Article Found Online at: http://scrumcrazy.wikispaces.com

Basic Story Testing Styles

Summary: In order for a team to get really good at User Stories, they need to learn to be able to efficiently express their story tests. Story tests are also known as "User Story Acceptance Tests", "Acceptance Tests", "Confirmations", and "Test Confirmations." In this article, I list the most basic styles for expressing Story Tests:

- Style 1: Bullet Points
- Style 2: "Test with..."
- Style 3: "Test that..."
- Style 4: Given/When/Then
- Style 5: Specification By Example - Conceptual Form
- The Best Style: Mix and Match!

Style 1: Bullet Points

Description

- Jot down bullet points on a Story card or a Wiki to remember what the actual and expected system behavior is.
- Because this is a shorthand notation, team conversation context is extremely important. Said another way, you the reader of this article, may not get a lot out of seeing the below bullet point examples, because you weren't there for the conversation and its associated context.

Good for

- Small Stories, where the effort involved is ~2-3 days(which is the recommended size for Sprint Level User Stories).
- Tests that are somewhat obvious (in the example below, the old password obviously needs to match what's on file)
- Tests that the team will likely remember anyway, so they only need to jot down simple reminder notes.

Bad for

- Tests that are not likely to be remembered later by the team just reading bullet points.
- Tests that have logic or complexity that might be forgotten.

Example Tests

- old password
- new password
- confirmation password (must match new pw)
- new password complies with password rules

Style 2: "Test with..."

Description

- Jot down notes about what scenarios and/or data to test with.
- Because this is a shorthand notation, team conversation context is extremely important. Said another way, you the reader of this article, may not get a lot out of seeing these example tests, because you weren't there for the conversation and its associated context.
- Optionally, start the test with "Test with..."
- Optionally, add a "...verify..." clause

Good for

- Small Stories, where the effort involved is ~2-3 days(which is the recommended size for Sprint Level User Stories).
- Simple tests.
- Tests where the expected behaviors/outputs are fairly obvious
- Tests where the expected behaviors/outputs are simple (generally only happy/sad paths)
- Tests that the team will likely remember behaviors/outputs anyway, so they only need to jot down some data scenarios to create the different expected behaviors/outputs

Bad for

- Tests that are not likely to be remembered later by the team just reading simple "Test with" notes
- Tests where the expected behavior is not fairly obvious
- Tests where the expected behavior varies a lot by data scenario
- Tests that have logic or complexity that might be forgotten

Example Tests

1. Test current password with incorrect, correct, blank
    - with optional "verify" clause
        - Test current password with incorrect, correct, blank - verify error msgs
    - with optional beginning of "Test with"
        - Test with incorrect, correct, blank current password - verify error msgs
2. Test with special characters in current password
3. Test confirmation password with blank, matching, non-matching
4. Test with admin users, manager users, end users
5. Test with 0, 1, 3, and 5 companion products

Source

- User Stories Applied (Cohn)

Style 3: "Test that..."

Description

- Start off a sentence with "Test that..." and finish it by describing in prose what you want to test to verify that actual system behavior is consistent with the expected system behavior.
- This style requires a fair amount of prose, but it is also the easiest to remember later.
- These tests are better expressed conceptually, most often without specific test data.

Good for

- Beginning acceptance test writers, as this is probably one of the easiest styles to learn
- Simple tests
- Catch all tests -tests that can't seem to be represented well by other acceptance testing styles.

Bad for

- Tests where expected behavior depends on a lot of different test inputs or test setup.
- Tests where there is a lot of logic.

Examples

1. Test that, when a user enters an incorrect old password, they get an error message indicating incorrect credentials.
2. Test that three incorrect submissions of the old password within 1 hour results in the user being logged out from the system.
3. Test that when a user is logged out after 3 incorrect submissions that they also receive a message telling them to call customer service, along with the customer service phone number.
4. Test that, when a user clicks on the "Continue Shopping" link, it takes them to the home page.
5. Test that the order confirmation contains:
    - An order number
    - Estimated arrival date
    - Customer Service email address

Source

- User Stories Applied (Cohn)

Style 4: Given/When/Then

Description

- Word a test in 3 parts:
    - **Given**
        - Preconditions, test setup, test inputs, system state
    - **When**
        - Trigger or State Transition event
    - **Then**
        - Describe system behavior, expected outputs, and/or next system state

- Some people think Given/When/Then is good for testing work flow or state transitions. I think a picture of a hand drawn state chart or flowchart on a whiteboard would be a better solution to describe flow or state transitions.
- This style can get pretty wordy/prosy pretty quickly.
- This style also integrates well with one of the leading Acceptance Testing tools, Cucumber.
- Optionally, you can append each section with "AND" or "OR" to specify other conditions(See example 2 below)

Good for

- Tests that require a lot of preconditions, setup conditions/logic.
- Tests that require setup that is important or easily forgotten
- Tests that have a specific trigger.

Bad for

- Simple tests
- Tests that have unimportant preconditions
- Test that have simple or obvious preconditions
- Tests where a single Given/When/Then only describes one of numerous very similar test scenarios

Examples

1.

- **Given**
  - A user who has submitted an incorrect old password 2 times in the last hour
- **When**
  - The user submits an incorrect password (for the 3rd time)
- **Then**
  - The system logs the user out
  - The system displays:
    - the customer service phone number.
    - a message telling them to call customer service.

2.

- **Given**
  - A user that is logged in AND
  - the user is an admin user OR the user's account has been flagged by Fraud
- **When**
  - The user submits an incorrect password (for the 3rd time)
- **Then**
  - The system logs the user out AND
  - The system generates an email to the production support team with the following info:
    - user id of the user AND
    - the user's phone number on file

Source

- http://dannorth.net/introducing-bdd/

Style 5: Specification By Example - Conceptual Form

Description

- Create a table of testing scenarios that specify test inputs and expected test outputs.
- For the conceptual form, avoid using specific data, but instead describe the data.
- Similar to the "decision table" concept

Good for

- Tests where expected behavior depends on a lot of different inputs, conditions, or system states
- Tests where there are numerous different expected behaviors
- Tests where there are multiple different inputs and multiple different outputs
- Any test where it seems like a table would be useful to:
  - describe the test better, or
  - help explore all of the possible inputs and outputs for a test.

Bad for

- Simple tests
- Test where there is really only one input or precondition.

Examples
Expected System Behavior for sending emails through a third party service.

Service A is Up?	**Service B is Up?**	**Is attachment more than 150Kb?**	**Expected Primary Email Service**	**Expected Backup Email Service**	**Expected output if backup fails**
Y	Y	Y	B	None - queue for later sending	-
Y	Y	N	A	B	display error
Y	N	Y	None - queue for later sending	-	-
Y	N	N	A	None - display error	-
N	Y	Y	B	None - queue for later sending	-
N	Y	N	B	None - display error	-
N	N	*	None - display error	-	-

* = all possible values
- = Not Applicable

In order to understand what this table represents, here are the example business rules(You have to assume that there are good business reasons for these rules, otherwise, the complexity would probably not be justified):

1. Don't allow the system to ever try to send emails to a service that is down.
2. If both services are down, always display an error.
3. Never send emails with attachments more than 150Kb to Service A.
4. It is ok to send emails with attachments more than 150Kb to Service B. If Service B is down or fails for an email larger than 150Kb, then queue the email for later sending.
5. If Service A is up, it is the primary service.
6. If sending the email through the primary service fails, use the backup service, if there is one, and if the backup service is up.
7. If the Backup service is attempted and fails, display an error message.

The Best Style: Mix and Match!

Truly, the best style is to mix and match. You might have a story with a simple test and a really complex one. So, for one of the Story Tests you might use "Test that...", and another story test in the same story, you might use "Specification By Example - Conceptual Form." In addition, you can use the styles together. You can do Given/When/Then, where the Then statement says "Then Test that...". You could also put a "Specification By Example - Conceptual Form" table in the "Then" part of a Given/When/Then. Try to use the most efficient style for the kind of test you're trying to represent.

In Closing...

In all reality, if you adhere to the User Story Vision and keep all of your Sprint Level User Stories to 2-3 days in size, you'll probably use Styles 1-3 80% of the time. The remaining 20% will be expressed in Styles 4-5 or some other style, maybe one that you come up with. Try to pick the most efficient style, and always prefer the simpler styles over the more complex ones. Said another way, only use the more complicated styles (Styles 4 and up) when it just seems to make total sense to do so. If you find yourself writing tests that start to look like long run-on sentences or paragraphs, chances are there's a more complicated style that would represent your tests better.

I hope this gives you a good start towards writing good acceptance tests. Sometimes after story implementation begins, you'll think of a new acceptance test. If the effort(communicating the test, implementing code, implementing test) involved in the new acceptance test is really small(immaterial time), just go ahead and add it to the story. If it takes anything more than immaterial time, thus making the original story estimate inaccurate, then create a new story with the new acceptance test. This story can then be sized, prioritized, and implemented in the next iteration or sprint.

As always, be sure to inspect and adapt your practices. Try to think of what your view of the story was when implementation began, and compare that vision with how the story ended up after being implemented. If there are major differences between the two views, then your practices probably need tuning.

May the Sprint be with you!

A Look Ahead...

In my next article, **Advanced Acceptance Testing Styles**, I plan to cover:

- Style 6: Specification By Example - Concrete Form
- Style 7: Flowcharts
- Style 8: State Diagrams

Article Found Online at: http://scrumcrazy.wikispaces.com

Author: Charles Bradley, CSM, PSM I, Scrum Coach

Author email: chuck@emailchuck.com