Agile Testing Documentation. How much is enough?

A POINT OF VIEW
BY TEST ADVISORY SERVICES TEAM
Copyright Information

This document is the exclusive property of Mindtree Limited (Mindtree); the recipient agrees that they may not copy, transmit, use, or disclose the confidential and proprietary information in this document by any means without the expressed and written consent of Mindtree. By accepting a copy, the recipient agrees to adhere to these conditions to maintain the confidentiality of Mindtree’s practices and procedures.
Introduction

Agile software development has now become the norm, and over time, software testing has also evolved to firmly embed itself in the agile process. Some intrinsic activities pertaining to software testing have remained largely unchanged with suitable adaptations to align towards agile (such as test case design and execution, defect logging and triaging). However, one important aspect that has been heavily impacted (due to short release cycles and paucity of time) is documentation. This paper highlights some important considerations for documentation in the context of agile testing.

Waterfall vs agile – A comparison

The waterfall approach to software development was relatively easier as each activity pertaining to software testing was individually bucketed (i.e. test requirement gathering, test design phase, test execution phase, defect logging, etc.). Teams also had the luxury of being able to take weeks (if not months) to complete each activity. Consequently, each phase came with robust documentation as part of the deliverable - to be used in the subsequent phase. Every project was typically supported by a detailed test strategy document, comprehensive test plan, detailed test cases, test execution report, defect triage report, etc. (see Figure 1).

As the waterfall model spans a longer period between releases, it makes good sense to capture all details through relevant documentation. However with agile, one needs to consider the cost benefit analysis of the need for documentation vis-à-vis the productivity gain/loss it offers - considering the quick release cycles. If we were to map the above documentation as-is into the agile framework, the documents and the timelines would look something like what you see in Figure 2.
In contrast, 2-3 week sprints beg the question whether such extensive test documentation is needed. In fact, some evangelists of the agile methodology are of the opinion that documentation is a waste of time and not needed at all. On the other end, traditionalists believe that developing software without the necessary documentation for posterity, regardless of the development methodology, is a bad practice. The reality lies somewhere in the middle. Documentation cannot and should not be eliminated from the agile process. At the same time, teams cannot afford to dedicate time to document each and every activity in an already compressed release cycle. So what is the correct approach for test documentation in agile? The answer lies in addressing the following questions.

**Who will read the documents that are developed?**

Documents should not be developed just because they ought to be there. It’s important to first identify the users of the documents and then develop it based on the need.

**How much time is needed and who should contribute to the documentation effort?**

In the waterfall approach, each document has a single owner. In the agile world, it’s both time and effort consuming to follow a similar approach. Typically, a test artefact should be a live document with contributions from multiple stakeholders. The template should be consistent and the onus of keeping it updated must lie with multiple contributors, based on their responsibility in the QA cycle (both at sprint and release level). From a time standpoint, the effort spent on documentation should be factored into the velocity and should ideally account for a small component – approximately 5% of the sprint effort.

**How long will the document be in use?**

This is a key question that needs to be answered before embarking on documentation for agile. For example, the test strategy document must be prepared early on to ensure a lifetime that spans the entire project across multiple sprints/releases. It will, of course, need minor updates during the course of the project. On the other hand, artefacts such as test summary report are needed only for the day/run. Consequently, the question of how much effort should go into these reports arises. Should this be standardized at a daily level or can/should this be automated with no manual effort? Test plans are typically developed at a sprint level based on the features and should ideally be a work-in-progress document across sprints.

**What should be the size of the document?**

Bigger is not always better. The waterfall approach needs comprehensive documentation because the timelines are longer, multiple requirements are addressed at the same time, and there’s a good chance of multiple people will use/require the same document due to factors such as team size, attrition etc. Documentation in agile, on the other hand, must be concise and should only have the information needed by the user - no more. Both waterfall and agile can be supported by a test plan but the content will/should be vastly different. The test plan for a waterfall project is usually voluminous as it is meant to contain information on hundreds of test cases. For an agile project, the focus is on a particular sprint and documentation should ideally take approximately 5% of the sprint effort. Such an approach will result in precise and concise information that is alive and updated as and when required.

**Conclusion**

Each project is unique and so are documentation requirements. Based on the type of project and maturity of the team, some specific types of test artefacts should be created. Agile does not mean ‘no documentation’. It means creating selective and concise documents that aid the team in the present
while maintaining records for posterity. Following the practices listed below will ensure that testing teams create the right set of documents:

a. Address only the consumers.
b. Be precise and concise
c. Create a live document and accept inputs from multiple members
d. Lastly, create documentation only if there is a definite need based on the requirements of the project.