Workshop : Effective User Stories

Fran O’Hara
fran.ohara@inspireqs.ie
Inspire Quality Services

www.inspireqs.ie - Agile Training & Coaching
- CSM, PSM, PMI ACP, etc.
- ISTQB Software Testing

Agenda

• Understanding User Stories
• Workshop exercise
• (Roles, Backlogs and Planning)
What’s Wrong with ‘Requirements’?

– Mandatory, Fixed, Hard to Change
– Feature Centric rather than Value Centric
– Specify the What, not the Why
– Critical bits hidden in the detail
– Expensive
– Slow us down
– Easy to verify, Hard to validate

Communication Modes

http://www.agilemodeling.com/essays/communication.htm
What is a User Story?

- A Written Card
  - For Planning
  - A Promise to Talk
    - Spec by Example
  - Acceptance Criteria
- It Defines
  - The Actor/User/Persona
  - The Goal/Action/Task
  - The Benefit/Value

Why User Stories?

- **User Centric** – what’s important to your customer
- **Story** – the power of narrative – emphasizes verbal communication and defers details
- Focus on the benefit, the value, what’s important
- Supports ‘pull’ of information as its needed
  - Iterative development and planning
**Vision**

**Theme**
- Strategic Objective
- May be 1 or more per release

**Feature/Epic**
- Large, Uncertain Stories

**Story**
- Sub-Sprint
  - Book flights
  - Rebook a flight I take often
  - Cancel up to 24 hours before with no charge
  - Emailed confirmation

**User Stories - CCC**

As a `<role>` I need `<action>` so that `<result>`

<table>
<thead>
<tr>
<th>Card</th>
<th></th>
<th>Card</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a customer I can search for products so that I can view their details</td>
<td>Value: Med</td>
<td>Risk: Low</td>
</tr>
<tr>
<td></td>
<td>Estimate: 3 pts</td>
<td></td>
</tr>
</tbody>
</table>

**INVEST**
- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

---

*Remember - different users/personas may have different ways of working and different objectives*
User Story Example – Email Attachments

Email Attachments.
As a user I want emails with attachments to go faster so that I can work more efficiently

Confirmation:
• User notices emails with attachments go at least twice as fast
• Works with attachments up to 10MB
• Works with up to 50 attachments

CONVERSATION:
• What if attachment already compressed?
• What if it’s a small file to start with?
• Should we store the compressed version?
• Should we allow user select compression options?
• Would up to 100 attachments be enough?
• Can each attachment be up to 10MB?

User Story Example – Hotel Reservation

Reservation Cancellation
As a user I want to cancel a reservation so that I avoid being charged full rate

Confirmation:
• Verify a premium member can cancel the same day without a fee
• Verify a non-premium member is charged 10% for same day cancellation but otherwise not charged
• Verify an email confirmation is sent to user with appropriate information
• Verify that the hotel is notified within 10 minutes of a cancellation

CONVERSATION:
• What if I am a premium member – do I have charges?
• When is a non-premium member charged and how much?
• How do these vary depending on when cancellation occurs?
• Do we need to send the user confirmation by email?
• When does the hotel need to be notified?
• What if the user has paid a deposit?
Another sample User Story

**Card:**
As a registered user, I want to log in, so I can access subscriber content

**Conversation:**

![User Login diagram](image)

**Confirmation (conditions of satisfaction):**

**Success** Valid user logged in and referred to the home page
a)Valid user name and password
b)"Remember me" ticked –Store cookie/automatic login next time
c)"Remember me" not ticked –Manual login next time
d)Password forgotten and a correct one is sent via email

**Failure** Display message:
a)"Email address in wrong format“
b)"Unrecognized user name, please try again“
c)"Incorrect password, please try again“
d)"Service unavailable, please try again“
e)"Account has expired –refer to account renewal sales page”

Right level of detail?
Purpose of confirmation/acceptance criteria

- define the boundaries for a user story/feature
- help the product owner answer what she needs in order for this feature to provide value (typically these are the minimum functional requirements)
- help the team gain a shared understanding of the story/feature
- help developers and testers to derive tests
- help developers know when to stop adding more functionality to a story

Level of detail in confirmation/acceptance criteria

- Try to keep relatively high level
- Detail goes into
  - Memory for immediate implementation
  - Internal team documentation (notes) – can include wireframes and screen mockups, validation rules, etc. – attach to story
  - (Automated) Acceptance tests (ideally defined to be understood by all)

Note: examples/scenarios make excellent acceptance criteria (see Specification by Example by Gojko Adzic & BDD)
Vertical Slices - High Level

Stories should represent a vertical slice through the system and should be completed in one iteration -

![Diagram of vertical slices through software layers](image)

What about Constraints/Non-functional requirements?

- Interoperability: The system shall be written in Java and all configuration data stored in XML files
- Performance: 90% of database searches will return results to the screen in less than two seconds
- Maintainability: automated unit tests must exist for all components, and run in their entirety every 24 hours
Dealing with Constraints/Non-functional requirements

- Write a ‘constraint’ story – drives creation of a test rather than a feature and a test
- Include the test in the regression suite
- If it relates to a specific story put into acceptance criteria of that story
- If it relates to all stories, put it into DoD
- Consider your traditional approach but adapted for incremental development e.g.
  - Performance:
    - Performance profiling in early sprints
    - Develop key functionality that can be performance tested early and repeat

Writing good stories - summary

- To identify stories, start by considering the goals of each user role in using the system
- Where possible write stories that cut through all layers of the application
- Create NFR cards and post beside team board or write NFR tests to ensure they are not violated
- Write smaller stories for functionality that will soon be implemented and high level stories for functionality further out
- Keep the user interface out of the stories for as long as possible
- Write stories in active voice
- Have the PO, rather than the developer, write most of the stories
- Keep stories short - their purpose is as a reminder to hold conversations
- Use acceptance criteria/examples/scenarios to clarify stories and know when you are done
Exercise

In teams of 3-4 write user stories for a system of your choice e.g
- A new customer focused airline website
- A bug tracking package
- A packaged holiday website
- A coin flip app for a mobile phone
- Your current system/product!

Procedure:
1. Firstly pick your system and consider the different users (personas) and include at least 2 in your stories (5 mins)
2. Write stories including acceptance criteria (20 mins)
3. Practice taking a large story and breaking it into smaller stories (5 mins)
30 mins, 10 mins feedback discussion

Product Backlog

- A list of all potential changes to the product (as PBIs) – owned by Product Owner
  - Shared across teams if multiple teams on a product
- Product Backlog Items (PBIs)
  - THE single source of requirements - usually a combination of
    - feature/story-based work (“let user search and replace”)
    - task-based work (“improve exception handling”)
    - NFRS (“10,000 transactions a second”)
    - bugs
- List is ordered by the Product Owner
  - Ordering considers value, cost, risk, dependencies
  - Top ordered items are well understood and agreed upon
- Only the people who do the work get to estimate it
- Initial backlog and ongoing backlog refinement
Q&A

Fran O’Hara
InspireQS
www.inspireqs.ie
fran.ohara@inspireqs.ie

Lean Competitiveness Fund
Enterprise Ireland
Funding Support for SMEs
Training, Consulting,
Coaching in Lean