

CSE608 L3
Agile Requirements
Techniques:
Delivering Value Quickly

Agenda

- Understand why **Stories** are used for requirements in Agile.
- Appreciate the discipline needed to effectively **refine** Stories based on their priority.
- Understand how to **decompose** Stories into finer grain work items.
- Learn how to slice stories **vertically** through the software architectural layers to provide increments of value.
- Learn how to capture **nonfunctional** requirements via Stories.
- Learn what the different types of Stories there are in order to capture different user needs.
- Get exposed to the concept of a Minimal Viable Product and why it accelerates value delivery and learning.

Requirements

50%

- What percentage of overall **project time** is spent gathering, elaborating, and communicating product requirements?

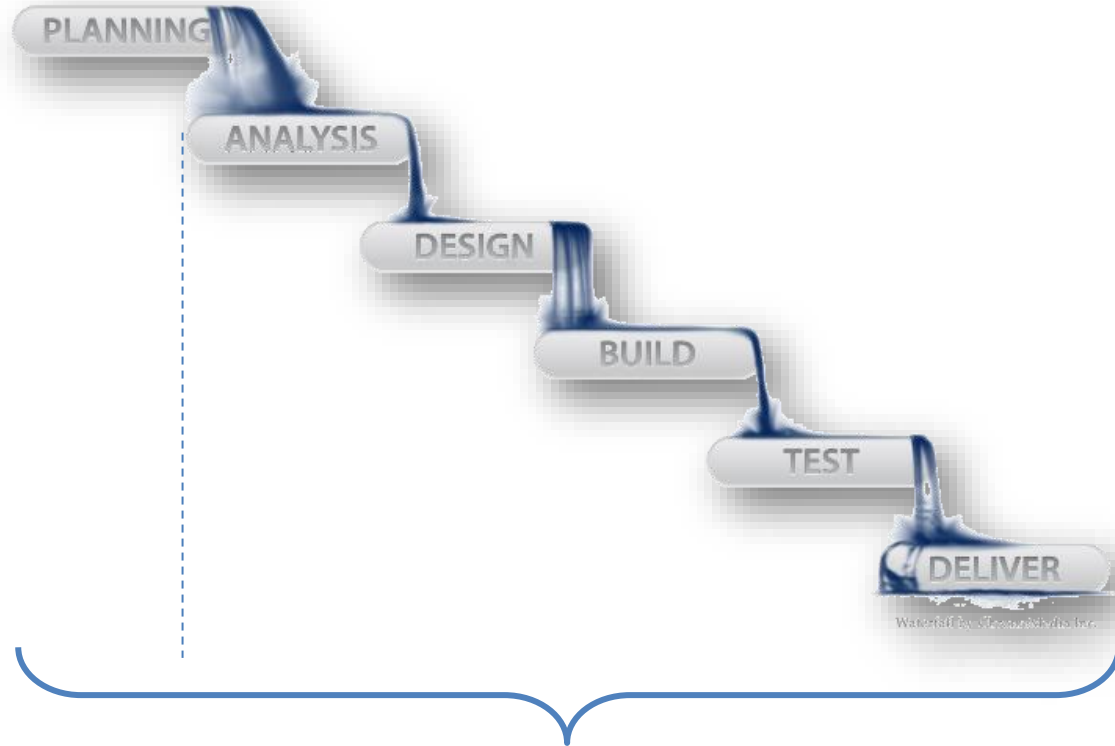
35%

- What percentage of **requirements**, as originally defined, change during the course of the project?

45%

- What percentage of features, as ultimately delivered, are rarely or **never used** by the product's end-users?

Delivering Value Sooner



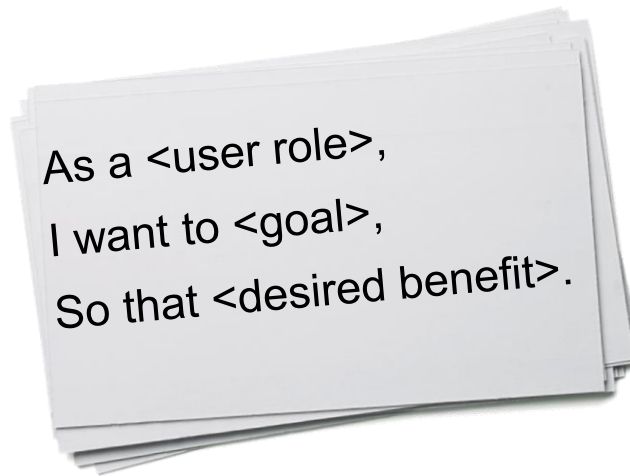
The 5 Levels of Planning

The 5 Levels of Planning



What is a User Story

A brief, simple requirement statement from a user's perspective.

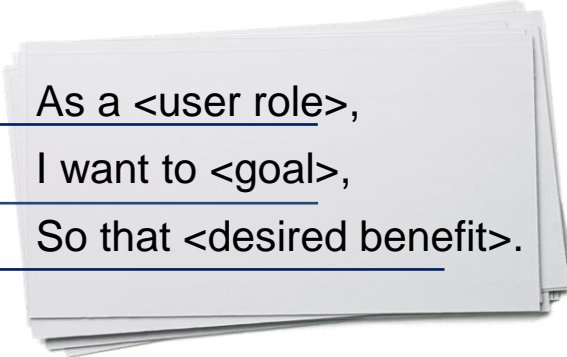



As a <user role>,
I want to <goal>,
So that <desired benefit>.



Acceptance
Criteria

Parts of a User Story

- 
- 1 Who As a <user role>,
 - 2 What I want to <goal>,
 - 3 Why So that <desired benefit>.



4 Acceptance
Criteria

Information Associated with a Story

Title: a
conversation
starter

Title: Traveller wants to book a trip so that they can go to their destination

Body

Story points : 3

Assigned to : Tom

Acceptance tests:

1. User can edit an airline booking
2. User can edit a car rental booking
3. User can edit a hotel booking
4. User can start editing from a screen that shows a booking

Other information:

- Attachments
- Screenshots
- UML
- Discussion
- Non-goals
- Additional details

Benefits of User Stories

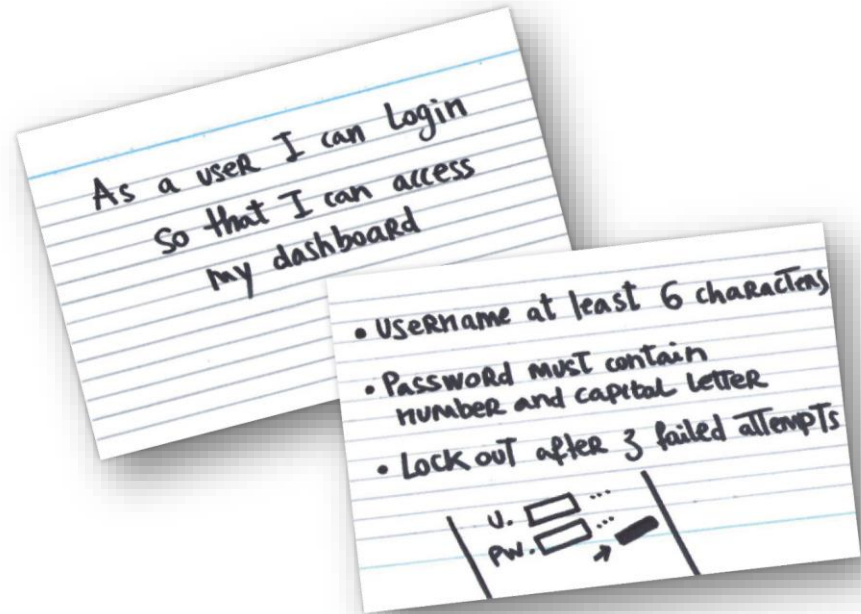
- User Stories emphasize verbal rather than written communication.
- User Stories are comprehensible by both customers and developers, encouraging greater levels of customer participation.
- User Stories are the right size for planning.
- User Stories are well suited for iterative development.
- User Stories force requirements validation by stating both WHO will use a feature & WHY it is desired.

What Are User Stories Used For?

- User stories are the basis for all work
- All development work should be based on user stories, no matter where those developers physically sit
- All test plans should be based around user stories, no matter who is doing the testing
- User stories may be insufficient for documenting the current architecture or functioning of the system
 - In this case, documentation for this specific purpose may be used
 - This documentation should not be used as the basis for development and testing, only as reference material or for auditing purposes

Acceptance Criteria

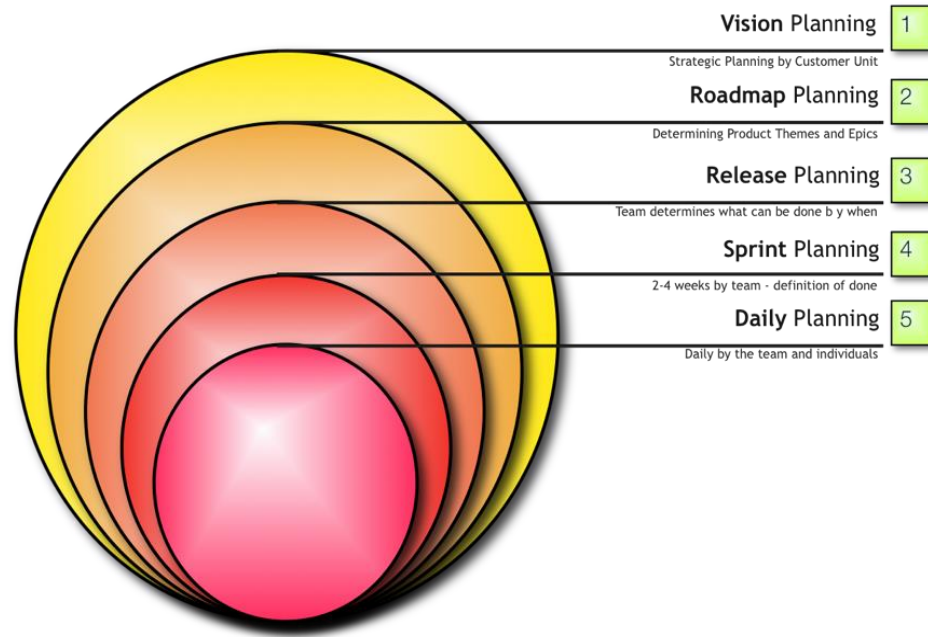
- **Acceptance Criteria as “Conditions that a software product must satisfy to be accepted by a user, customer or other stakeholder.”**
- Acceptance Criteria are a set of statements, each with a clear pass/fail result, that specify both functional (e.g., minimal marketable functionality) and non-functional (e.g., minimal quality) requirements



The 3 C's of a User Story

1. **The Card** - A 3x5 index card forces brevity. Only capture the topic of the item, a high level description of the desired system behavior, and why it is important.
2. **The Conversation** - A User Story is not enough. Consider it a placeholder for conversation. Detailed requirements are only discovered once the story has been targeted for a sprint.
3. **The Confirmation** - On the back of the card capture Acceptance Criteria. They outline specifications from the Product Owner and will allow the team build functionality for acceptance.

Where User Stories Fit In



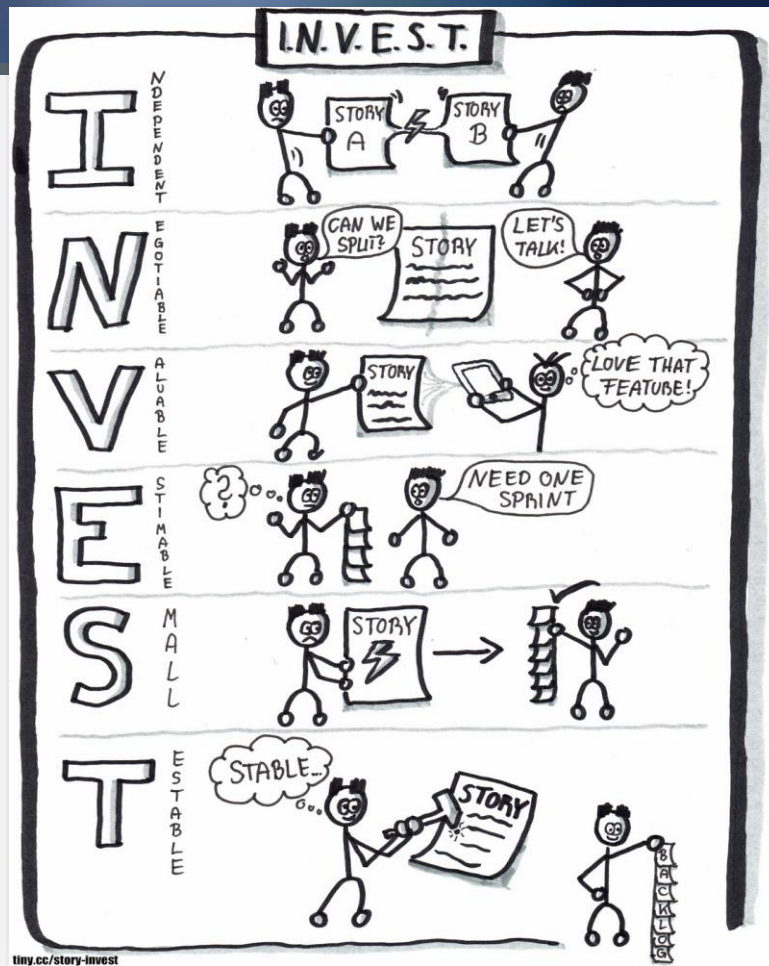
← **Must** have User Stories written, estimated & prioritized **prior to** Release Planning.

Product Backlog

- A prioritized list of all user stories that may be delivered
- New items can be added at any time to the Product Backlog
- Items are defined and prioritized by Product Owner with input from others
- Team members estimate items in Product Backlog relative to each other using predetermined scale (story points)

I.N.V.E.S.T.ing

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

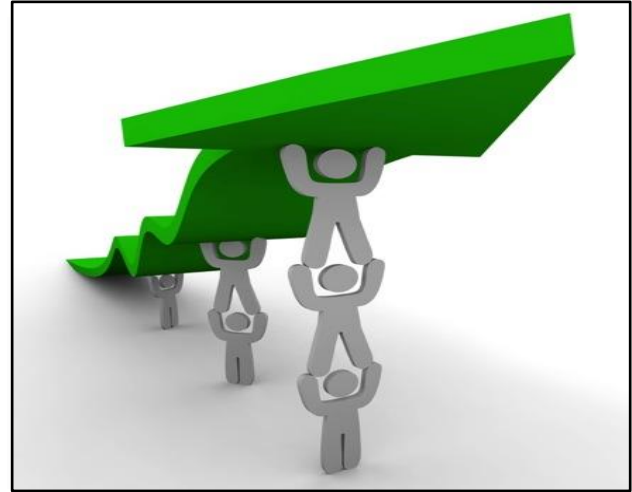


Who writes stories?

Everyone.

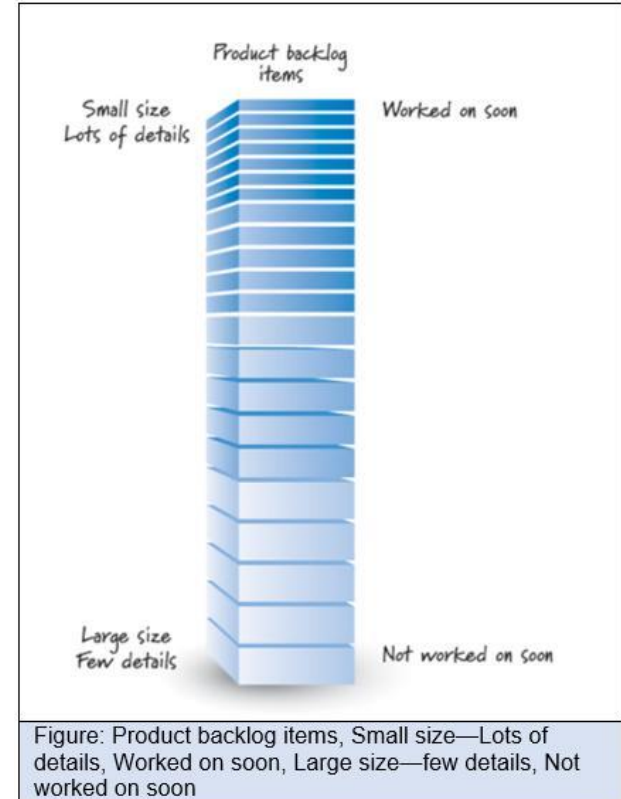
- Driven from Product Owner
- Assisted by the Team
- Requires Collaboration

Together as a team you can be successful.



Prioritizing the Backlog

- Financial Criteria
- Decision Matrix
- MoSCoW
- KANO



User Roles

- Why are User Roles important?
- Unique perspectives change requirements and acceptance criteria
- Who are your target customers?
 - What do they use the software for?
 - How do they use the software?
 - What are their priorities?

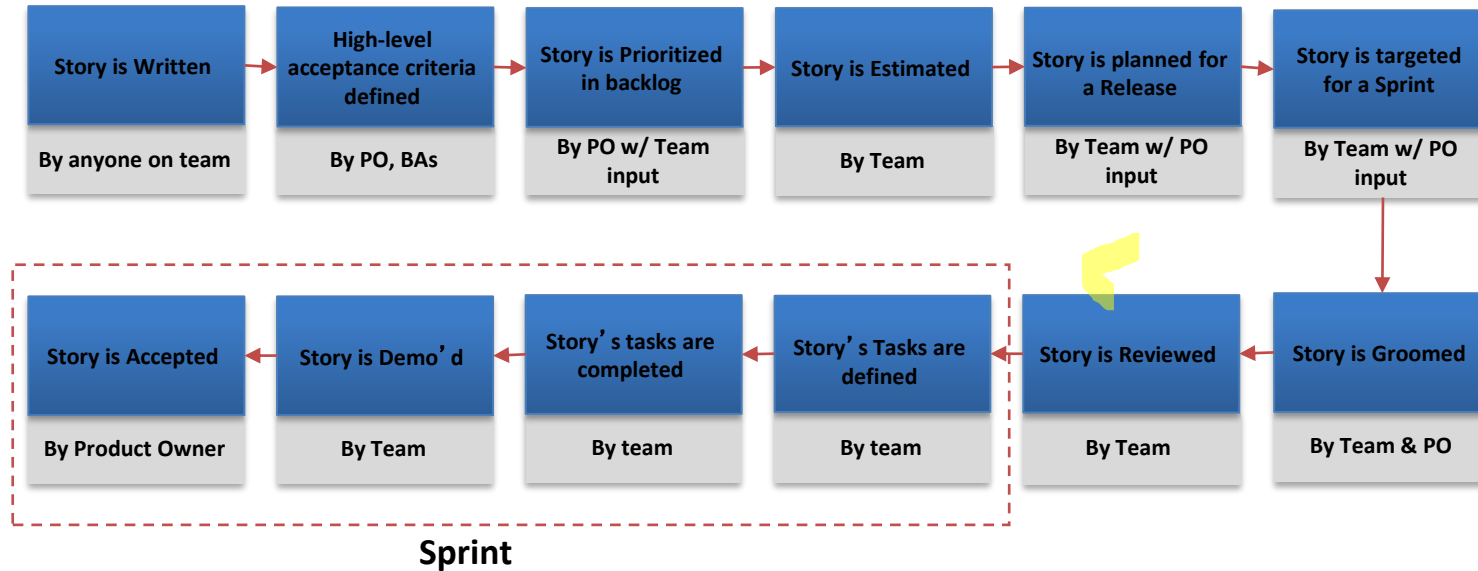


Story Mapping



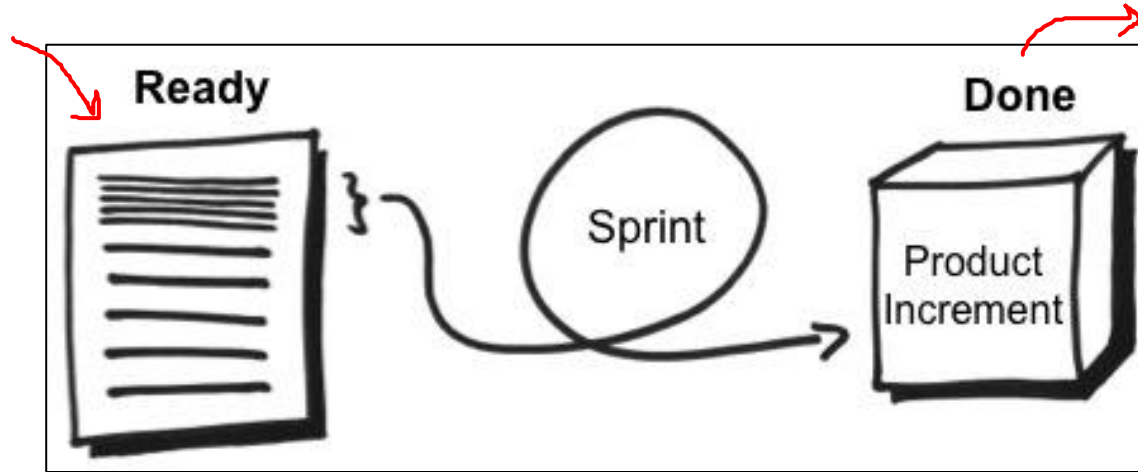
Example story map created by Steve Rogalsky
<http://winnipegagilist.blogspot.com>

Lifecycle of a Story



- Davisbase

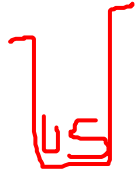
Defining Ready



Typical Ingredients in Definition of Ready

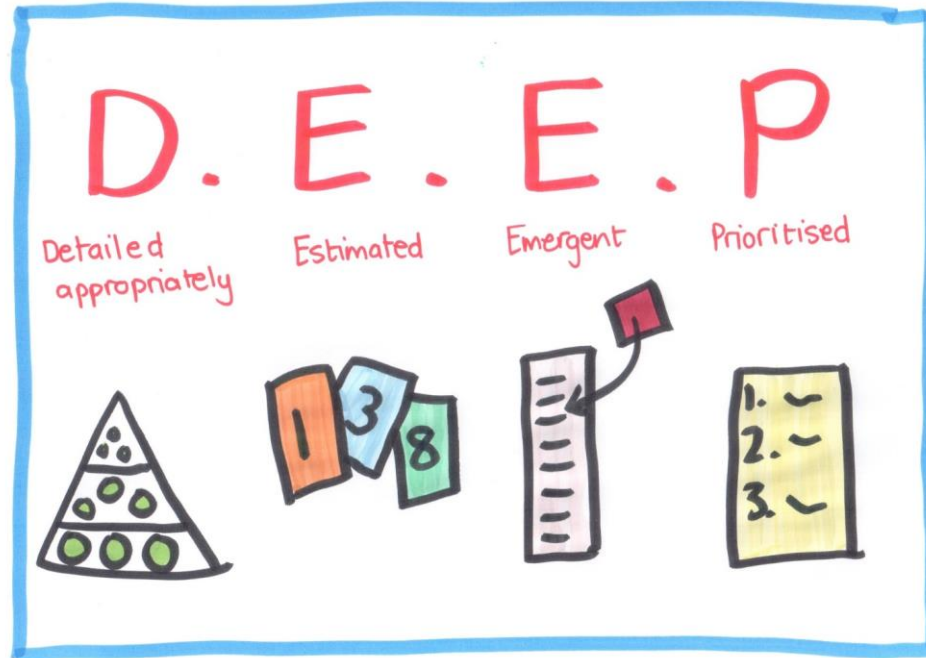
- Meets the INVEST criteria
- Has acceptance criteria
- Very little to no research, or all research
 - If a lot of research is required for a story, create a research-only story and time-box it
- The story is estimated
- UAT is well understood
 - Preferably fully stated as part of story... or...
 - QA person proxy for UAT tester
- **Whole team** feels comfortable that they know what it takes to get story to “done”
- The **whole team** has contributed to the grooming/estimation of the story

Backlog Refinement

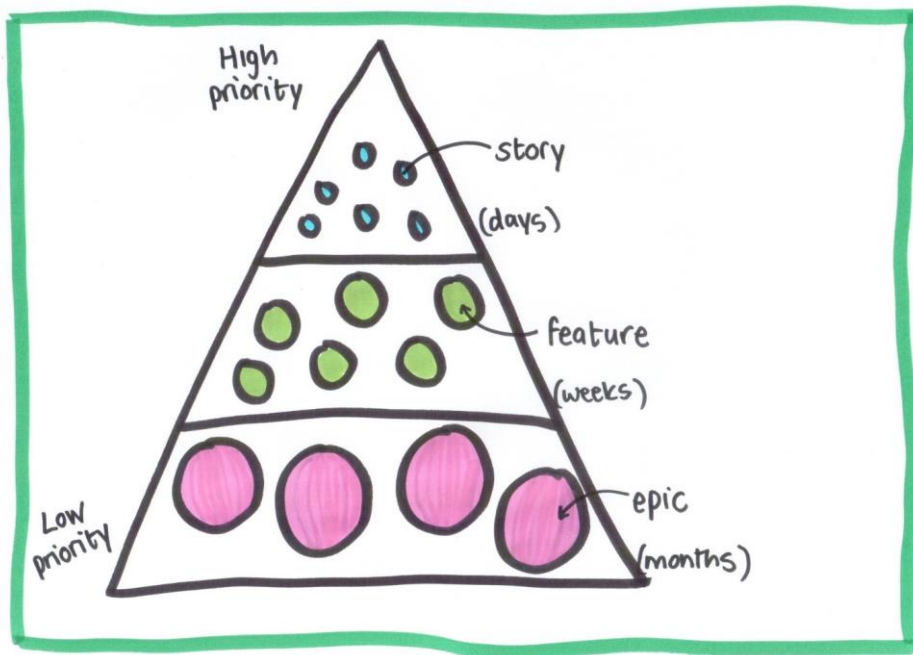


Product **backlog refinement**—sometimes called product backlog grooming in reference to keeping the backlog clean and orderly—is a meeting that is held near the end of one sprint to ensure the backlog is ready for the next sprint.

Refining the Backlog

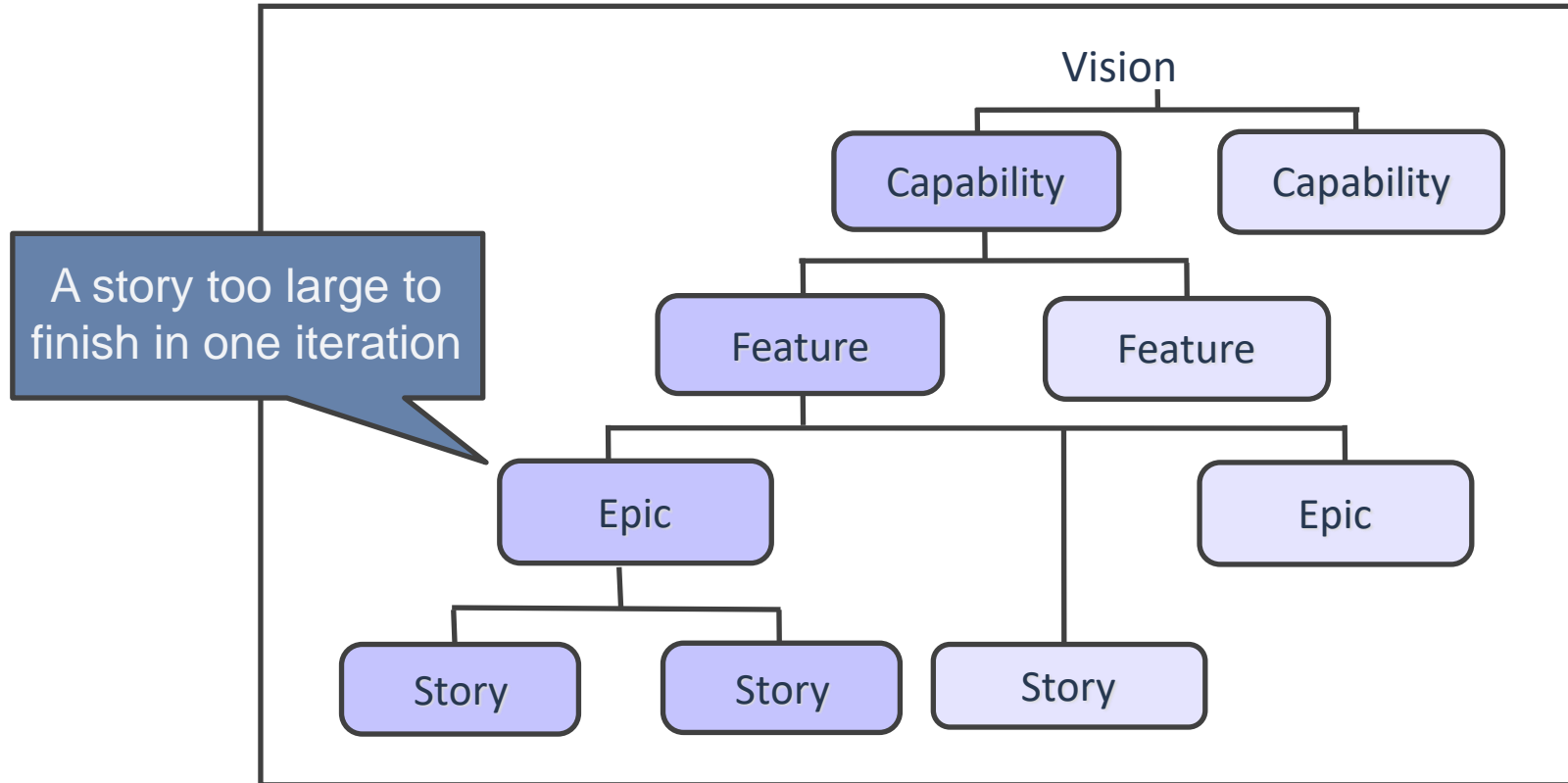


Decomposing Stories



- leanpub.com

Levels of Agile Requirements



Slice it Up

Take slice of the whole rather than individual layers.

“An authenticated member can post a recommendation for a book”



- ✗ An authenticated member can fill out a recommendation form
- ✗ Information on a recommendation form is written to the database
- ✓ A member can post a written review about the book
- ✓ A member can post a rating about the book
- ✓ A member can post references to similar books

Open vs. Closed Stories

- Open stories often have no end in sight
 - “As a Publisher, I want to **manage** the ads I have placed”
- Closed stories show an achievable, meaningful accomplishment
 - “As a Publisher, I want to **change** the expiration date of an ad”
 - “As a Publisher, I want to **delete** an ad that is no longer relevant”
 - “As a Publisher, I want to **measure** how many times an ad has been clicked through”



Additional Approaches to Split Stories

- Splitting by Acceptance Criteria
- Splitting by User
- Splitting by Items in a List
- Splitting by Create/Read/Update/Delete or the Word “Manage”
- Splitting by Keyword
- Splitting with Lists
- Splitting by Test Scenario

Size the Story to the Horizon

- Focus attention on most critical areas first
- Write stories at levels based on the implementation horizon

If stories are further out, they can be Open/Goal Stories

- “...Guest Member I want to register for an account...”
- “...Authenticated Member I want to post a recommendation...”
- “...Pioneer Member I want to submit feature suggestions...”
- “...Critic I want to post reviews of a book...”

Size the Story to the Horizon

Once a story is close to being started, break it down.

“As an Authenticated Member, I want to log into the system so that my information can only be accessed by me.”

...I want to log in with my username and password...

...I want to change my password...

...I want the system to warn me if my password is easy to guess...

...I want to be able to request a new password so that I am not locked out if I forget it

...I want to be notified if there have been three consecutive failed attempts to access my account...

Non-User Stories

- Technology foundation stories
 - At times these can be stated in customer terms
- Dependencies from external teams
- Creative elements
- Spikes
- Other types of stories... defects, maintenance, training, etc.

As a developer, I want to upgrade to the latest version of the database software so that we have a supported product

Spike: As a developer, I need to investigate a semantic search algorithm to facilitate natural language searching of the person's financial record.

Non-User Stories: Constraints

- Constraints often do not represent user functionality.
- Should be documented and remain visible for team, but does not go into the product backlog.
- Should be stated in measurable terms and be testable



As a patient, I want the system to function like the other systems in the suite so that it is familiar and easy to use.



As a stakeholder, I want page load times to conform to current standards so that patients will be able to use the system on a dial-up connection.

Functional vs. Non-Functional

- Functional - Captured through User Stories

As a <user role>,
I want to <desired FUNCTION>,
so that <desired benefit>.

Acceptance Criteria could also elaborate on functions.

- Non-Functional - Captured in Several Ways
 - Acceptance Criteria
 - Definition of Done
 - Constraints of the Product

Guidelines for Good Stories

1. Start with **Goal** Stories
2. Slice it Up
3. Open vs. Closed Stories
4. Size the Story to the Horizon

Outputs from Story Types

- **User Stories** ➡ Demonstrable working software for acceptance by the Product Owner.
- **Foundational** ➡ Working software, infrastructure, or systems that enable User Stories to be completed.
- **Spikes** ➡ Information or a decision.

What to Watch Out For

Mike Cohn's 'Catalog of Story Smells'

- Stories that are too small
- Stories too big....too many being split later
- Interdependent stories
- Goldplating
- Too much detail
- Interface detail too soon
- Thinking too far ahead
- Lack of customer participation, writing and prioritizing



References

