Agile Software Development

L3 – Requirements management in ASD

Mariana Falco
mariana.falco@ut.ee
Agenda

1. Quick recap
2. Requirements in ASD
3. Agile program
Recap

• Agile terminology
  • Mindset, Values, Practices, Methods
  • Miro
  • Principles

• TDD
Your ideas
Summarized as...

- Communication
- Project planning + management [time, budget, deadlines, deliverables, iterations, teams]
- Expectations and requirements
- Stakeholders [involvement, consensus, feedback]
- Teams [knowledge, training, seniority, comms, collab]
- Tech, toolkits
- Research + industry
- Product validation + short feedback loop + user feedback
- Scalability, maintainability, performance, testing
Summarized as...

- Communication
- Project planning + management [time, budget, deadlines, deliverables, iterations, teams]
- Expectations and requirements
- Stakeholders [involvement, consensus, feedback]
- Teams [knowledge, training, seniority, comms, collab]
- Tech, toolkits
- Research + industry
- Product validation + short feedback loop + user feedback
- Scalability, maintainability, performance, testing
Requirements in ASD
Requirements in SE

• Software requirements is a communication problem

• When the business side dominates...
  • it mandates unrealistic functionality and dates
  • it ignores if developers understand exactly what is needed

• When the developers dominate the communications...
  • technical jargon replaces the language of the business
  • the developers lose the opportunity to learn what is needed by listening
Requirements – Use Cases

• An artifact that defines a sequence of actions that yields an observable result of value

• Can be represented as a **graphical element** in a diagram and as a **use-case specification** in a textual document

https://en.wikipedia.org/wiki/Use_case
Use Cases – Example

![Use Case Diagram]

- **Waiter**
  - receive order
  - place order

- **Chef**
  - Order Wine

- **Client**
  - facilitate payment
  - pay
  - accept payment

- **Cashier**
  - Pay for Food
  - Pay for Wine

- **Order Food**
  - extend
  - confirm order

- **Serve Food**
  - extend

- **Cook Food**
  - if wine was ordered

- **Serve Wine**
  - if wine was served

- **Eat Food**
  - extend

- **Drink Wine**
  - extend

- **Pay for Food**
  - if wine was consumed
Requirements in ASD | User Stories
An agile program
An agile program

collections of epics that drive toward a common goal

large bodies of work that can be broken down into a number of smaller tasks (called stories)

short requests written from the perspective of an end user
An agile program

organization goal that drive the creation of epics and initiatives
User Story

• The smallest **unit of work** in an agile framework

• A USs is a general explanation of a software feature written from the **perspective of the end user**

• In Scrum, **USs are added to the sprints** and burned down over the duration of each sprint

• USs are the **building blocks** of larger agile frameworks like epics and initiatives
User Story Structure

As a [persona]
I [want to]
so that [benefit]

https://www.atlassian.com/agile/project-management/user-stories
User Story Structure

As a [persona] I [want to] so that [benefit]

Who are we building this for?
Their intent - not the feature they use
What is the big problem that needs solving?

+ acceptance criteria

https://www.atlassian.com/agile/project-management/user-stories
Example

As a manager,

I want to **be able to understand** my colleagues progress,

so I **can better report our success and failures**

https://www.atlassian.com/agile/project-management/user-stories
Role identification

• Most project teams consider only a single type of user □ this leads to software that ignores the needs of, at least, some user types

• Identify the different **user roles** who will interact with the software
  • Some user roles benefit from being described by **personas** □ A persona is an imaginary representation of a user role

• **Brainstorm, Organize, and Consolidate** the roles
Where are the details?

As a user
I want to search for a job
Where are the details?

As a user
I want to search for a job

• Many details can be expressed as additional stories
• It is better to have more stories than large stories
• User Story Splitting
Assess the Quality

A good user story should be:

- Independent (of all others)
- Negotiable (not a specific contract for features)
- Valuable (or vertical)
- Estimable (to a good approximation)
- Small (so as to fit within an iteration)
- Testable (in principle, even if there isn’t a test for it yet)

https://www.agilealliance.org/glossary/invest/
Quality criteria: INVEST

<table>
<thead>
<tr>
<th>INVEST by Bill Wake</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Independent</strong></td>
</tr>
<tr>
<td>• Of order of user story delivery</td>
</tr>
<tr>
<td>• Of internal and especially external dependencies</td>
</tr>
<tr>
<td><strong>Negotiable</strong></td>
</tr>
<tr>
<td>• Flexible scope</td>
</tr>
<tr>
<td>• None specific language</td>
</tr>
<tr>
<td>• Explain the intention, not the implementation</td>
</tr>
<tr>
<td><strong>Valuable</strong></td>
</tr>
<tr>
<td>• Value is clear to everyone</td>
</tr>
<tr>
<td>• Persona matches Benefit &amp; Goal will deliver the benefit</td>
</tr>
<tr>
<td>• Avoid technical / role specific language</td>
</tr>
<tr>
<td><strong>Estimatable</strong></td>
</tr>
<tr>
<td>• Clear and concise explanation</td>
</tr>
<tr>
<td>• Avoid technical / role specific language</td>
</tr>
<tr>
<td><strong>Small</strong></td>
</tr>
<tr>
<td>• Easily fits into a Sprint. i.e. &lt; 20% of velocity.</td>
</tr>
<tr>
<td>• Definitely not &gt; 33% of velocity</td>
</tr>
<tr>
<td><strong>Testable</strong></td>
</tr>
<tr>
<td>• Can be automated</td>
</tr>
<tr>
<td>• Avoid external testing / long test suites</td>
</tr>
</tbody>
</table>

https://www.agilealliance.org/glossary/invest/
User Story Splitting patterns

1. Prepare the input story
   • Check INVEST
   • Check the size

2. Apply the splitting patterns
   • Operations, Data, Workflow steps...

3. Evaluate the split
   • Are the new stories equal in size?

https://agileforall.com/patterns-for-splitting-user-stories/
HOW TO SPLIT A USER STORY

1. PREPARE THE INPUT STORY
   - Does the big story satisfy INVEST (except, perhaps, small)?
     - NO
       - Combine it with another story or otherwise reformulate it to get a good, if large, starting story.
     - YES
       - You're done.
       - Continue. You need to split it.

2. APPLY THE SPLITTING PATTERNS
   - WHEN YOU APPLY THE OBLVIOUS SPLITS, WHICH STORY COMES FIRST?
   - DOES THE STORY HAVE A SIMPLER CORE THAT PROVIDES MOST OF THE VALUE AND/OR LEARNING?
   - CAN YOU GROUP THE LATER STORIES AND DO THE EASY PART FIRST?
   - CAN YOU SPLIT THE STORY TO HANDLE DATA FROM ONE INTERFACE FIRST AND ENHANCE WITH THE OTHERS LATER?
   - DOES THE STORY HAVE A COMPLEX INTERFACE?
   - WHEN YOU APPLY THE OBVIOUS SPLITS, WHICH KIND OF INTERFACE COMES FIRST?
   - CAN YOU SPLIT THE STORY TO HANDLE DATA VIA MULTIPLE INTERFACES?
   - DOES THE STORY HAVE THE SAME KIND OF INTERFACE?
   - CAN YOU SPLIT THE STORY TO HANDLE DATA FROM ONE INTERFACE FIRST AND ENHANCE WITH THE OTHERS LATER?

3. EVALUATE THE SPLIT
   - ARE THE NEW STORIES ROUGHLY EQUAL IN SIZE?
     - YES
       - Try another pattern on the original story or the larger, less split story.
     - NO
       - Is there a simple version you could try?
       - Write that story first, build it, and start again at the top of the process.

4. BREAK OUT A SPIKE
   - CAN YOU DEFINE THE 1-3 QUESTIONS MOST HOLDING YOU BACK?
   - CAN YOU DEFINE THE 1-3 QUESTIONS MOST HOLDING YOU BACK?
   - CAN YOU DEFINE THE 1-3 QUESTIONS MOST HOLDING YOU BACK?


https://agileforall.com/patterns-for-splitting-user-stories/
MTAT.03.295 | Agile Software Development | © Mariana Falco 2022

UNIVERSITY OF TARTU
Organizing User Stories

How to envisage the entire product or service as a series of tasks which the user completes?

Two mechanisms:

• Kanban board
• User Story Mapping
Kanban Board

• It visualizes work and the process it goes through

• Generally more sophisticated than “mere” task boards
Story Mapping

- **Story mapping** consists of ordering user stories along two independent dimensions
  - **horizontal axis**  
    - order of priority (or “the order in which you would describe activities to explain the behaviour of the system”)
  - **vertical axis**  
    - it represents increasing sophistication of the implementation
- The first horizontal row represents a “**walking skeleton**“, a barebones but usable version of the product
- Working through successive rows fleshes out the product with additional functionality
User Story Map in 7 steps

1. Frame the journey
2. Build your story backbone
3. Identify and group activities
4. Break large tasks into subtasks
5. Fill in the blanks
6. Prioritize tasks and subtasks (but leave your backbone as is)
7. “Slice” groups of tasks into iterations

https://plan.io/blog/user-story-mapping/
User Story Mapping

https://plan.io/blog/user-story-mapping/

MTAT.03.295 | Agile Software Development | © Mariana Falco 2022
User Story Mapping

https://plan.io/blog/user-story-mapping/
MTAT.03.295 | Agile Software Development | © Mariana Falco 2022
User Story Mapping

The backbone is also the narrative flow!

https://plan.io/blog/user-story-mapping/
The backbone is also the narrative flow!

https://plan.io/blog/user-story-mapping/
User Story Mapping

The backbone is also the narrative flow!

https://plan.io/blog/user-story-mapping/
User Story Mapping

The backbone is also the narrative flow!

Priority +

Priority -

https://plan.io/blog/user-story-mapping/

MTAT.03.295 | Agile Software Development | © Mariana Falco 2022
User Story Mapping

The backbone is also the narrative flow!

Priority +

Priority -

https://plan.io/blog/user-story-mapping/
User Story Mapping

The backbone is also the narrative flow!

https://plan.io/blog/user-story-mapping/
### User Story Mapping: Example

```plaintext
<table>
<thead>
<tr>
<th>Organize Email</th>
<th>Manage Email</th>
<th>Manage Calendar</th>
<th>Manage Contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Email</td>
<td>File Emails</td>
<td>View Calendar</td>
<td>Create Contact</td>
</tr>
<tr>
<td>Compose Email</td>
<td>Read Email</td>
<td>Create Appt</td>
<td>Update Contact</td>
</tr>
<tr>
<td>Delete Email</td>
<td>View Email</td>
<td>Update Appt</td>
<td>Delete Contact</td>
</tr>
<tr>
<td>View File</td>
<td>Move Emails</td>
<td>View Appt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search by Keyword</td>
<td>Move Emails</td>
<td>View lis</td>
<td>Create basic</td>
</tr>
<tr>
<td></td>
<td>Send Emails</td>
<td>ti of appts</td>
<td>Open basic email</td>
</tr>
<tr>
<td>Create sub folders</td>
<td>Send RTF</td>
<td>Delete email</td>
<td>Open RTF email</td>
</tr>
<tr>
<td></td>
<td>email</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create RTF appt</td>
<td>Add</td>
<td>Accept/Reject/Tentative</td>
</tr>
<tr>
<td></td>
<td>View Formats</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Limit Search to one field</td>
<td>Send HTML email</td>
<td>Empty Deleted Items</td>
<td>Create HTML appt</td>
</tr>
<tr>
<td></td>
<td>Open HTML email</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Open Attachments</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set email priority</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mandatory/Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update data</td>
<td>Delete Contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update Address Info</td>
<td>Delete Contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search attachments</td>
<td>Get address from contacts</td>
<td>View Weekly Formats</td>
<td>Get address from contacts</td>
</tr>
<tr>
<td>Search sub folders</td>
<td>Send Attachments</td>
<td>Search Calendar</td>
<td>Add Attachments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>View Attachments</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example story map created by Steve Rogalsky
http://winnipegaglist.blogspot.com
```

---

MTAT.03.295 | Agile Software Development | © Mariana Falco 2022
Releases

• A release is made up of one or more iterations

• Release planning refers to determining a balance between a projected timeline and a desired set of functionality

• Iteration planning refers to selecting stories for inclusion in this iteration

• The customer team and the developers are both involved in release and iteration planning
Team activity

Your team has been hired to add new features to a social network

1. Choose your team’s favourite social network
2. One of you must play the role of Product Owner (PO). Others can play different roles (e.g. Engineering, Sales, Marketing, Finance...)
3. Agree on some epics to add (3 should be enough)
4. Create a user story mapping to plan at least the next release
5. Identify gaps, dependencies, technical requirements, and alternatives.
6. Share your results on Moodle!