Scrum

XP, Poker Planning и куча страшных слов

Netcracker
An NEC Company

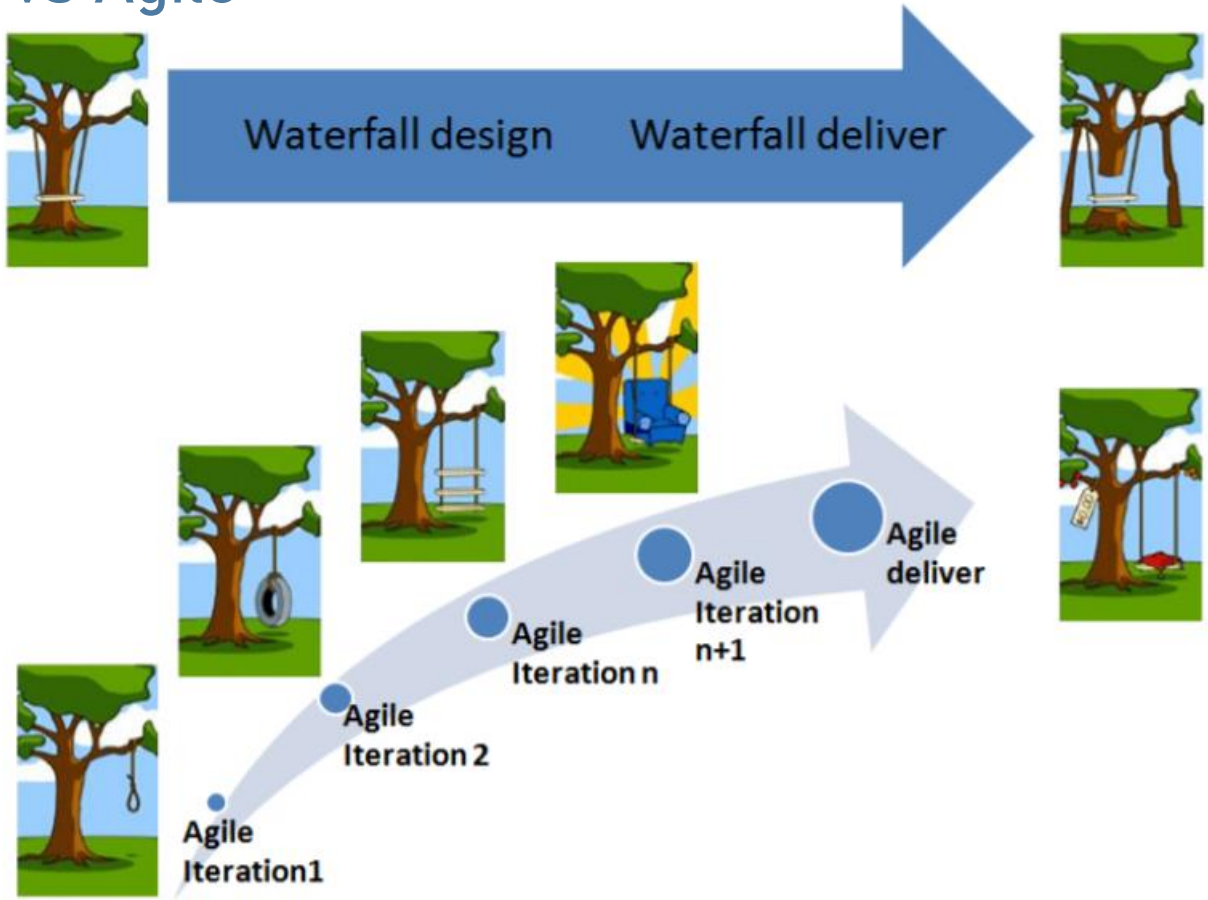
План

1. Scrum
2. Story, как часть XP
3. Poker со своим блэк-джеком и Фибоначчи.
4. Домашнее задание

Agile Manifesto Principles

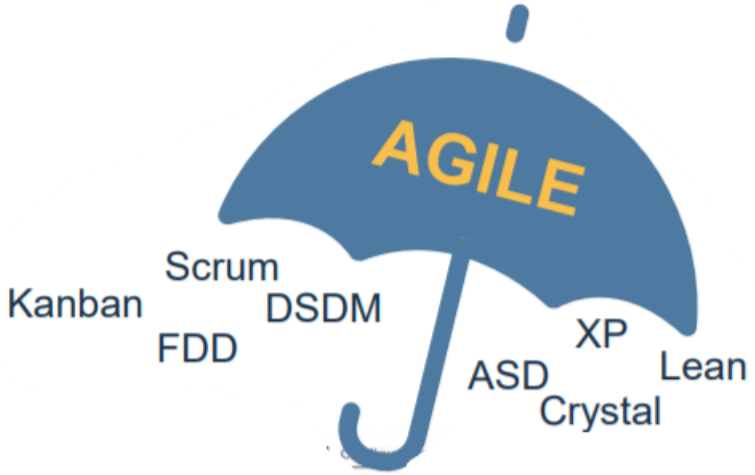
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Working software is the primary measure of progress.
4. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
5. Business people and developers must work together daily throughout the project.
6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get
the job done.
7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain
a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Waterfall vs Agile



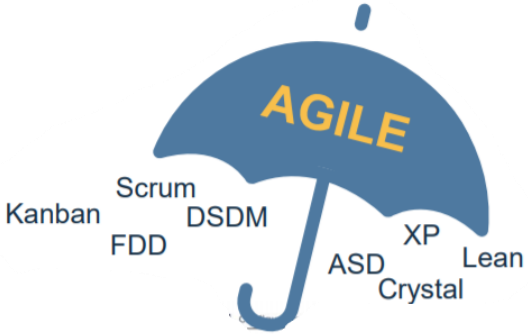
SCRUM

Скрам — это фреймворк, который помогает решать изменяющиеся в процессе работы задачи, чтобы продуктивно и творчески поставлять клиентам продукты с максимально возможной ценностью.



SCRUM

Скрам основан на теории эмпирического управления (эмпиризме). Согласно этой теории, источником знаний является опыт, а источником решений – реальные данные.



Эмпиризм утверждает, что знание приходит с опытом, решения принимаются на основании того, что является известным

SCRUM. «Три кита»

Прозрачность

Инспекция

Адаптация

SCRUM. Три кита – прозрачность

Прозрачность

«Прозрачность» означает, что значимые характеристики процесса должны быть известны тем, кто отвечает за его результат. Прозрачность требует: эти характеристики должны быть обозначены общими соглашениями, чтобы все участники одинаково понимали происходящее.

- терминология, имеющая отношение к процессу, должна быть общей для всех участников.
- понимание Критериев Готовности должно быть общим для исполнителей работ и тех, кто инспектирует результаты.

SCRUM. Три кита – инспекция

Инспекция

Участники процесса должны регулярно инспектировать артефакты Скрама и свой прогресс в движении к Цели Спринта , чтобы вовремя обнаружить нежелательные отклонения. Частота проведения проверок не должна мешать работе. Проверки приносят наибольшую пользу, когда выполняются профессионалами с соответствующими навыками.

SCRUM. Три кита - адаптация

Адаптация

Если в результате инспекции выясняется, что одна или несколько характеристик процесса выходят за допустимые пределы, и это приводит продукт в неприемлемое состояние, то процесс или обрабатываемый материал необходимо изменить. Чем раньше будут внесены изменения, тем меньше риск дальнейших отклонений.

SCRUM. Ценности

Успешность использования Скрама напрямую зависит от того, насколько хорошо люди придерживаются ценностей Скрама (преданность, смелость, сфокусированность, открытость и уважение).

- Каждый участник предан целям Скрам-команды.
- Все обладают смелостью действовать правильно и работать над решением сложных задач.
- Каждый участник сфокусирован на целях Скрам-команды и на их достижении в рамках Спринта.
- Заинтересованные лица и Скрам-команда соглашаются быть открытыми друг с другом в работе, несмотря на возникающие трудности.
- Участники Скрам-команды уважают профессионализм и самостоятельность друг друга.

SCRUM-команда

SCRUM-команда. Product Owner

Владелец Продукта – единственный человек, который отвечает за управление Бэклогом Продукта . Управление Бэклогом Продукта включает в себя:

- описание Элементов Бэклога Продукта ясным и понятным образом;
- управление порядком Элементов Бэклога Продукта для наилучшего достижения целей и миссий;
- оптимизацию ценности работы, выполняемой Командой Разработки;
- обеспечение доступности, прозрачности и ясности Бэклога Продукта для всех участников процесса. Бэклог Продукта при этом отражает, над чем Скрам-команда будет работать дальше;
- гарантию, что Команда Разработки в достаточной степени понимает Элементы Бэклога Продукта.

SCRUM-команда. Development team

Команды Разработки обладают рядом характеристик:

- Они самоорганизующиеся
- Они кросс-функциональны: команда обладает всеми навыками, которые необходимы для создания Инкремента Продукта.
- Разработчик — единственная роль для членов Команды Разработки, независимо от типа задач, которые он выполняет.
- Скрам не признает подкоманд в Команде Разработки, независимо от областей, над которыми необходимо работать (например, тестирование, архитектура, эксплуатация или бизнес-аналитика). Это правило не имеет исключений.
- Команда Разработки несет коллективную ответственность за создание Инкремента Продукта.

SCRUM-команда. Scrum-master

Скрам-мастер оказывает Владельцу Продукта следующие услуги:

- обеспечивает условия, при которых Скрам-команда как можно лучше понимает цели, объём работ и предметную область;
- помогает найти наиболее эффективные техники для управления Бэклогом Продукта;
- объясняет Скрам-команде необходимость кратких и понятных Элементов Бэклога Продукта;
- объясняет особенности планирования продукта в эмпирической среде;
- помогает Владельцу Продукта упорядочить Бэклог Продукта, чтобы получить максимальную ценность продукта; • способствует лучшему пониманию гибкости и её применения;
- фасилитирует события Скрама при необходимости.

SCRUM-команда. Scrum-master

Скрам-мастер оказывает Команде Разработки следующие услуги:

- коучит Команду Разработки быть самоорганизующейся и кросс-функциональной;
- помогает Команде Разработки создавать продукты с высокой ценностью;
- устраняет препятствия, мешающие прогрессу Команды Разработки;
- фасилитирует события Скрама при необходимости;
- коучит Команду Разработки в тех частях организации, в которых Скрам еще не полностью понят и принят.

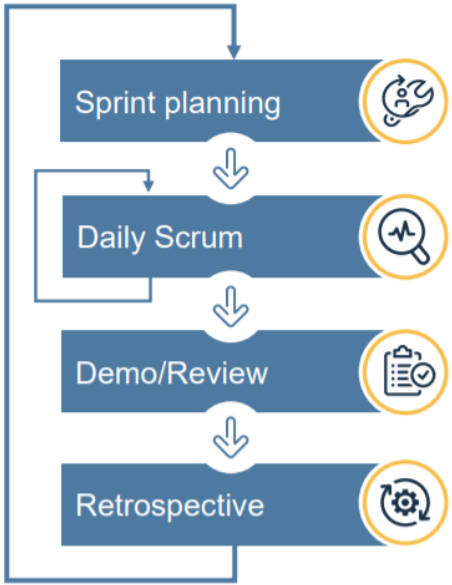
Скрам-мастер оказывает услуги Организации:

- направляет и коучит организацию при внедрении Скрама;
- планирует переход на Скрам в организации;
- помогает сотрудникам и заинтересованным лицам понять теорию и практику Скрама, правильно реализовать принципы эмпирической разработки продуктов;
- способствует изменениям, направленным на повышение продуктивности Скрамкоманд;
- сотрудничает с другими Скрам-мастерами для повышения эффективности применения Скрама в организации.

SCRUM. События Скрама

Спринт — контейнер для остальных событий.

Спринт состоит из Планирования Спринта, Ежедневного Скрама, разработки, Обзора Спринта и Ретроспективы Спринта



Каждый Спринт можно считать проектом, который длится не более одного месяца. Спринты, как и проекты, нужны для достижения целей. Каждый Спринт включает цель, концепцию реализации с адаптивным планом по её достижению, исполняемую работу и Инкремент продукта как результат работы

Существует единственное основание для отмены Спринта — потеря актуальности Цели Спринта.

SCRUM. Планирование Спринта

Тема первая: что будет сделано?

Тема вторая: как будет выполнена работа?

Во время Планирования Спринта Скрам-команда также формирует Цель Спринта.

Цель Спринта – это ориентир для Команды Разработки. Чтобы его достичь, команда должна использовать технологии и реализовывать функциональность.

К концу Планирования Спринта Команда Разработки должна уметь объяснить Владельцу Продукта и Скрам-мастеру, как она намерена работать в рамках самоорганизации, чтобы достичь Цель Спринта и создать ожидаемый Инкремент.

SCRUM. Ежедневный Скрам

- Что я сделал вчера, что помогло Команде Разработки приблизиться к Цели Спринта?
- Что я сделаю сегодня, чтобы помочь Команде Разработки достичь Цели Спринта?
- Вижу ли я какие-либо препятствия, которые могут помешать мне или Команде Разработки достичь Цели Спринта?

SCRUM. Обзор Спринта (Demo)

Ключевые элементы Обзора Спринта:

• в число участников встречи входят Скрам-команда и ключевые заинтересованные лица, которых приглашает Владелец Продукта;

• Владелец продукта объясняет, какие Элементы Бэклога готовы, а какие нет;

• Команда Разработки рассказывает о том, что получилось во время Спринта, какие возникли проблемы и как они были решены;

• Команда Разработки демонстрирует готовую работу и отвечает на вопросы об Инкременте;

• Владелец Продукта описывает текущее состояние Бэклога Продукта. При необходимости он прогнозирует возможные даты завершения разработки Продукта, основываясь на текущих показателях прогресса;

• все присутствующие обсуждают, над чем стоит работать дальше. Таким образом Обзор Спринта предоставляет ценные данные для следующего Планирования Спринта;

• проводится обзор, как изменения рынка или потенциальное использование продукта могли изменить то, что нужно сделать в первую очередь;

• выполняется обзор сроков, бюджета, возможностей и позиций на рынке для будущих релизов или возможностей продукта.

SCRUM. Ретроспектива Спринта

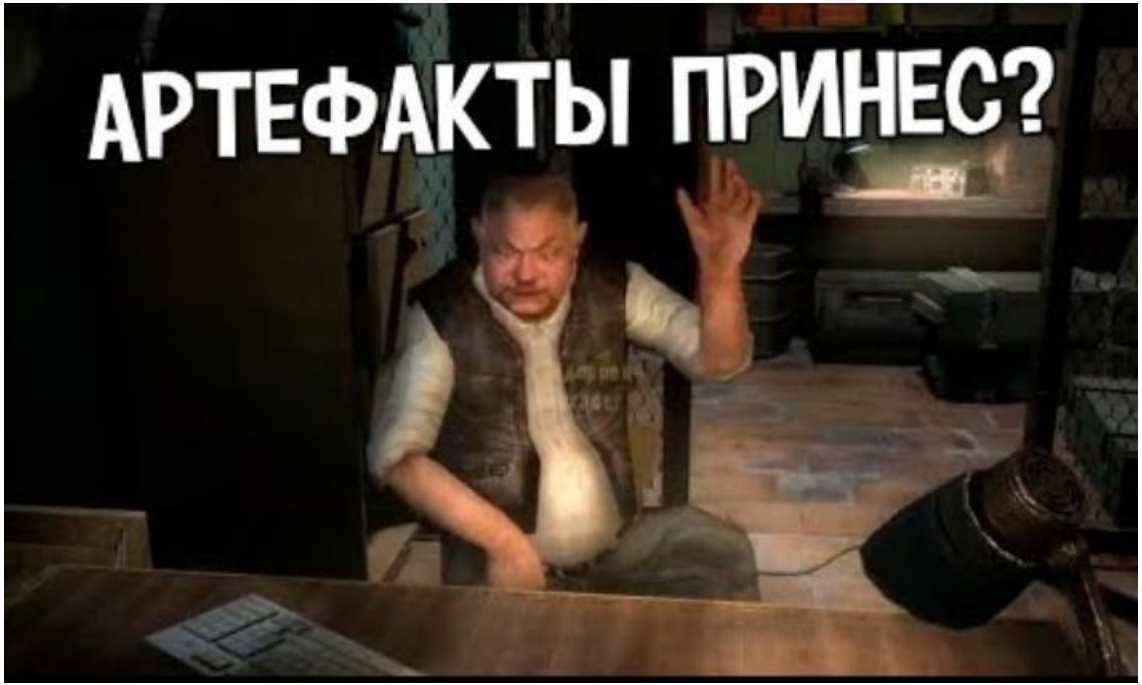
Цели проведения Ретроспективы Спринта:

• инспекция прошедшего Спринта применительно к людям, отношениям, процессам и инструментам. Обнаружение и упорядочение того, что прошло хорошо и того, что нуждается в улучшении;

• создание плана внедрения улучшений в процесс работы Скрам-команды

Ретроспектива Спринта – формальная возможность сконцентрироваться на инспекции и адаптации

SCRUM. Артефакты Scrum



SCRUM. Артефакты Scrum

Бэклог Продукта

Бэклог Спринта

Инкремент

SCRUM. Критерии Готовности

Story

As a <role>

I can <activity>

So that <business value>

(Roles can be people, devices, or systems)

Story

User Story Card Examples

As a Consumer, **I want to be** able to see my daily energy usage **so that** I can lower my energy costs and usage

As a technical support member, **I want** the user to receive a consistent and clear message anywhere in the application **so that** they can fix the issue without calling support.

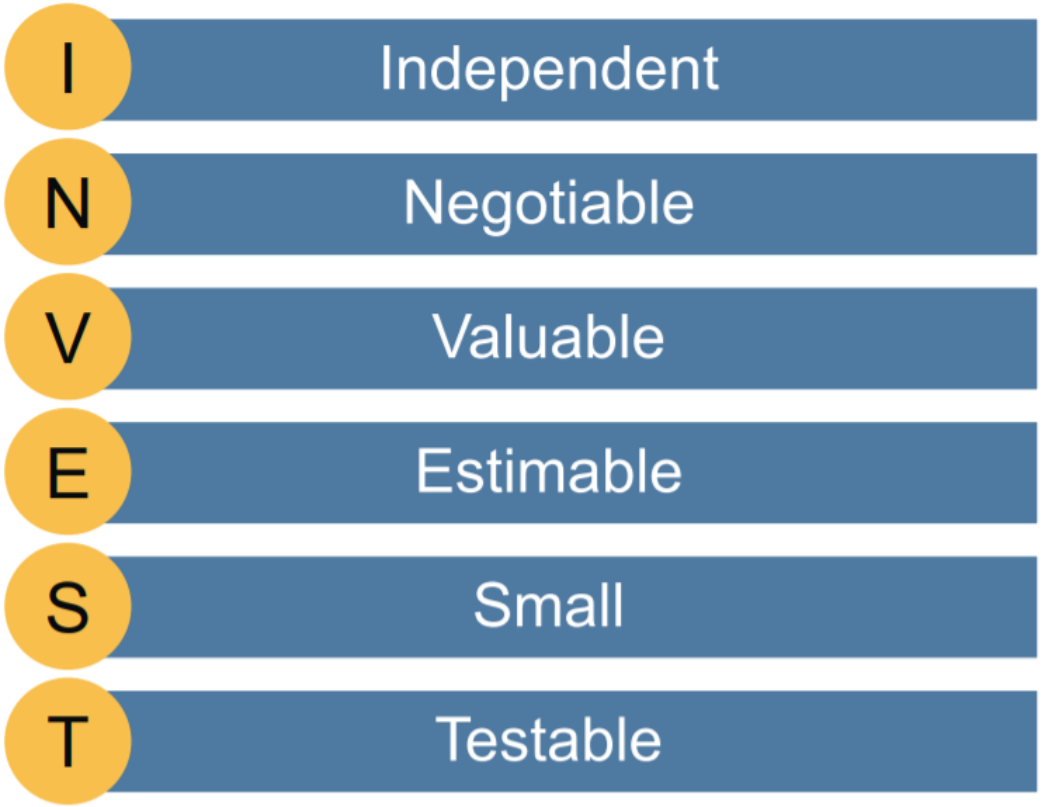
As an app user, **I want** to add profile photos **so that** more people write to me about how awesome I am

As a utility Marketing Director, **I can** present users with new pricing programs **so that** they are more likely to continue purchasing energy from me.

As an administrator, **I can** set the consumer's password security rules **so that** users are required to create and retain secure passwords, keeping the system secure.

As a driver, **I want** to get a receipt after fueling **so that** I can expense the purchase.

Story

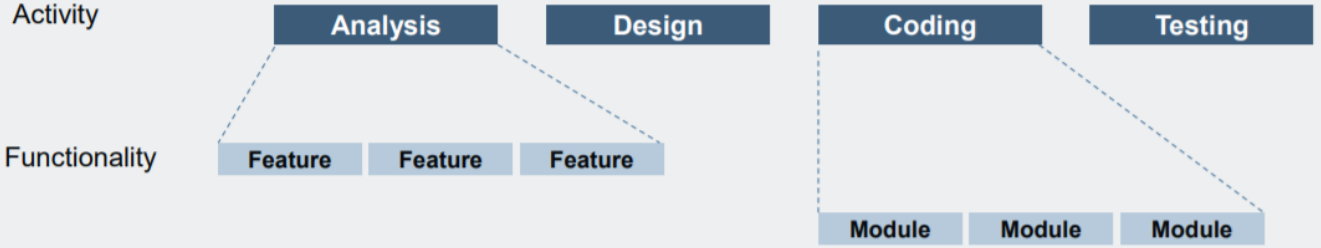

I
Independent
N
Negotiable
V
Valuable
E
Estimable
S
Small
T
Testable

Story

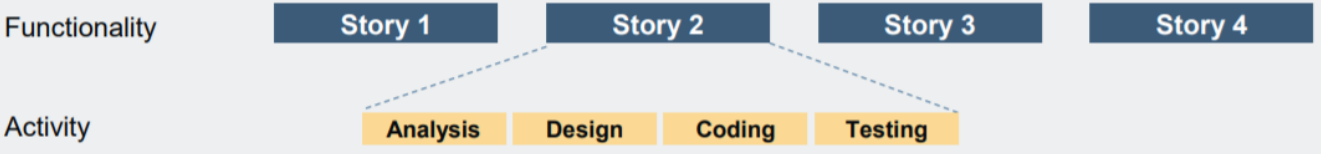
User Stories are **Independent**

User stories define what will be delivered,
not the work steps that will be performed

Do not break stories in a Work Breakdown Structure



A Story includes an entire slice of functionality



Story

User Stories are **Negotiable**

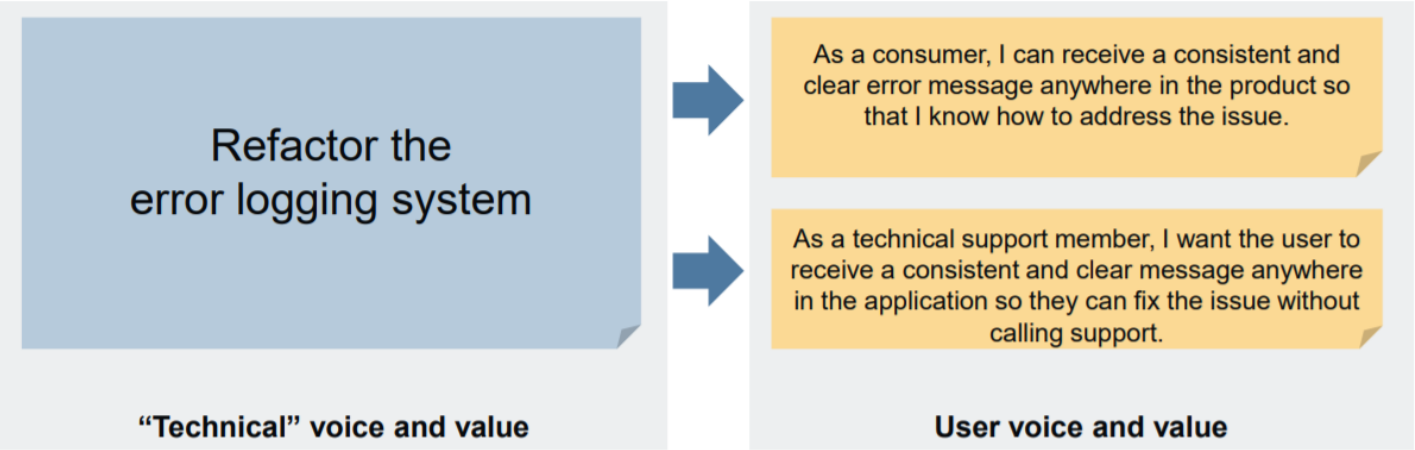
- User Stories are statements of intent, not contracts or detailed requirements
- Too much detail gives impression of false precision or completeness
- Flexibility drives release schedule and goals

A story is not a contract.
A story IS an invitation to a conversation

Story

User Stories are **Valued** by Users

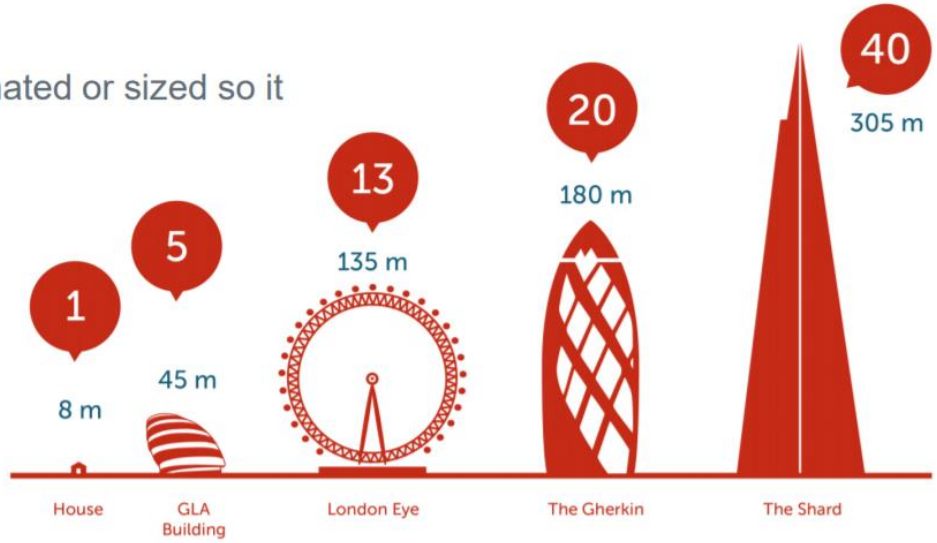
- Write stories in the voice of the customer
- Write for one user



Story

User Stories are **Estimable**

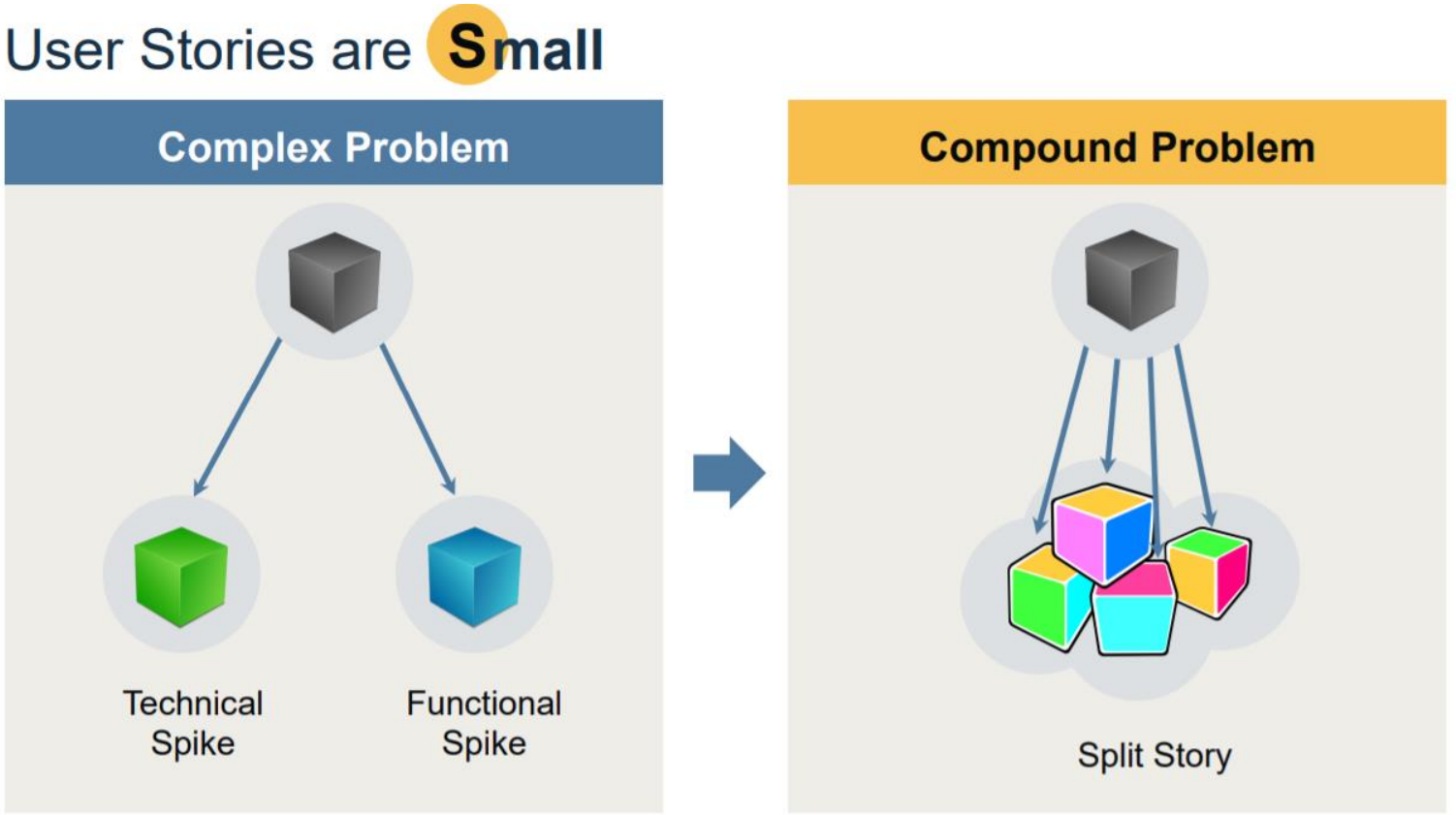
A story has to be able to be estimated or sized so it can be properly prioritized



Estimating may be difficult because …

- Developers lack the **domain knowledge** to know what is to be done
- Developers lack the **technical knowledge** to know how to do something
- The story is too big or vague

Story


User Stories are Small
Complex Problem
Technical Spike
Functional Spike
Compound Problem
Split Story

Story

User Stories are Testable

- Write stories that are testable
- Include Acceptance Criteria for each story

As a power generation company salesperson, I want my search results to return quickly so that I can find relevant contacts for the information I am searching

Not testable

As a power generation company salesperson, I want to receive the first page of search results within 3 seconds so that I can find relevant contacts quickly

Testable

Story

Story writing is a shared responsibility of the team

- Involve users, don't just ask them:
  - Users may not know what they want or need
  - Users can't communicate these needs effectively
- Include the whole team in story writing:
  - Product Owner, developers, testers, doc writers, users, etc.
  - Estimate and prioritize stories later
- Anyone can write stories and acceptance criteria:
  - Otherwise, the product owner will become a bottleneck

Story

Acceptance Criteria

Acceptance Criteria define how the team will know that the intent of a story has been achieved

As a consumer,
I **want** always see current energy pricing reflected on my portal **so that** I know that my energy usage costs are accurate and reflect any utility pricing changes

Acceptance Criteria:

1. The current pricing is always used and the calculated numbers are displayed correctly on the portal (See attachment for format.)
2. The pricing and the calculated numbers are updated correctly when the price changes.
3. The "current price" field itself is updated according the schedule time.
4. The info/error message when there is a fault in the pricing. (See approved error message attached.).

- Cover relevant aspects, including:
  - System behavior.
  - Non-functional requirements.
- Are mainly defined at Sprint planning, but also can be defined before (at Backlog Refinement) and after (within the DBT cycle).
- Serve as a starting point for story acceptance tests.

As a driver, **I want** to get a receipt after fueling **so that** I can expense the purchase
Acceptance Criteria:
When driver asks for the receipt then it is printed and includes: Amount Fueled, Amount Paid, Tax, Vehicle Number, Date and Time

Netcracker
An NEC Company

Definition of Done (DoD)

Acceptance criteria are largely the domain of the Product Owner.
The Definition of Done is owned by the whole team.

Example: a user story is done when ...

- Acceptance criteria met
- Unit tests coded, passed and included in the BVT
- Cumulative unit tests passed
- Coding standards followed
- Code peer reviewed
- Code checked in and merged into mainline
- Story acceptance tests written and passed (automated where practical)
- No must-fix defects
- Nonfunctional requirements (NFRs) met
- Story accepted by the Product Owner

Planning Poker

Planning Poker

Description and Setup

- Estimating poker combines expert opinion, analogy, and disaggregation for quick but reliable estimates
- Participants include all team members
- The product owner participates, but does not estimate
- Each estimator is given a deck of cards with 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞, and ?

Estimating Process

Get Cards → Read Story → Select Card Privately → Turn Cards Over → Discuss → Re-Estimate

- For each backlog item to be estimated, product owner reads the description
- Questions are asked and answered
- Each estimator privately selects a card representing his or her estimate
- All cards are simultaneously turned over so that all participants can see each estimate
- High and low estimators explain their estimates
- After discussion, each estimator re-estimates by selecting a card
- The estimates will likely converge. If not, repeat the process.
- Some amount of preliminary design discussion is appropriate. However, spending too much time on design discussions is often wasted effort.

Planning Poker

Common mistakes made when using Story Points

- Equating Story Points to just complexity, uncertainty or value
- Translating Story Points to hours
- Averaging Story Points
- Adjusting Story Point estimates during sprint
- Adjusting reference PBI's each sprint
- Story Pointing unfinished issues again
- Adjusting Story Point estimate because a specific developer will work on it.
- Never adjusting reference PBI's
- Not discussing incorrectly Story-Pointed issues in retrospective

Let's go!