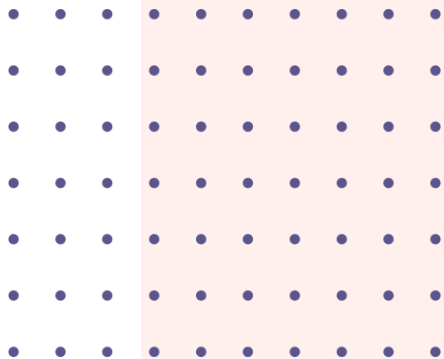


Урок 12. Естимація.

Структура заняття

1. Поняття естимації
2. Основні техніки естимації
3. Poker planning



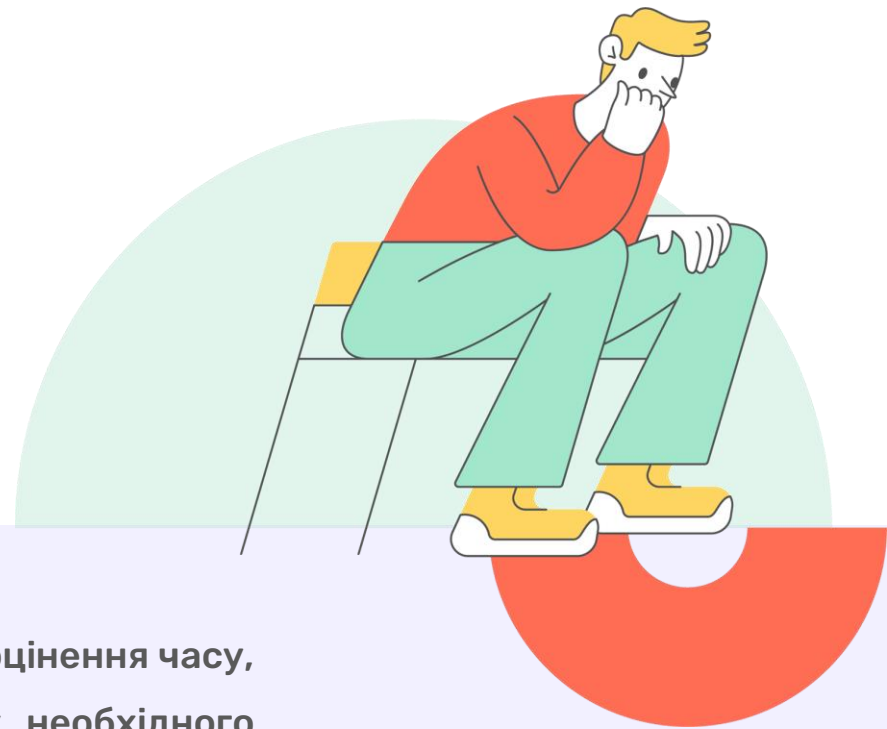


Естимація (estimation)

Оцінка складності задачі, з урахуванням всіх активностей, що необхідні для її завершення.

Для естимації задач тестування необхідно врахувати всі етапи підготовки – перевірка вимог, написання тест-кейсів, підготовка тестових даних, виконання тестів та заведення багів тощо.

Для чого потрібна естимація



Якщо ми говоримо безпосередньо про **QA effort estimation**, то це оцінення часу, який потрібен QA-інженеру для завершення певної задачі, і часу, необхідного для всіх активностей, в яких спеціаліст братиме участь, виконуючи її. Маємо пам'ятати, що естимацію ми зазвичай робимо, коли ще немає повних вимог або вони не погоджені. Тому тут є багато питань, як правильно та ефективно цей час оцінювати.

Для чого потрібна естимація



Розберімося спочатку, навіщо нам взагалі оцінювати.

1. Бо до нас часто приходять проджект-менеджер або бізнес-аналітик чи хтось із команди розробників із питанням «Коли це завдання буде завершено?».
2. На проєкті з'являються нові люди, яких потрібно заонбордити, і ми ще не розуміємо, наскільки швидко спеціаліст працює. І ми теж запитуємо: скільки часу тобі необхідно, щоб протестувати задачу? Нам треба навчити людей правильно оцінювати свої зусилля і планувати час.
3. Найважливіше — оцінювання нам потрібне для того, щоб визначити дедлайни та надати клієнту інформацію про тривалість проєкту.

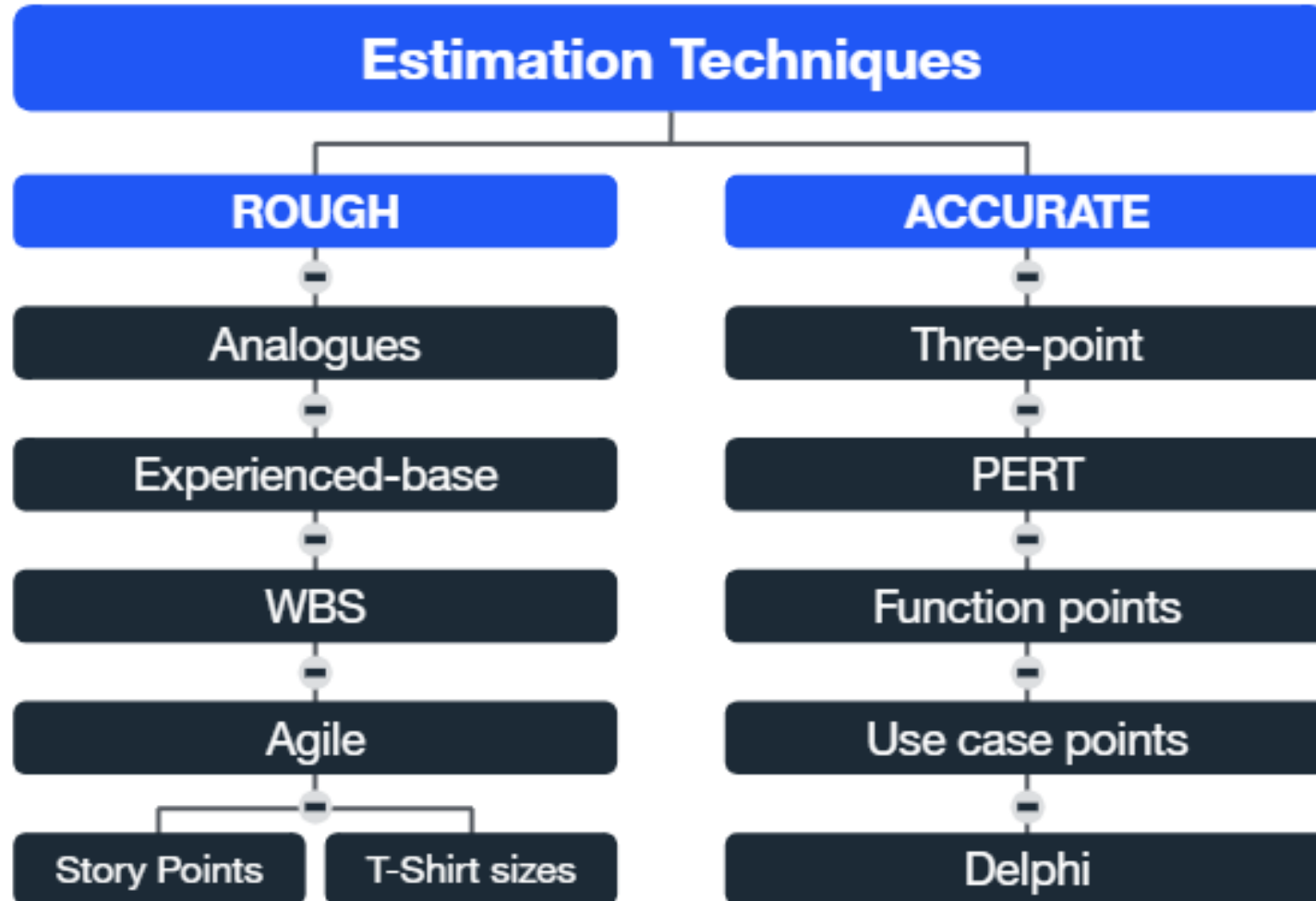
Алгоритм оцінки



1. **Scope estimation** – рахуємо загальний об'єм роботи;
2. **Effort estimation** – знаючи, що потрібно зробити, можна скласти список задач, які потрібно реалізувати;
3. **Schedule estimation** – графік, коли ці роботи можуть бути виконані;
4. **Risk estimates** – оцінюємо ризики;
5. **Resource estimates** – оцінюємо ресурси з урахуванням коли люди підуть у відпустки і є перерви у роботі, коли потрібно збільшувати команду.
6. **Budget estimates** – отримавши попередні оцінки, можна прорахувати бюджет

Техніки оцінювання і коли їх використовувати

Для того щоб проводити оцінювання, нам потрібні певні інструменти – техніки естимації. Глобально їх поділяють на Accurate (точні) та Rough (грубі).



Accurate (точні)

методи використовуються, коли немає досвідчених спеціалістів або потрібно працювати з новою технологією. Або також коли є готове Технічне завдання (специфікація) з усіма вимогами

Three-Point estimation (метод 3-х точок)

методика, яка використовує оптимістичну та песимістичну оцінку для визначення ідеальної оцінки виконання проєктного завдання. На основі цього складають 3 варіанти оцінки (оптимістичний, реалістичний та песимістичний) та використовують формулу:

$$E = (a + m + b) / 3$$

A Function Point (FP) Аналіз функціональних точок

це одиниця вимірювання для вираження обсягу бізнес-функціональності, яку інформаційна система (як продукт) надає користувачеві. FP вимірюють розмір програмного забезпечення. Зазвичай приймає участь вся команда.

Delphi Method (Метод Делфі)

метод структурованої комунікації, який спирається на групу експертів. Експерти відповідають на анкети в два або більше раундів. Після кожного раунду ведучий надає анонімне резюме прогнозів експертів з попереднього раунду з обґрунтуванням своїх суджень. Потім експертам пропонується переглянути свої попередні відповіді у світлі відповідей інших членів групи.



Rough (грубі)

методи використовуються, коли недостатньо інформації про тестований продукт (неповна специфікація або не всі вимоги погоджені, неповні user stories тощо).

Естимація за аналогією

Кожну задачу нового проєкту зіставляємо із задачами аналогічного попереднього по пунктах. Порівнюємо схожість у відсотковому співвідношенні й можемо використовувати ту оцінку, яку дали аналогічному проєкту, з урахуванням відмінностей. Якщо у нас його немає, ця техніка не спрацює.

Experience-Based техніка

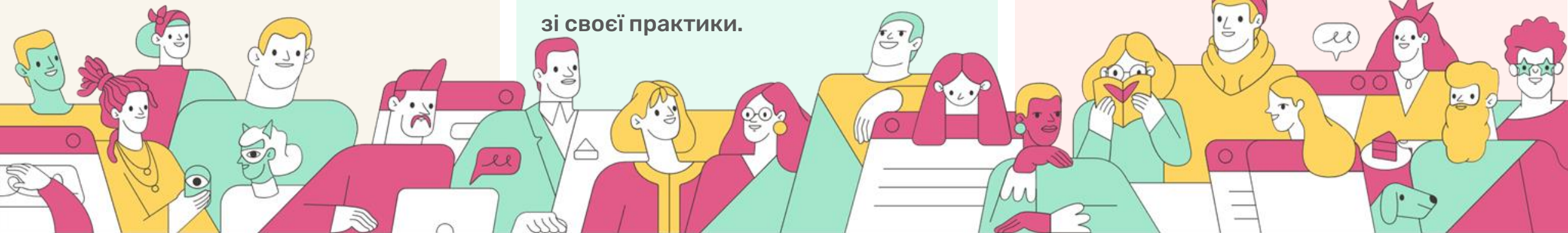
Оцінюємо час на задачі, базуючись на нашому досвіді на попередніх проєктах.

Тут усе залежить від вашого досвіду, та якщо досвіду небагато – в компанії завжди знайдеться досвідчена людина, до якої можна звернутися з питаннями й за експертною оцінкою. Він може допомогти з оцінюванням і розповісти ризики та tricky moments зі своєї практики.

Work Breakdown Structure (WBS)

Це підхід до естимації, суть якого полягає в декомпозиції. Складну задачу ділимо на підзадачі доти, доки не зможемо максимально ефективно і точно її оцінити. Переваги:

- можливість дати більш точну оцінку, адже ми бачимо весь скоуп робіт, і це допомагає при подальшому їх плануванні та виконанні;
- ми майже на 100% можемо бути впевнені, що не пропустили жодної функціональності.



Work Breakdown Structure (WBS) приклад

	A	B	C	D	E	F	G	H
	WBS Level 1:	% of Total		WBS Level 2:	% of Total		WBS Level 3:	% of Total
1							1. Bicycle	
2							1.1 Frame Set	
3							1.1.1 Frame	7
4				1. Bicycle			1.1.2 Handlebar	2
5				1.1 Frame Set	15		1.1.3 Fork	3
6				1.2 Crank Set	5		1.1.4 Seat	3
7				1.3 Wheels	30		1.2 Crank Set	5
8	1. Bicycle	100		1.4 Braking System	5		1.3 Wheels	
9				1.5 Shifting System	5		1.3.1 Front Wheel	13
10				1.6 Integration	35		1.3.2 Rear Wheel	17
11				1.7 Project Mgt	5		1.4 Braking System	5
12					100		1.5 Shifting System	5
13							1.6 Integration	
14							1.6.1 Concept	3
15							1.6.2 Design	5
16							1.6.3 Assembly	10
17							1.6.4 Testing	17
18							1.7 Project Mgt	5
19								100
20								

Agile estimation techniques

Story points техніка

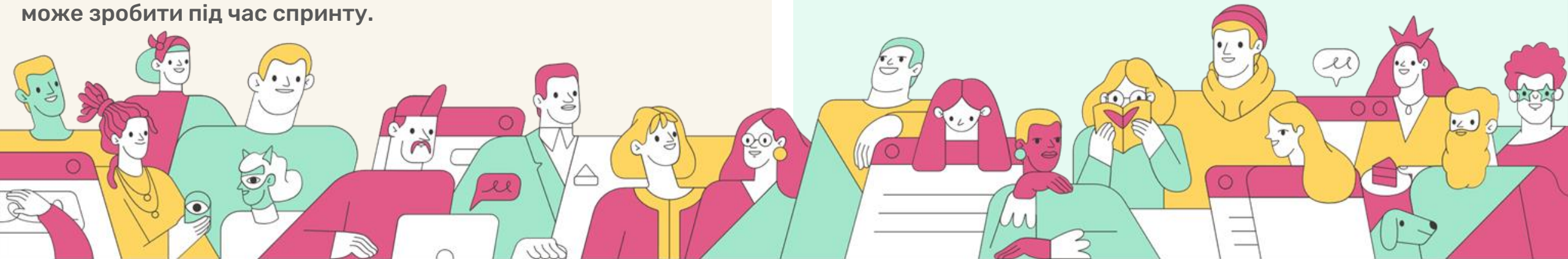
Техніка оцінює не час на виконання задачі, а її complexity, наскільки вона складна. Ми беремо за взірець маленьку юзер-сторі, якій присвоюємо один сторі-поінт. Відповідно до цього за складністю естимуємо наступні задачі.

У проєктах, де використовується ця техніка, команда має своє velocity — це та кількість сторі-поінтів, які команда може виконати за один спринт. Це значення має базуватися на результаті кількох спринтів. Наприклад, ми заестимували чотири User Stories, і загальна кількість Story points для QA вийшла 24, але під час спринту ми змогли виконати тільки три User Stories. У такому разі на наступний спринт у Sprint Backlog мають плануватися User Stories у тій кількості, яку завершили в попередньому. За результатом кількох спринтів ми матимемо velocity: ту кількість Story points, яку команда може зробити під час спринту.

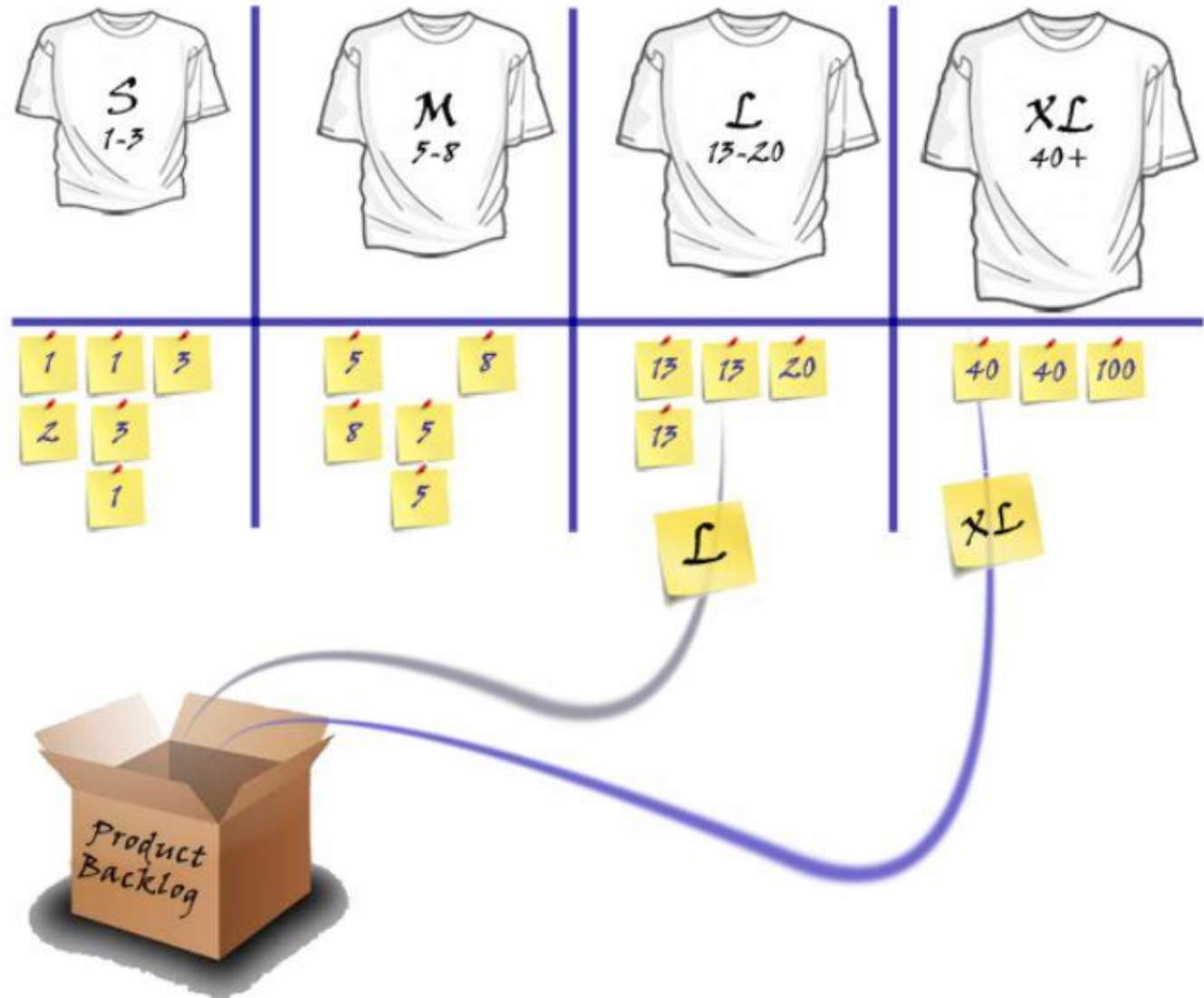
T-Shirt sizes

Використовують для оцінювання великих беклогів наперед. Ми отримуємо невелику юзер-сторі та визначаємо її «розмір» як size S. Що означає S? Це знов-таки складність. Ми присвоюємо S і відповідно до складності цієї юзер-сторі естимуємо всі наступні й теж присвоюємо їм «розміри. Після оцінювання маємо визначену складність задач, які надалі можемо планувати в спринт-беклог згідно з пріоритетами.

Є два підходи, як потім використовувати складність. Можна зробити співвідношення, скільком годинам дорівнюватиме S. Або вираховувати velocity на основі кількості stories різних розмірів, які було взято в спринт. Наприклад, 10 (S) + 12 (M) + 15 (L) = 37. Також можна вираховувати velocity у сторі-поінтах, присвоюючи кожному розміру свою кількість story points: S = 1–3 story points, і вже від цього відштовхуватися.



T-Shirt sizes

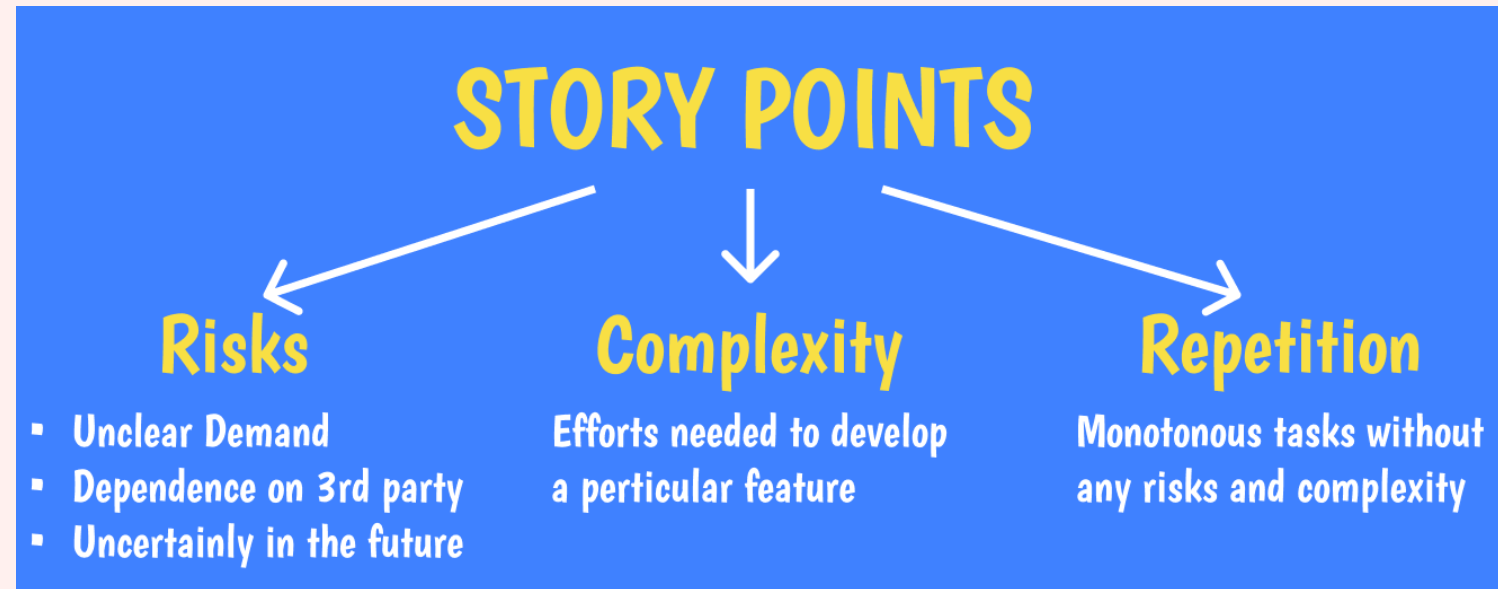


Story points





Спосіб оцінити кількість зусиль, необхідних для завершення user story у вашому product backlog.

Як правило вони враховують 3 фактори, які можуть вплинути на обсяг завдання та зусилля, і цінність story point відповідно зростає. Оскільки story points є відносними, ви знайдете їх цінність (value), взявши до уваги ці деталі та порівнявши схожі завдання між собою.

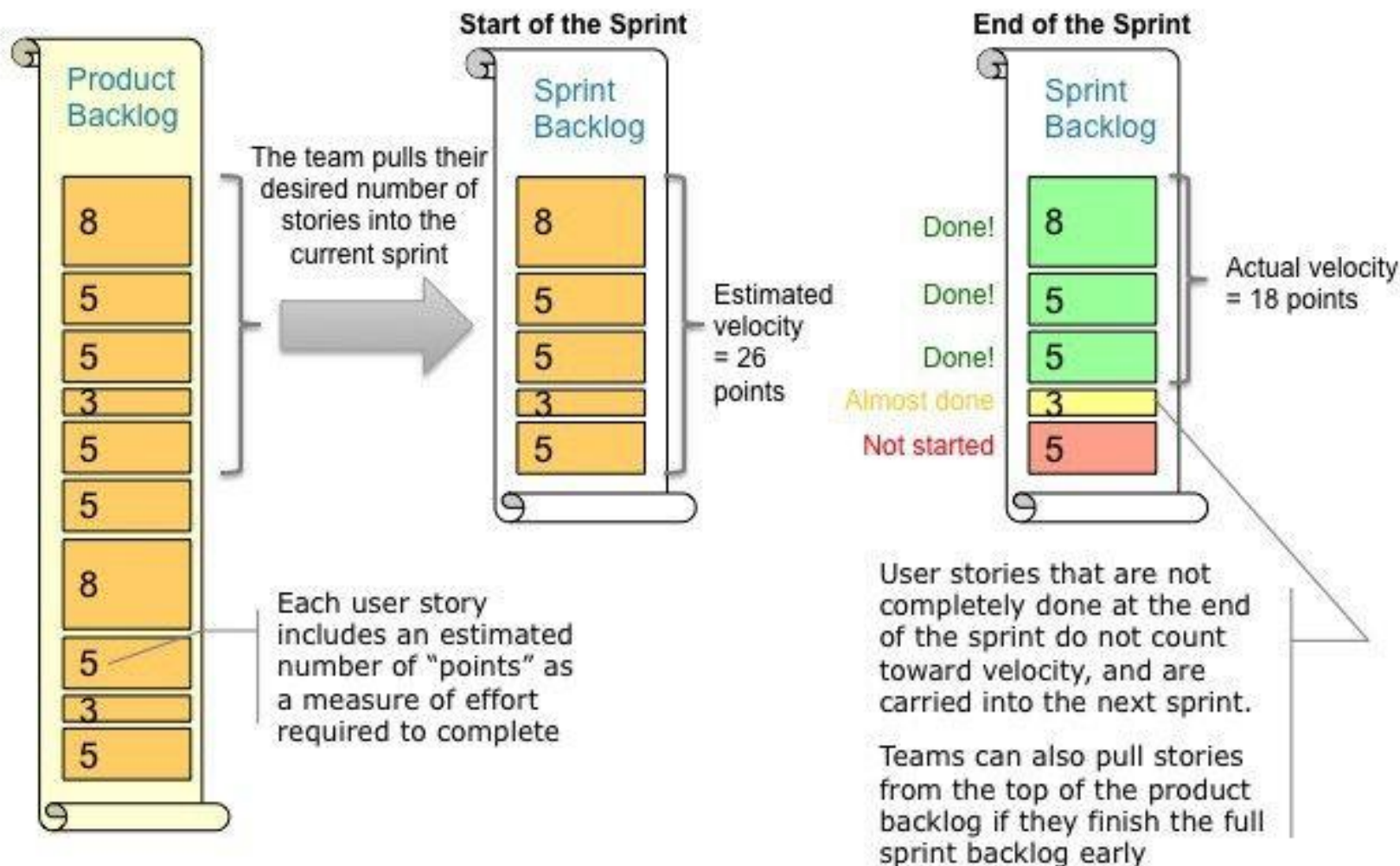
- Ризик (risk) – кількість загального ризику або невизначеності, пов'язанні із завданням (якщо завдання включає треті сторони, підрядників або зацікавлених сторін – рівень ризику збільшується)
- Повторення (repetition) – це досвід команди з подібними завданнями
- Складність (complexity) – це рівень складності завдання та наскільки чіткі цілі завдання.



Story points

Story Points	Work	Risk*	Dependencies**	Unknowns
0.5 ⁺	Piece of cake	-	-	-
1 ⁺	No-brainer	-	-	-
2	Some	Little	Could be	-
3	Much	Some	Some	Could be
5	A lot	Much	Many	Some
8 → Split/Spark	Huge	A lot	A lot	Many
13 → Epic				

"Velocity" is the Key Metric in Scrum



Poker planning

Ігровий метод для оцінки складності виконання задачі в розробці програмного забезпечення.

1. Щоб розпочати сеанс планування покеру, власник продукту або клієнт читає історію користувача agile або описує функцію для оцінювачів.
2. Кожен оцінювач тримає колоду карток «Планування покеру» зі значеннями на кшталт 0, 1, 2, 3, 5, 8, 13, 20, 40 і 100, і це послідовність, яку ми рекомендуємо. Значення представляють кількість очок історії, ідеальних днів або інших одиниць, у яких команда оцінює.
3. Оцінювачі обговорюють функцію, за потреби ставлячи запитання власнику продукту.
4. Після повного обговорення функції кожен оцінювач приватно вибирає одну картку, яка представлятиме його чи її оцінку.
5. Тоді всі карти відкриваються одночасно.
6. Якщо всі оцінювачі вибрали однакове значення, воно стає оцінкою. Якщо ні, оцінювачі обговорюють свої оцінки. Високі та низькі оцінювачі особливо повинні поділитися своїми причинами.
7. Після подальшого обговорення кожен оцінювач повторно вибирає картку оцінки, і всі картки знову відкриваються одночасно.





Домашнє завдання