Quality in Agile Software Development

Handouts session on XP-days 2011

authors:
Johan Peeters
Alexander Helleboogh
Nelis Boucké

Technique 1: User stories

What?

A **user story** is one or more sentences in the everyday or business language of the stakeholder that captures what the stakeholder wants to achieve. User stories capture features. Each user story is described concisely, and should fit on a small paper note/card to ensure that it does not grow too large. It does not aim to be a complete specification. Instead, it is an invitation to a conversation with the stakeholder. Each user story identifies how to verify that it is done (acceptance criteria).

Template?

"As a **<role>**, I want **<goal/desire>** so that **<benefit>**"
"As a **<role>**, I want **<goal/desire>**" (short version)

Example?

“As a user, I want to search for my customers by their first and last names so that I can find them without knowing their customer number.” (Functional requirement)

“As a user, I want customers queries to deliver results in 2 seconds so that I can easily and fastly serve enough customers.” (Quality requirement: performance)

“As a customer, I want to be able to run your product on all versions of Windows from Windows 95 on.” (Quality requirement: portability)

References?

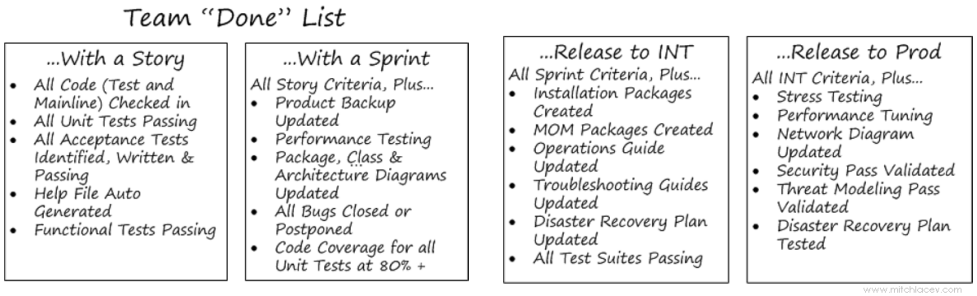
- Intro to user stories:
  - http://www.agilemodeling.com/artifacts/userStory.htm
- User stories to model quality requirements:
  - http://blog.mountaingoatsoftware.com/non-functional-requirements-as-user-stories
  - http://www.methodsandtools.com/archive/archive.php?id=113

Technique 2: Definition of done (DoD)

What?

Stakeholders' interpretations of when a feature is "done" might differ. For a developer, done could mean that the feature is implemented and ready for integration, optimization and user acceptance testing. For a customer/product owner this could mean the feature can start generating money.
The Definition of Done (DoD) defines the criteria to decide when a feature (or sprint or release) is done. It is an agreement between all stakeholders on what "done" means for a feature.

Template?



Example?

"A feature should never increase search response time" (Google, performance)
"A feature should be compiled to run both on Linux and Windows" (Portability)

References?

- What is DoD?
  - http://www.scrumalliance.org/articles/105-what-is-definition-of-done-dod
  - http://agilesoftwaredevelopment.com/2006/05/definition-of-done
  - http://www.rallydev.com/help/definition-done

Technique 3: Acceptance criteria

What?

Acceptance criteria are the conditions that must be met before a specific user story can be considered as complete. Acceptance criteria are usually noted on the back of the card describing the user stories. They are often used in BDD as a basis for automating the user stories tests in test driven development.

We use scenario based acceptance criteria following the format proposed in Behavior Driven Design (BDD).

Template?

Given: precondition (Environment)
When: actions or triggers (Stimulus)
Then: consequences (Response)

Example (for an online bookstore)?

User story: As a user, I want to search for books so that I can retrieve the book that I want to buy.

Functional acceptance criteria:

- Given a book title is offered by our system When a user performs a search for that title Then the result should contain the book.
- Given a book title is not offered by our system When a user performs a search for that title Then the result is empty.

Quality acceptance criteria:

- Given a normal weekday, When a user searches for a book Then a response should be provided in 1 seconds.
- Given peak load (e.g. around Christmas), When a user searches for a book Then a response should be provided in 4 seconds.

References?

- http://dannorth.net/whats-in-a-story/
- http://dannorth.net/introducing-bdd/

Technique 4: Abuser stories

What?

Abuser stories are similar to user stories, but instead of describing an interaction the system enables, it relates something that must be avoided. While user stories allow us to dream up our ideal system, abuser stories contain our nightmares. Like user stories, abuser stories should have a clear focus, fit on a small index card and be clear on how to verify that the nightmare scenario can be avoided.

Template?

"As a **<role>**, I want to avoid **<anti-goal/nightmare>** since this would cause **<damage>**"
"As a **<role>**, I want **<anti-goal/nightmare>**" (short version)

Example?

"As a user, I do not want to be known to the content provider whose content I use in order to assure my anonymity." (Quality requirement: security)

"As a content provider, I do not want the SuD's response time to drop below 1s as otherwise I cannot guarantee a sufficiently low overall response time." (Quality requirement: performance)

References?

- Abuser stories are to abuse cases as user stories are to use cases. These are the papers that pointed out that, sometimes, it is not sufficient to state what is desirable and the undesirable also needs to be described in software development:
  - McDermott, J., "Abuse-Case-Based Assurance Arguments", Proceedings of the Annual Computer Security Applications Conference, ACSA Publications, Silver Spring, 2001, also available at http://www.acsac.org
  - McDermott, J., and C. Fox, "Using Abuse Case Model for Security Requirements Analysis", Proceedings of the Annual Computer Security Applications Conference, ACSA Publications, Silver Spring, 1999, also available at http://www.acsac.org
- Coining the term: http://johanpeeters.com/papers/abuser%20stories.pdf
- http://www.gettingagile.com/2007/09/16/develop-architectural-needs-through-abuse-user-stories/