How to Find the Level of Quality Your Sponsor Wants

Sue Bartlett
STEP Technology
Sbartlett1 @ qwest.net
503.244.1239

Abstract
The level of quality a product will attain usually comes down to a business decision. While testers understand how to ensure a top-notch product, the customer sponsoring the development may balk once time and labor start translating into dollars. This paper's purpose is to help you recognize customer needs — whether stated or implied — and turn them into an effective software testing process. It explores requirements gathering techniques, test efficiency, communication skills, and persuasion tactics.

- Do you have "quality" goals?
- How to turn fuzzy demands into concrete test cases
- A methodical approach to prioritizing testing around quality goals

for STAR West 2001

Introduction

There's this project. The technical lead has a PP, an RS, and a WBS for this DRM test tool that's going to be built in .net using C# and XML and he's really excited. And you have NO CLUE what he is talking about, but you are supposed to figure out how to test it.

In this paper, I'll talk about how to focus yourself to do effective testing. I'll cover where to start, what to look for, what and who are your resources. There is a level of understanding you need about the quality goals for a project. I'll give some examples of ways to figure that out without even asking questions. Or, who to ask if you could and then what to do with the information to be effective in your testing efforts. Some of this is similar to risk based testing, which you can use in a complimentary fashion. The way it is different is in its focus on the sponsor's expectations. Regarding .net and C# - sorry, you are on your own.

1 The Start

1.1 Where do you start? What do you have to work with? What deliverables do you receive? What do you normally do when you start a new project? How do you determine what to test? This depends on what you have to start with. What information do you have about the project? Do you have written requirements? If so, are they current? How complete are they? Do you have a "customer" or "user" you can query directly? Do you have knowledge and experience in the domain of the software you will be testing? If it is a follow-on to an existing project, do you have access or knowledge of previous customer complaints? Or, worst case, are you given the software with no documentation: an over-the-wall release to you, and being told to test it?

1.2 As you can see, there are lots of possibilities – a continuum of insufficient knowledge to do your job perfectly. But still you persevere. If you are lucky, you start with some written requirements that reflect the sponsor's reality. Or at least the lead programmer's view of reality. Or not. So, you either extract or make up the testable requirements or functionality that you plan to test.

1.3 You take that information, break it down into functional pieces you can write test cases from and test everything. Right? There are hundreds of books that tell you how to do that. (See the reference section at the end for a few.)

1.4 Life intrudes. You can't test everything. They won't even give you time to test what you would consider adequate. You've given them all the ritual arguments and they have said, "There's no extra time – it's a hard deadline. It goes now with X or it waits until the next release cycle in 3 months which is not really an option here." More testers to help? "No, so sorry, but Kristy's out on maternity leave, John's on sabbatical, and well, there is that guy in shipping who could probably give you a hand if you want." "Well then, can I get some parts of it early on to spread the testing out more?" "Sure! We can give you the database a week before the scheduled integration! Will that help?" "Ah, thanks…. Every LITTLE bit helps."

1.5 So, here's plan B. You figure out what the most important quality goals are for the sponsor of the project. As soon as you can articulate them, start by emphasizing the priorities to the team, to management and in testing. If you don't get to everything, well, some things will ship without being tested or wait for the next maintenance cycle. This is what the rest of the paper is about.

2 **Test Scoping:** What is a “quality” goal or requirement? Why do we need them?

2.1 You’ve just been presented with several reasons why you need quality goals in the previous section. There’s a bigger one that hasn’t been addressed yet. It’s the one that really defines the whole test scoping issue. The argument of how much testing do you do and what is really good enough. James Bach has written much on the subject, and I suggest taking a look at some of his arguments if this subject is new to you. But as much as the perfectionist in us all hates to admit, when it comes to the amount of testing to do, it all comes down to a business decision.

2.2 It’s pure business. How much time can we afford to spend in development? How much time can any one tester be applied to this project versus another project? How much revenue will this project bring us? What’s the tradeoff between one level of quality and another? (Note: this does not address the level of quality built into the product by the programmers – there are many techniques that can be used there, but that is outside the scope of this paper. See the reference section for ideas.) How many technical support calls will we get if we don’t fix this bug and how much does each call cost us? What happens if we don’t look here? How many customers will we lose? How many customers will we gain if we get this piece of functionality in, even if it isn’t perfect? What’s the relative market share of this particular configuration? What about the editors that review our products – how will they react? Is it still within “competitive quality”? And so on.

3 **Sponsors:** Looking at the business decision from the Quality point of view, the arguments can be limited a little more by being very clear on what is important to the sponsor of the project. Who your sponsor is will depend on your company and your business. One way to determine that is to “follow the money” or the power structure.

3.1 For example, when I worked at Tektronix, the argument could be made that the sponsor was whoever was responsible for allocating the budget for our division. However, in this case the sponsor for each color printer I worked on was a combination of the marketing product manager, the project manager and the director of engineering. That’s one group of people who directly decided which printer projects would be staffed and what new functionality would be developed. One could also argue that the sponsor was the customer who ordered or paid for the printers. Occasionally the service organization would attain enough visibility that individual customer complaints would enter the company consciousness. For many products in the “consumer” category, marketing people develop a composite view of customers for the purposes of forecasting future needs and addressing complaints.

3.2 Other types of companies, like the one I currently work for, have several different kinds of sponsoring people or companies. My team works specifically on the turnkey software projects, and that means that a sponsor can be one person who specifically has the full authority to hire us, define requirements, and make acceptance decisions. Generally though, the person we work most closely with is a person delegated by the sponsor. We also have clients whose sponsor is a little less direct. The sponsor might actually be the head of one division, who is allocating money to another division specifically for this project, giving some decision making authority to a representative, but holding full and final authority themselves. This makes it a little more difficult to divine the intent of the sponsor since often decisions come filtered through a chain of command and direct communication is only invoked when issues are elevated. I’m sure you will agree this is not the best time to discover true intent.

3.3 There are political situations that hide the state of sponsorship. Many times you think you are dealing directly with the final decision maker, only to discover after the project gains a little visibility, that there is someone else in the organization with veto power who has been left out of the process. This actually happened once to a client and the poor guy hadn’t even realized this was a possibility, till he was the recipient of the veto.

4 A quality goal is…

4.1 You have identified the sponsor, so now what? You need to figure out what that sponsor wants. What is a quality goal? It is some statement or requirement that describes your sponsor's idea of which aspects of quality are important to her. So think about some of your most favorite quality goals and I'll remind you of a few that rank right up there with those oxy-morons like, "military intelligence", "presidential dignity", and "intense apathy". How about, "easy to use and intuitive UI"? Then there is, "fast performance" and "high quality", "scalable", "doesn't crash".

4.2 I'm not saying that generalities are not useful. "Useful error messages" is actually downright specific in these days of unintelligible shutdown errors. It tells you that a) the sponsor wants some error handling and b) they want the user to be able to understand what is happening. That's heads and shoulders above what many programs give you these days. For a list of books to help you understand good requirements (not in the scope of this paper,) see the reference section.

4.3 For our purposes here, we are not looking for especially testable quality requirements, but rather clues toward the sponsor's idea of quality priorities. Some requirement specialists suggest using two questions to help prioritize requirements: 1) How important is this functionality and 2) How bad would it be if this were not implemented? This is similar to your task. Here are a few reasons why quality goals can be vague. A) A sponsor may not have the vocabulary to translate generalities into testable requirements. If she mentions performance and scalability – you know there is an issue there, and you have to delve a little deeper to find out what that means to her. B) A sponsor may not understand the limitations of the technology (sometimes neither do the programmers.) C) Sponsors may not understand software products, let alone the need for testing. D) Entire subjects may not have come up in the discussion of the project.

4.4 Another problem with unspecific goals is that people seem to use them as a litany and don't really think about them specifically in relation to their needs. They are also generally the first ones out the window when time and cost enter the picture.

4.5 Additionally, quality assumptions are rarely addressed. Some things are never mentioned because the sponsor just expects a certain level of quality. Maybe they've never had to articulate it before. Many of us labor under the misconception that we all know it when we see it. But here's the rub: What we all recognize is the absence of quality, rarely the presence of it. You assume a certain level of quality too, but you, as a quality professional are probably more aware of your assumptions since you've probably had to defend most of them.

4.5.1 For example: I think it should be obvious to everyone that a program shouldn't crash. A program gets confused in the middle of an operation, and an OS error message appears, talking about a forced shutdown. I am confident that most of you, sometime during your career, have had to convince someone else that these sorts of things must be fixed before shipping.

4.6 Another thing I should mention is something that comes into the realm of cover your ass (CYA). This is an unmentionable part of the business decision. There are always going to be areas of testing that an experienced tester knows will come back to haunt her if she doesn't test them. In addition, there are certain organizations that get into blaming. For those organizations you may have to get creative to get more time for testing because only you will be able to determine how much testing must be done and what bugs must be documented to keep blame from falling on to you. One consultant I know used to talk about having a horrendous looking bug that he kept in his back pocket for when he needed more time to test and management was hell bent to ship. It was one of those old OS bugs that would guarantee a "blue screen of death" with the right keystrokes. I don't know how to do that on the current OS's, but if you work in "one of those" organizations, it's not a bad

idea to find one. This could also be a bug in the current system that you know for a fact will take no time or risk to fix, but looks bad – just don't forget to get it fixed before it ships.

5 **Deriving quality goals.** How do you derive useful quality goals? In the appendix, there's a list of questions to help give you some ideas of things you can ask yourself or others on the project. Right here, though, I'm going to give you some examples of ways quality goal epiphanies have come to other colleagues and myself.

5.1 With web applications, supported browsers are a big deal since much of the time you have to write the same functionality separately and specifically for each browser. Many clients seem to start out wanting all versions supported and are beat down to a realistic grouping when you start explaining the dollars involved. However, one client seemed to think MS®IE5 on MS®Windows was enough for the first release and we were caught off guard when one of us overheard a mention that one of the client's investors used a Macintosh®. We scrambled. It crashed. We fixed it. So there's a question for your sponsor if investors are involved: "What configurations do your investors use and will they be monitoring the progress of this project on those computers?"

5.2 We had a client that wanted a sample application that used their software. It would be both a marketing tool as well as a tool to jumpstart their customer after purchase. It was clear to us that since it was a web application, browser and OS combinations would be very important. As you are no doubt aware, the combinatorics will kill you in testing. However, trying to negotiate down to a small set was getting us nowhere fast. He wanted them all and didn't understand why it should mean more time or cost. So my colleague put together a matrix with all the operating systems and browser versions he thought relevant and talked them into categorizing them all into a five step rating system from high to low priorities. The deadline was a drop-dead date for incorporation into another company's release so there was no missing it. The matrix allowed him to test as far down into the matrix as he had time for and also gave us a way of determining which were the most important bugs to fix.

5.3 There was a very complex system, previously developed. The client had a long list of issues to be addressed, particularly error handling additions, but a fixed budget. Neither the client, the programmers, nor management seemed to understand the fact that the complexity of the system meant that every change that was made had a chance of unintended repercussions. Two things happened that helped relieve the situation. First, the QA lead decided to map the admin system out in Visio®. No one had done that before. Even the original developers were surprised at how some things were connected. All of a sudden, people started recognizing the impact of their changes, not to mention additional places to make changes. After sifting through a series of meeting notes and emails, our QA lead decided to ask the client what were the most critical issues. Data issues, was his response. That began a search through all the requests to determine what those issues were. We found that the integrity of the data was being compromised during uploads and downloads, particularly if they happened too close together. The end result was prioritized testing around data integrity, new rules around how data was allowed to be moved and a concentration of the programming effort to the error handling around the data movement, leaving other areas unrelated to data integrity to a later contract. The lessons: what should have been obvious from the beginning was obscured by the pure volume of issues and lack of focus; and there's no better way to make your point than with a visual aid.

5.4 Have you ever had a client that was so easy to please it was aggravating? One client company didn't really care about the quality at all. What they were interested in was proving out their idea and selling it to someone else. Yes, that was in early 2000 when everything was overvalued. The problem was that we didn't figure this out right away and we didn't understand why we were getting conflicting messages around the quality goals.

5.5 In several instances, clients have said that performance wasn't that important. Sometimes it's at the end of the project and resources are scarce, sometimes it's at the beginning. Here's where assumptions will get you in the end and CYA testing is important. Of course performance is important – they just don't know how to rate it with other risks. And that's because there is an underlying assumption that it will work well enough. Unfortunately, that designation is different for everyone. Often your sponsor or direct client, the one who pays you or allocates your budget, is not the actual user of the system. The user of the system has a certain expectation of what "good enough" performance is. Much of the time the user is not involved in defining the system. Or there are too many users to discern particular preferences. In one instance, the user who complained was the client's technical support. The sponsor had not sought out input from technical support for the project. What we have learned from this is that no matter what the client says, we must do at least one round of CYA performance testing. We make sure it operates at a level of performance we think should be expected. Perhaps we bring particular bottlenecks up to the client, to allow them to estimate their own risk and to document the issue. This helps makes the risks real to them.

5.6 There are systems I've read about in the news. Part of it includes a handheld device like a Palm or PocketPC. It includes an application that allows a doctor to write a prescription and electronically send it to a computer, which uploads it to the patient's record and to a pharmacy of the patient's choice. As part of that app, it does a check for the other medications the patient is taking and warns of possible drug interactions if relevant. Well, here's a possible conflict. Let's say the doctor doesn't buy this device. Instead, let's say it is supplied by a drug company, or at least is subsidized by a drug company. Another incentive for the doctor to use it is that insurance companies give discounts to doctor's on their malpractice insurance. There are a lot of documented deaths from drug interactions as well as prescriptions filled at the wrong dosage due to doctor's legendary illegible handwriting. The drug company is motivated to do this by the promise of being able to have their drug information and ads automatically downloaded to the doctor's handheld device. I see three possibly conflicting sets of requirements there: one sponsor is the company who makes the software that goes on the device along with other services they are trying to sell. Another is the subsidizing drug company. The last are the end users themselves – the doctors.

6 **Process:** You have the quality goals. You have the priorities. You have a time-line, and (limited) resources. What do you do with that information? This is where great QA resources really differentiate themselves. You develop the right process, which focuses on critical pieces of the product at the highest leverage point in the product development life cycle. In some cases you can figure out quality goals where design issues are involved and use that knowledge to test for those issues when design changes can still be made easily, or even better, make sure it is taken into account before the changes have to be made.

6.1 For example, if internationalization is important, checking for UNICODE builds and non-embedded strings at regular intervals during development would be a high leverage QA activity.

6.2 As another example, if the product must support both MS IE and Netscape, testing even the mockups in both, is a continual reminder to the developers that both implementations must be considered. If it's not designed it in from the beginning, it's a major headache and deadline killer at the end. It's amazing how many developers ignore it (probably hoping Netscape will get dropped from the requirements.)

6.3 Methodical approach based on heuristics

6.3.1 Figure out what you have.

6.3.2 Who do you have access to for: a) domain knowledge; b) competitor information; c) sponsors; d) user expectations; e) political issues and assessment of the project or the sponsor.

6.3.3 Who do you trust, or rather, what's your assessment of each contact's level of realism?

6.3.4 Determine minimal CYA test activities.

6.3.5 Assign priorities based on sponsor risk or areas of interest.

6.3.6 Sketch out a complete test plan, but focus first on high priority, CYA activities and only minimally address the rest in the beginning. Flesh low priority items out as you get to them or in moments of idleness, like between builds.

6.3.7 If there are issues you can't resolve – remember quality is a business decision. Lay it out as you see it, including the risks, to the team and the sponsor and let the sponsor decide.

6.3.8 Adjust and revise as you go along. Remember that they can cancel functionality too. You'll need to red flag it if it's on the your high priorities list in case someone else isn't paying attention to the sponsor. Pay attention to the sponsor's questions. They will often reflect her interests, anxieties, and view of the risks.

Conclusion

I've talked about why you want quality goals, some ways to figure out what they might be and how to incorporate them into your testing focus and the team's focus. Hopefully this paper has given you new ideas and a more productive focus in situations with tight schedules and hard end dates.

A sponsor's time and cost issues discussed here seem obvious – but when you are thinking about data architecture and difficult design issues or failing third party software – how easy is it to always remember the basic customer needs? When we do communicate quality goals ahead of the detailed planning, design or coding, we have a better chance to avoid a quality mismatch at the end. That means meeting schedules, covering costs and making a profit will have a better chance of success. I want to be clear that I am not advocating "less than adequate" testing. However, there are gray areas and choices have to be made every day. You still have to use your best common sense and your awesome negotiation and persuasion skills to get the time you need.

Disclaimer time: Is THIS a silver bullet? It can't be, it's based on your judgement and it is particular to each project. And guess what? You can be wrong. But you can also be right and no matter which, you'll be more aware than you were before of the nature and focus of the sponsor. This can significantly increase your positive effect on a project. It's another tool in your QA toolbox.

Appendix

Here is a list of questions, in no particular order, which you can consider or ask in your grand quest for the elusive Quality Goal.

How do you think the software will be used?
What are the users needs apart from the sponsor's?
How does the sponsor say the software will be used?
Who will be supplying the funding for this software?
Who is paying you and your organization or allocates the budget?
What kind of data are they collecting?
How important is loss of data and at what level? All data or just some? Is any particular stage of data transformation more important than others?
What possible data issues could be underlying other sets of functionality?
How is the data to be backed up?
What are possible security issues for the sponsor, for their customer, for the end user?
What issues could there be that might end in a lawsuit and how likely is that?
Are they close to doing an IPO?
Are they at a make or break point for the company with this product?
Is a magazine editorial review more important to them right now than the target user?
What possible issues are there around connection/ feedback/ response times?
What environment will the application be used in?
What versions of the OS/ browsers are expected to be used, by end users, sponsors, others?
Have they done any market analysis that might dictate some quality goals?
What are the possible compatibility issues they might have?
Would Netscape be "nice" or is there really more of 10% of their market using it?
What part of this product is the actual differentiation from its competitors, and does it have a different set of quality goals from the main product?
How many users are on the system now? How many do they expect on what timeline? Is it realistic or possible?
What configurations do your investors use and will they be monitoring the progress of this project on those computers?
Are you seeing evidence of conflicting goals from the sponsor? What is your best guess?
Are you aware of the sponsor's market strategy or positioning?
Maintainability is an often overlooked, assumed quality goal because it is difficult to articulate.
Is the sponsor's view of the customer profile similar to yours?
Is the technology you are using bleeding edge and are there unidentified risks associated with it?
CYA issues could be: when a sponsor is not aligned with end user's or their customer's goals; sponsor's ignorance of their true market; not recognizing the political shakiness of their situation; haven't thought through their business model.

Further Reading References

Testing books:

Testing Computer Software, 2nd Edition 1999, by C. Kaner, J. Falk, H. Q. Nguyen, Publ: Wiley
Testing Applications on the Web, 2000, by Hung Quoc Nguyen, Publ: Wiley
The Craft of Software Testing, 1997, by Brian Marick, Publ: Prentice Hall

Requirement books:

Exploring Requirements, 1989, D. Gause, G. Weinberg, Publ: Dorset House
Mastering the Requirements Process, 2000, S. Robertson, J. Robertson, Publ: Addison Wesley

Process Improvement Books:

Almost any book by Gerald M. Weinberg, particularly his *Quality Software Management* series.
Software Engineering: Theory and Practice, 1998, by S. L. Pfleeger, Publ: Prentice Hall

Sue Bartlett

Sue Bartlett is the QA Manager at STEP Technology, a software company that builds high performance, scalable, eBusiness and Enterprise applications. She has over 18 years of experience in software quality assurance and testing, software process improvement and software development. Her domain experience ranges from Computer Aided Design (CAD) to color printers and most recently, web, and other business applications. Over half of her career has been at a technical lead or managerial level.

Sue has a B.S. in Computer Science from the University of Wisconsin, Madison. She is a past President of the Pacific Northwest Software Quality Conference. She has taught software testing as an adjunct for Portland State University, OCATE and Oregon Graduate Institute in Portland Oregon.