Testers Have Requirements Too!

Moira Tuffs
moira.tuffs@3dsystems.com

Abstract

Requirements from the testing team are often overlooked when business or user requirements are the focus of a program. Business or user requirements contain quantifiable objectives such as speed improvements of 30%, or the user should see plain English error messages. However, there are underlying assumptions that make these business specifications measurable. Using the speed improvement as an example - where is the 30% measured from? What is the baseline to show this requirement has been met?

Therefore there is a new demand for documented requirements from the testing team. In order to show a 30% speed improvement, the testing team must first establish a baseline and be able to measure the improvement against that baseline.

In this paper I will highlight the testing requirements based on familiar business or user-based requirements, along with some of the lesser requirements when the non-standard paths through the application are considered.

By the end of this paper, you will have the essential tools to critically examine what types of testing requirements should be added to enhance the business or user requirements relevant to your work, thus resulting in a more complete set of requirements. And we all know that complete requirements are the starting point to a great product.

Biography

Moira Tuffs is the Manager of the Embedded Software Quality Assurance team at 3D Systems. For more than 30 years she’s been involved in software and hardware products on both sides of the Software Engineering and Quality Assurance fence, gravitating to the QA side for the last 12 years. She has experience in projects from a wide variety of domains including spacecraft control systems, signal integrity and PCB design systems, as well as embedded systems.

Her career goal is to push the Quality and Reliability message into every aspect of product generation, such that Quality Assurance becomes everyone’s responsibility to attain and police.

Copyright Moira Tuffs August 2021
1 Introduction

This informative brief has come about from a simple question I posed at the beginning of testing our project on an existing printer. The question was, “How do we show evidence that the home position is calibrated for the print platform when we select the calibrate button on the GUI?” Another way in which to word this question is, “How do we know the button does what it is intended to do?”

For some background on this particular project, this printer started with some hefty marketing requirements for speed and reliability improvements. The customers of 3D Systems need their printed jobs to be created correctly, quickly, without need of repeats, and the printed parts must be dimensionally accurate. Not all of these specifications are within the testing roles of the Software Quality Assurance (QA) team; however, our job as QA is to make sure the firmware and hardware deliver to the requirements requested for the program. We do our best to relate to our customers or, “put our customer hat on” if you will and test the product like a customer would. Not only does this mean that the products we sign off on are user friendly, but it also means that when customers come to us with problems, we have likely already walked through such a scenario in our own testing. This process enables our team to deliver a better product that we are confident will meet the customer’s expectations.

As a team we dug into the information we needed to answer the original question, and the conclusion that we collectively came to as the QA team became, “We have requirements too!”

2 Why do we need testing requirements?

Testing of embedded systems comes with its own unique set of problems stemming primarily from the use of a closed system. Unfortunately, one of the headaches we find ourselves dealing with time and time again in the position of Quality Assurance is the ability to show evidence of testing. Log files can provide a wealth of information that may be useful to the developers. But they often have cryptic debugging information which you then need the keys to interpret them. Additionally, log files have not always been targeted as a QA resource, and they are often an afterthought to be a tool which the QA team can use. This creates a disconnect between what developers are producing and what the QA team can see of that product.

For example, sometimes the developers flood the log files with data, and it becomes a tedious scavenger hunt trying to identify the specifics of a task; when a task started or ended or when a set position was reached by a mechanical component before moving off again.

There are other times when the QA team finds a cryptic message: “Sent ack”. Sent to who? Did they receive it? What message was the “ack” associated with? There are many questions that are based off this one clue, creating unnecessary clutter that now someone in QA has to spend time and resources diving through to answer a part of the bigger question. While precious time is being spent on a scavenger hunt of sorts, there still remain deadlines and testing expectations that need to be met.

When put in such a position, it becomes important to define a set of testing requirements to hold ourselves and others accountable, not to mention avoiding unnecessary confusion in the future. Defining a set of testing requirements not only helps with testing down the line, but it also helps get the testing team in on the action earlier and starts the conversation flow.

Below shows a chart from the Journal of Information Systems Technology and Planning, paper by Dawson et al. showing the relative cost of fixing defects.
The results of the research performed by the IBM team have often been attributed to an increased focus on the upfront requirements and design phases. Adding testing requirements forces the testing thought process up stream thus saving all of us another potential lecture to ensure bugs can be found early on in the product development life cycle. It is generally accepted industry practice that complete clear and concise requirements will lead to an overall more complete product (Kar and Bailey 1996).

3 What types of testing requirements should be considered?

The types of testing requirements will vary for each problem domain and application, be it a simple software program with few paths through, to the complex embedded systems used with 3D printers. Requirements can also be resource driven as well as data driven. The following are highlighted as general areas to be considered; however, the overarching theme is to approach each program requirement with the same question – What information will the QA team need to be able to verify this requirement has been met at the end of the program?

3.1 Requirements that will allow the QA team to collect evidence of testing.

Let’s start by going back to the log file example. Often in the world of 3D printers some part of the printer needs to move to a set known position. For example, the build plate moves to its lowest position to begin printing its first design. The question we now need to ask is, how do testers know that a specific position was achieved? From there of course there are more questions that can be answered. In fact, the more detail that testers give on the products they are testing, the more evidence there is to use in future.
problem solving. Therefore, we ask ourselves, where is that evidence that you can post to a testcase to prove the results with evidence of testing?

In this specific problem, the answer to that second question is in the log files. The testing requirement would be to log the expected set position, alongside the actual position achieved, in units that make sense for the types of procedures that the user can perform.

How many times as a tester have you set a timer on a watch or a phone to track the time an operation takes? Wouldn’t it be helpful to have a summary of that operation once it’s completed in the log files?

Another solution to problems we face lies in the functions that our customers would also like to see in our products. There are some functions such as printing that the customer would like to anticipate when the task will finish, so they can plan any resource usage appropriately. The testing task for a print job is to verify reported start and end times are correct. Another testing task is to evaluate the accuracy of the estimated total print time versus the actual total print time. In the log files start and end times are reported, but no overall print time. That drives my QA team crazy, because now they must search the logs to match the start and end job names, note the timestamps, and do the math to answer the question of how long a build took to complete and how accurate the estimate was based on the total build time.

**Testing Requirement:** At the end of the operation report a summary of time start, end and duration.

### 3.2 Requirements for testing improvements from a Baseline metric

Releasing updated versions of printer firmware often includes increasing the speed to certain processes. For this example, we can go back to the original project that sparked the idea for this paper. Part of the requirement was to have a target X% speed improvement for printing. The first question to ask here is, what is the baseline? Without a baseline there is nothing to measure speed against, and therefore any increase or decrease that we make to the speed of the processes is irrelevant to the requirement. More examples of questions the QA team could ask next include: What material? What print job are we looking at specifically? Is it every print job that must improve? Can we average the print jobs to get an overall percentage improvement? The devil is in the details for this requirement, and the QA team is often the one to see the need for those details to perform their job function fully.

In this specific instance, the engineering team had already identified some processes that were more than ready for speed improvements. Great – testing this speed improvement should be simple. But come back to that question – what is the baseline? And remember the customer runs the printer as a system, they are not interested in the separate processes that combine to make the overall print time. So, putting on our “customer hats” so to speak, the customer will notice if their print job times are faster or slower, not that a certain piece is moving a fraction of a second faster than before.

**Testing Requirement:** Specify the set of print jobs, and the specific material for those print jobs to validate the speed improvements.

### 3.3 Requirements for testing the customer intent system.

Take a moment to consider what requirements there might be for testing the full product version of the hardware and software, and to be able to test the full system. On one occasion the QA team that I work with were asked to validate an updated version of a hardware part for the printer. We were told that no software change was needed to support this hardware change.
The purpose for the updated hardware part was to make a cleaning process more effective using a different wiping component. Unfortunately, the footprint for the updated part was larger than the existing part by only a few millimeters. This tiny change was not considered to impact any action that the user or software would take when interacting with the cleaning process.

Even with the previous hardware part, one step of the cleaning process instructs the user to remove the updated part for cleaning itself. The QA team found the new hardware part extremely difficult to remove as the location of a cross beam was now holding the new part in because of those few millimeters in extra length. The mechanical team had tested the removal and insertion of the part, but not as part of the full customer intended process. Just a few millimeters were what impacted the ability for a customer to clean the product. Thankfully, there was a specific test for the complete cleaning process. Therefore, a software change to move the crossmember before the cleaning process started was implemented, and tested, to ensure the customer intended process would go smoothly.

In early phases where a project contains mechanical, electrical and embedded systems, not all the parts (hardware, electronics, firmware and software) are available as a single testable system. Meaning that there may be further requirements on testing with a simulator or emulator environment that will depend on the individual situation. However, for most of the projects, testing the customer intent system is a critical component of the testing cycle.

**Testing Requirement**: Ensure hardware and software are updated as a system for regular testing cycles.

### 4 How to get the testing requirements into the project.

Programs have a defined content, and the schedule and end delivery date are driven from that content. As such, it is very important to get the testing requirements in early so they can be scoped alongside the development work and be incorporated into the schedule. In a continuously changing environment testing requirements would be added as stories and tasks to the backlog, to be pulled into the planning for the sprint cycles.

However, in most cases, the program requirements are defined in the Marketing requirements document. This document identifies the customer’s need and possibly the business’ case for the requested feature or product. A typical program life cycle has the engineering team then respond to the Marketing requirement, with their set of features and identified use case scenarios. In our agile world, these are listed in the JIRA project to create a complete backlog.

It remains advantageous as the QA lead on a program (or the QA manager) to review early drafts of the Marketing Requirements to build your understanding of the types of requirements that need firmly defined baselines. This provides a streamlined process that then can be organized properly to fit schedules and deadlines as well as bringing the QA team into the conversation of requirement documentation.

As the engineering team on a project is digging into these program and technical requirements, it is advantageous to find some allies on the development team. Working closely with the scrum master or technical program manager to make sure the testing requirements are understood reduces the risk of miscommunication later in the process. Also, scheduling some time in one of the planning meetings gives an opportunity to educate the engineering team as to some specific requirements the QA team have to enable them to show evidence of testing.

As the project then moves into the design phase, I recommend pairing a QA and Software Engineer to walk through the use cases. At this point it is important for the QA engineer to ask the very pointed questions, “How can we test what you are planning to implement?” and “What data will be logged as part of this operation?” and “Can you make this information simple to find?” These are important to ask given
Let's step through a simple example to elaborate on this process.

**Marketing Requirement:** User should be able to see the completed print time on the UI at the end of a print job

The breakdown of this requirement into architecture and design is outside the scope of this paper. Taking a shortcut, it’s not hard to end up with a set of JIRA items describing the development work as below.

- **JIRA Story:** UI needs to display print end time when a print job completes
- **JIRA Story:** Control code should report print job end time to the UI when a print job completes

The QA team can now ask the question how do they validate the end time on the UI? Which leads to the more in-depth question about the control code, and how that data point can be validated. Having a QA team member work the development team member on the control code story should conclude that additional information is logged at the end of a job in a format and log file that the QA team can find easily. Using keywords or tags is often a neat solution for these scenarios. At some point a JIRA item should be added for the additional work required to help the QA team verify the above JIRA items. An example of such an item could be

- **JIRA Story:** At the end of a print job, the control code should record in a log file:
  - Print start time
  - Print end time
  - Print duration

With the implementation of this additional JIRA item the validation of the original marketing requirement becomes an exercise in log file collection, search and matching the data to that shown on the UI.

## 5 Conclusion

Having a complete set of requirements for consideration at the start of a program is the central theme to this paper. The QA team is often the last to get a voice in this requirements process, which can result in a struggle to have the right tools and data to show sufficient evidence of testing. Making a well-informed decision on product readiness to ship has been impacted by missing early testing requirements. The goal of this paper has been to suggest some options and techniques to change that.

In my experience, the QA team need to be involved early in the design of the project to review requirements for testability and suggest additional functionality to aid in the QA testing process.
updated study from Dawson et.al is as relevant today as it was in 2010. Finding bugs early in the design process is a cost savings for any program. Well defined requirements from all roles in the team will lead to a cleaner overall program that benefits all parties.

Showing evidence of testing from engineering log files has long been problematic for the QA team. Identifying those features where the information in the log file would be useful to the QA team in proving a feature has been fully implemented and is working as expected, is a step to adding testing requirements into the development life cycle for the program. Giving the QA team a voice in that early identification adds the formal request for work to be included in the scoping and planning process, and clearly states the need for detailed information for QA to complete their task.

Identifying the marketing or program requirements where baseline information is needed to prove that requirement has been met shows another option to include the QA team early in the process and allow a complete set of requirements for the team to consider when gathering the full scope needed for implementation and test.
References


Kar, Pradip and Bailey, Michelle, "Characteristics of Good Requirements", Proceedings of INCOSE 1996, Volume II.

Acknowledgments

I would like to thank my reviewers Vanessa Tuffs, Amith Pulla, and Philip Daye for their unending patience, words of wisdom and encouragement. Without their help, this paper would still very much be in bullet points without a complete sentence in sight.