28TH ANNUAL

PACIFIC NW SOFTWARE QUALITY CONFERENCE

OCTOBER 18TH – 19TH, 2010

ACHIEVING QUALITY IN A COMPLEX ENVIRONMENT

Conference Paper Excerpt from the

CONFERENCE PROCEEDINGS

Customer-Driven Quality

John Ruberto
Intuit, Inc
John_Ruberto@intuit.com
blog.ruberto.com

Abstract

Today, developing software is much more complex than in the past. In this complex world, how do we define quality? We define quality goals with an ever increasing number of "ilities". We perform all kinds of testing: acceptance, performance, functional, concurrency, stress, exploratory, stability, build verification, and finally, regression testing.

Given all of this complexity, it can be difficult as a quality team to define quality. The true definition of quality, however, lies with our ultimate quality consultants, our customers.

One of our values at Intuit is "Customers Define Quality", and this paper shares how we interact with our customers at every stage of the life-cycle to understand their definition of quality. This paper describes the preparation, infrastructure, and practices for the Customer-Driven Quality frame-work.

Biography

John has been developing software in a variety of roles for 25 years. He has held positions ranging from development and test engineer to project, development, and quality manager. He has experience in Aerospace, Telecommunications, and Consumer software industries. Currently, he is the Quality Leader for QuickBooks Online, a web application which helps small business owners manage their money. He received a B.S. in Computer and Electrical Engineering from Purdue University, an M.S. in Computer Science from Washington University, and an MBA from San Jose State University.

Prepared for the Pacific Northwest Software Quality Conference, October, 2010

Introduction

A long time ago, in a conference room far away, our QA organization got together for an offsite meeting, a chance to get away from the hustle and bustle of our day-to-day projects. A chance to step back and think about what is important and perhaps think of new ways to improve our quality.

The first exercise as a team was to define quality. We brainstormed in separate teams, writing dozens of yellow sticky notes. Then, as a group we reviewed all of the ideas, grouped similar items into themes, and started pulling together a comprehensive definition.

We ended up with a large pile of definitions; each one seemed valid by itself. One pile was around meeting a number of "Quality Attributes," another pile all about following processes and meeting criteria. Another pile was all about defects (more precisely, the lack of defects). And on and on.

The resulting definition was very comprehensive, and complex. One example, we ended up with 37 different quality attributes[1], and that was just from 1 pile of sticky notes.

The next question, how does this definition help us build better software? Where should we focus our efforts? Some of the definition parts must be more important than others, but which of the 37 quality attributes were part of the vital few? Did we truly have a workable definition, one that would guide us? Well, in asking these questions, we kept coming around to the pile that was all about customers. Some of the ideas were "increased customer satisfaction," "lack of customer support calls," and "high net-promoter." (Net-promoter is a metric we use to measure customers willing to refer their friends to our product)[2].

One slip had written on it, "Customers Define Quality." The reaction was yes, this is obvious, but not too helpful. It has that feeling like "I know it when I see it." It may be true, that our customers will be the ultimate judge of quality, but we only know if they accept it when it's too late, after the project is complete. We needed a definition that helps guide our efforts before, during, and after the project.

However, by thinking "how can we include our customer in each stage of the life-cycle to improve quality," we end up with a number of customer centric practices. We call these practices "Customer-Driven Quality."

Customer-Driven Quality

Customer-Driven Quality is a set of practices for developing software applications to ensure that the product quality meets or exceeds the customer's expectations. Figure 1 shows a graphical view of Customer-Driven Quality.

In Figure 1, customers are the center of the process, surrounded by an iterative software development life-cycle.

Define - Activities related to defining the content of a software project or release. These include ideation, requirements definition, planning, and creating user stories.

Build - Design and Coding phase of software development.

Test - Ensuring the product meets or exceeds the customer's definition of quality.

Support - Helping the customers to use the product and gather feedback to funnel into the next round of product definition.

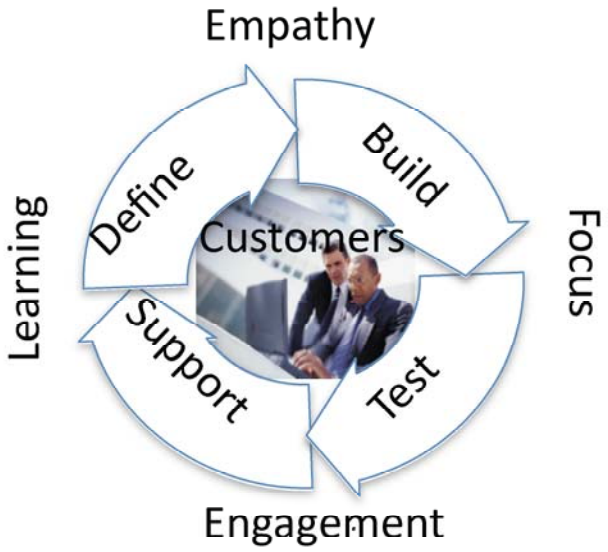


Figure 1. Graphical view of Customer-Driven Quality

Independent of the particular life-cycle stage, Customer-Driven Quality also involves 4 key focus areas. These are shown surrounding the life-cycle, as these activities occur independently from particular projects or life-cycles.

Empathy - Empathy with customers, gaining a deep understanding of the customer's context, including technical skills

Focus - Focus on customers, deciding to invest time and energy in customer-driven activities

Engagement - Direct and unfiltered engagement between customers and the development team

Learning - Understanding actual behavior of customers

Why use Customer-Driven Quality?

Customer-Driven Quality is meant to be applied in addition to, not instead of, all of the traditional software quality practices. Without adding this focus, it can become easy to get wrapped up in the craft of software quality, paying attention to the tools and methodologies, and forgetting why we are developing software in the first place, to satisfy customers.

Looking at the bookshelf, conference programs, and web sites, we tend to pay a lot of attention to the craft of testing and quality. Books, papers, websites, training, certifications, consultants and tools all help with the techniques and methods of building and measuring quality. But, not as much explicit focus is applied to the customer. The following lists shows where customers fall in a word frequency analysis for several software quality publications.

PNSQC Proceedings 2009	Better Software Sept 2009	Software Test and Performance January 2010
test	software	testing
software	testing	software
testing	product	test
team	agile	testers
quality	test	team
code	project	people
time	build	work
data	time	story
product	team	business
development	system	development
system	development	web
process	owner	agile
requirements	work	product
project	quality	use
use	better	performance
work	scrum	career
teams	business	quality
management	code	technology
defects	requirements	code
case	management	company
.	.	.
.	.	.
.	.	.
Customer #159	Customer #510	Customer #32

Table 1, Word Frequency Analysis of several documents, showing the top 20 most frequent words (words like conjunctions, pronouns, names, and articles have been omitted)

This table shows that we frequently discuss the craft of software quality, much more so than discussion customers. Test or testing are the most frequently appearing words, while the word "customer" appears in 159th place in the 2009 edition of the PNSQC Proceedings[3].

Naturally, the word "customer" is #1 in this paper. [4]

Customer-Driven Quality is about getting deeper insights about customer's wants and needs built into our process. It's to build customer empathy in every team member, not just customer representatives.

Looking at software quality literature, there are two rough schools of thought, "old school" and the "new hotness." The "old school" is largely a waterfall process, and is concerned with things like Requirements, Process, Inspection, Verification, Validation, and Traceability.

In "old school" software development, getting customer input is the job for product managers or business analysts. Their job is to understand the customer needs (requirements) and write those in formal documents. Then, the customer comes back into the picture after the code has been released, where the support team is the only interaction. The development team is shielded from the customer by organizational design and practice. This protection is in place to reduce confusion in the development team and protect their time. The developers need to stay focused on implementing the requirements document to maintain end-to-end traceability.

In the “new hotness,” where Agility rules, practices are in place to get more frequent interaction with customers. Practices such as Scrum and User Stories address customer interaction directly, while others like Extreme Programming, Test-Driven Development, Continuous Integration, and Exploratory Testing are focused on the craft of developing software. The iterative nature of these practices guarantees interaction and feedback from customers, or customer representatives.

Customer-Driven Quality is a set of practices that can be used in addition to those already in use. The Customer-Driven practices can be added to any product development life-cycle to supplement your current process.

Preparing For Customer-Driven Quality

This paper describes Customer-Driven Quality practices in two main sections, the “Preparing For...” and a “Life-cycle view.” The “Preparing For...” section describes a set of practices that can be used independent of a project life-cycle. These practices provide a foundation for Customer-Driven Quality.

Team Composition and Focus

Everything starts with the team building your software. Without the team, the software wouldn’t exist to delight customers. So, efforts to prepare for Customer-Driven Quality should start with the team.

In addition to technical, process, and leadership skills, your team should also have some customer advocacy mindset built in. A very fertile area to recruit customer advocates is from the customer care organization. What they **may** lack in automation or coding skills they compensate with deep knowledge and empathy for your customers. In the customer care role, they were talking all day directly with customers, helping them through problems or training them in how to use your software. Who better to help build the next generation of your products than the people who supported the previous generation?

At Intuit, we have a win-win scenario for including customer care people in the development life-cycle. Many of our products are seasonable in nature. For example, the vast majority of our customers use our tax products during a few months of the year. This seasonable demand for care agents provides an opportunity during the off season, the time when we are building next year’s product. We are able to supplement the test organization with the best support agents testing the product. This levels the staffing needs of the whole company, while improving quality of the next year’s product.

At release time, the support agents return to their support roles already well trained in the product. This is also provides a career path for individuals in the customer-care organization. Many of our best quality analysts and engineers took this path to the QA organization.

Goal Setting

Organizations differ in the degree of formality of Performance Management systems. Some have very formal goals that are frequently updated, others are more informal. Regardless of the culture of goal setting or monitoring progress towards these goals, the activities associated with Customer-Driven Quality should play a large role.

Everyone in the organization can participate in the customer-driven activities. Many have customer interactions as an explicit part of their job, while others need to take the initiative to engage in customer interaction. I expect everyone on the team to participate, even the test automation framework developers.

One of the most powerful ways to influence any activity is to model that behavior yourself. If your team sees you investing your time talking to customers and gaining insights from that activity, they are very likely to participate as well.

Building Empathy

Most software products are developed for "real people" not software developers. Sometimes those of use that are creating software every day, using technology in every aspect of our lives and are savvy computer users sometimes forget that our customers don't have the same skills and thought processes.

For example, I'm working on the QuickBooks Online team, a Software-As-A-Service (SAAS) offering to help small business owners manage the money aspects of their business. Our customers are focused on their craft, whether it is running a restaurant, an independent bookstore, a medical practice, or an automobile service station. They are not working with a variety of browsers and operating systems. They may not think to open the preferences to see what options are available. They often select the default choice presented in our UI. In short, they are experts at their field, and expect us to be experts at ours.

These are a few practices that have proven useful for building empathy with our customers. For learning how they think, and what they think about, and what they expect to be taken care of for them.

Customer Care rotation for new engineers

All new college hires go through a rotational program designed to start their career with customer empathy, by spending a couple of months in Customer Care. In this program, they are provided the same training that our care agents receive then get on the phone to help our customers. The learnings are amazing. After 4 years of academic learning, steeped in algorithms and theory, they get exposed to the day-to-day issues that our customers encounter. An example of an insight that an engineer passed on to me, "I was amazed how many people didn't know about the right-click menu."

The Customer Care rotation program is outstanding, but it's also pretty expensive. It's mostly been applied to new college graduates, and it's combined with a more general orientation to our company and corporate life. Another method that is used by experienced professionals as well as long time employees is to listen into customer calls.

Listening to calls

Many of our customers call in for support. As we all know, "those calls may be monitored for quality control purposes." The monitoring infrastructure provides an outstanding opportunity to get engineers and testers exposed to these calls. In fact, listening to both sides of the conversation while not an active participant gives some deeper insight to the customer's issue and how we resolve those issues.

Many of our staff meetings start with listening to a call, followed by a discussion about the insights each of us learned by the process.

Follow Me Home

The Follow Me Home process started a long time ago, when the main distribution channel for our products was a retail store. Engineers and marketers would hang out at a local retailer and watch people shop for our products. Marketers were interested in how the brand new customer chose our product from the alternatives on the shelf, but the fun stuff happened when they "followed" the customer home.

The team would approach the customer and ask to follow them to their home or place of business and just observe the first time use process. Did they read the instructions? Have trouble installing? Accept defaults or investigate each option?

The process continues, but is usually set up ahead of time via email contact. It's very instructive to observe the customer use our products, in their environment. You notice things that are not spoken or written in surveys or support tickets.

One example insight, on a follow me home that I conducted with a customer that had complained about our service being slow. We checked the logs; all the transactions were in normal tolerances. We checked her computer, network connection bandwidth, network latency to our data center, etc. No slowness there.

In observing her, though, she had a stack of bills that needed to be paid. This stack was on a credenza behind her desk. She would take a bill, swivel her chair around and enter the data on our page. Then hit save, swivel and get another bill. She said "there, see, the screen is not yet ready to enter a new bill. It used to be ready by time I swung back." Ah, we now have a new definition on how fast is fast enough: it can't impede her work flow. (The real insight, why are we making this poor woman type in information, the bills should be automatically imported.)

Usability Studies

Web design is about a lot more than fonts and style sheets. Ultimately, it's about ease of use. To test ease of use, and gain insights about customer behavior, the UI designers invest a lot of time in formal usability studies. They will invite customer (or potential customers) in house and observe them using our product or prototypes. This is an excellent opportunity to get an engineer or tester some insights as well. Make friends with the design team, and get some invitations to observe as well.

Voice of the Customer

Our customer care team handles many calls, emails, and online chat sessions each week. They produce a call drivers report, which lists the top 10 issues that generate a support call, with the goal of fixing product or process so the call is not required. This aggregated and abstracted data helps focus priorities for product changes, and the raw data that feeds these reports are a great resource for gaining insights about our customers. We call this raw data the Voice of the Customer.

Our support system has a feature that pulls out 50 random issue summaries, and emails it to the entire team. The first item on our weekly development staff meeting is to review the call drivers and discuss insights learned from reading the verbatim comments.

Net Promoter Call Backs

Our marketing team conducts a customer satisfaction survey every quarter, called the Net Promoter survey. (Net Promoter is a measure of the willingness of our customers to recommend our product to their friends and colleagues). Part of this process is to call every customer that is a detractor (not willing to recommend our product), and a random sample of the promoters.

The marketing team loves getting help in making these follow-up calls. The team that builds and tests the product is an excellent choice for this task. Each engineer and tester is expected to make 3 of these calls per year. It's not a huge investment in time and the insights gained are very valuable.

Product Challenge

One way to understand our customers is to act like one. We hold frequent "challenges" where we put ourselves in the shoes of our customers and try to use our product. This is a great way to introduce new team members to the product.

In QuickBooks, we have a virtual "shoebox" of receipts and invoices that a customer may have when they decide to get better organized and buy our product. The challenge participants are given this shoebox, plus a CDROM of our software, and asked to produce financial reports and some insights about where the money is going.

Infrastructure

Preparing for Customer-Driven Quality also involves some tools and organizational infrastructure that most likely already exists in software development organizations. Here are a few of the main components.

Customer Care Organization

The customer care organization has already been mentioned many times in this paper. It's extremely important to build relationships with the care organization and have a deep engagement with them.

Feedback Mechanisms

Besides the customer care channel, it's very useful to get direct feedback from customers in the context of using our products. We've build a feedback widget that can be placed on any of our pages.

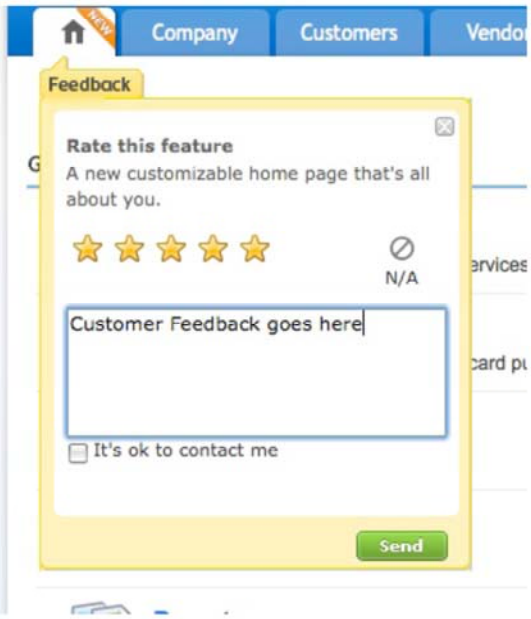


Figure 2. Screen shot of a feedback widget, which allows customers to provide feedback directly for the feature that is currently active on their window.

The feedback widget allows our customers to give the feature a rating, from 1 to 5 stars, and to type in some comments, and optionally contact information if they are willing to have a conversation.

The engineers and testers who created the feature monitor the feedback. This has been an extremely useful mechanism to learn what is working, and find opportunities to improve the feature.

Here is some example feedback, where customers provide valuable insights in how we can make the feature even more useful. [5]

Rating	Customer Comments
4	I love this feature! But I would like to be able to view notes like the old view allowed.
4	I would like to be able to do a statement for an individual client from this screen. Like we can do now.
4	Just started using this screen. Looks great so far.

Figure 3. Screen shot of a typical sample of customer feedback.

Another example, in the View My Paycheck service, the quality team was so passionate to get customer feedback, that the quality engineer did the coding necessary to integrate a third party feedback service.

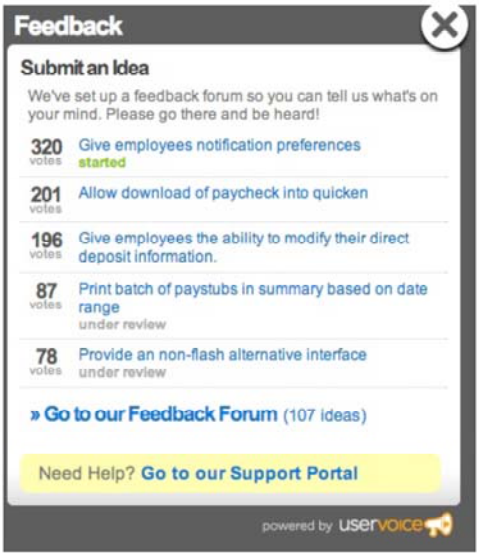


Figure 4. Screen shot shows another example of a customer feedback mechanism used in product. This one allows customers to vote on other customer's feedback, helping to identify which of the new requests are most popular with existing customers. [6]

Text Analytics

Much of the feedback is in text form. The feedback widgets, support emails, support chat transcripts, and call summaries. While useful to read the raw input to build empathy, it's also important to aggregate the raw data to gain insights on the most important issues faced by your customers.

We are lucky to have a lot of customers, so we get a lot of feedback. Automating the analysis of feedback is efficient and gives an opportunity to analyze it more frequently.

A tool that we've found to be very useful is Sentiment Analysis. Sentiment analysis is a semi-automated process that measures both the frequency of occurrence of each issue type, but also measures the intensity.

Text frequency analysis allows us to count the number of times our customers have a particular type of problem (i.e. how many calls are about login). Where sentiment analysis shines is adding customers intensity. For example, "I really hate your login" is generally worse than "Your login is difficult."

Monitoring

Our software has many monitoring mechanisms built in to make sure its operating properly and to alert the support team in the event things start to go wrong. There are action logs, exception logs, and performance monitors. Each of these logs also can provide some insights into actual customer behavior and experience.

The QA teams learn what logs are available, how to get access, and how to analyze these logs. One example where we used the log files, we pulled the action logs across 30 days of use, and identified the top 10 transactions. This information helped us to prioritize the test automation program.

Engage with Customers

The incoming support calls, feedback widgets, and customer usage logs are great ways to get information about your customers, but there are a lot more opportunities to engage directly with customers.

Social Media

Product specific blogs allow conversations with customers. New features can be announced, articles written to help customers, and conversations happen through the comment feature of popular blog engines. One example where we used blogs effectively in QuickBooks Online (http://blog.quickbooksonline.com) was to influence out customers to upgrade their browser. One quality constraint that we were living with was the support for Internet Explorer 6. A lot of special code had to be written for IE6, and testing multiple browsers reduced the amount of test on any specific browser. Newer alternatives had better security, faster performance, and better compatibility with standards.

We wrote a series of articles on the blog to influence our customers to upgrade. Each time a new article was published, we saw a reduction in the usage in the web analytics logs. Influencing our customers to take some action at the client side is an example of customer-driven quality.

Facebook and Twitter are other venues for learning about customer's usage of our products, and an opportunity to communicate with them. Often, customers are posting on these sites their frustrations and joys with our products. It's pretty simple to have an agent providing real time search for our product name.

Search Engine Strategies

We also use search engines (Google, Bing) to search for our products and what our customers are potentially saying online about our products. Doing this exposed several community forums dedicated to using our product.

Customers will post their likes and dislikes about our products, questions about how to perform a task, and desired new features. This is a good way to gauge the customer's opinion of our quality.

The search engines have a feature called alerts, which will automatically perform the search specified, and email the results in either a daily digest, or real time. To monitor the "blogosphere," we've set up alerts with the popular search engines.

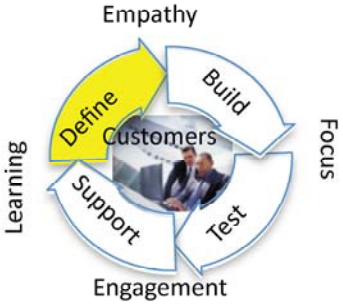
Customer-Driven Quality: Life-cycle View

The previous sections described a number of practices that provide the foundation for Customer-Driven Quality. This section provides a simplified product development life-cycle and describes the associated customer-driven practices.

This simple life-cycle is described in 4 stages: Define, Build, Test, and Support.

Define

The Define life-cycle stage is where the product requirements and scope are defined. The customer-driven practices are intended to help build the right product for our customers.



Investment level

One of the most powerful decisions that can be made during the definition phase is deciding how much of the limited development bandwidth to apply towards three broad categories: product infrastructure, solving current customer pain points, and building new value for current or future customers.

These questions are not easy to make, but each category must be considered. Often, the product definition phase is dominated by building value for new customers, but product infrastructure and current customer pain points are important as well.

The product infrastructure must be maintained and improved to prevent future pain points, such as performance or availability issues. Solving current customer pain points are one of the reasons we gather and analyze all of the feedback.

A few ways of managing this prioritizing the current customer activities include:

- Applying a "tax" of development time before considering new features; keeping some development resources in reserve so they can concentrate on satisfying current customers.
- Managing requirements and features in a common prioritized backlog along with infrastructure and customer issue. Having a common repository will help the team make these tradeoffs.
- Periodically, a "customer love" or "net promoter" release can be useful, where the entire focus is just solving pain points.

Ideation Process

Customer-Driven Quality does have a natural predator, the HIPPO. Not the aquatic mammal from Africa, but the "Highly Paid Person's Opinion". The HIPPO is a term we use to remind ourselves that it's the customer's opinion that counts, not the boss's.

Instead of building the feature set that is desired by the influential people in the organization, instead the customers should decide what we build. We call this process the Ideation Process.

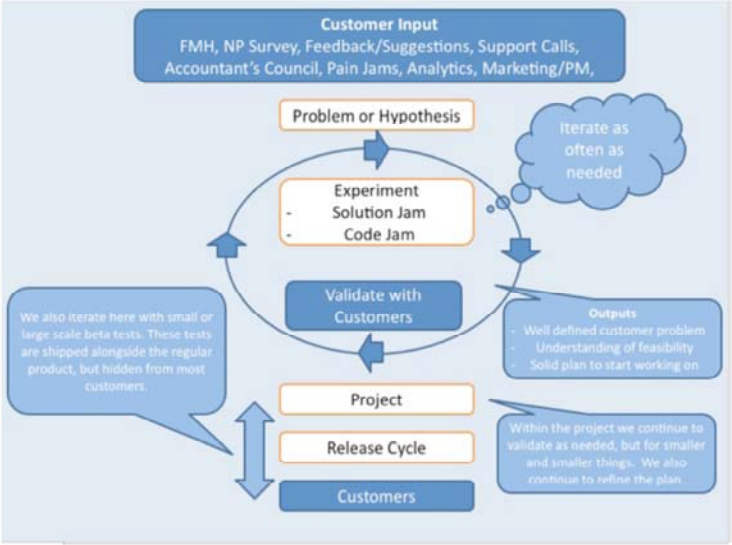


Figure 5. The "Ideation" process, where features are defined while working closely with a set of customers.

The Ideation Process involves understanding the customer's pain points, brainstorming ideas to solve the pain, and testing the ideas in the market with real customers. To emphasize the experimental nature of this process, the ideas are called hypotheses.

In the center of the diagram of Figure 5, the core of the Ideation process is a Solution Jam & Validation with customers. The product team brainstorms solutions with customers, then will work with designers or engineers to produce a lightweight prototype (paper or a very quick Flash application) to validate the solution. Solutions that are deemed correct then go to the next stage of validation.

Since the prototyping is done with a limited set of customers, and these customers know they are part of the experiment, the solution then goes to the next stage, testing in market.

The prototype and list of user stories that result are the input to a product development cycle to produce a working product. This working feature is then put into the next round of testing, where its exposed to actual customers. These experimental features are exposed to several dozen to several thousand customers. Feedback and analytics are monitored closely to judge the effectiveness of the solution. Winning solutions become part of the product and are widely deployed. Others are either quietly abandoned before getting too much exposure or the team goes back to the drawing board.

The Ideation process can be thought of as a product management process, and being tangentially related to "quality." However, a better way to think about it is testing the requirements before the product is built.

User Stories not Requirements

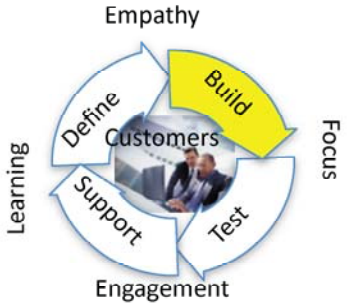
In the traditional model of software development, a special team of people, called business analysts or product managers, talk to customers, understand their needs, and document those needs in a formalized language. This formal language is called requirements.

Great care is taken to write the requirements in a manner that facilitates traceability, completeness, and abstracts any hint of implementation. When reading these formal requirements, it's frequently easy to miss the point completely.

Documenting the product requirements in the form of User Stories helps keep the focus on customers, and removes an opportunity for ambiguity. User Stories can also be read and understood by customers.

Build

The build stage of the life-cycle is where design and coding takes place.



Personalized Development

We have several methods for developers to interact directly with customers during the implementation phase. These boil down to putting a name, face, and engagement with actual customers to build the right product in the right way for our customers.

The Adopt a Customer approach has a developer choosing a small group of customers. The developers then build the feature collaborating directly with the customers. The developers have access to customers to ask questions, give frequent informal demonstrations, and brainstorm ideas. This process provides focused problem solving, rather than guessing intent from the requirements (or asking HIPPOs).

We maintain a list of customers willing to provide frequent interactions, through the Inner Circle program. Customers opt in to this program, and are available for consultations.

Default Behavior

One customer behavior that has been learned through web analytics is that the vast majority of customers do not change the default choice when provided multiple choices. This behavior drives many of the design decisions. For example, a user interface design where the customer should choose from multiple options (State, for example), should not pre-select a default, especially if the list if alphabetical.

Reviewing the default options is a specific line item in the design review checklist.

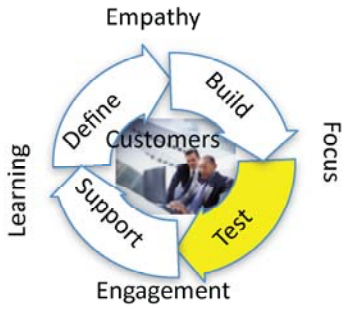
Build with Customer's Platform

Many self inflicted errors come from cross platform development. To the degree possible, this should be avoided. The product should be built on the platform used by customers.

In the current state of web application development, most developers prefer to develop with Firefox, because of the superior development and debugging plug-ins. These plug-ins were developed by developers, for developers (talk about Customer-Driven Quality). However, most customers use Internet Explorer as their browsers. Ideally, the web UI should be developed with Internet Explorer, or at least tested by the developers before they check in.

Test

The test phase is where we ensure that the product meets the defined requirements and is suitable for release.



Beta Test

Releasing product as a beta is a time tested practice to include customers' feedback in the development process. One observation about beta test, some customers are not willing to test a beta product, so we may miss out on their feedback. One benefit of an iterative development cycle, we always have a chance to include customer's feedback.

Rolling Deployment

In the first day after fully deploying our application, we see several orders of magnitude more product usage than in the entirety of our testing cycle. To make sure we have a smooth deployment, we deploy in stages. This helps us ensure we didn't miss anything in our development process.

We deploy to approximately 10% of our customer base the first evening. Then, we monitor the feedback closely from these customers, looking for any new or unique issues being reported by these customers. The balance of customers see the new release a couple of days later. This process gives us the chance to correct any bugs that were missed by the test team.

Analytics Driven Test Plans

Several members of the QA team are also participants on the Web Analytics team. The Web Analytics team is largely used by the marketing team to understand customer's behavior for the purposes of finding opportunities to optimize customer value, but we've found a lot of useful information to inform the test strategy.

One example where this added value, we have an automated test suite that duplicates the top 12 customer workflows. This test suite is executed every day.

Performance Testing in the Real World

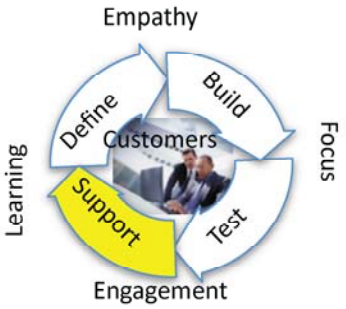
Performance and scalability testing is both supplemented and calibrated by using a remote testing service. The remote testing service executes test scripts from many locations across the world, which provides information on how our application is running from our customer's perspective, not just from inside our firewall.

The service that we use allows us to test performance from many locations in the world, both at network backbone locations and on actual customer's machines, which may be connected by dial-up, DSL, or Cable modem.

This service has helped us optimize performance using Content Delivery Networks, and optimizing content download time.

Support

Since this is a circular life-cycle, there is always customer learning that can be applied to future iterations.



The Customer Care organization is all about interacting and satisfying customers. This section will provide a few ideas to incorporate the support phase in defining quality for future iterations.

Development Team Support

When we develop and release new features of significant complexity, a very useful process is to include the development team on the front line for customer care. This practice helps by supplementing the customer care team, as the call load is likely to be greater for a new feature that is not yet understood. By putting the developers in the support role, they get that direct interaction which helps them tweak the design.

Analytics & Feedback

In addition to the feedback channels mentioned above, the support team is an excellent source of knowledge. Since they interact with customers as their main task, they are able to give customer focused insights to the development team.

Social Media

Interacting with customers through social media, like Twitter and Facebook, were mentioned earlier as way's to build empathy with customers and discover their definition of quality. Likewise, it's a great way to interact with customers in a support capacity. Customers will frequently post their problems to their friends. Replying directly, proactively, and helping solve the problem usually results in a positive experience for the customer.

Conclusions

Customer-Driven Quality has helped our team provide focus on what is truly important, and help cut through the clutter of complexity facing our software development decisions. The following checklist distills many of the practices of Customer-Driven Quality:

Customer-Driven Quality Checklist
Does your team have a customer advocate?
Do your team members communicate directly with customers?
Are Customer-Driven Quality activities part of the goals for your team?
Is the support case feedback available to the development team?
Does your team regularly analyze and act upon the support case feedback?
Are members of the team engaged with customers on social media? (blogs, forums, Twitter, Facebook)
Are you searching for new mentions of your product on a regular basis?
How much effort is being allocated to delighting current customers?
Are requirements determined by actual interactions with customers or by industry trends?
Do you track actual usage, and use that information in your quality strategy?
How does the development platform differ from a typical customer's platform?
Is customer acceptance testing performed with actual customers?
Are you calibrating your test/quality strategy based on actual customer metrics (usage analytics, performance, platforms, etc.)

Notes and References

[1] Chithambaram, Raju, Non-Functional Quality Attributes, Intuit Life-time Design Template, April 2010

[2] Net Promoter is a customer satisfaction metric, where customers are asked their likelihood to recommend a product or service to their friends. The responses are classified as "Promoters" or "Detractors." The metric is calculated by subtracting the percentage of detractors from the percentage of promotors. For more information on Net Promotor see http://en.wikipedia.org/wiki/Net_Promoter or http://www.netpromoter.com

[3] The text frequency analysis used 3 documents, the Conference Proceedings of the 2009 Pacific Northwest Software Quality Conference, the September 2009 edition of Better Software Magazine, and the January 2010 edition of Software Performance and Test Magazine. The text frequency analysis was performed by converting the PDF documents to text, and using a word counter program. Words like articles, pronouns, conjunctions, and proper names were eliminated, as these were not relevant.

[4] This document was analyzed using the same process as the other documents, and the top 10 words in descending order are: customer, quality, product, team, development, software, driven, process, test, and support.

[5] This example screenshot is taken from QuickBase, a Software As A Service (SAAS) database offering which allows easy database creation.

[6] This example uses another service, UserVoice.