Bridging the Gap Between Acceptance Criteria and Definition of Done

Sowmya Purushotham, Amith Pulla

sowmya.sudha@gmail.com, amith.pulla@intel.com

Abstract

With the onset of Scrum and as many organizations are trying to adopt Agile test processes and tools, the test resources on the Agile teams need to help the product owner define Acceptance Criteria during backlog grooming or the Iteration planning process. Acceptance Criteria may include a collection of Story tests that need to be passed for Story acceptance. The Acceptance Criteria and Story tests can be confined to the Story and may not consider all the aspects of the application or product.

The Definition of Done (DoD) is a wider and more comprehensive criterion that needs to be defined and met at a Story, Sprint and Release level in addition to Story acceptance criteria for the new set of features or capabilities to be released to production or users. Story acceptance criteria can include a series of functional tests to ensure the functionality meets the business or user needs, but does not necessarily define all conditions to say the Story is done and no more work needs to be performed by the team. For instance, the acceptance criteria can have list of Story tests or test cases that cover functionality, but a Story DoD can cover non-functional and technical requirements of a User Story like performance testing, browser compatibility, stakeholder acceptance, unit testing, regression testing and automation scripting.

The Scrum team(s) needs to collaboratively define DoDs at Story, Sprint and Release levels. The Story, Sprint and Release has to meet all the conditions defined in the DoDs for the Scrum team(s) to say the Story, Sprint and Release is Done and no further activity is needed.

This paper discusses the need for a clear Definition of Done at Story, Sprint and Release levels and sheds some light on few examples that are common in software applications and products, with a goal to help Agile and Scrum teams to better define their DoDs as a team and improve the overall quality of the application and releases.

Biography

Sowmya Purushotham is a software quality engineer with Clinicient Inc. based in Portland, Oregon. Sowmya has an extensive background in Agile testing practices and tools. As a software quality engineer, Sowmya works with a large global application development team in an Agile environment.

Amith Pulla is a QA Manager at Intel Corp, currently working at the Intel site in Hillsboro, Oregon. Over the last decade, Amith has been involved in developing software testing strategies and processes for applications mainly focused on sales and marketing. Amith also worked on developing test methodologies and techniques to meet the business's needs and timelines. As part of his Agile QA Manager role, Amith focused on improving and refining QA processes and standards for efficient software testing. Amith also worked as ScrumMaster and Project Manager on short assignments.

Amith has a M.S. in CIS from the New Jersey Institute of Technology; Amith got his CSTE and PMP certifications in 2006. Amith got his CSM certification in 2012.

1. Introduction

Acceptance Criteria which are defined by the product owner and then elaborated and further broken down into Story tests by the Scrum development team can be confined to a specific Story. In Scrum, the development team will have developers, testers and others working together to design, build, test, and deliver a product or application. The Acceptance Criteria defined for a Story does not take into account all the other stories, both under development and in production, of the application or product. For instance, if a particular Story under development may impact the other Stories, features or capabilities in the application, certain regression testing and other activities are necessary to make sure this new Story has no negative impact to the product, but works as intended. The acceptance criteria defined for a Story may not cover the regression testing aspects necessary to successfully integrate the new Story into the application, but the Story Definition of Done needs to cover the regression testing aspects.

A Definition of Done (DoD) is typically defined at Story, Iteration or Sprint and Release levels by the entire team, and DoDs are applicable to all Stories, Sprints and Releases. In Agile, "Done" means all the activities necessary to deliver a specific Story are completed. The Story Definition of Done captures all the conditions that needs to be met, once DoD is met it means no further tasks or activities are needed for that Story. A DoD looks at all aspects of the application or product and defines specific conditions that need to be met for the team to say the Story is Done. This may include regression and performance testing of the entire application, cross browser testing, test automation scripting, user acceptance, end user training or horizontal capabilities like BI.

In this paper, we'll discuss Acceptance Criteria and Definition of Done with a few examples, illustrating the importance of defining, documenting and agreeing on DoD. We'll offer some guidance on what to look for when defining Definition of Done for a Story, Iteration and Release.

2. Acceptance Criteria

2.1 Definition

Acceptance Criteria are defined as conditions that a User Story must satisfy to be accepted by a user, customer or other stakeholder.

In an Agile process, product owners write User Stories and discuss them with development team. The User Story is presented and a review process is started. The team collectively decides on the Acceptance Criteria and further elaborates into specific Story tests like functional test cases with Pass/Fail conditions. Acceptance Criteria can be a set of statements or conditions describing both functional and non-functional aspects of the product or application. Typically, there is no partial acceptance of a specific condition; it is either satisfied or not satisfied.

Depending on the product and application under development, Acceptance Criteria can also define boundaries and constraints of the User Story that determine whether the Story is working as expected and ready to be accepted. Acceptance Criteria needs to be defined before the Iteration and development on the Story starts, but can be refined during the Sprint based on the product owner's feedback. A User Story in itself usually doesn't provide enough details on the functionality and feature to the development team to start designing and coding; developers use the User Story and Acceptance Criteria to start any development work.

Generally in Scrum, Acceptance Criteria are discussed and defined as part of the product backlog grooming sessions where the whole team meets including the product owner and ScrumMaster. Sometimes backlog grooming sessions are also called Story Time sessions. A product backlog grooming session is not a formal Scrum ceremony like Sprint Planning or Daily Stand Up (DSU). Ken Schwaber, one of the founders of Scrum, recommends that teams dedicate five percent of every Sprint to this process (Scrum Backlog Grooming, CollabNet). There are other activities that happen during the backlog grooming process like ordering (prioritization) and estimation, but defining Acceptance Criteria is a key activity.

Acceptance Criteria needs to be clearly written in a language that customers, product owners, and the development team can easily understand. In Scrum, development team is the cross-functional team that has developers, testers, QA, UI designers, test automation experts and other roles. When defining Acceptance Criteria, there should not be any ambiguity on the expected outcome and expected behaviors. Well written Acceptance Criteria should be easily translated into one or more manual or automated Story tests.

2.2 An Example

Here is an example of a User Story:

As a User (Account Manager), using the letters feature, the User should be able to fax letters to a contact directly from the system instead of printing them out and manually send fax to the contact.

Questions and discussion for the product owner by the development team may include:

Will the User be able to fax multiple letters from the application to same contact?
Where does the User choose content to fax?
Can the User edit a letter for faxing?
Does the User need to manually enter fax number?

The issues and ideas raised in this session are captured in the Story's Acceptance Criteria.

The Acceptance Criteria could include:

User is able to see a 'Fax' button under Actions > Create a Letter menu
User should be able to choose a Client, Account and Contact
User can select a letter template and edit as needed.
User needs to click 'Fax' button to fax a letter.
User should see the fax chart notes pop up on clicking 'fax' button to choose a Fax cover page.

2.3 Story Acceptance

In Scrum, when the development team has completed the development and testing of the Story, the product features and functionality that are part of the User Story are demonstrated to the product owner showing how each item in the Acceptance Criteria has been satisfied. The product owner accepts the Story by verbal approval or by updating the project management, Agile Lifecycle Management (ALM) or QA tool such as Rally, Quality Center, RTC or VersionOne.

Typically, Acceptance Criteria is written as part of the Story in ALM tools; ALM tools have indicators that will show if a Story is accepted or not and who accepted it.

3. Definition Of Done

The "Definition of Done" (DoD) is the project or team's agreement stating that all the tasks or activities are completed for a Story, Sprint or a Release. It is a set of common conditions across all Stories, Sprints and Releases that states that no more work is left to be done for a Story, Sprint or Release. The primary purpose of a good DoD is for the Stakeholders and teams to better understand when a product increment can be called "Done." The DoD makes sure that the team adheres to a shared understanding of completion. Unlike Acceptance Criteria, Definition of Done is not specific to a Story or a feature, but defined such a way that it's applicable to all Stories. It also defined at a Sprint and Release level and is applicable to all Sprints and Releases that the team works on.

The DoD can also serve as a contract between a development team and its stakeholders. It can be seen as the team's standard of excellence or quality. The Definition of Done is defined and agreed upon as a collaborative effort between the ScrumMaster, development team, product owner and other stakeholders. The Scrum Guide (Ken Schwaber, Jeff Sutherland 2011) describes the Definition of Done (DoD) "as a tool for bringing transparency to the work a Scrum Team is performing." The Definition of Done is more about the quality of an overall product and user experience and less about the functionality and features. Typically, the Definition of Done will be written and reviewed before the first Iteration or Sprint starts, but it is regularly revisited, reviewed and updated in Retrospectives to meet customer, product owner and quality expectations.

Like the Acceptance Criteria, the Definition of Done needs to be clearly written in simple language that the development team, product owner and stakeholders can clearly understand. The Definition of Done should be written as a set of conditions like a checklist that the team can check off using the known information in the project management or tracking tools. The Definition of Done is usually posted in a prominent place in the online collabration and project management tools so that everyone on the team sees it on a regular basis and fully understands all the conditions. Having a clear DoD can foster team collaboration, common and consistent development practices, as well as better visibility for customers which leads to better overall product quality.

The book "Agile Testing: A Practical Guide for Testers and Agile Teams" talks about the importance of technology-facing tests in addition to business-facing tests. It says "Technology-facing and business-facing tests that drive development are central to agile development." (Lisa Crispin, Janet Gregory, 2009). When defining good Definition of Done statements at a Story level, the team needs to consider both technology-facing and business-facing tests that are not already defined in the Story Acceptance Criteria and add them to Story DoD as needed.

Some suggestions on writing good DoDs: (Rally Publications, 2013)

- Use conditions like, "all code checked in" or "unit test coverage > 80%".
- Use "Code review completed" instead of "code review".

3.1 Technical Debt and Quality Risk Considerations

A good Definition of Done is important for the velocity of the agile team and the quality of the delivered product. A poorly defined or incomplete Definition of Done can lead to gaps and defects in potentially-shippable software as development practices vary from Story to Story. The team needs to decide whether an activity or condition belongs in the DoD for a Story, a Sprint or a Release. As we move the conditions from a Story to Release level, the team is temporarily creating technical debt and adding to the risk. The team should try to keep as many conditions or activities as possible at the Story level and move them up to Sprint or Release level only if it's inefficient to do it at Story level. There are many factors that will influence these decisions. The team should ask themselves if a task can be done at the Story level, and if not, move it to the Sprint DoD. If it cannot be done at the Sprint level, it must be done at the Release level and needs to be added it to Release DoD.

For instance, if the teams want test automation scripting as part of the Iteration or Sprint Definition of Done, the team should consider whether test automation scripting can be added to Story DoD and completed with the Story. This means test automation scripting is removed from Sprint DoD and added to Story DoD, so that team completes test automation scripting before they call each Story done.

Performance or load testing can be a good example too. Performance or load testing can be added to Story DoD, but if it's inefficient or expensive for the team to run performance or load testing for each Story individually, the performance or load testing can be moved from Story DoD to Sprint DoD. This allows the team to run a single performance or load testing cycle towards the end of each Sprint on multiple Stories developed in that Sprint.

As part of defining the Definition of Done, the team should consider and discuss all the impediments that could prevent them from meeting the DoD. Based on the impediments, conditions can be moved from Story DoD to Sprint or Release DoD.

3.2 Examples of Definition of Done

3.2.1. For a Story:

- Code Completed and Reviewed
    - Code is refactored (to support new functionality)
- Code Checked-In and Built without Error
- Unit Tests Written and Passing
- Release Configuration Documentation Completed (if Applicable)
- Acceptance Tests written and Passing
- Pass all Non-Functional Requirements if Applicable (Cross browser compatibility tier 1, 2)
- Product Owner Sign Off /Acceptance
- User Acceptance
- Manual regression scripts updated
- Test Automation Scripts Created and integrated
- Localization (truncation, wrapping, line height issues, string array issues, etc.)
- Analytics (Non-Functional Requirements) integrated and tested
- Story level device support (big browser, tablet, mobile device) tested

3.2.2. For Iteration:

- Unit Test Code Coverage >80%
- Passed Regression Testing
- Passed Performance Tests (Where Applicable)
- End user training team hand-off
- UAT (User Acceptance Testing)
- Production Support Knowledge Transfer done

3.2.3. For a Release:

- Regression tests completed in an integrated environment
- Performance or Load Tests completed
- Open defects reviewed by stakeholders and production support team
- Workarounds documented
- UAT and end user training completed

3.3 Definition of Done for Enterprise-Class Agile Development

Many Agile teams today in large enterprise companies work in Scrum of Scrums setup or adopt an Enterprise-class Agile development method like SAFe (Scaled Agile Framework) or DAD (Disciplined Agile Delivery). In the Scrum of Scrums setup, usually there are two or more Scrum teams working on a common platform or different aspects of the same product. The teams need to adhere to common enterprise architecture, release cadence, UX design and platform constraints.

When multiple Scrum teams are working together, it’s important that the Definition of Done is same for each of the Scrum teams involved to avoid inconsistencies in the development and testing practices and quality of the application or product. Generally in Scrum of Scrums set up, multiple Scrum teams work in the environment as separate Scrum teams but release the code on a same platform and in same cadence; this means the Release DoD must be shared by all the Scrum teams in the Scrum of Scrums setup.

3.4 SAFe Definition of Done Example

(Scaled Agile Framework, Dean Leffingwell, 2013)

This is an example of Definition of Done from Scaled Agile Framework (SAFe), one of the most popular Enterprise-Class Agile Development methods adopted in the software development industry today. In this example, the Story DoD is defined at Story, Feature and Releasable Feature Set.

All items defined for Story DoD needs to be met before the Story is declared Done and then it will be considered ready to be added to a Feature. In the same way, all items defined for Feature DoD needs to be met before the Feature is declared Done and only then it will be added to Releasable Feature Set.

Story	Feature	Releasable Feature Set
Acceptance criteria met	A stories for the feature done	All features for the releasable set are done
Story acceptance tests written and passed (automated where practical)	Code deployed to QA and integration tested	End-to-end Integration and system testing done
Nonfunctional requirements met	Functional regression testing complete	Full regression testing done
Unit tests coded, passed and included in the Build Verification Tests (BVT)	No must-fix or Showstopper defects	Exploratory testing done
Cumulative unit tests passed	Nonfunctional requirements met	No must-fix defects
Code checked in and merged into mainline	Feature included in build definition and deployment process	End-to-end system, performance and load testing done
All integration conflicts resolved and BVT passed	Feature documentation complete	User, release, installation, and other documentation complete
Coding standards followed	Feature accepted by Product Owner or Product Manager	Localization and/or internationalization updated
Code peer reviewed		Feature set accepted by Product Management
No Showstopper or must-fix defects open		
Story accepted by the Product Owner		

3.5 Meeting the Definition of Done

Stories are marked or stated as Done by the team anytime during the Sprint or Iteration when all the items in the Story DoD are satisfied.

For Sprint DoD, at the end of each Sprint the development team conducts a Sprint review attended by the whole team. During the Sprint review the team reviews the Sprint DoD and checks off the Sprint DoD list.

The Release is stated as Done once all the activities to deploy the code to production are completed and new features are ready for the Users. The Story, Sprint or Release is not done until all activities and conditions are met and satisfied.

4. Bridging the Gap

The QA Leads and Test Engineers in Agile teams play an important role in bridging the gap between the Acceptance Criteria and Definition of Done. QA resources, with their extensive background in product quality and testing, can help their teams define the optimal Definition of Done that will help the team build higher quality software.

Below are the items to consider for the teams when discussing the Definition of Done.

4.1 Horizontal Capabilities

Every product or application has some horizontal capabilities. This is especially true for enterprise applications. Below we'll discuss few examples.

4.1.1. BI (Business Intelligence) and Analytics

In today's applications or products, BI and Analytics are an important aspect. For an application, the analytics may be tracking site usage, page views, user behaviors or transactional data. For a web application, each page can have some analytics code that sends data back to a central repository to be used for BI. In a client or mobile native app, generally there are several built in reports and transactional data tracking tools that are used for system analytics.

4.1.2 UI (User Interface) Design

If the team is building a large product or application that has multiple components or sub-sites, it's important to have a consistent UI design and UX (User Experience) across the site. Creating a common Definition of Done at a Story or feature level for all stories within a product can ensure that all teams use the same UI design patterns and guidelines.

4.1.3 Training and Help Files

In enterprise applications, the end user training material and documentation is usually created by a separate team within the IT organization. The training and help documentation must be updated for every production Release. Capturing this activity in a Sprint or Release DoD ensures that users have the most up to date documentation and training available after each Release.

4.2 Agile Engineering Practices

For Agile teams, it's important to adopt Agile engineering or development practices that will enable the team to have stable velocity while meeting the quality standards. Engineering practices need to use common tools and frameworks for greater efficiency and tracking. Typically, the engineering practices are captured in DoDs.

4.2.1. Unit Testing and Code Reviews

A unit test is a piece of code written and maintained by the developers that exercises other areas of the code and checks the behavior. The result of a unit test is primarily binary, either pass or fail. Developers write a large number of unit tests based on the functions and methods in the code. Unit tests are usually automated and can be run frequently as the code changes. Unit testing frameworks are used to write, maintain and execute the unit tests in an application. Unit tests are written as part of development or coding and this activity is typically part of Story DoD. NUnit and JUnit are two popular tools for unit testing. Some Agile teams refer to unit testing as developer testing.

Code reviews and pair-programming are popular in Agile development teams; many Agile experts believe that effective use of code reviews and pair-programming can improve the overall quality of the application by identifying and eliminating defects during the coding or development phase itself. The book "How Google Tests Software" emphasizes the importance of code reviews in the development process. The book says "Google centers its development process on code reviews. There is far more fanfare around reviewing code than there is about writing it" (James A. Whittaker, Jason Arbon, Jeff Carollo, 2012).

4.2.2 Test Automation

Building a robust test automation framework is critical for Agile teams to deal with regression issues. Test automation tools are used to automate functional, integration and system tests. Automated regression tests can find issues faster and help the team move faster by saving time spent in manual testing. Automated tests can be created as soon as the Story is ready for testing. The team can keep this activity as part of the Story DoD.

4.2.3 Continuous Integration (CI)

Test Automation in and of itself doesn't offer much value to Agile teams if the tests are not run frequently to find defects. A Continuous Integration model allows teams to check in code and relevant automated tests frequently, sometimes multiple times a day. Once the new code is checked in and a working build is created, the automated regression tests can run on the build and find the defects faster. As soon as the Story and a relevant test automation scripts are developed, they can be checked into the CI model. Again, this activity can be tracked in the Story DoD. The Story is not done until the automated tests are created and added to the CI and regression tests are run without any failures.

4.2.4 Test-Driven Development (TDD)

Test-driven development (TDD) is a software development process in which the developer writes an automated test case that states the desired behavior; the test should be initially failing. Then the developer writes the minimum amount of code to pass that test. The new code should be refactored to acceptable coding standards. TDD, which is intended to build the code right, as an engineering practice is encouraged in some Agile circles for better design and improved quality.

4.2.5 Acceptance Test Driven Development (ATDD)

ATDD (build the right code) is a complete paradigm shift from other Agile software development practices. In ATDD, developers build the application code based on User Stories and acceptance tests and automated tests are run on them to capture feedback from Users and product owners as the development is still in process. The automated tests are defined by product owners and Users using a WiKi mechanism, and then an ATDD framework like FitNesse or Cucumber is used to integrate the tests to working code using fixture code. ATDD integrates developers, testers and product owners and Users into the development effort in a kind of forced collabration. ATDD encourages team's acceptance of quality as everyone's responsibility.

4.2.6 Automated Deployments

Automated deployment or infrastructure automation is an emerging engineering practice in Agile teams and is considered a key aspect of DevOps. DevOps is a set of practices to improve collaboration and alignment between development and operations. Automated deployments allow teams to push working software or shippable increments to production in a matter of minutes. Automated deployments use frameworks and tools that can deploy and test code on servers in a fraction of time compared to traditional deployment methods; sometime these tools can deploy code to multiple severs simultaneously. These tools integrate with version control systems and the team needs to change the configuration and setup to work with a planned production release. The setup and configuration activity can be tied to the Release DoD.

4.3 Integration Testing

In large scale enterprise applications, there are multiple platforms and applications in the ecosystem that need to communicate with each other to bring the right data and experience to the Users. The application may need to communicate with several common enterprise User authentication, Business Intelligence (BI) and security systems. Integration testing ensures that all the integration points and data flows are happening as expected. Integration testing needs to be done for each release. This activity can be tracked as part of the Release DoD.

4.4 Performance and Load Testing

Performance and load testing are important aspects of testing that ensure the application meets the performance expectations of the Users under typical production load. It's not always efficient to run performance tests for every Story, but they should be run for key features to ensure that response times are below the expected thresholds as defined in the Acceptance Criteria. Performance tests can be run at the end of each Sprint when a set of stories are completed by the development team. Performance testing activity can be part of Sprint Definition of Done.

4.5 Mobile Device Compatibility (if applicable)

In today's mobile era, the application functionality often needs to be delivered not only for traditional form factors (desktops and laptops) but also for mobile form factors (tablets and phones). The Story DoD can capture activities that can enable a product's features for mobile devices.

Any mobile development and testing activities that are common for all Stories and features for a product can be added to Story or feature DoDs.

5. Conclusion

Acceptance Criteria and Definition of Done are two important aspects of Agile development that will help teams deliver quality to the users or customers. Teams should invest time and collaborate to define and document these and make them accessible and visible to the overall team including stakeholders, development team members and management.

The QA team members or Testers on Agile teams, with their knowledge of product quality and experience in developing test strategies, can help the team define and implement good Definition of Done conditions at Story, Sprint and Release levels, bridging the gap between Acceptance criteria and Definition of Done.

References

Lisa Crispin and Janet Gregory. 2009. Agile Testing: A Practical Guide for Testers and Agile Teams

James A. Whittaker, Jason Arbon and Jeff Carollo. 2012. How Google Tests Software

Rally Publications. 2013. Agile Definition of Done Guide
https://www.rallydev.com/sites/default/files/defining_done_guide.pdf

Dean Leffingwell. 2013. Scaled Agile Framework
http://scaledagileframework.com/

Ken Schwaber and Jeff Sutherland. October 2011. The Scrum Guide

CollabNet. Scrum Backlog Grooming.
http://scrummethodology.com/scrum-backlog-grooming/