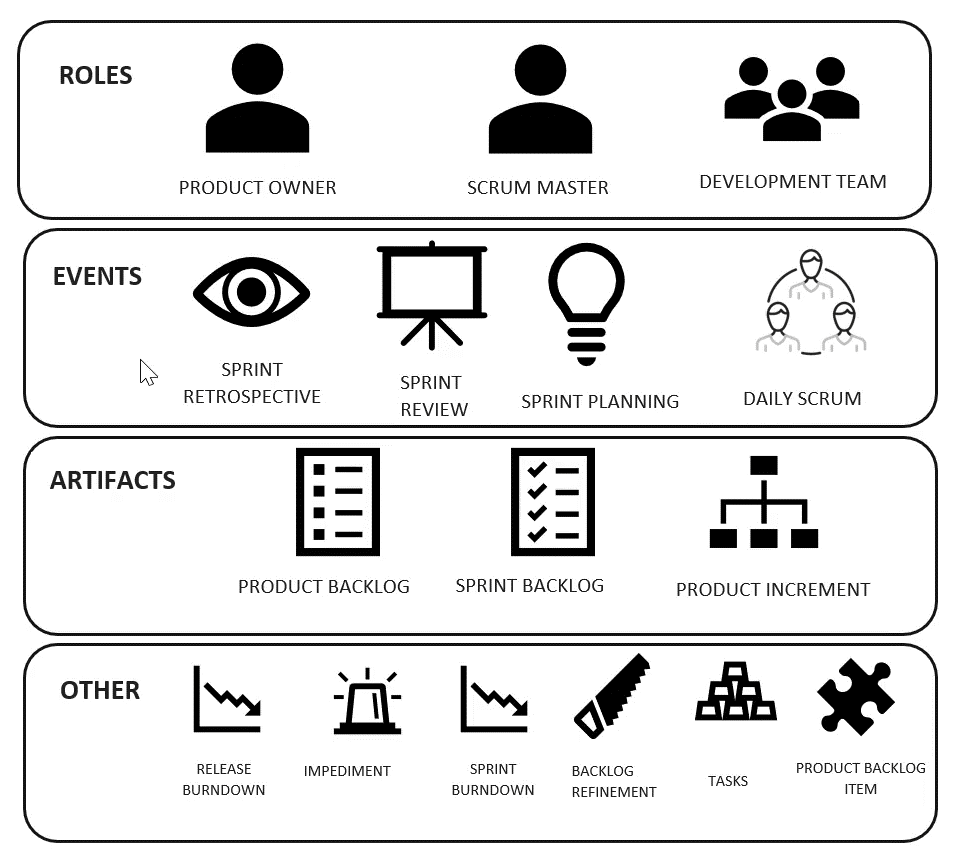
- Sprint Planning

- Daily scrum (Standup) – обсуждение что я сделал, что буду делать, что мне мешает

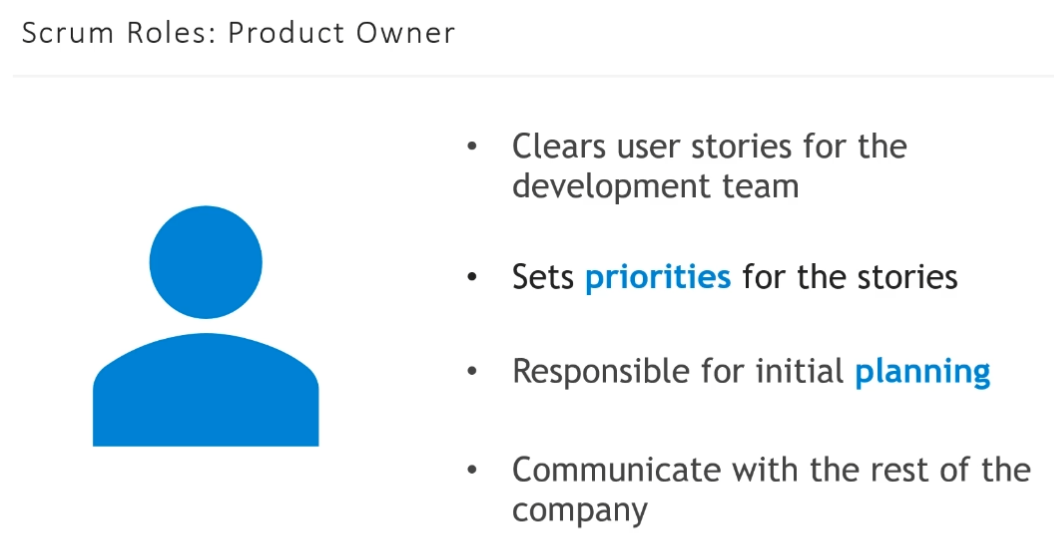
- Sprint Review

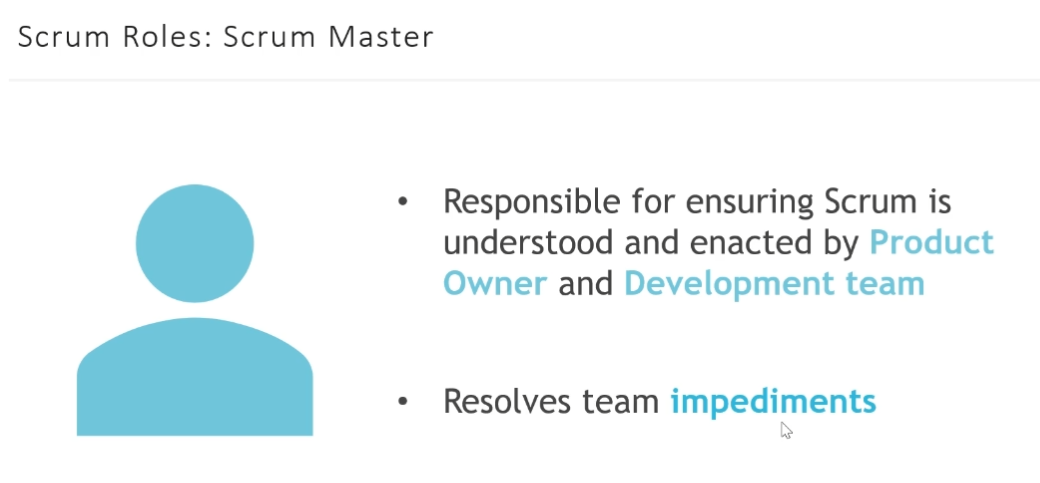
- Sprint Retrospective

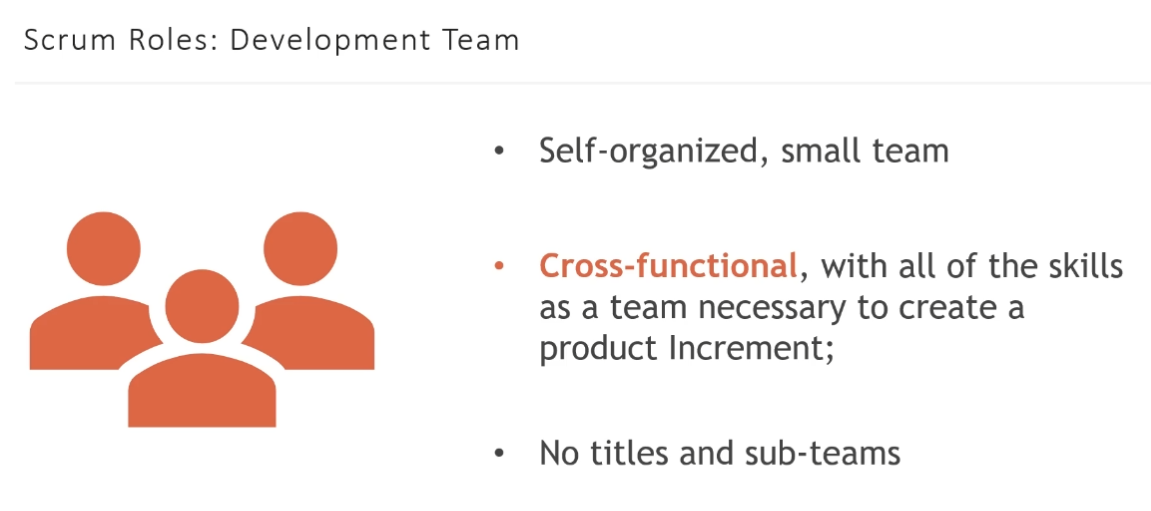
**Элементы Scrum:**



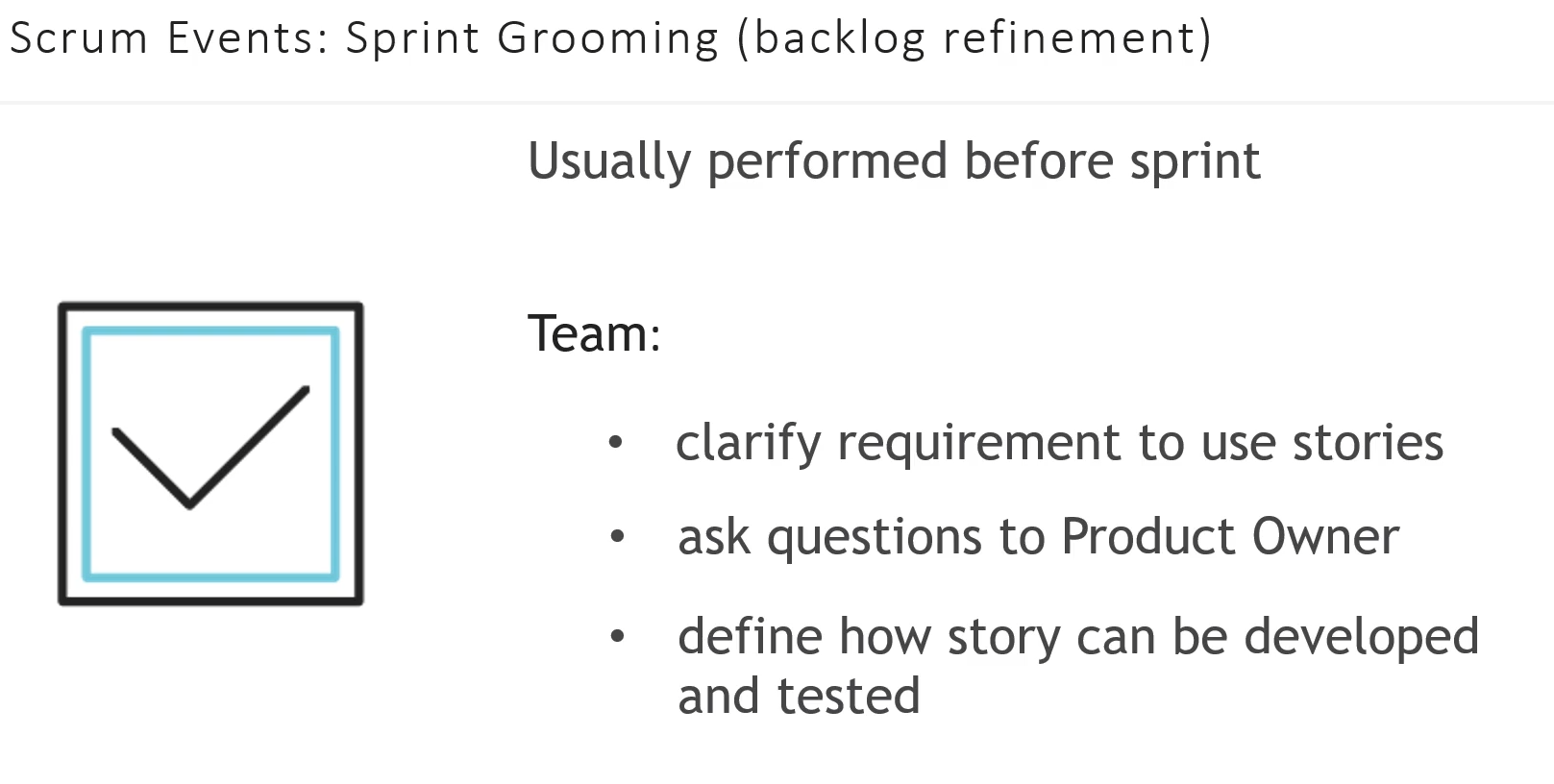
**Роли:**

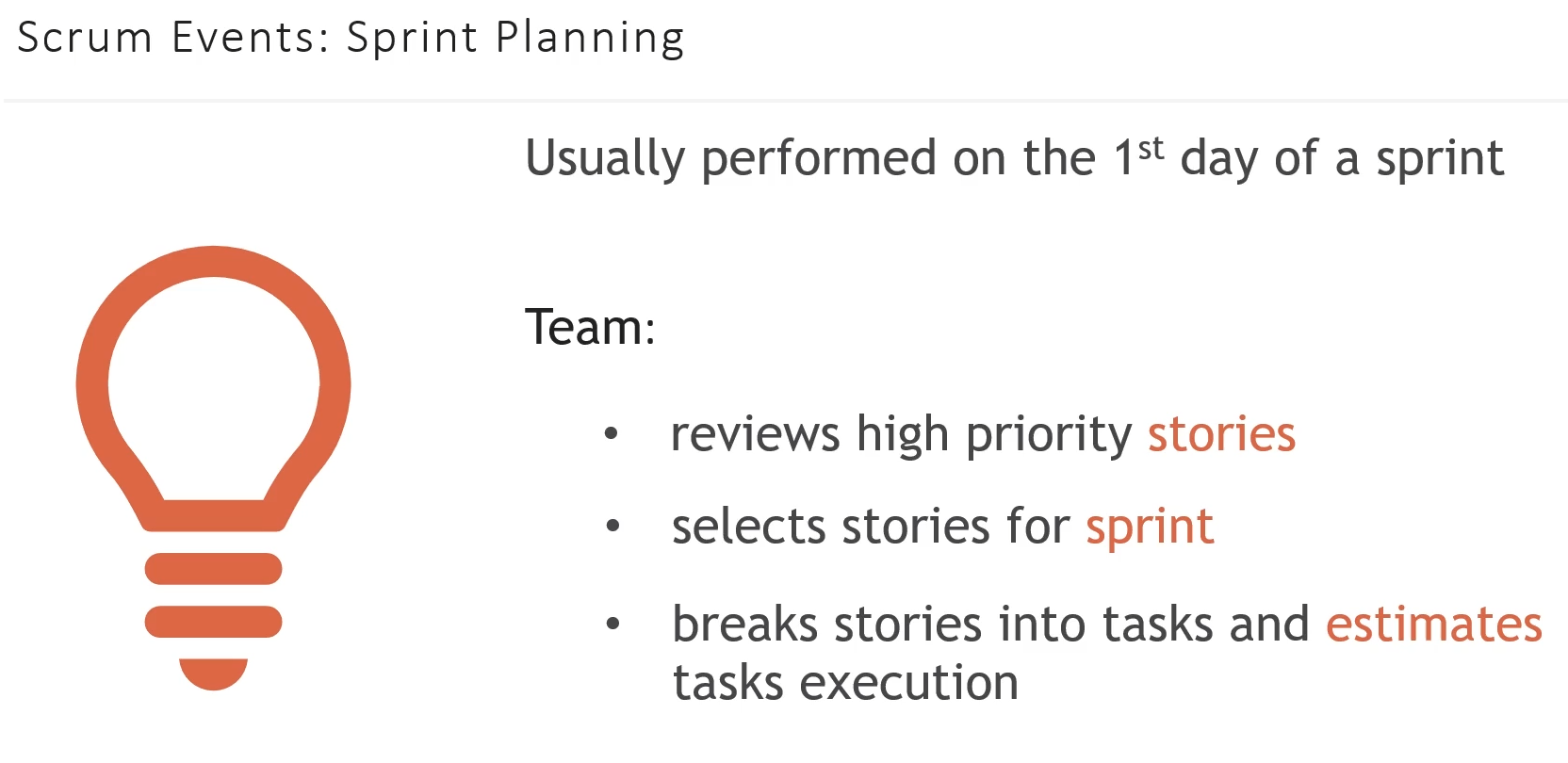


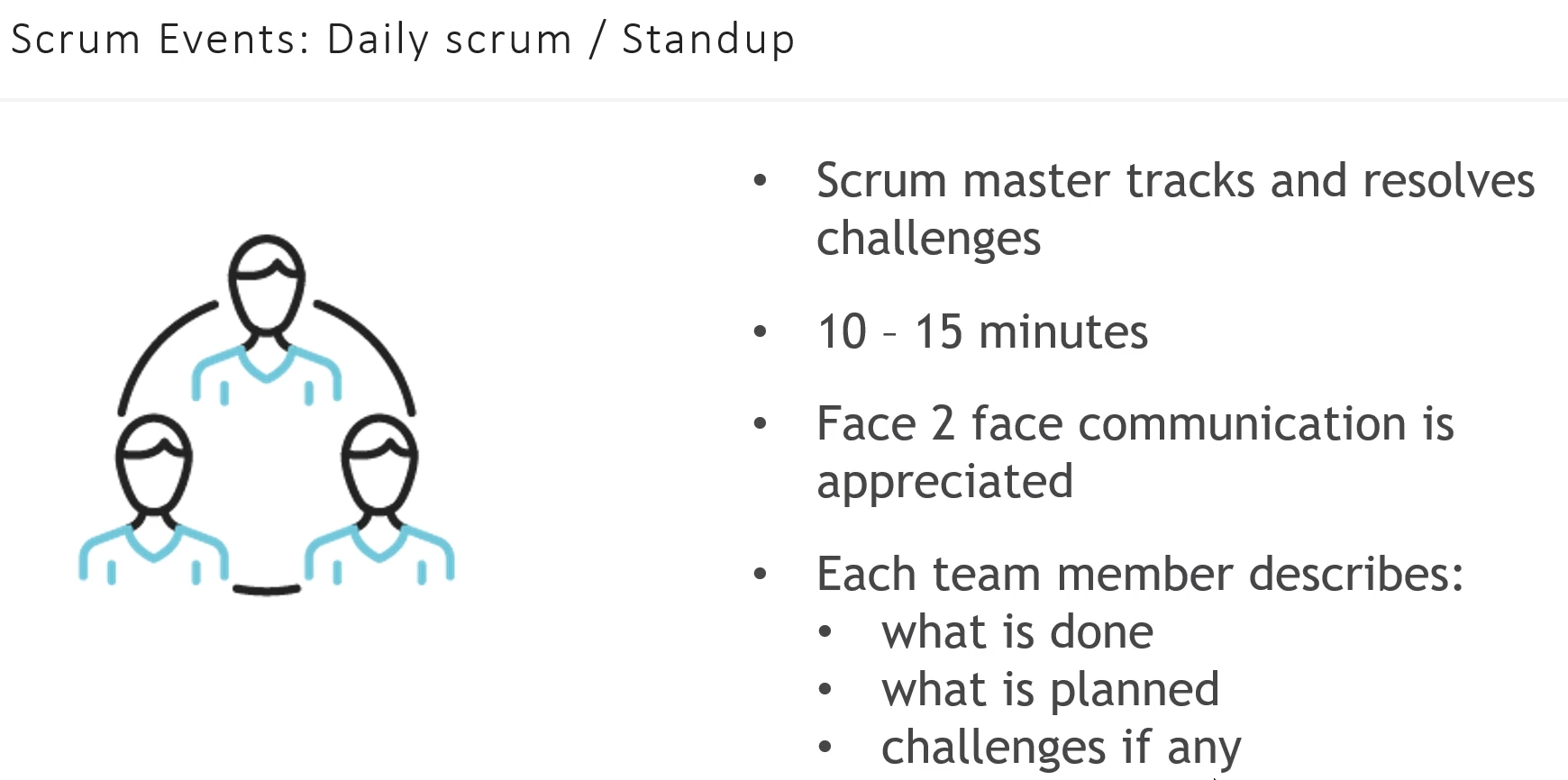


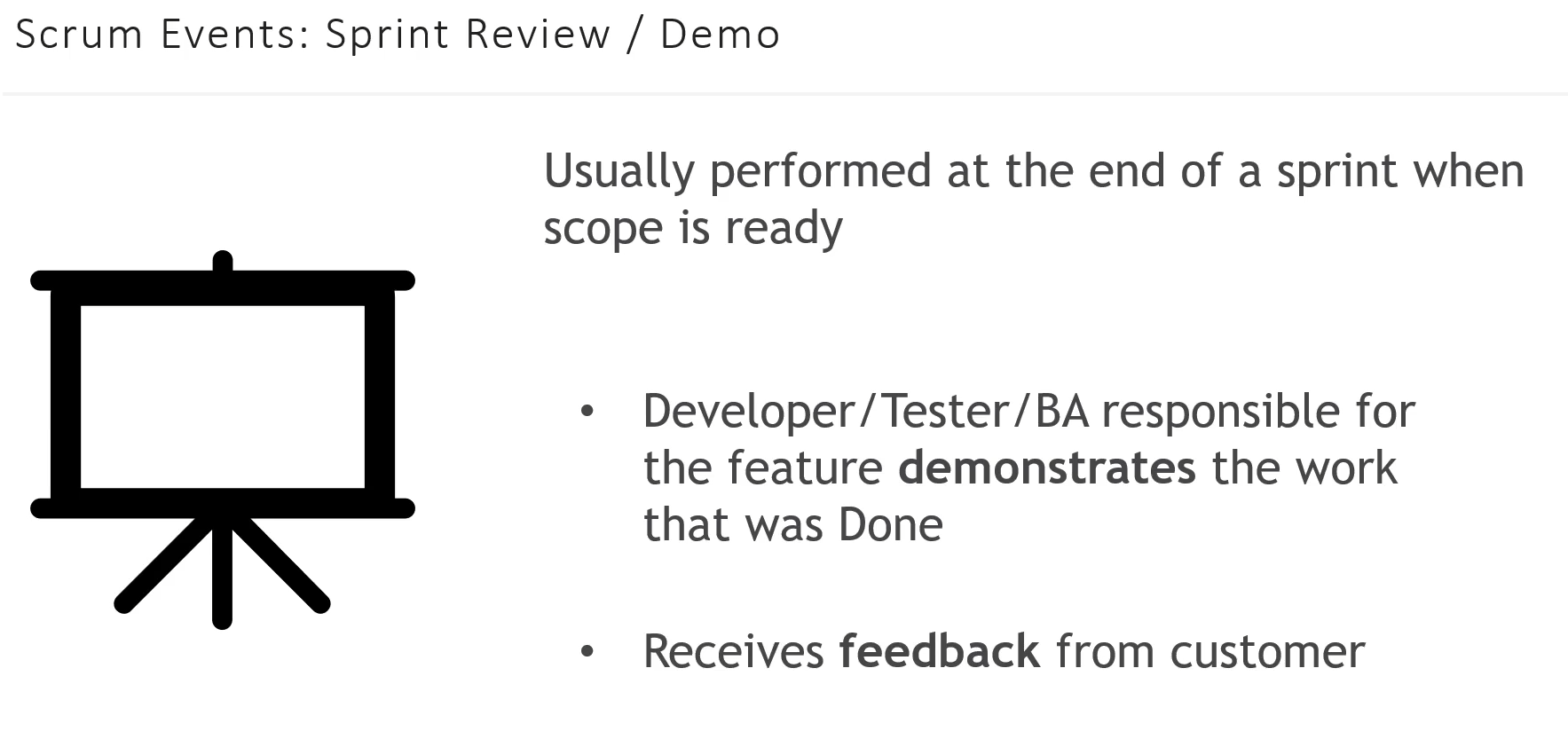


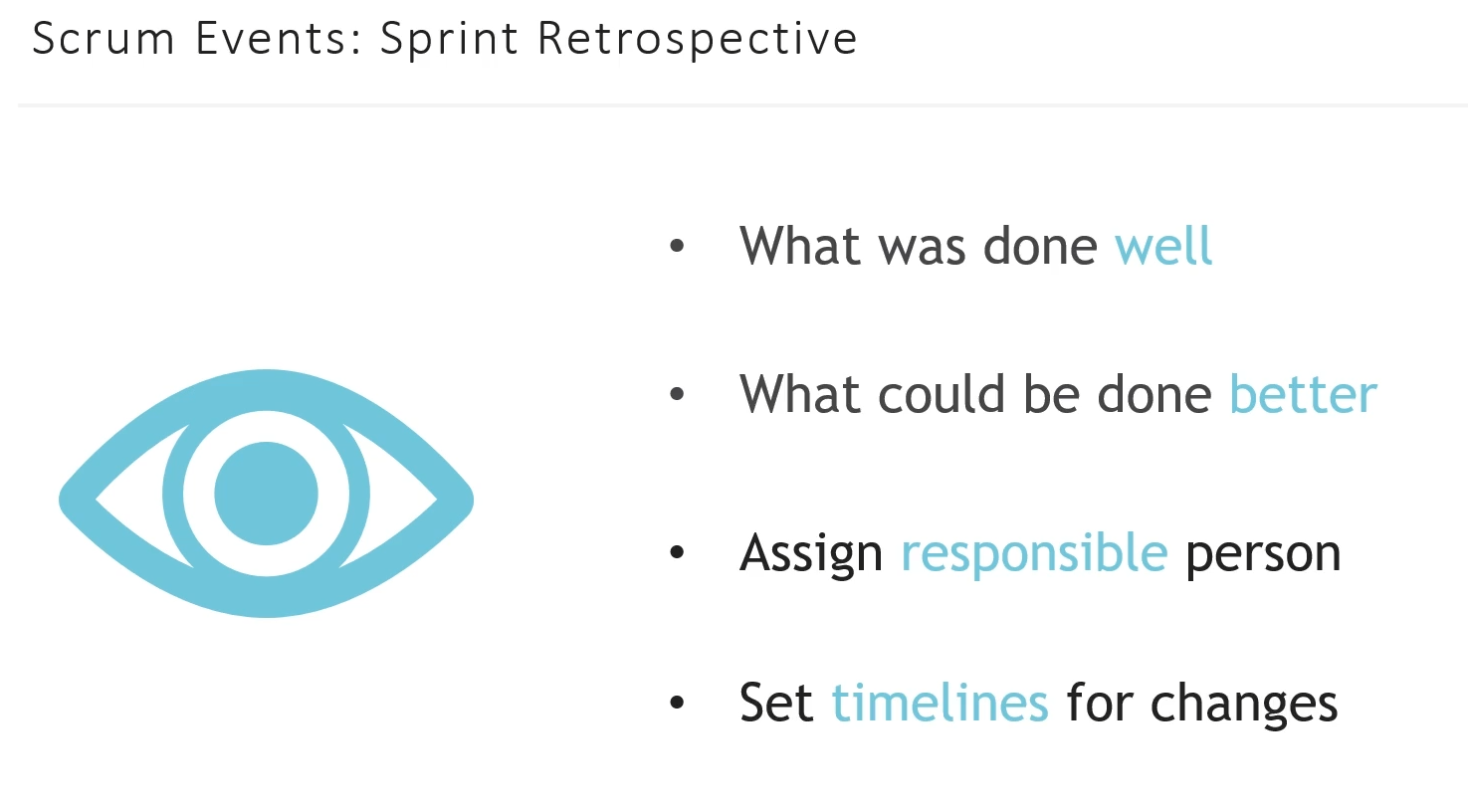
**Events:**











# **Kanban**

Метод управления разработкой, реализующий принцип “точно в срок” и способствующий равномерному распределению нагрузки между работниками. (Изобретён компанией TOYOTA)

Особенности:

* Визуализация задач на доске при помощи карточек
* За этапы реализации 1 таска могут отвечать несколько человек
* Над задачей трудятся столько времени, сколько надо для её решения
* Ограничьте число одновременно выполняемых задач на 1 сотрудника

Плюсы:

* Члены команды могут видеть текущее состояние задачи на любой момент времени
* Гибкость планирования
* Меньшая длительность итерации
* Быстрое выявление проблем

Минусы:

* Ограничение по размеру команды
* Краткосрочность планирования

# **Extreme Programming**

Один из фреймворков Agile подхода, который позволяет создавать программное обеспечение высокого качества, при этом поддерживать высокое качество жизни у членов команды разработки.

Цель – справится с постоянно меняющимися требованиями к программному продукту и повысить качество разработки.

13 Практик экстремального программирования:

1. Вся команда
2. Игра в планирование
3. Частые релизы версий
4. Пользовательские тесты
5. Коллективное владение кодом
6. Непрерывная интеграция кода
7. Стандарты кодирования
8. Метафора системы
9. Устойчивый темп
10. Разработка, основанная на тестировании
11. Парное программирование
12. Простой дизайн
13. Рефакторинг

Преимущества:

* Работаем не больше 40 часов в неделю
* Быстрые изменения в коде и добавление новых функций
* Код постоянно тестируется
* Быстрый темп разработки
* Высокое качество кода
* Затраты на разработку ниже

Минусы:

* Успех проекта зависит от вовлечённости заказчика
* Трудно предугадать затраты времени на проект
* Успех ExtPrg зависит от уровня программистов
* Парное программирование
* Недостатки структуры и документации

Принципы:

- Простота

- Коммуникация

- Обратная связь