Involving customers early and often in a software development project

Level: Introductory

Laura Rose, Quality Engineering Manager, IBM

16 Jan 2006

from The Rational Edge: Iterative development offers built-in opportunities for customer engagement throughout the development lifecycle, yet many organizations neglect to involve customers effectively in the process. Directly incorporating customer involvement into the iteration planning and the other simple techniques described in this paper can greatly improve project success.

A large study presented at the XP 2002 conference by James Johnson of the Standish Group shows that 45 percent of the features coded into most applications are never used (see Figure 1).[1] *It seems absurd to spend time on things no one will use, so where do all these features come from?*

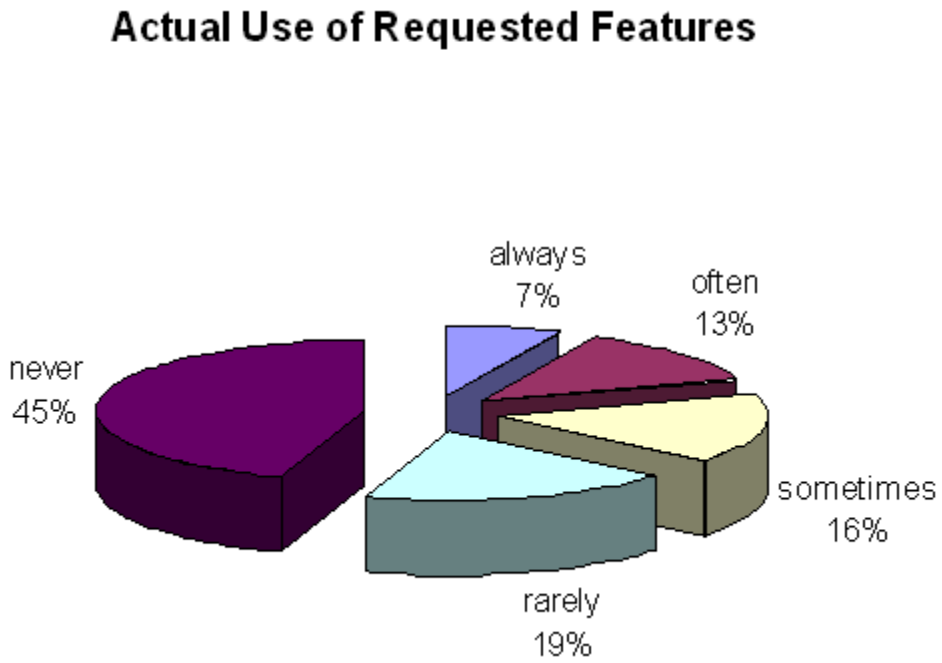

Figure 1: Usage of coded features presented by Standish Group, at the XP 2002 conference

Feature lists come from many places. Some come from the business analyst, who wants a visible representation for how the business is staying competitive. This

sounds both reasonable and important. We want competitive differentiation. But the way we go about it may not be adding value for the customer. Consider, for instance, all the Consumer Reports *analyses, or even your own analyses based on comparisons from technical sources. Typically, these reports include charts that list more than 100 features compared across the various brands of a certain tool, with each item given a checkmark across the columns if that brand offers that feature. At first glance, we probably assume that the brand with the most checkmarks is the best solution.*

What isn't as obvious is that, of those 100 features, a total of 64 percent are rarely or never used by any customer. But to stay seemingly competitive to the other brands, we incorporate superficial solutions to the unneeded 64 features, allowing us to "checkmark" those rows (making our product more complicated and frequently more difficult to use).

In iterative and agile development organizations, adaptive and client-driven planning is forefront. Client-driven development implies that the choice of features for the next iteration or release comes from the client. The focus is on whatever the client perceives as the highest business value. Still, many development organizations do not consult the client, but instead isolate the choice of features to the business analyst or product marketing teams. Prioritization of features is done by the product managers via their interpretations of the market trends and competition. Unclear and vague requirements are passed to the development teams, who typically have even less contact with the customer. They code the features to their perceptions (which are far removed from the customer's viewpoint). Clients often do not see the application until after code freeze or during a formal beta cycle, which is too late.

In this paper, I offer suggestions for how we can change this cycle in our organizations and get early and continuous customer involvement. I provide examples of common problem scenarios based on my personal experience in software development projects. And although these examples illustrate a software development lifecycle, the techniques I describe can be used by anyone responsible for verifying and gathering requirements.

Obstacles to success: Imaginary and real

Learning how to discover actual requirements and achieve higher software project success may not be easy for many development organizations. Some limitations are imaginary, some are real. In this section, I present what I consider the most common types of limitations we are likely to encounter.

Removing imaginary obstacles

Consider the following example, which illustrates development teams working in isolation from the customer.

A performance test tool was designed to emulate thousands of users to stress servers and other applications under test. Diagnostic information gathered by the tool was used to tune the servers to meet those applications' required response and performance goals. The performance tool supplied the right diagnostic information to cover the specific and overall design requirements. But the customer took four hours to locate, extract, interpolate, and manipulate that data into a form they could hand off to a different task team who actually tunes the server. The development team assumed that the group using that data would be the same team using the performance tool. Because the tool could not automatically export that data, the product failed customer acceptance.

How did this happen when we met the original requirements? Well, in the first place, the original requirements were vague. The developers focused on the data that needed to be collected. They collected and logged in the most expedient method for the project schedule. They created real-time, console-displayed graphs and tables. Data was console-displayed both during and after the run. Real-time graphing during runtime was a high-profile "new feature." In the second place, the development team did not realize that, from the business process viewpoint, the customer test teams running the tests were different from the customer IT groups using the data to tune the back-end server. The teams using the data to analyze what needs to be done on the server were not sitting in front of the performance test tool. Therefore, the addition of "real-time" reporting during the run was invisible to the teams that needed the data.

In fact, the actual imperative feature was the ability to export and share the right reports. Had the original requirements been reviewed against the customer's

business processes, this important fact would have been highlighted clearly. We would have avoided this "error" had the customer been consulted in the early requirement-review and walkthrough phases.

In the previous example, the problem boiled down to the *perception* that developers can't have contact with the customers because there are too many obstacles in the way. Indeed, many obstacles may *seem* to prevent the development team's involvement with customers. Some are real, but the following ones are imaginary:

- As a developer, I don't have access to customer contact information.
- I can't take the time to meet and talk with the customer.
- Customer contact is someone else's job. I need to trust they are doing their job, and I need to focus on my job.

These are really just excuses, not real obstacles. Let's expose each one.

"As a developer, I don't have access to the customer contact information."

As developers and testers, we work on customer defects every day. Those defects have customer contact information. Starting this relationship is as easy as picking up the phone. Once you have started the relationship, there are other techniques to keep customers engaged and involved.

"I can't take the time to meet and talk with the customer."

This might be true today, but it won't be true tomorrow. Developers should make their intentions known and work with their managers to schedule time in the next cycle to accomplish specific customer interaction goals.

"This is someone else's job. I need to trust they are doing their job, and I need to focus on my job."

A developer's[2] job is to create quality code, applications, and products that are valuable to the customer. Finding out what is valuable and how the customer actually uses your application should be considered an important part of the developer role.

Customer interaction is a never-ending activity. There are thousands of ways to interact with customers. There are activities that are owned by certain roles and groups. But there are literally hundreds of other opportunities that have no owner. Even the people that own certain roles can use the assistance.

Adaptive planning

When we go shopping, sometimes we have the "wanted" item in mind. Sometimes we don't. And sometimes we change our minds after we see it. We need to actually see, feel, and try the product before we are sure we want to buy it. Software customers feel these same needs: They don't always know what they want, and they sometimes change their minds. Many organizations get their first customer feedback through beta versions of a nearly completed product.[3] While this is a widely accepted and useful practice, it provides late feedback. If the beta customer identifies critical usability or missing functionality at this stage, you have a difficult decision: Release with the critical defect or delay the release in order to redesign.

Having detailed discussions with your customer about early versions of your product design verifies that you are on the right track. Customers also need to see something concrete before they can give useful feedback. By providing quick prototypes, earlier design review, and short iterations to delivery (in addition to the normal beta release), you reduce the risk of the customer finding critical usability issues at the later stages.

Lastly, business needs are often dynamic. Delivering partial solutions to parts of the problem sooner is better than the entire solution a year later. If your full solution to the customer's needs is a year away, the customer will find another solution to specific, critical needs in the meantime. Provide mini-versions of your applications,

and allow the customer to continually readjust your next iteration. This increases their involvement, investment, and loyalty. On-site visits and arranging early deployment to help customers reinforces and aligns your vision with theirs. By continuing to mold your tool to their specific business processes (versus making them change their processes to fit your product), you transform your developer-to-consumer relationship into a true creative partnership.

Adaptive planning for each iteration directed by the customer is fundamental to an agile programming style, and the advantages can easily be exploited in other development environments.

Understand the limitations

Granted, it may not be possible for some projects to focus solely on what the customer or set of customers wants because often there are many different kinds of users and stakeholders. Competitive factors, corporate-mandated security-compliance features, and other overhead requirements may force us to generate features that the end-user never uses. In these cases you should identify your "release-defining"[4] feature for each release, then limit and designate a customer (or set of customers) who represents the majority of your users for that specific target. With each subsequent release, you may have a different "release-defining" feature highlight; and you may have a different set of customers represented.

This selected set[5] reviews each iteration's deliverable (requirements, prototypes, demos) throughout the development lifecycle. By doing this, your resulting product will have more of the features that actually get used, and you learn which features to spend your time on. Even if you have 100 features in your product, by constantly communicating and interfacing with your customers you know which 36 features to really focus on and which 64 features to assign lower priority. So, even though you may not have influence on how the business teams list their required features, you at least know what's going to make your customer most satisfied by the end of the project.

How do we actually do that?

Applying the practice

With the right level of commitment, combined with frequent customer interaction and a willingness to remain flexible in implementing a solution, software development teams are much more likely to provide the features that make a software project successful. Naturally, satisfied customers will return to your organization for more projects in the future. Let's take a look at the basic ingredients for this success.

Make the commitment

The first step on any project is to actually commit to creating and increasing your client list. There are many ways you can learn more about your customers.[6] Here are some of the more obvious things you can do:

1. Call the clients who have previously reported defects on your product. If they have reported a problem, you know they are using your application.
2. Attend the training classes on your product, where you can meet the people planning to use your application or product.
3. Attend your organization's user group meetings and conferences. Those events are sponsored for your customers.
4. Present for and attend local professional organizations that align with your application or product.
5. Seek out internal customers for your application. Get other teams within your organization to use your product. They can provide "friendly criticism" under real-world use.

And here are some less-obvious things you can do:

1. Talk with the field experts on a particular account. Learn that account's actual customer user flow, business flow, or processes. Use those user flows in your

testing and design.
2. Partner with your field experts to tag-team on proof-of-concept engagements.
3. Offer to train a class on your application or product.
4. Invite the customer to your test labs.
5. Offer to go to the customer's labs to help them with their projects or train them on your application.
6. Offer to be the support contact on your beta program.
7. Visit the customer on-site so you can see how they work in their natural environment.

Then make the commitment to contact and interact with at least three customers this year (or whatever seems reasonable in your organization). Make it official and place it in your documented performance evaluation goals as a forcing function to success. Report the results in your quality assessment reports in your product reviews.

Learning to contact the customer

You have to assume that, in their own job roles, your customers want to perform at their highest possible levels. They want tools and applications that help them efficiently and accurately accomplish their goals. Your goal is simple: Make them successful.

When your intentions are presented that clearly, customers will bend over backwards to help you build what they need. Here are three techniques that will help development team members make good on their commitment to improve customer contact.

- **Provide options**

Once you have contacted the customer, offer options on how to proceed. For instance, my team has several customer programs in place, including **customer roadshows** (visiting customer at their location to just chat), **transplant testing** (bringing customer applications, data, test cases, customer procedures into our test labs so that we can more effectively test the product like the customer would use it), **implant testing** (putting our utilities and procedures in their labs), and a **residency program**.

In the residency program, the customer spends five days working in our test labs, using our products with our testers -- while our developers watch. We conduct real-time discussions of their concerns and questions. We set up meetings with the product managers, documentation teams, and tech support members so that all the development lifecycle stakeholders share the same awareness.

By offering options on several levels, the customer can choose how to interact and drive the interchange without feeling forced. These programs are also progressive in nature: With each success, we have the opportunity to progress to the next level of involvement.

For example, presenting at a user group meeting or attending a customer training class may lead to a customer trip. A customer trip or roadshow often leads to the gathering and transplanting of the customer applications, data, test cases, and business process into your own labs.[7] Residency programs often lead to beta and early alpha commitments by the customer for the next release. And alpha and beta customer involvement often leads to client-driven iterative development meetings throughout the development cycle (not limited to the alpha or beta release).

The momentum soon becomes self-perpetuating, with the progression to each level of customer involvement naturally evolving without much effort on your part.

- **Prepare for template-like activities**

Once you start engaging with customers, you will find that most forms of customer interactions, however different, require similar preparations. For instance, we typically need signed nondisclosure agreements, certificates of originalities, or other paperwork prior to any external customer interaction. Creating a few checklists

and templates can reduce the risk that you've eliminated an important consideration or step. A sample checklist can be found in the Appendix.

- **Incorporate customer interaction into daily business**

Yes, this takes time. The mistake is to consider this as an "optional" activity. The moment we determine that customer interaction is above and beyond the call of duty, we're doomed.

As a quality engineering and test manager, I create test plans for various programs and products. I incorporate transplant testing and residency visits into our routine and send out my testers to customer test labs explicitly as their "test activities." These activities become part of my test plans; their results become part of my iteration exit criteria. Incorporating in this manner eliminates the need to "find the time" to perform activities that otherwise might seem extraordinary.

For instance, my test plans include well-defined success criteria for each iteration. Some examples are:

- Customer X's review and signoff on requirements
- Customer X's scheduled demo planned and executed
- Customer X's positive evaluation of iterations, deliverables, and quality criteria
- Successful residency visit by Customer Y, ending in a positive evaluation of product
- Published prioritized list of customer issues and concerns in defect database

Once again, the best way to be successful in these activities is to very publicly state your intentions and success criteria. Incorporating these activities along with your success criteria in the team's test and development plans changes the priority, focus, and tenor of the entire product.

In today's lean organizations, confusion constantly ensues from juggling and balancing among adding new features, expanding to a new platform, fixing the backlog of defects, and improving usability. Having customers continuously involved provides a tangible and very simple team goal: Make this customer successful.[8] Everyone can understand this goal. Priorities and focus automatically follow. And at the end, we're not looking for a success story, because we've been creating it all along.

Defining your customer interaction success criteria

Many projects undergo some level of customer interaction or beta program. Often they are scheduled at the end of the product's development cycle, after the product is deemed "feature complete." As mentioned above, this is extremely late for the first customer preview. I've actually reviewed a beta plan where the beta was scheduled to run from October 3 through November 7, and the product was scheduled to be released to the general public on November 5 (prior to the beta exit). Obviously, the results of the customer beta were irrelevant to this particular release. Their input will go into the next release, but just not in the current release schedule. And this isn't necessarily bad, but we need to understand and agree upon the reason and success criteria for any given beta cycle.

For instance, if this were a small maintenance release for a well-established product, we might already have confidence that the product meets the customer needs. The corrective contents have come directly from the customer, and there really isn't a need to wait for the beta feedback. We're essentially just supplying an early release of the product to our best customers and sales teams.

On the other hand, if we are releasing a relatively new product or technology, or moving to a new platform, and we are uncertain about how the changes will be received, customer evaluations need to be conducted throughout the development process. Reaction time to that feedback needs to be properly scheduled.

In his book *Seven Habits of Highly Effective People*, Steven Covey often stresses that we need to *begin with the end in mind*. Before embarking on any customer interaction like a beta program, we need to define and agree upon the goal and success criteria. Imagine yourself at the end of the customer exchange, beta event, or

residency: What explicitly do you want to have in hand as you exit?

An evaluation of your product can help define what constitutes a successful customer trip, residency, or beta program. An evaluation might include a customer rating of the following categories (using the 1-5 scale provided in Figure 2):

1. Usability
2. Out-of-the-box experience
3. Has the features that are important to accomplishing my goals and they work
4. User assistance
5. Others?

Categories	1	2	3	4	5
Usability					
Out of the box					
Feature value					
User assistance					

Figure 2: A product evaluation might include a customer rating of these categories.

As you develop categories for your product evaluation, you should also develop a consistent rating system focused on the customer perspective. In other words, the numerical ratings should mean something like:

1 -- Forget about releasing this any time soon.

2 -- Product has some major problems and would be high risk for me (customer) to actually try to implement in my organization.

3 -- Only one or two major issues with it, but lots of minor stuff that would make it annoying to use. I can still get my work and my goals accomplished, but it would be annoying to do so.

4 -- Only some minor quirks -- low risk. Good.

5 -- I'll buy it today!

And you will need to weigh your results across the entire spectrum of customers who evaluate the product. For instance, the success criteria might be defined as "an evaluation of 3 or better from 85 percent of the customers surveyed."

Conclusion

Iterative development offers built-in opportunities for demonstrating incremental advances on the solution being implemented. These early milestones are important customer engagement opportunities; unfortunately, many development organizations focus on iteration planning and other technical aspects of iterative techniques rather than on the critical need to involve customers. Directly incorporating customer involvement into the iteration planning and the other simple techniques described in this paper can help balance the team's motivation. Teams will start to approach customer interaction as an integral part of their development lifecycles.

Customers will not only be excited about the solution being implemented, but also look forward to future collaboration.

Additional reading

Laura Rose, "Twelve Tips to Realistic Scheduling." *The Rational Edge*, July 2005.

Craig Larman, *Agile & Iterative Development: A Manager's Guide*. Addison-Wesley 2003.

Notes

[1] See The Standish Group paper, *What Are Your Requirements?* Standish Group International, Inc. 2003 (based on 2002 CHAOS Report).

[2] The term "developers" refers to programmers, testers, technical writers, and all those involved in developing the product.

[3] That is, the product is virtually complete. All the features are coded and integrated, with only system testing still left to accomplish.

[4] "Release-defining" feature means that if this feature isn't done, the product doesn't ship. There may be other enhancements and features scheduled for this release, but if they aren't completed on time, the product can still ship. A successful strategy strives for only two or three "release-defining" features per release.

[5] Working with business partners or consultants that deal with various types of customers in your solution/industry is a way to get good feedback on various environments with one contact point.

[6] Brainstorm with your teammates and cohorts on all the possible ways to contact and interact with your clients.

[7] This could also be done by assigning a tester from the team to temporarily work in the customers' labs and bring that knowledge back so that the product can be tested according to how the customer will use it.

[8] Remember that "this customer" is the "customer or set of customers" that represents the majority of the users you are targeting for just this "release-defining" feature. The next release, you might have a different product emphasis or feature. Therefore, your "customer set" will be different for the next release. Your current release may not make everyone who will ever use your product successful. But it will make the target group for this release successful.

Appendix

Once you start engaging with customers, you will want to create various checklists and templates, which reduce the risk that you've eliminated an important consideration or step. Below are some sample checklists.

Planning:

Item	Description	Owner	Planned date	Actual date	Status

1	Create general agenda for customer activity or discussion. This agenda will be used for talking to perspective customers about the details about the program.				
2	Find customers available for your specific activity.				
3	Agree on visit date: Get agreement with customer and IBM stakeholders for exact date of interchange.				
4	Agree with customer on goals of interchange. Identify the success criteria for the interchange.				
5	Request confidential disclosure agreement (CDA) or any other legal forms required.				
6	Get tech support data on problems/issues customer is having as background information.				
7	Get other groups involved, such as developers, tech writers, testers, product management, development managers, etc.				

Prep:

Item	Description	Owner	Planned date	Actual date	Status
1	If this is a several-day event, prepare overall agenda and first-day agenda and book the needed rooms. The agenda is just a template that will be continually revised by the customer in real time.				
3	Prepare your in-house participants -- topic of discussion, available time slots, book meeting rooms, schedule meeting times. Include staff members from quality engineering (QE), dev, doc, training, tech support, product management, and RE.				

Setup:

Item	Description	Owner	Planned date	Actual date	Status
1	Prepare lab and machines.				
2	Make sure your staff actually works in the lab during your customer's visits to the lab. Even though one tester or developer is always with your customer, having your entire team in there also provides additional help when needed.				

Execution:

Item	Description	Owner	Planned date	Actual date	Status
1	Hold welcome meeting and exchange business cards and first-day agenda.				
2	Create next day's agenda to hand out the following morning. This allows for the greatest flexibility as people's schedules do change and unexpected things do come up. This also gives the coordinator the ability to customize the program as more is learned about the participants and their interests.				
3	Have a checkpoint with customer each day to make sure session is headed in expected direction.				
4	Consider taking the customer out to dinner one evening.				
5	Prepare evaluation questions and certificates for longer customer exchanges like residency visits.				
6	Wrap-up meeting (normally customer, coordinator and site QE manager). Discussion about session. Hand out evaluation forms and certificates if appropriate.				

Wrap-up

Item	Description	Owner	Planned date	Actual date	Status
1	Enter customer issues in your organization's defect tracking tool.				
2	Create report for management, include completed evaluation forms.				
3	Give presentation to QE leadership team, QE team, and other interested ID and dev groups.				

Follow-up:

Item	Description	Owner	Planned date	Actual date	Status
1	Contact customer two to six weeks after their visit (depending on severity of visitation). Provide follow-up on items not complete at end of initial interchange.				
2	In addition to discussing new/improved features, invite customer to review the design: provide feedback on prototype, take part in usability session for the design, etc.				
3	Schedule on-site customer visit. Observe customer(s) using the application in their own environment.				
4	Work customer feedback into program so next session will be better than the first.				
5	Follow-up, be available for new problems.				

Don't	Do
Commiserate with customers about how bad the product is.	Acknowledge specific problems to focus on solutions.
Mention release dates, prices, or announce specific features in a specific release.	Collect feature and enhancement requests, to forward to product management and place in your organization's defect tracking tool.

Make any enhancement commitments	Follow up customer exchange with a thank-you note confirming that you've passed along concerns and requests to the proper folks.
Mention/discuss competitor applications (either positive or negative).	Discuss what customer likes about competitor products, if you find yourself in that type of situation.

Sample customer questionnaire for IBM Rational Performance Tester tool:

Below are sample questions you might ask customers:

- What are your top performance testing problems or obstacles (independent of the performance test tool that you are using)?
- What are your particular challenges when you run your performance tests?
- What are the typical scenarios and environments that you run?
- What are the five protocols most used in the field (Ex. Citrix, SAP, Web, Siebel, DB/2, etc.)?
- What performance test tool have you used? «product X»
- What do you really like about «product X»?
- What do you *not* like about «product X»?
- What capability or feature do you need that is not in «product X»? Specifically, if we implemented any of the following use cases, would you switch more quickly to RPT?

IBM Rational Performance Tester (RPT) questions:

- How easy was RPT to learn?
- How easy is RPT to use?
- What things did you find yourself doing over and over again, things that you feel could have been made easier?
- Where do you spend the majority of your time with a performance test tool? (For instance: Recording? Editing the test script? Running the test? Monitoring the agents? Analyzing the results?)
- Did RPT satisfactory accomplish your performance test goals (i.e., stressed your system)? If not why?
- What changes would you make to RPT's interfaces, usability, reporting, etc.?
- What new features and improvements in RPT would you like to see the most? (Please list in priority of importance.)
- What improvements in user assistance, user manuals, and help would you like to see most? (Please list in priority of importance.)
- Would you be interested in a closer working relationship with RPT to exchange information and help?

About the author

Laura Rose is the quality assurance manager responsible for automated performance test tools at IBM Rational. In addition to leading projects in both software programming and testing environments, she has thirteen years of programming experience and ten in test management. She has been a member of the American Society for Quality, the Triangle Quality Council, and the Triangle Information Systems Quality Association, and has published and presented at various test and

quality conferences. You can reach her at llrose@us.ibm.com.
