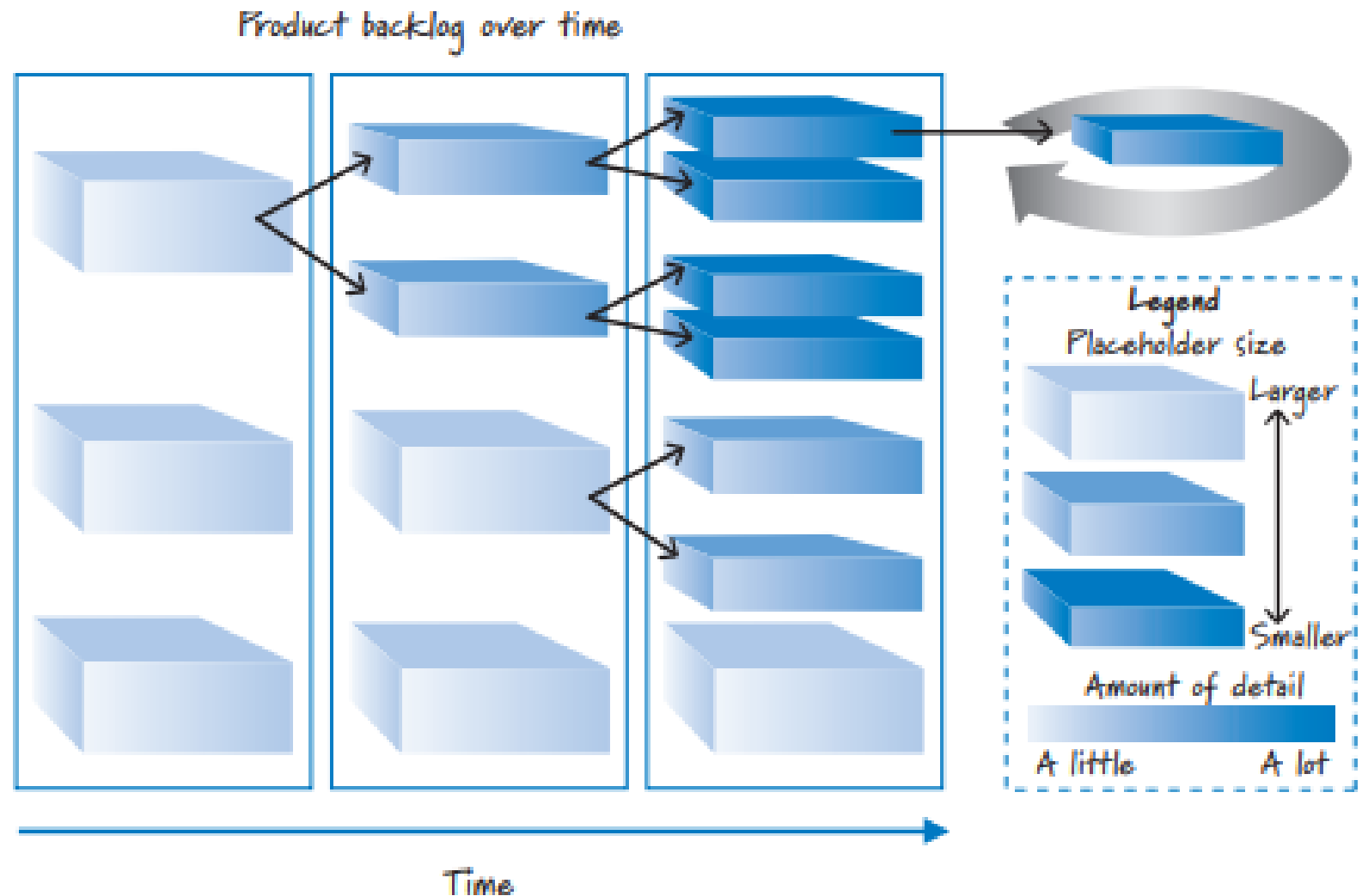


# Produkto reikalavimų inžinerijos procesas: agile RI

Dr. Asta Slotkienė

# Requirements hierarchy



# Product backlog

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint



This is the  
product backlog

# Definiton of Product backlog

- **The product backlog is a prioritized list of desired product functionality.**
- It provides a centralized and shared **understanding of what to build and the order in which to build it**

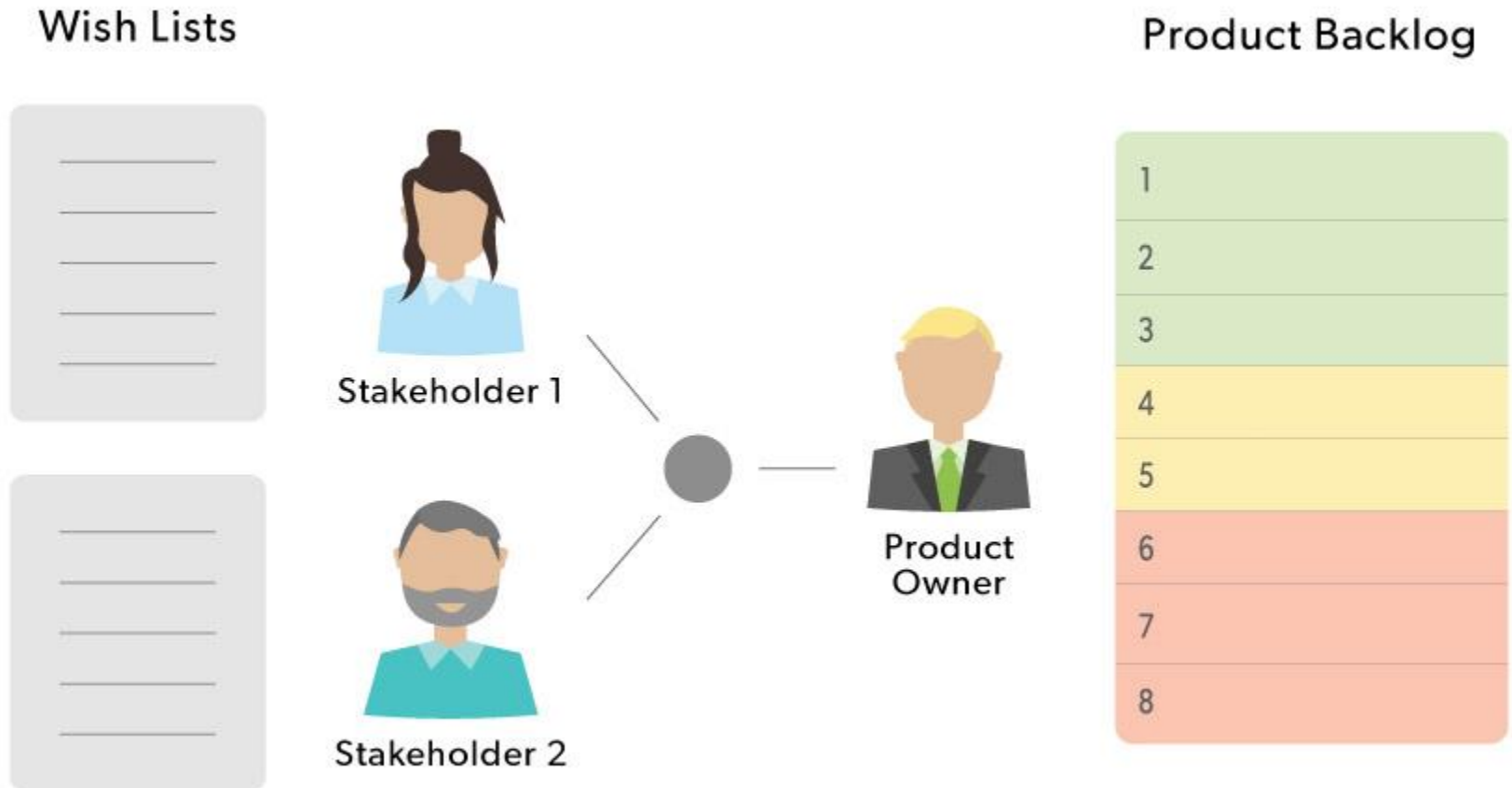
# Definiton of Product backlog

- The product backlog is a master list of all desired functionality for the system under development
- A living document – stories are added, changed, and/or removed throughout project duration

# Definition of Product backlog

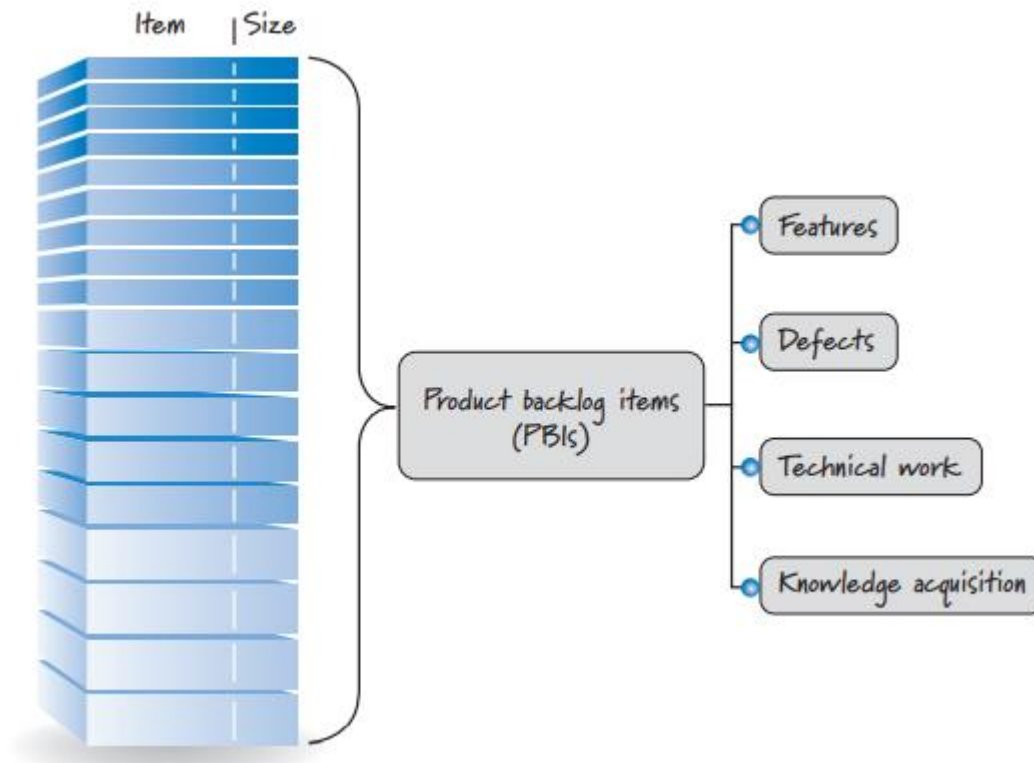
- Essentially the requirements, captured as a list of **user stories**
  - User stories are prioritized and assigned to sprints progressively
  - Stories are reprioritized at the start of each iteration

# Product backlog



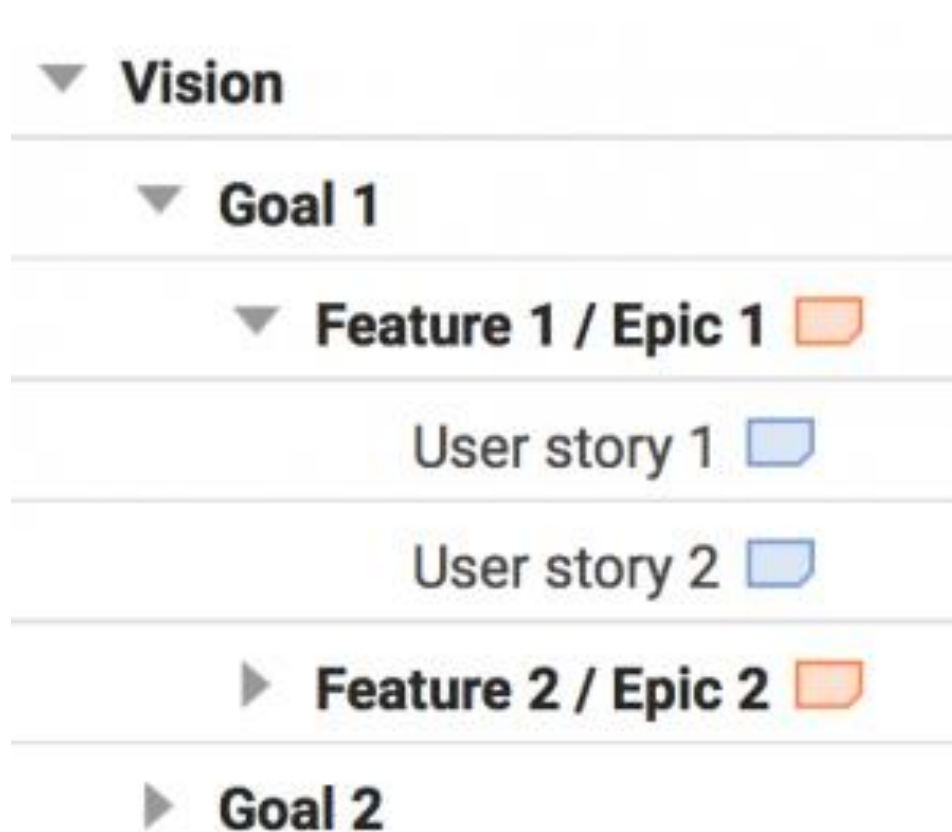
# Product backlog items

- The product backlog is composed of backlog items, which I refer to as PBIs
- Most PBIs are features, items of functionality that will have **tangible value to the user or customer.**

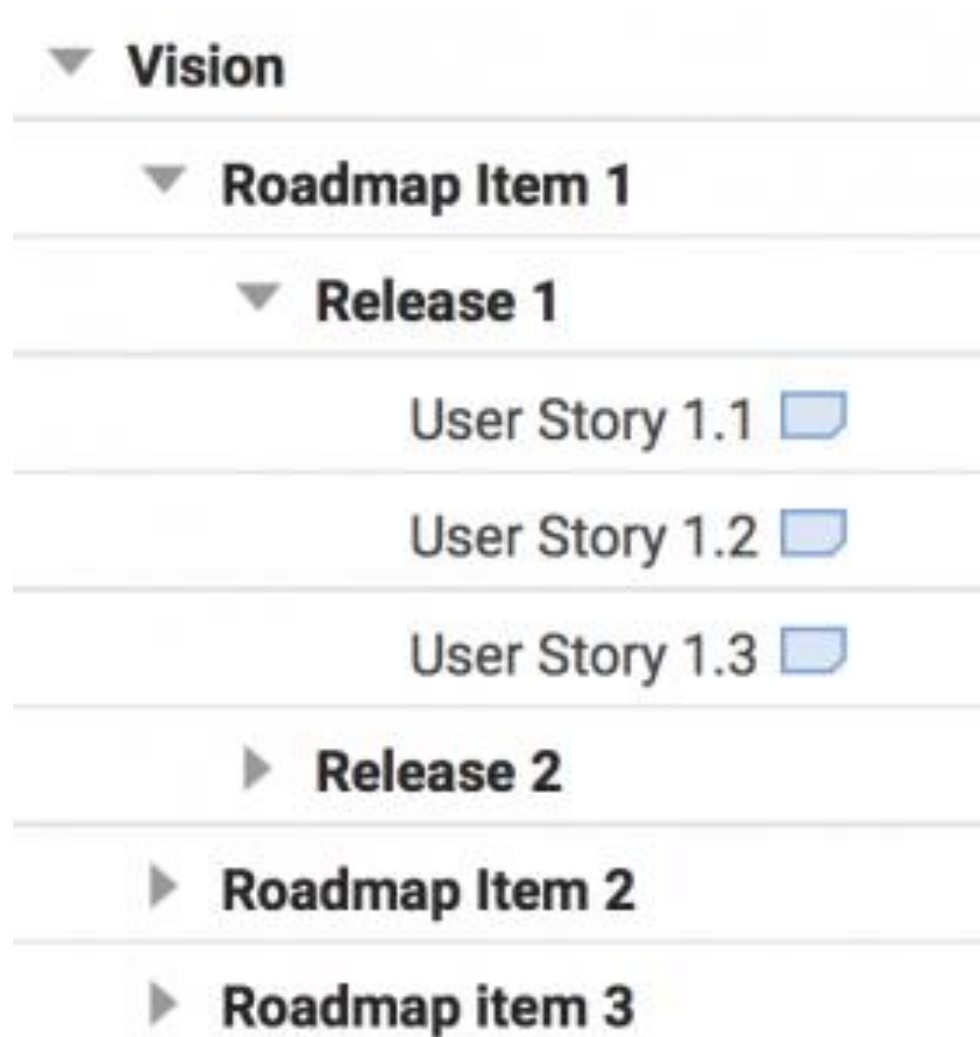




# Version of product backlog: Goal



# Version of product backlog: Planning



# Version of product backlog: component

## ▼ Product 1

---

### ▼ Component 1

---

#### ▼ Sub-system 1.1

---

Change 1

---

Change 2

---

#### ▶ Sub-system 2.1




---

### ▶ Component 2

---

## ▶ Product 2

# Version of product backlog: Team

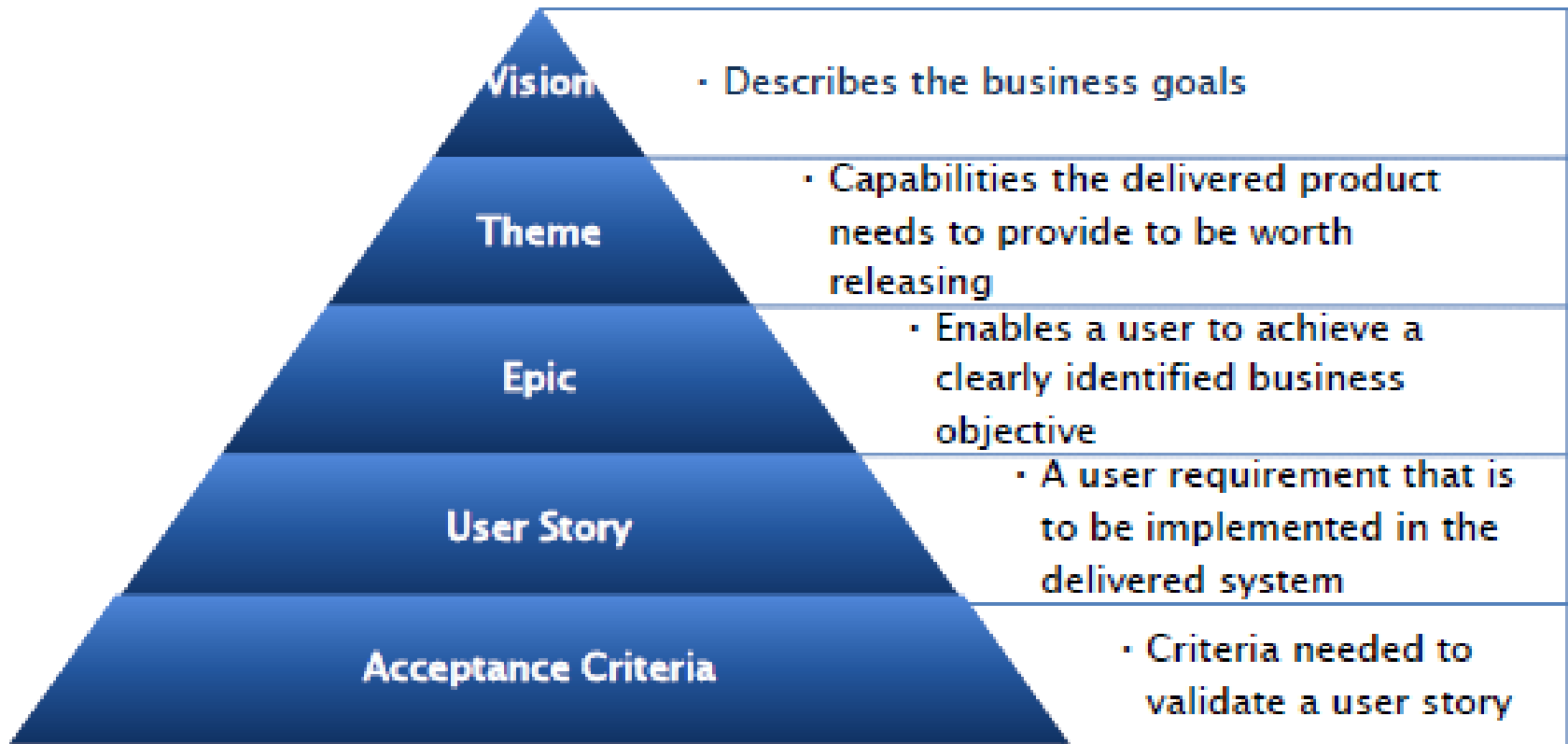
▼	<b>Area 1</b>	Delegated to: Product Owner
▼	<b>Team Yellow's Backlog</b>	Delegated to: Team Yellow
	User Story 1	
	User Story 2	
	User Story 3	
►	<b>Team Blue's Backlog</b>	Delegated to: Team Blue

# Version of product backlog: Stakeholder and processes

## ▼ Funnel

- ▶ **Wish List 1** Delegated to: Stakeholder 1
- ▶ **Wish List 2** Delegated to: Stakeholder 2
- ▶ **Product Backlog** Delegated to: Product Owner
- ▶ **Verification / Release Ready**
- ▶ **Done** Archived

# Hierarchical requirement: User stories



# User story: Epic level

- **When a story is too large** it is sometimes referred to as an *epic*.
- Epics can **be split into two or more stories** of smaller size.

# User story: Epic level

- When a story is too large it is sometimes referred to as an *epic*.
- Epics can be split into two or more stories of smaller size.



# User story: Epic level

- An epic should be introduced by a short explanation.
  - *What is the problem we want to solve and why? (Problem statement)*
  - *Who are we doing it for? (Personas)*
  - *How do we want to improve it? (Goal/Solution)*

# Example of the epic

- Create products **Epic?**
- Create products online **Epic?**

# Example of the Epic

**As a customer , I want to be able to **CRUD** my products on my desktop PC, so that I have all product information in one place**

# Example of the epic

- User story:
  - *A user can search for a job*

could be split into these stories:

- *A user can search for jobs by attributes like location, salary range, job title, company name, and the date the job was posted.*
- *A user can view information about each job that is matched by a search.*
- *A user can view detailed information about a company that has posted a job.*

# Example of the epic: not splitting

- User story is a very reasonable and realistic story.
  - *A user can view information about each job that is matched by a search”*

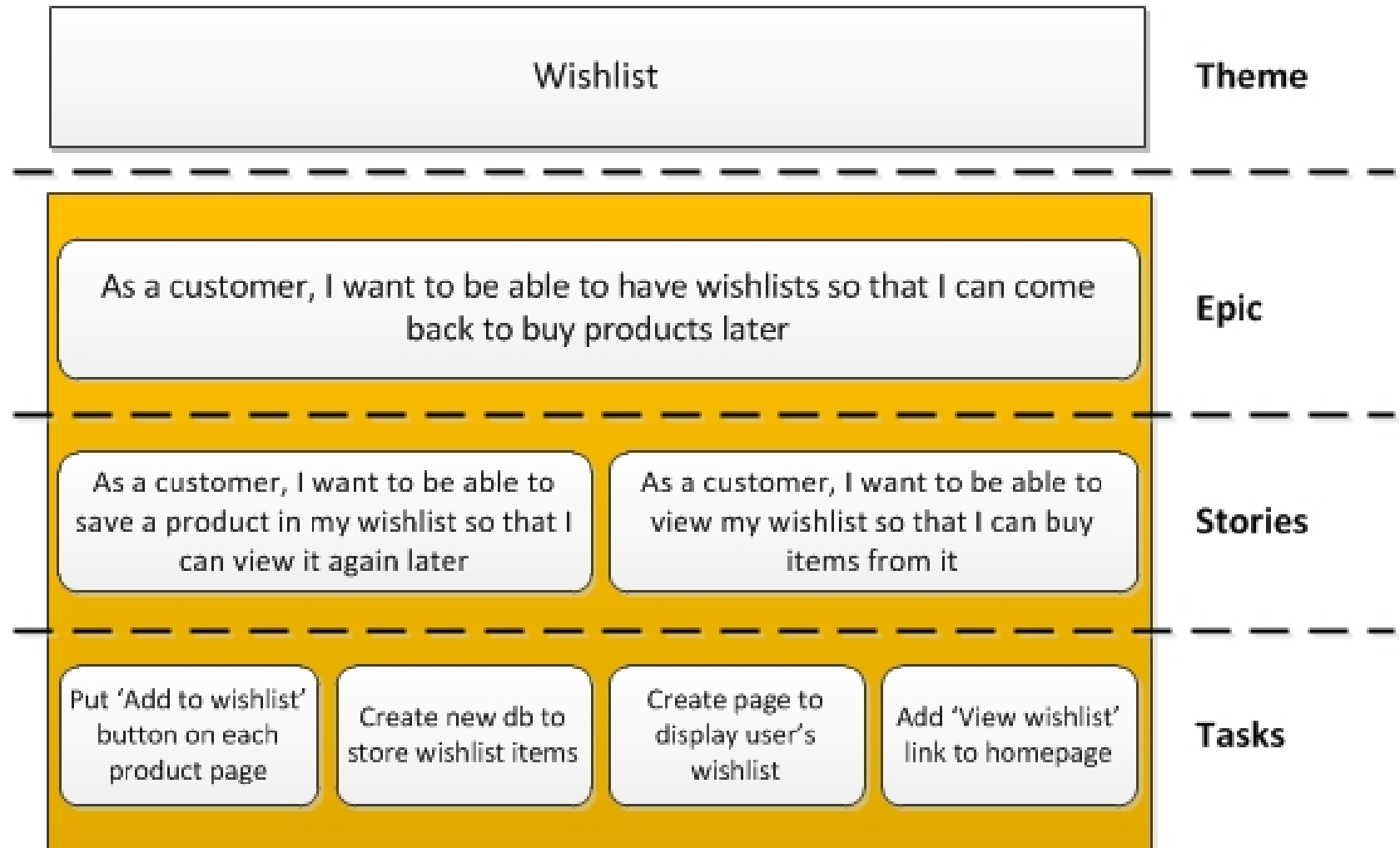
We do not need to divide it into smallest US:

- *A user can view a job description.*
- *A user can view a job’s salary range.*
- *A user can view the location of a job.*

# Themes

- A collection of related user stories.
- A theme provides a convenient way to indicate that a **set of stories have something in common**, such as being in the same functional area

# Example



# Feature

- **A distinguishing characteristic or capability of a software application or library:**
  - performance,
  - portability,
  - functionality and etc.



Portfolio Item

Story Level Item

Theme

Initiative /  
Epic

Initiative /  
Epic

Feature

Feature

Feature

Story

Story

Story

Story

Story

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Years

Months

Weeks

Days

Hours

# Features vs. Epics

- A Feature is not a User Story.
- An Epic is a User Story.
- A User Story may be an Epic.
- A User Story can contain many Features.
- A Feature can fulfill 1 to many User Stories.

# Definition of User story

- A user story **describes functionality** that will be **valuable** to the a user or customer of a system or software
- Simple stories that describe **what the users world must look like**
- The basic user story template is simplistic, it **helps us remember a need while providing context.**
- A brief, simple **requirement statement from the perspective of the user**

# Definition of User story

- ...A user story is nothing more than an **agreement that the customer and developers will talk together about a feature.**

[Beck et al, 2001]

...understandable to customers and developers,  
testable, valuable to the customer and small  
enough so that the programmers can build half a  
dozen in an iteration

# User Story

As a <user role>

I want <goal>

so that <benefit>.

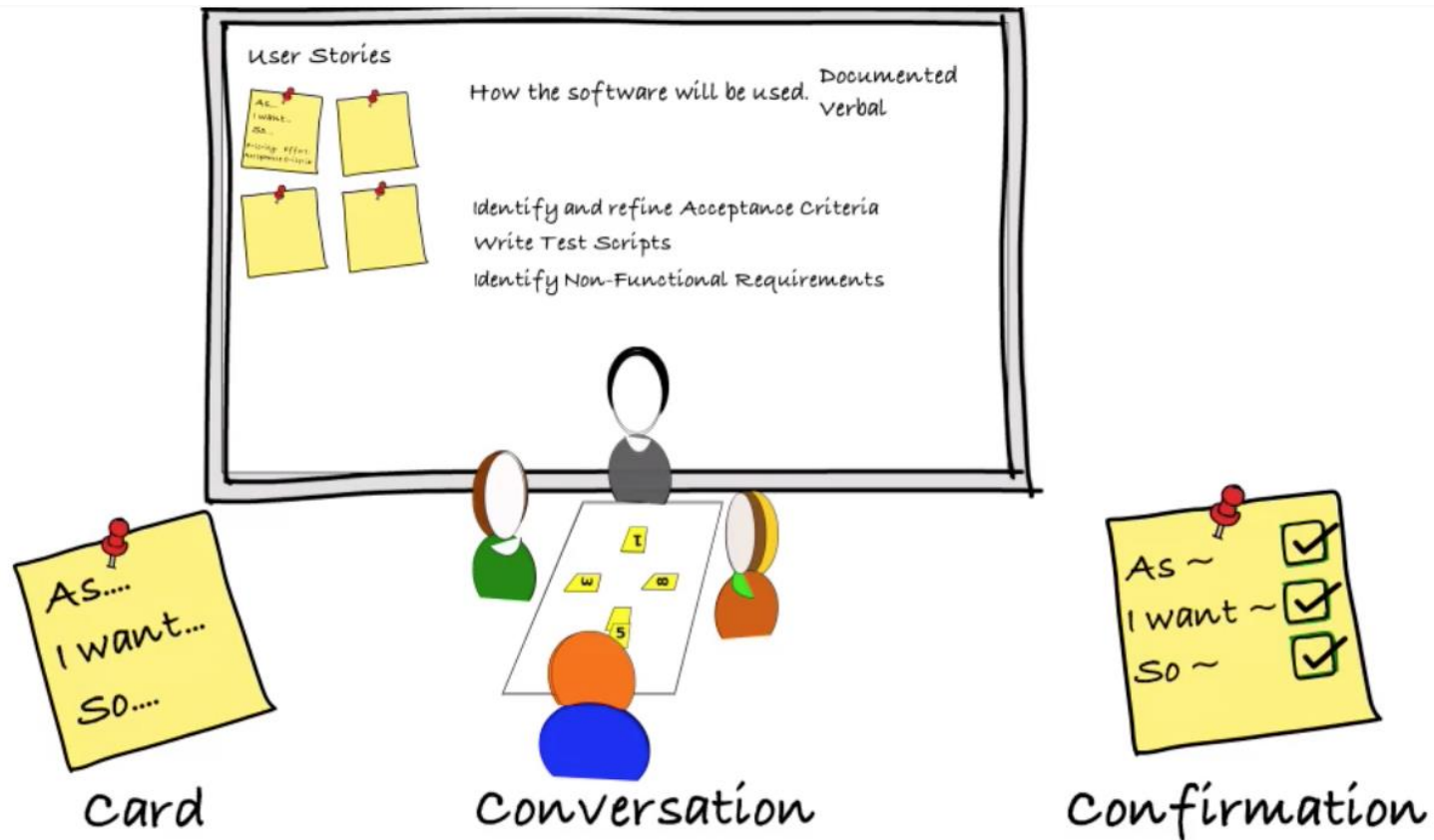
# User Story

<b>WHO</b> are we building it for? Who is the user?	As a <type of user>
<b>WHAT</b> are we building? What is the intention?	I want <some goal or objective>
<b>WHY</b> are we building it? What is the value for the customer?	So that <benefit/value>

# Purpose of US

- Communicate stakeholder need
- Describe product feature on a high level of abstraction
  - Can be understood by ALL stakeholders
- Reminder for future conversations
- Used for planning
- Provide value to the business

# The Three C's of User Stories





# The Three C's of User Stories

1. a written description of the story used for planning and as a reminder
2. conversations about the story that serve to flesh out the details of the story
3. tests that convey and document details and that can be used to determine when a story is complete

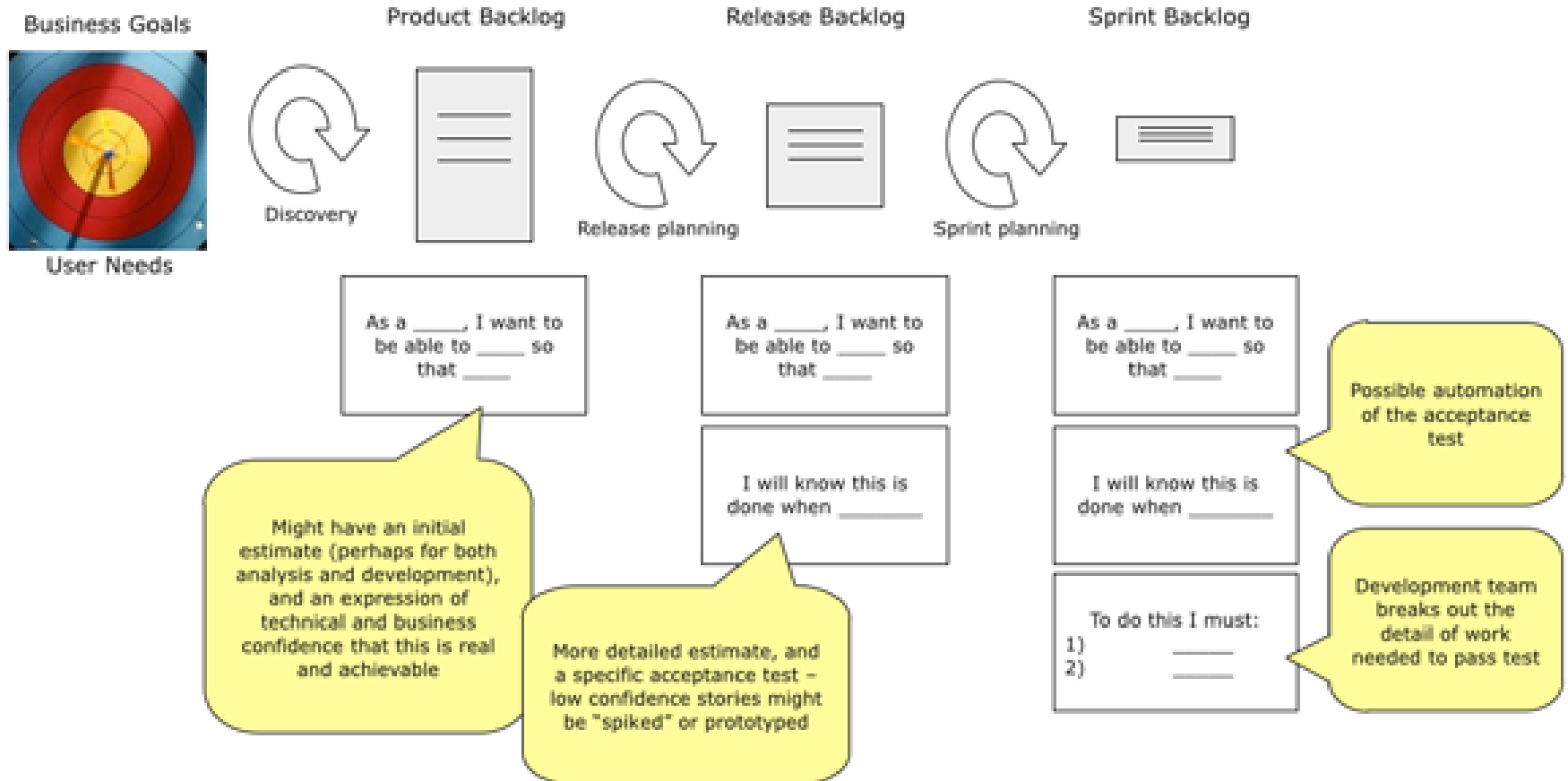
# US Refinement

- Refinement ensures the highest priority Stories meet the **Definition of Ready (DoR)**
- Product Owner and the Development Team collaborate to ensure that items on the Product Backlog:
  - **are understood the same way by all involved**
  - have a size estimate for the (relative) complexity and effort of their implementation,
  - are ordered according to their priority in terms of business value and effort required.

# US Refinement



# User Story Lifecycle



<https://agilefaqs.com/services/training/user-stories>

# Component of user stories

- Title— this is a short handle for the story.
  - A present tense verb in active voice is desirable in the title.
- Acceptance test—this is a unique identifier that will be the name of a method to test the story.
- Priority
- Story points—this is the estimated time to implement the user story.

# Task

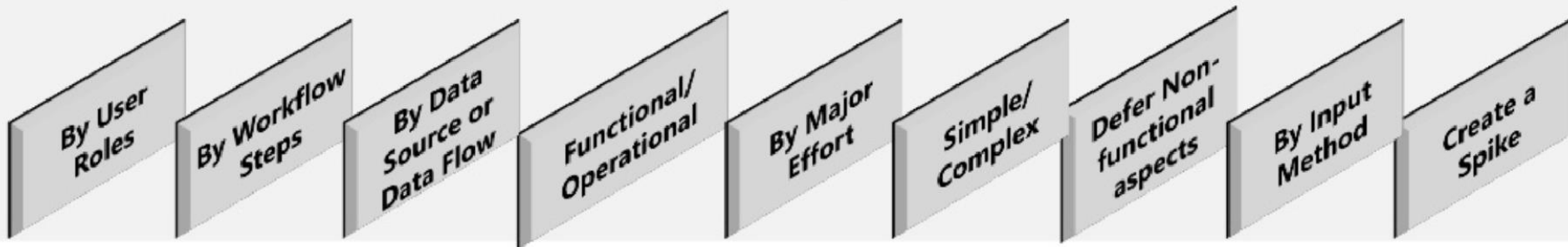
- What are the activities we need to perform in order to deliver outcomes (user stories)
- Tasks are individual pieces of work
- Main question – HOW?

# Writting user stories

- Developing user stories is an “iterative and interactive” process.
- The development team also manages the size of stories for uniformity:
- too large—split, too small—combine).

# Splitting US

## Patterns of Splitting User Stories



### User Story Detailing Levels





# Example of US

- As a
  - *register user*
- I want
  - *change my password*
- So that
  - *keep my account secure*




# Example of US

- As a
  - *register user*
- I want
  - *remember my password*
- So that
  - *I have possibility to login in the system, when I don't remember it*



# Example of US ?

- As a
  - *admin user ?*
  - *administrator of IS department ??*
- I want
  - *disable users ??*
  - *create rights of the users in the system ?*
- So that
  - *prevent unauthorized logins ?*
  - *ensure secure of my systems ?*



As a <role>  
I want <goal>  
So that <benefit>

Acceptance criteria:

...

# Example of US ?


- As a
  - *user*
- I want
  - *view information about each job*
- So that
  - *is matched by a search ??*
  - *Is best possibilities for my future ??*



# Example of US

- As a
  - *guest*
- I want
  - *register in the system*
- So that
  - *admin has list of user*

Bad example, WHY?



As a <role>  
I want <goal>  
So that <benefit>

Acceptance criteria:

...

# Example of US

- As a
  - *register user*
- I want
  - *to find a book about Java programming language*
- So that
  - *has possibilities to improve my programming skills*

Bad example, WHY?

As a <role>  
I want <goal>  
So that <benefit>

Acceptance criteria:  
...

# Example of US

- As a
  - *student John*
- I want
  - *to get notifications about changes of timetable*
- So that
  - *I could arrange better my activities*

Bad example, WHY?

As a <role>  
I want <goal>  
So that <benefit>

Acceptance criteria:  
...

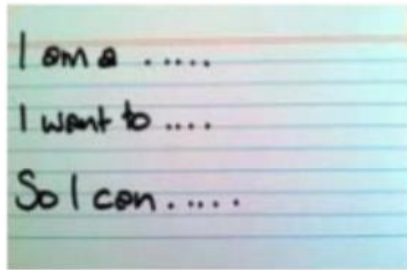
# Example of US

- *As a programmer I want to write the software code in C++ So that this language is OO*
- *As a developer I want use SQL database So I can store information without duplication*
- **Why these stories are wrong?**



# The Three C's of User Stories

## Card



Stories are traditionally written on note cards

## Conversation



Details of a story come out during conversations with Product Owner

## Confirmation



Acceptance tests confirm the results are what the requestor wants.

# User story template

*Story Title*

**Estimate:** ... story points

**As a** ... — who? fill this in with a stakeholder type

**I want** ... — what? fill this in with the problem they want solved

**so that** ... — why? fill this in with the value they will gain

**Additional detail:**

... — what else do we need to know to understand the story?

**Acceptance criteria**

We'll agree it's been completed when:

- ... — add a list of easy-to-verify true/false statements
- ... — concrete examples are best
- ...
- ...
- ...

# Acceptance test

- Acceptance testing is the process of verifying that **stories were developed** such that each works exactly the way the customer team expected it to work
- Tests should be **written as early** in an iteration as possible
- Acceptance tests also **provide basic criteria that can be used to determine if a story is fully implemented.**

# Acceptance test

- Valid for a specific user story
- Describes when value is realized for customer
- Should be testable
- Basis for estimations
- Can change over time
- Has to be fulfilled before story is moved to testing stage
- Provide basic criteria that can be used to determine if a story is fully implemented

# Acceptance Criteria Goals

- To clarify what
- To ensure everyone has a common understanding of the problem
- To help the team members know when the story is complete
- To help verify the story via automated tests

# User Story with Acceptance Criteria

## User Story:

- As an online banking customer, I want strong a strong password, so that my credit card information is secure

## Acceptance Criteria:

1. The password must be at least 8 characters
2. The password must contain at least 1 character from each of the following groups: lower case alphabet, upper case alphabet, numeric, special characters (!, @, #, \$, %, ^, &, \*)

# User Story with Acceptance tests

- User Story:
  - *A user can pay for the items in her shopping cart with a credit card*
- Acceptance tests:
  1. Test with Visa, MasterCard and American Express (pass).
  2. Test with Diner's Club (fail).
  3. Test with a Visa debit card (pass).
  4. Test with good, bad and missing card ID numbers from the back of the card.
  5. Test with expired cards.
  6. Test with different purchase amounts (including one over the card's limit).

# User story with Acceptance tests

User story:

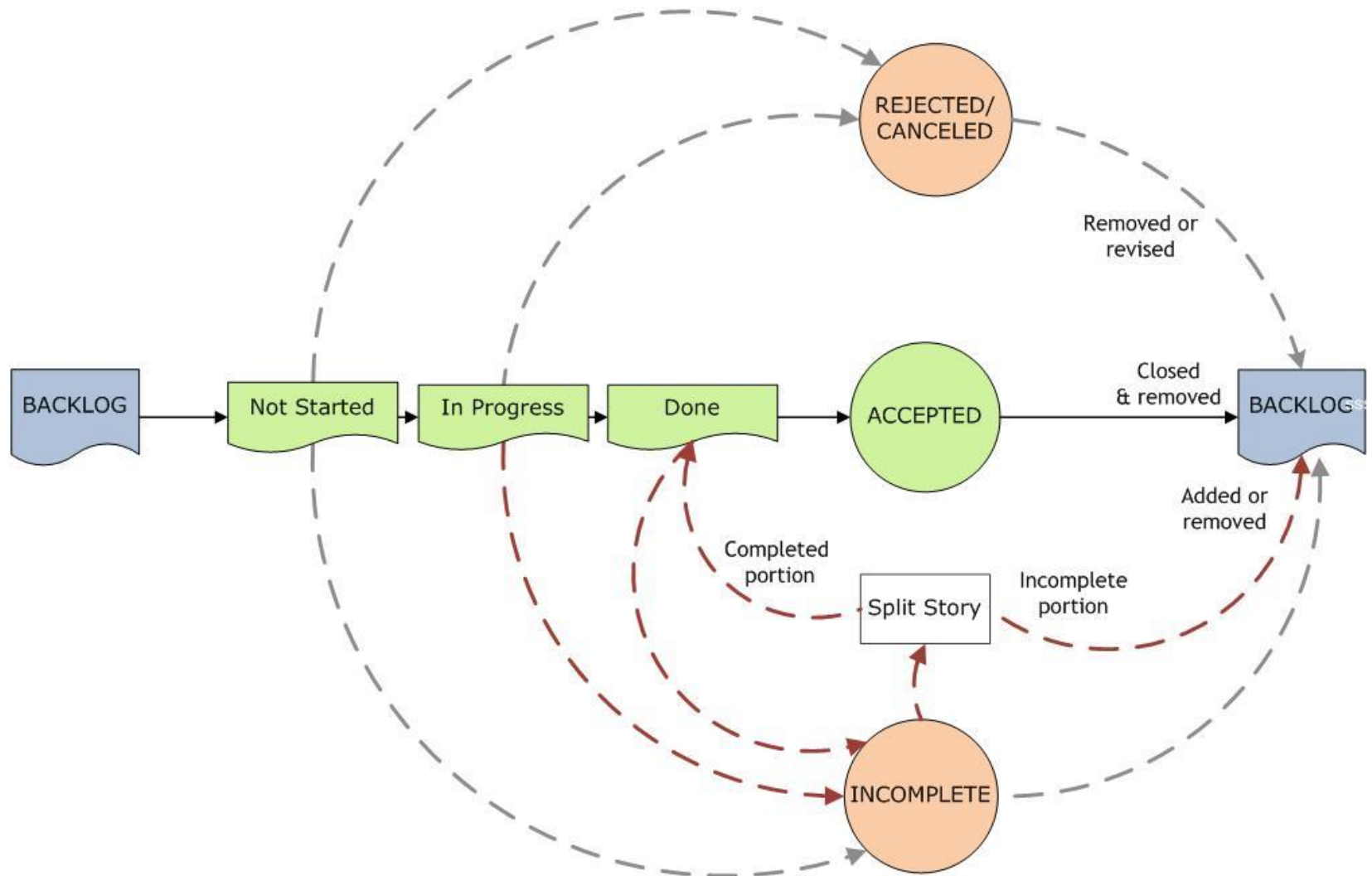
*As an employer I want to post a job so that others people can find it.*

Acceptance tests :

1. Verify that only an authorized user with a valid employer account can post a job.
2. Verify that a duplicate job posting cannot be entered.
3. Verify that the posting date is past today's date.
4. Verify that the positing expiration date within 90 days.
5. Verify that the screen fields pass our standard field format rules (link here to doc).
6. Verify that all required fields are entered (list them or link to UI Prototype).



## State diagram for a user story



# **PRACTICAL VIEWPOINT: USER STORY**

# Epic level, examples

1. As a administrator, I want to administration users in e-bank system, so that I can do all actions with user data
2. As a administrator, I want to see user list, who are register/banned in the system
3. As a administrator, I want to generate various reports up to 10 s, so that I could get it quickly
4. As a administrator, I want to get data form database by Rest technology ?

# User stories level: examples

1. As a bank users, I want to find near ATM of my address, so that I can quickly to perform action with my deposit
2. As a administrator, I want to add new user, so that I can to fill true information about users

# Task level: examples

1. As a administrator, I want to change types of user, so that I can to have correct users list
2. As a administrator, I want to get user list by each criteria
3. As a administrator, I want to get user list by course of study
4. As a administrator I want to choose the role of user of the ComboBox ?

# Epic level/User story level/Task level ?

1. As a administrator, I want to generate log of action for 1 day, so that I can to find any mistake
2. As a register user, I want to buy, edit or cancelling my order, so that I can manage my order list
3. As a administrator, I want to make any change of user profile
4. As a register user, I want to change my address, So what my order to send by it

# INVEST model

Follow the INVEST  
guidelines for good  
user stories!



# User stories: INDEPENDENT

- **US should be self contained, in a way that there is no inherent depends on other user stories**
- For large systems this is nearly impossible, but, minimizing, identifying and prioritizing, **dependencies can result in a better backlog**
- **Dependencies between stories lead to prioritization and planning problems.**



# User stories: independent

Are these US independent?

- 1. As a user I want to register, so that I could to perform test of knowledges*
- 2. As a user I want to check my knowledge, that I would like to know my level of skills*
- 3. As a register user I want to get report of my test*

# User stories: independent

1. *A company can pay for a job posting with a Visa card.*
  2. *A company can pay for a job posting with a MasterCard.*
  3. *A company can pay for a job posting with an American Express card.*
- Why these stories are splitting not correctly?

# User stories: independent

1 solution: Combine the dependent stories into one larger, but independent story

- *A company can pay for a job posting with a credit card*

2 solution: Find a different way of splitting the stories

- *A customer can pay with one type of credit card.*
- *A customer can pay with two additional types of credit cards.*

# User story: Negotiable

- Stories are not written contracts or requirements that the software must implement.
- Story cards are short descriptions of functionality
- US are not explicit contract and should leave space for discussions

# User story: Negotiable

1. *As as driver I want to get shortly directions to my destination, so that I get there quickly*
  2. *As as driver I want to get directions to my destination on map, so that to use Google maps*
- Where isn't negotiable?

# User story: Valuable

- Each story must be valued by the users
- Many projects include stories that are not valued by users.
- Keeping in mind the distinction between
  - *user*: someone who uses the software
  - *purchaser*: someone who purchases the software
- The best way to ensure that each story is valuable to the customer or users is to have the customer write the stories

# User story: Valuable

- 1. As a user I want to have my previous orders stored in the database So they will be there permanently*
- 2. As a customer I want 75% off all purchases, So I can save money*

# User story: Small/Estimable

- User story should not be so big as to become impossible to plan/task/prioritize with certain level of certainty
- You must be always estimate the size of US



# User story: small? Estimate?

- 1. As a customer I want to perform e-services of my bank account, so I can do all action with my account*
- 2. As a customer I want to find an ATM near the address , so that I can make deposits outside of banking hours*

# User story: Small/Estimate

Divide to small US:

- *As a bank user I want to see my canceled operations online*
- *As a bank user I want to make the money transfer online*
- *As a bank user I want to plane an transfer which bank will do later*

# User story: Testable

- Description of US must be provide the necessary information to make test cases
- Solutions include adding acceptance criteria or better defining the story
- If the story cannot be tested, how can the developers know when they have finished coding?

# Untestable user stories

- They show up for nonfunctional requirements, which are requirements about the software but not directly about its functionality.
  - *A user must find the software easy to use.*
  - *A user must never have to wait long for any screen to appear.*

# Untestable -> Testable user stories

- *A user must find the software easy to use.*
- ***A novice user is able to complete common workflows without training***
- *A user must never have to wait long for any screen to appear.*
- **New screens appear within two seconds in 95% of all cases**

# User story: Testable 1

- *As a user I want to good GUI, so that quickly understood it*
- *As a user I want to simple GUI, so that quickly understood it*
- Acceptance criteria:
  - All text is dark colour on light background
  - Only two different fonts used in GUI

# User story: Testable 2

- *As a user I want to get confirmation of my order, so that I will be guaranteed*
- **Acceptance criteria:**
  - Copy of Confirmed order send by email through 1 hour
  - Decreasing selected items (of order) in database

# Requirement vs User Story

- REQUIREMENT =
  - User story (end users) +
  - Acceptance Criteria (non –functional requirements) +
  - Supporting/technical requirements



# Summary of US

- Ideally, stories are independent from one another. This isn't always possible but to the extent it is, stories should be written so that they can be developed in any order.
- The details of a story are negotiated between the user and the developers.
- Stories should be written so that their value to users or the customer is clear. The best way to achieve this is to have the customer write the stories.
- One of the best ways to annotate a story is to write test cases for the story
- Stories need to be testable.

# Requirement+ Test Plan

- REQUIREMENT =
  - User story (end users) +
  - Acceptance Criteria (non –functional requirements) +
  - Test cases +
  - Supporting/technical requirements

