universidade de aveiro
deti departamento de eletrónica, telecomunicações e informática

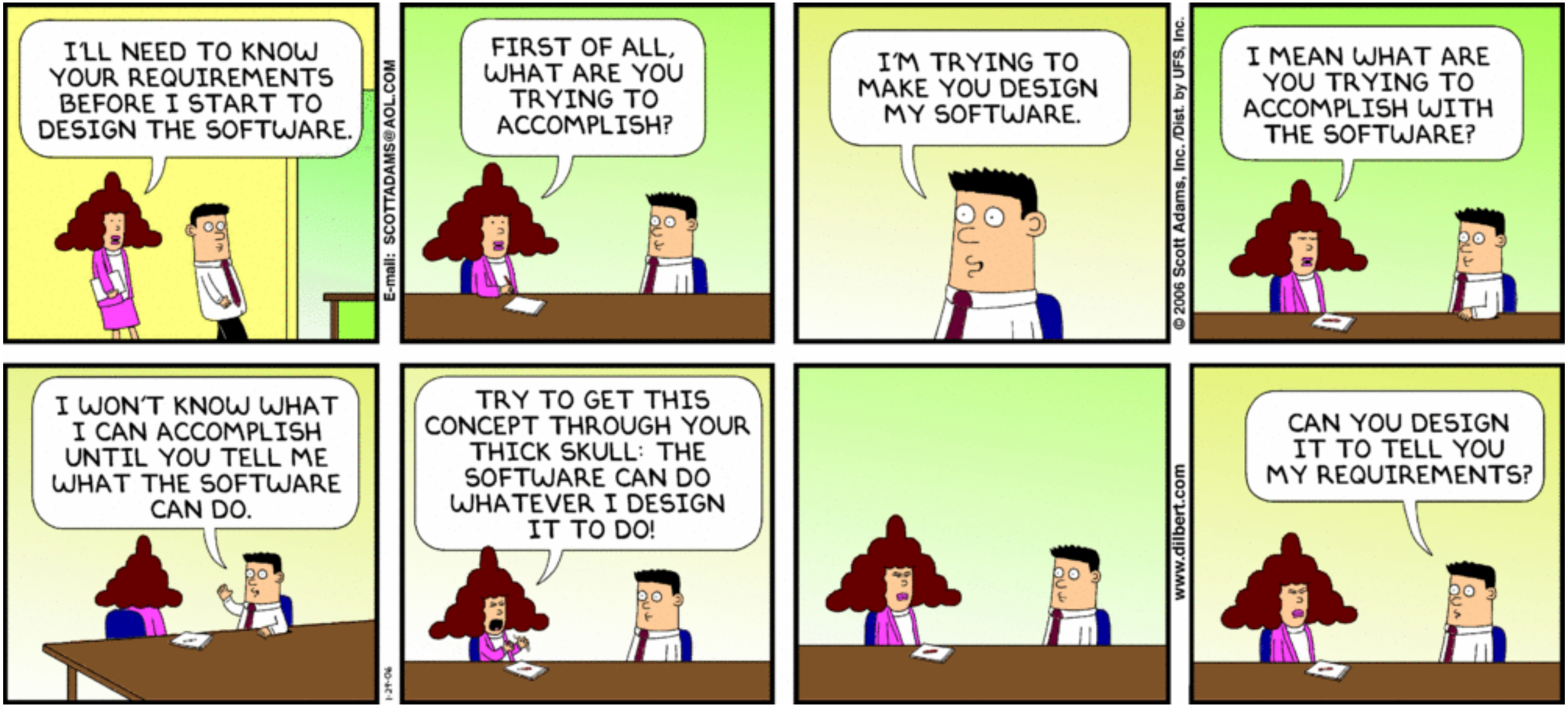
Requirements planning in the DevOps cycle

Software Engineering

2025/2026

Summary

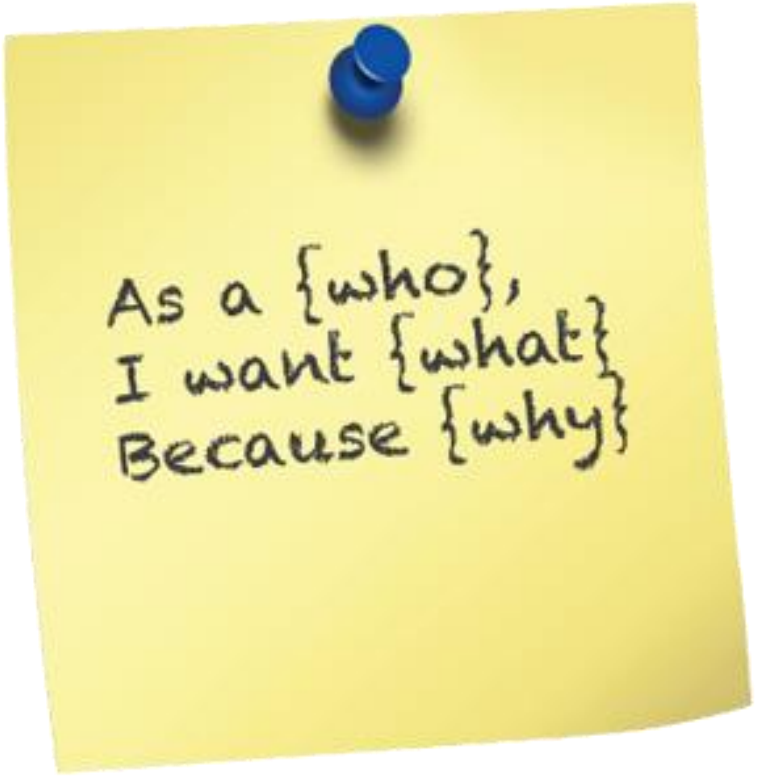
- User stories
  - Good practices
  - Acceptance criteria
  - Epics
  - Initiatives
- Estimation
  - Story points
  - Poker planning
  - Story points vs. hours
- Requirements
  - Functional
  - Non-functional


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc. /Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
1-31-06
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

User stories

User story

- A user story is a key tool in agile software development that captures a feature from the end-user's perspective.
- It identifies **who** the user is, **what** they want, and **why** they need it.
- By focusing on the user's needs, a user story provides a clear and simplified description of a requirement.



User story

Definition

- A user story often follows the following “equation”:

  As a ‹type of user>, I want < some feature> so that ‹reason>

- A simple example of this could be:

  As a **customer**, I want **to add an item to my cart**,

  so that **I can purchase it**

User story

Template

WHO are we creating it for? Who is the user?	**As a <type of user>**
WHAT are we creating? What is the intention?	**I want <some objective>**
WHY are we creating it? What is the value for the customer?	**So that <benefits>**

User story

Checklist

- ❑ Keep it short
- ❑ Keep it simple
- ❑ Write from the user's perspective
- ❑ Highlight the value or benefit
- ❑ Focus on a single functionality (split if needed)
- ❑ Write stories collaboratively
- ❑ Define acceptance criteria to outline the MVP

Acceptance criteria

- Acceptance criteria (or "conditions of satisfaction") define the detailed scope of a user's requirements.
  - They clarify the value of the story, align the team on expectations, and establish when the work can be considered complete.

- Goals
  - Clarify what the team should build before starting work
  - Ensure a shared understanding of the problem
  - Define when the story is complete
  - Enable verification through automated tests

Acceptance criteria

Example

- User Story:
  - As a registered user, I want to reset my password so that I can regain access to my account.

- Acceptance Criteria:
  - User receives a password reset link via email
  - The link expires after 24 hours
  - The new password must meet security requirements
  - A confirmation message is shown after successful reset

It can be seen as the "Definition of Done" within Agile methodology.

Acceptance criteria

Template

- Given–When–Then (BDD style)
  - **Given** some initial context
  - **When** the user takes an action
  - **Then** the expected outcome occurs

- Example:
  - Given a registered user requests a password reset,
  - When they submit their email address on the “Forgot Password” page,
  - Then they receive a password reset link via email.

Acceptance criteria

Template

- Given–When–Then (BDD style)
  - **Given** some initial context
  - **When** the user takes an action
  - **Then** the expected outcome occurs

- Checklist style:

  User receives a password reset link via email

VS

- Example:
  - Given a registered user requests a password reset,
  - When they submit their email address on the “Forgot Password” page,
  - Then they receive a password reset link via email.

- The Given–When–Then style works well for automated tests
  - while the **checklist style** is simple and quick for teams.

Acceptance criteria

What your checklist should cover

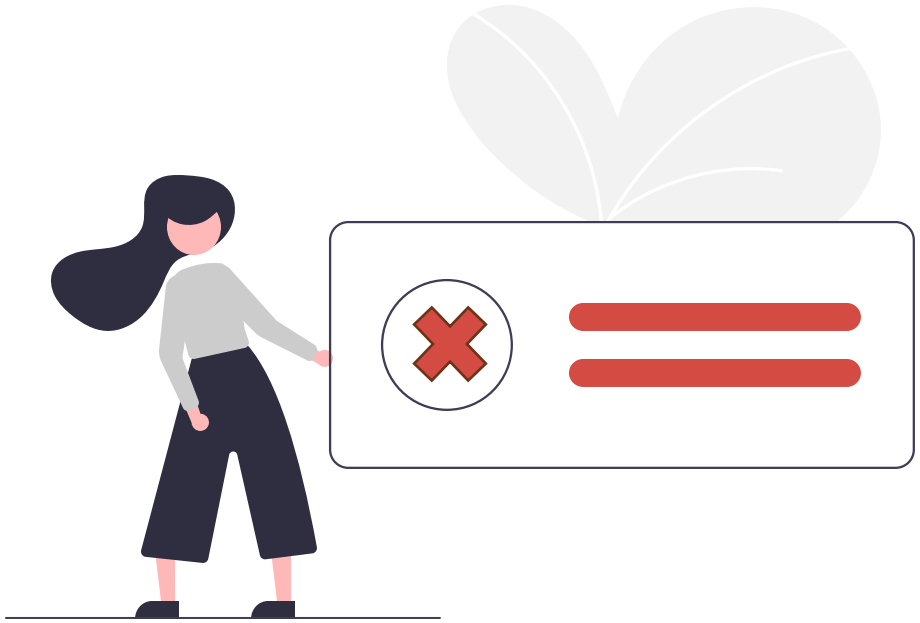
- ❑ Negative scenarios of the functionality
- ❑ Functional and non-functional use cases
- ❑ Performance concerns and guidelines
- ❑ What system/feature intends to do
- ❑ End-to-user flow
- ❑ The impact of a user story to other features
- ❑ UX concerns

Acceptance criteria

What your checklist should not include

- ❑ Code review was done
- ❑ Non-blocker or major issues
- ❑ Performance testing performed
- ❑ Acceptance and functional testing done

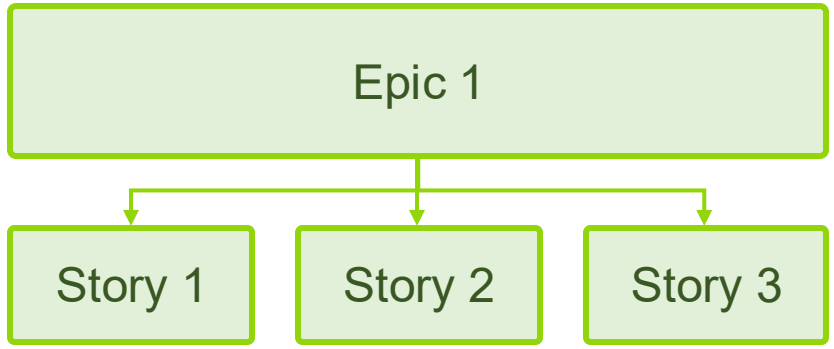
- ➢ The team should already share a clear understanding of the DoD



Epics

Definition

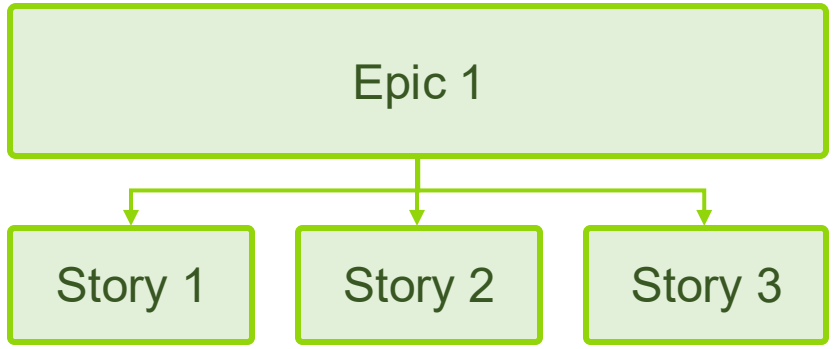
- A large user story that can’t be completed in a single sprint.
- Represents a significant feature or initiative.
- Can be broken down into smaller, manageable user stories.



Epics

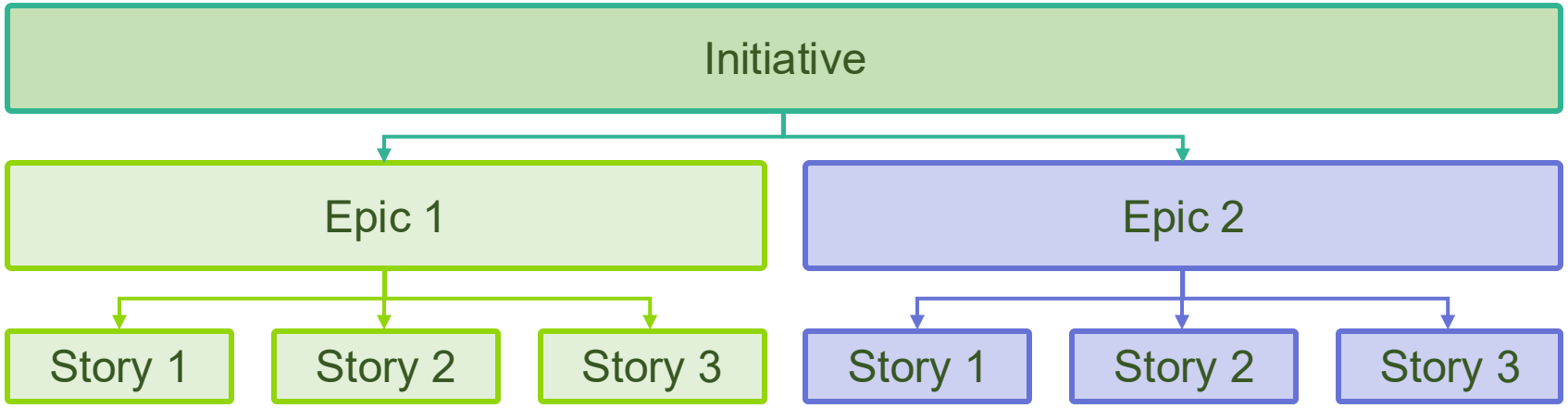
Characteristics

- Covers a broad scope and multiple teams or sprints.
- Focuses on a bigger business goal rather than technical details.
- Serves as a container for related user stories.
- Managing epics
  - Break epics into smaller stories for easier planning.
  - Adjust scope as priorities and requirements evolve.



Initiatives

- A high-level objective that aligns with strategic business goals.
- Larger than an epic and often spans multiple epics.
  - Focuses on outcomes, not just features.
- Drives multiple epics across teams or departments.



Estimation

Estimation

- Estimation is a forecast — much like predicting the weather.
  - It can be right or wrong due to both *known unknowns* and *unknown unknowns*.

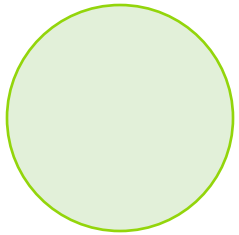
Estimation

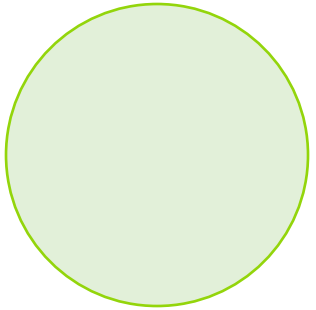
Challenges

- Humans often struggle to accurately estimate values.

- We often confuse an item's proportional worth with its exact worth.
  - While a 50% discount feels significant, the absolute value of the item might still be more than you should pay.

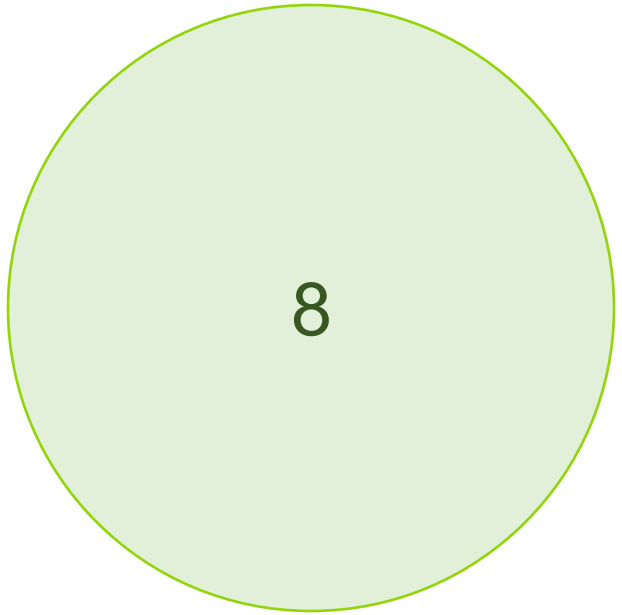
- We struggle to estimate the value of something we have little to no experience with.
  - A seasoned contractor can give a far more accurate estimate for a renovation project than a homeowner who has never done one before.






3
4


3


8

Story points

- A Story Point (SP) is a relative unit of measure.
  - Used by Scrum teams to estimate the effort required to complete a requirement.
- Estimates are not absolute time but relative comparisons (e.g., one story is twice as complex as another).
  - Teams define their own scale (e.g., Fibonacci sequence: 1, 2, 3, 5, 8, …).
- Helps in planning, forecasting, and measuring team velocity.

Planning poker

- Consensus-based estimating technique used in Agile teams.
  - Can be applied with Story Points, Ideal Days, or any other estimating unit.

- Process:
  1. Product Owner presents the story.
  2. Each team member selects a card privately.
  3. All cards are revealed simultaneously.

Planning poker

- It's crucial that everyone participates in the estimation process.
  - Developers, designers, testers, deployers
  - Estimates are most accurate when made by the people who will actually do the work.
- Discuss differences → repeat until consensus is reached.
- Encourages collaboration, shared understanding, and avoids anchoring bias.

Story points vs. hours

- Developers feel pressure to match the estimated hours exactly.
- Exceeding the estimate may suggest poor performance.
- Finishing under the estimate may indicate a flawed original estimate.
- Advantages of story points:
  - Estimates are independent of the estimator's skills or experience.
  - Team velocity is tracked over time.
  - No need to re-estimate if the team's capacity changes.

Requirements

Requirements

What is a requirement?

- Detailed specifications that describe what the system must do
- Can be functional (features, behaviour) or non-functional (performance, security)
- Serve as input for design, development, and testing
- Stories are important!
  - But they are too high-level for developers to implement directly

Requirements

Requirements of requirements

- Clear and Unambiguous
  - Every requirement should have a single, precise meaning.
- Testable
  - Must be possible to verify through inspection, demonstration, or testing.
- Complete
  - Covers all necessary functionality and constraints.
- Consistent
  - Does not conflict with other requirements.
- Feasible
  - Can realistically be implemented within time, budget, and technology constraints.
- Traceable
  - Linked to its source, e.g., user story, epic, or business objective.

Requirements

Functional Requirements (FR)

- These define specific actions or behaviours of a system.
  - They directly support business processes and user tasks.

- Examples:
  - System logic:
    - The system shall calculate the tax based on the user's location and the current tax rate.
  - User interaction:
    - A user shall be able to log in using a unique email and password.
  - Data handling:
    - The system shall store a complete history of all user orders for a minimum of 7 years.
  - API/Integration:
    - The system shall expose a RESTful API endpoint to retrieve product data.

Requirements

Functional Requirements (FR)

- These define specific actions or behaviours of a system.
  - They directly support business processes and user tasks.

- Examples:
  - System logic:
    - The system shall calculate the tax based on the user's location and the current tax rate.
  - User interaction:
    - A user shall be able to log in using a unique email and password.
  - Data handling:
    - The system shall store a complete history of all user orders for a minimum of 7 years.
  - API/Integration:
    - The system shall expose a RESTful API endpoint to retrieve product data.

Requirements

Non-Functional Requirements (NFR)

- Non-Functional Requirements **directly impact architectural decisions**, technology stack choices, and testing strategies.

→ The system shall support scalability

Or

→ The system shall be secure

These are not

Non-Functional Requirements

Requirements

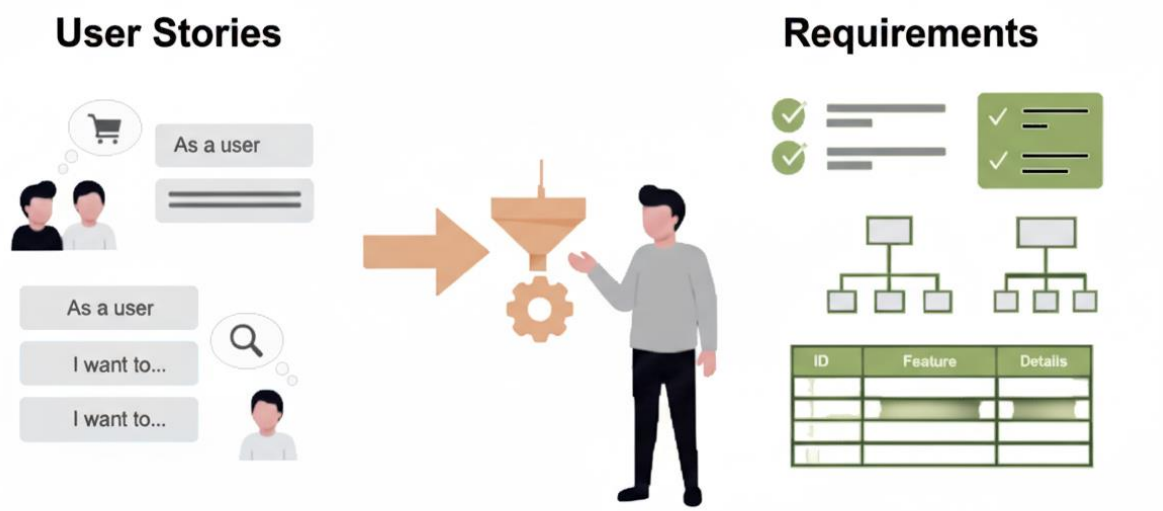
Non-Functional Requirements (NFR)

- Scalability:
  - The system shall support 10,000 concurrent users without a noticeable degradation in performance.
- Performance:
  - The user's dashboard shall load in less than 2 seconds.
- Security:
  - All user authentication data shall be encrypted at rest and in transit using industry-standard protocols.
- Maintainability:
  - The codebase shall adhere to a clean architecture and be easily extensible by new team members.
- Reliability:
  - The system shall have an uptime of 99.99%.
- Usability:
  - The checkout process shall require a maximum of three clicks.

Requirements

Convert user stories into requirements

- Review the user story and its context
- Identify functional needs
  - What the system must do
- Identify non-functional needs
  - Performance, security, usability
- **Define acceptance criteria**
- Document in a testable format



Requirements

Convert user stories into requirements

- User Story:

  As a registered user,
  I want to update my profile information,
  so that my details are always current.

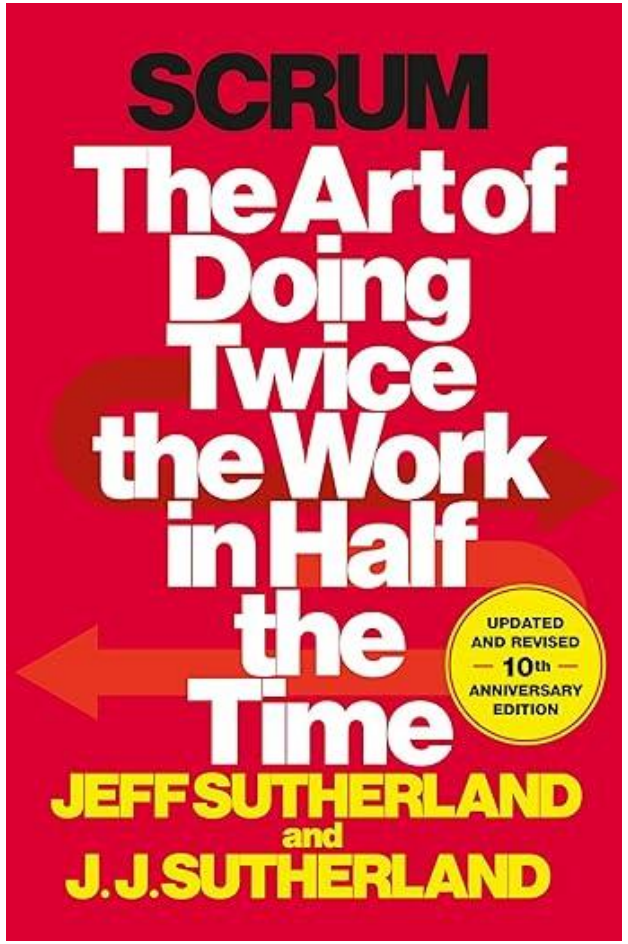
- Step 1: Breakdown into Functional Requirements (FRs)
  - FR-1: The user shall be able to view their current profile information on a dedicated page.
  - FR-2: The system shall provide input fields for the user's name, email, and password.
  - FR-3: The user shall be able to save changes to their profile by clicking a "Save" button.
  - FR-4: The system shall validate all input fields upon submission and display an error message for invalid data.
  - FR-5: The system shall display a success message upon successful update of the profile.

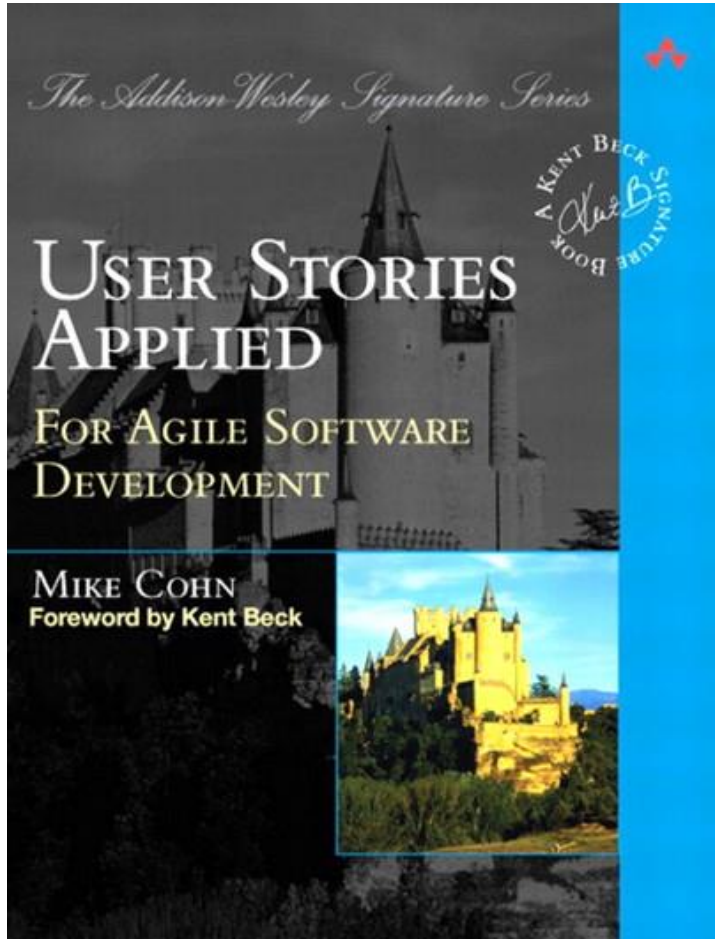
Requirements

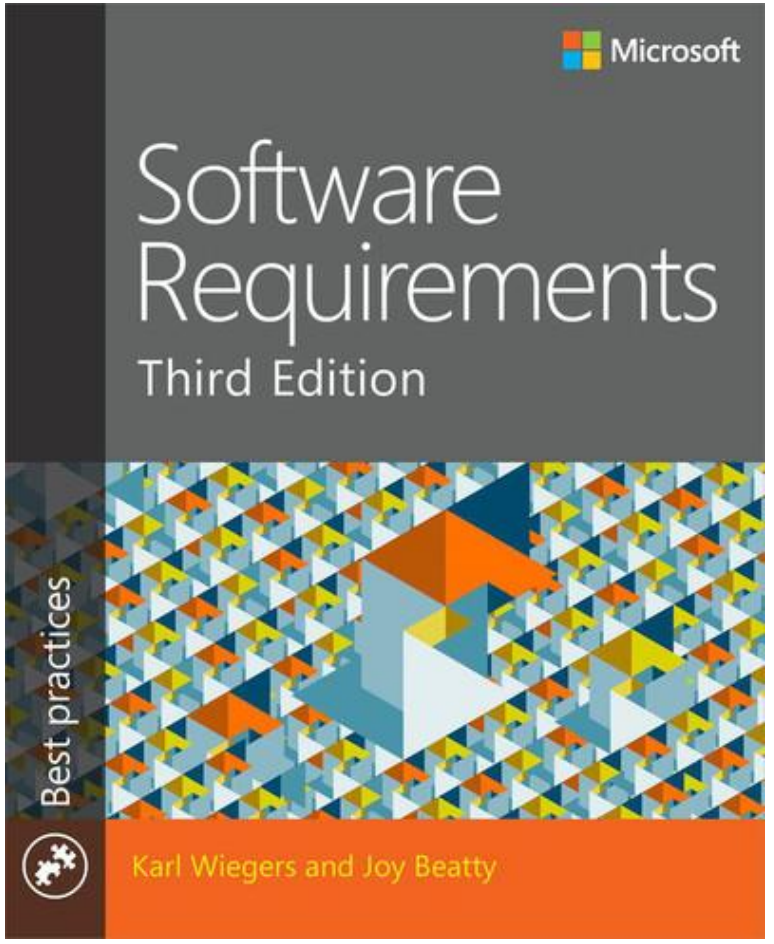
Convert user stories into requirements

- Step 2: Identify Non-Functional Requirements (NFRs)
  - NFR-1 (Security): Password updates shall require the user to re-enter their current password for verification.
  - NFR-2 (Performance): The profile page shall load in less than 1 second. The profile update operation shall complete in under 500ms.
  - NFR-3 (Usability): Input fields shall be clearly labelled and provide clear instructions.

References


SCRUM
The Art of Doing Twice the Work in Half the Time
UPDATED AND REVISED
10th
ANNIVERSARY EDITION
JEFF SUTHERLAND
and
J. J. SUTHERLAND


The Addison Wesley Signature Series
A KENT BECK SIGNATURE BOOK
USER STORIES APPLIED
FOR AGILE SOFTWARE DEVELOPMENT
MIKE COHN
Foreword by Kent Beck


Microsoft
Software Requirements
Third Edition
Best practices
Karl Wiegers and Joy Beatty