Effective Writing and Testing of User Stories

Agile Tour 2015

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

www.inspireqs.ie  - Agile/Lean Training & Coaching
- ISTQB Software Testing
Agenda

• Writing User Stories

• Testing User Stories

• Common Pitfalls
Product requirements – sample structure

Vision

Theme
Strategic Objective
May be 1 or more per release

Feature/Epic
Large, Uncertain Stories

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

Premium Frequent flyer benefits
Book flights
Cancel flights

© 2015 Inspire Quality Services
User Story

As a HomeOwner, I want to regularly trim my lawn so its neat and tidy.
Co-Design

Problem Space
- Customers
- End Users
- Domain Experts
- Product Owner

Innovation Space
- Uncertainty
- Ambiguity
- Conversation
- Social Objects

Solution Space
- Developers
- Architects
- UI/UX Designers

© 2015 Inspire Quality Services
Communication Modes

- Synchronous Bi-Directional
- Asynchronous Uni-Directional

Diagram showing a spectrum of communication methods from cold to hot, and from low to high effectiveness. Methods include:
  - Face-to-face at whiteboard
  - Face-to-face conversation
  - Video conversation
  - Phone conversation
  - Email conversation
  - Audiotape
  - Videotape
  - Social Networking
  - Wiki
  - IM

Legend:
- Synchronous Bi-Directional
- Asynchronous Uni-Directional

Copyright 2002-2005 Scott W. Ambler
Original Diagram Copyright 2002 Alastair Cockburn
User Stories – Three C’s

**Card**

As a customer I can search for products so that I can view their details

- Value: Med
- Risk: Low
- Estimate: 3 pts

**Conversation**

- I can find all products
- I can use any search criteria I need
- Once found I can view details
- ....

**INVEST**

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

Remember – different users/personas may have different ways of working and different objectives.
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....

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)
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?
Workshop: Test a User Story

- User Story: As a frequent user of an online games store I want to find a game by entering its title so that I can quickly select one for purchase
- Acceptance criteria
  - Check I can enter the game name (or part thereof) and get a list of games that match my search criteria
  - Check I get prompted with suggestions of relevant games that match my search text as I enter it
  - Maximum search string length is 30 characters
  - Check that if there are more results than can fit on one page, I will be able to view the list in pages or sections
  - Check up to 5000 searches can be performed simultaneously with results returned in less than 1 second

- In teams, identify some test conditions/cases... e.g.

<table>
<thead>
<tr>
<th>Input</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Valid game name not in database</td>
<td>Display game not found</td>
</tr>
<tr>
<td>2. .....</td>
<td></td>
</tr>
</tbody>
</table>
## Sample Test Conditions/Cases

<table>
<thead>
<tr>
<th>Input</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User navigates to search page</td>
<td>Search box is highlighted with greyed instructions re entering game name</td>
</tr>
<tr>
<td>2. Valid game name</td>
<td>List of matching games is presented alphabetically</td>
</tr>
<tr>
<td></td>
<td>As characters are entered (search criteria becomes more specific) the list of matching games reduces until only final matched game(s) displayed</td>
</tr>
<tr>
<td></td>
<td>Search criteria are highlighted in the name</td>
</tr>
<tr>
<td>3. Valid game name not in database</td>
<td>Display game not found</td>
</tr>
<tr>
<td>4. Enter upper/lower/mix of characters</td>
<td><strong>Search works</strong></td>
</tr>
<tr>
<td>5. Update/delete search characters from middle of string</td>
<td>List of matching books is updated as search criteria amended</td>
</tr>
<tr>
<td>6. Select a game from the returned match list</td>
<td>Game details displayed for purchase</td>
</tr>
<tr>
<td>7. Enter search string with 30 characters</td>
<td>List of matching games returned</td>
</tr>
<tr>
<td>8. Enter search string with 31 characters</td>
<td>Error reported on string length</td>
</tr>
<tr>
<td>9. Valid search producing results over multiple pages</td>
<td>Correct formatting of list appears over multiple pages</td>
</tr>
<tr>
<td>10. Performance test response time with simulated load equivalent to 5000 simultaneous searches</td>
<td>Search results correctly returned with Response time &lt; 1 second</td>
</tr>
<tr>
<td>11. Check GUI against style guideline</td>
<td>Guidelines followed</td>
</tr>
<tr>
<td>12. Enter invalid data (using std. company validation rules) e.g. null, special characters, SQL string, maths symbols, etc</td>
<td>Appropriate error message reported</td>
</tr>
</tbody>
</table>
So we just test the acceptance criteria right?

Typical: Unit, component integration, story acceptance, exploratory (with appropriate regression)

+ what about...
story interaction, workflow, feature, system, system integration... and Non-functional...

© 2015 Inspire Quality Services
Quality Risk Analysis

Done at both Release level & Iteration level

- Gather the team
- List backlog items
- Identify quality risks
- Assess each risk
- Determine extent of testing
- Select appropriate test technique
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 the Definition of Done... or on the wall beside team board!
- Consider your traditional approach but adapted for incremental development e.g.
  - Performance:
    - Performance profiling/benchmarking in early sprints
    - Develop key functionality that can be performance tested early and repeat
Common Pitfalls

Stories too large, too detailed, solution oriented, ... (INVEST)
Test cases written too late
Test analysis/design not separated from test implementation
Insufficient collaboration
  - E.g. Tests not shared with product owner or developer

Tests expanding on the story scope
Non-functional requirements inadequately defined
Non-functional stories tested too late
Narrow interpretation of ‘acceptance’ testing
  - Need to test beyond stories and address other quality risks

Inappropriate balance between automated, manual documented and exploratory testing
Summary

Collaboration in creating/refining user stories – defect prevention

Consider quality risk – release and iteration levels

Beware ‘acceptance’ testing

Small stories, Vertical slices

Post NFRs beside team board – test!

Test cases = examples = ☺

Early testing – analysis/design as the first task
Estimation & Planning

STORY POINTS, VELOCITY

TASKS (IDEAL) HOURS

Sprint Plan
Vertical Slices - High Level

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