Usability Testing and Risk Management
in a Multi-Developer Context

Marilyn Valentino

EPRI
Palo Alto, California

Abstract

Driving usability improvement in a company with over 100 different remotely-located software development organizations, each using independent development processes, presents specific Software Quality challenges. This paper describes the usability acceptance testing and usability risk assessment approaches that one organization instituted within this complex development environment, from the ground up. The steps taken are outlined, along with charts presenting results data. The data show a 100% increase in customer satisfaction, a reduction in customer-reported usability problems, and order-of-magnitude improvements in testing cost reduction and turn-around time over a three-year period. Key success factors are highlighted.

Overview

After an introduction to our business and development context, you will find these sections:

- Program Beginnings
- Usability Testing
- Customer Satisfaction Surveys
- Customer-Reported Problems
- Usability Risk Assessment Pilot Program

Each section contains three parts: What we did, Results, and Success factors. A summary section describes planned next steps and recaps key lessons learned.

Business Perspective

Our business

EPRI began in the 1970s as a government-funded think tank, the Electric Power Research Institute. Over the years, the business evolved into a non-profit organization funded first by the U.S. electric utility industry, and now by a diverse group of customer companies involved with energy production and distribution worldwide. Today we are a 330 million dollar business. We have about 900 employees at four main U.S. sites, and a number of offices in other countries.

Research has always been the soul of the institute: EPRI's core competencies lie in assembling technical experts and sponsoring knowledge breakthroughs. Due to this orientation, EPRI initially viewed software as an occasional side result of research, not as a primary product. However, software is growing in importance, and it constitutes an increasing portion of our total budget.

Our software development approach

EPRI uses an approach of 100% software development outsourcing. We contract our software development with over 100 different organizations, largely domain experts in science and engineering who are geographically dispersed throughout the country. Our developers range from single-person shops to ISO 9000 certified organizations, and each has a unique software development process.

We produce scientific software. Our applications have small user communities--typically under 50 users per title. We have a portfolio of over 300 personal computer software applications, which are mostly used for technical and economic analysis and simulation scenarios. We develop 70-80 software releases each year, including some new products and many upgrades of existing EPRI products. Three quarters of our software releases have budgets under $500,000.

Because our software requires extensive technical knowledge to develop, our selection process for developers emphasizes their research and technology expertise over their software development skills. Although specialized technical knowledge in the research area drives vendor selection, Software Quality has instituted an interview process for all prospective developers at the contract stage. An introductory interview covers the quality checkpoints and approaches that the developer plans to use for a given project, and also emphasizes that the software will be tested for user-friendliness before it is accepted.

Our Software Usability Challenge

Because of its highly technical content, EPRI software is extensively tested by developers, researchers, and user groups to ensure that calculated results are accurate. EPRI has always had a fully documented and audited Quality Assurance program for safety-related software. This program scrutinizes software for regulatory compliance, and applies specialized verification and validation processes to generated results.

In this context, user-friendliness has received a relatively low priority. Users have always prized EPRI software for its advanced and accurate mathematical results, but they have complained that the software was difficult to use. Installation could take several hours of work, menu items could appear inconsistently on different screens, or a program could require users to exit and re-start if entered information was rejected.

With the changing business environment due to utility deregulation, however, customers are placing increasing importance on user-friendliness. For commercial success, it is no longer enough to provide leading-edge technical answers without also providing the level of usability that customers have come to expect from off-the-shelf software.

Our Software Quality Team therefore emphasizes usability. We operate independently from the rigorously documented and controlled Quality Assurance program that tests calculated results. We concentrate on improving those aspects of software quality that lie outside the domain of science and engineering expertise. For example, we work to answer questions such as "How easy is this software to use?" and "Did we develop what customers wanted?"

Because EPRI uses so many different independent software development organizations, our key Software Quality challenge is to reduce the variability in the level of user-friendliness that we deliver to our customers. The following sections describe what we have done to address this challenge, our results so far, and key success factors.

Usability Program Beginnings

What we did

Motivated by comments from customers on the difficulty of using our software, EPRI management commissioned an independent consulting group to perform an initial software business study. The survey portion of the study included 560 written surveys and over 100 interviews with customers, developers, and EPRI management.

The good news from the initial study was that customers placed a high value on EPRI software. The bad news was that our performance did not meet customer expectations. The most frequent customer complaints were:

- Products not meeting committed delivery dates
- Missed expectations/some desired functions not present
- Difficult installation
- Weak user documentation
- Hard to use

The study presented several specific recommendations for improving the usability of EPRI software:

- Get more input from end-users during development
- Establish contractor and software standards
- Evaluate and qualify contractors
- Do acceptance testing that emphasizes usability
- Track customer feedback after release

Results: Actions Taken

Management Quality Policy
In 1996, EPRI senior management established and publicized a Quality Policy emphasizing software usability. Its key elements, with some expansions, continue today:

- Usability acceptance testing for all software, with "A" grade required for release
- Software Quality Manager review and approval of developers before development starts
- Software Quality Team guidance and advice for project managers and developers, at no cost to them
- Customer feedback on functions and features during beta test
- Executive (COO) review of a software grading report summary

EPRI management specifically included executive review and measurement requirements in the Quality Policy, because these were seen as critical to achieving the buy-in needed.

Success Factors in Software Usability Program Startup

We see the key reasons for the success of our usability program start-up as:

- Management proactively listening to customers and acting on customer concerns
- A clearly defined and publicized program--what we would do/would not do
- Executive sponsorship of the Quality Policy
- Management commitment to hire an outside resource as Software Quality Manager, when inside skill levels were not sufficient
- Measures linking software usability test results to bonuses for individuals, groups, and the entire business

Measuring the software usability grading results and linking these to bonuses was a particularly important success factor for the program start-up, because it provided incentives for people to work in new ways. Now that the usability testing program is well underway, "A" grades are expected and are no longer specifically rewarded.

Usability Testing

We started our usability improvement program with testing, as this was the easiest way to gain quick successes.

Because customers are the judges of quality, we have used customer satisfaction data to drive our program since its beginnings in the Quality Policy. Customer-identified concerns define the

core elements that we test, and information from ongoing customer surveys continues to guide us as we improve our usability testing process.

What we did

Defined our strategy and objectives
Because testing can address some issues much more easily than others, you need a strategy: what will you test, and what will you not test?

Our test strategy focused on the usability problems stressed by customers in the initial software study that could be tested most easily. These included installation, user documentation, and important user interface difficulties. Some customer desires, such as appropriate/useful features and functions, were deliberately omitted from our testing. Other important desires, such as timeliness, do not belong to software testing.

We defined the objectives for the usability testing process as:
- Narrow the existing wide range of software user-friendliness
- Address key areas of customer concern revealed by survey data
- Strengthen the business against growing commercial competition in software
- Improve the development process through feedback of lessons learned

Set up the test organization
Since EPRI lacked in-house software expertise, the Software Quality Manager selected an independent testing contractor after reviewing proposals from several organizations.

The main testing contractor selection criteria were:
- Cost of testing service
- Skills and experience of individuals and the business in software testing
- Commitment to measure and reduce turnaround time and cost of testing

The selected testing contractor offered an important advantage: the ability to adjust resources to minimize cost to us. We did not need a full testing staff always available.

The EPRI Software Quality Manager set up these metrics and requirements:

- Turn-around time per test: commitment to measure and reduce it
- Cost per test: commitment to reduce it
- Requirement to document the test process and lessons learned

Established testing scope and grading criteria
The usability testing process covered six evaluation areas, which we continue to use. These appear in Table 1 below.

Table 1: Usability Testing Evaluation Areas

Evaluation Area	Definition
Virus check	Are all media free of viruses?
Installation	Are user instructions complete and correct? Can software be installed in a non-default drive? Does software provide backup or warning if modifying system files or shared files?
User documentation	Is the User Manual complete and accurate in describing the software and menu options? Are system requirements present and accurate? Is accurate help provided (including on-line help)?
Example problems (test cases)	Are solved examples or a tutorial included in the user documentation? Do these work as described?
GUI (graphical user interface) consistency	Is screen layout logical and consistent (preferably Windows standard)? Do main functions operate without crashes, lockups, abnormal exits, etc.?
Year 2000 check	Does software meet the EPRI definition of Year 2000 Ready?

The testing contractor and the EPRI Software Quality Manager developed a set of usability grading criteria reflecting an "A" to "F" scale. Table 2 below summarizes the grading criteria; the complete criteria document appears in Appendix 1.

Table 2: Summary of Usability Grading Criteria

Grade	Cause	Action
A	No major usability problems identified; test report contains recommendations only	Approved for distribution
B	Any one of the following: Installation problems; important problems such as crashes, lockups, abnormal exits; virus detected; illogical or non-standard user interface; non-functioning menu items; insufficient or inaccurate user documentation; solved examples missing or do not follow descriptions; not Year 2000 Ready	Fix and resubmit software for testing.
C	Installation failure; frequent occurrences or combinations of items under "B" grade	Fix and resubmit software for testing.
F	Combination of two or more items under "C" grade	Fix and resubmit software for testing.

All software development contracts contain detailed information on the six usability test evaluation areas as well as the grading criteria. This ensures that developers and project managers are familiar with the standards for user-friendliness that we will use during acceptance testing. Our software development guidelines and requirements also appear on the EPRI Web site. (See the References section at the end of this paper for access details.)

Gathered data on lessons learned

Over the course of testing about 150 software applications for user-friendliness, the testing contractor collected a large number of lessons learned—examples of usability errors encountered in each of the six evaluation areas.

In order to alert developers and project managers to the most common problems, the Software Quality Team published an ongoing series of Lessons Learned documents. We discussed these at special seminars for developers and project managers, we posted them on our intranet, and we sent copies to anyone who asked how to get an "A" grade. As a result, we stopped seeing "F" grade software after the first year of testing. We saw almost no "C" grade software after the second year.

Brought the test process in-house

Reducing testing cost and turn-around time continued to be important goals. Over a 2 ½ year period, the testing contractor had been unable to lower the cost per test and had achieved only a 15% reduction in turn-around time, which remained at about 3 weeks per test.

We decided to bring the usability testing process in-house, primarily as a cost-saving measure. The extensive Lessons Learned documents enabled us to develop a detailed set of checklists that a team of junior-level testers could follow to create a repeatable usability testing process. The evaluation areas and grading criteria have remained the same.

User documentation, which includes both user manuals and on-line help, drives the usability testing process because we emphasize the end-user view of the software. Testers do detailed checks of the user instructions for installation and solved examples. All documented inputs, outputs, and screen shots are checked for consistency with the software. Testing also includes a checklist for a type of ad hoc testing we call "smart monkey testing." Here, the tester enters text into numeric fields, tries extreme values to test range checking, interrupts "save" commands, and otherwise tries to break the software. An example of one of the test process checklists--the User Interface Checklist--appears in Appendix 2.

The new test group, which resides within the development organization, has taken on an additional role besides acceptance testing. This role involves assisting developers with debugging delivered software. Under the new approach, final testing is halted and immediately discussed with the developer if a tester encounters problems that would lead to a grade lower than "A." Thus, a software package may rapidly make several attempts at final test before it achieves an "A" grade.

The independent Corporate Software Quality group reviews all final usability test reports in detail, and discusses these with the testers before making the final decision on release.

Results

We have improved customer satisfaction, as well as the usability testing process itself.

Customer satisfaction metrics

Our usability testing has had important positive effects upon customers. Data on the significant improvements experienced by customers appear in the following two sections of this paper: Customer Satisfaction Surveys and Customer-Reported Problems.

Test process metrics

The new internal testing process has created order-of magnitude improvements in testing cost and turn-around time, as shown in Table 3 below.

Table 3: Test Process Comparison

Metric	1997-1998 Test Process	1999 Test Process
Average cost per test	$3100 per final test	$325 per final test
Turn-around 10 days or less	5% of tests	47% of tests
Average turnaround time	20 days	6 days

Success Factors

We view the actions below as essential to our successes with usability testing:

- Established clear objectives and desired outcomes based on customer data
- Established clear testing contractor selection criteria emphasizing cost control
- Defined clear grading criteria for evaluating software usability
- Established and tracked metrics on the performance of the testing organization
- Documented Lessons Learned, as well as the testing process
- Developed detailed checklists to guide new testers and standardize the testing process

Customer Satisfaction Surveys

Customer satisfaction data drive our software usability improvement program. We have completed three customer satisfaction surveys: 1996, 1997, and 1999. We intend to continue

these surveys at approximately one-year intervals, because they help us to evaluate the effect of our improvement efforts, and to prioritize further action.

What we did

Defined objectives and desired outcomes
Our first goal was to establish a baseline before the improvement program had taken hold, in order to demonstrate progress made. Other objectives, which continue to guide our customer satisfaction measurement efforts, are:

- Define gaps between customer expectations and our performance in order to set priorities
- Assess the effectiveness of usability testing and other improvement actions
- Track trends year-to year

Survey method
We partner with an expert supplier to develop the survey questions, and also to implement the survey process itself. This approach has several advantages: we avoid the bias introduced by internal interviewers, we benefit from the supplier's skills in interviewing and data interpretation, and we get results faster.

Each year, we select 20-30 software products for the survey. The list is developed based upon a combination of order quantity and management choice. Interviewers complete 350-400 phone surveys, contacting at least 15-20 users per product. The large number of data points enables drawing conclusions with confidence, and avoids the uncertainties of small sample sizes.

We use telephone interviews with end users of the software, because this method allows interviewers to ask clarifying questions and to obtain valuable detailed comments. We insist upon talking with users, rather than managers or other company contacts who are often easier to reach.

The first year, we did use mail-in/fax-in surveys for about half of the software products surveyed. However, we discovered that the gain in the richness of information justified the increased cost of phone interviews, and we now use only telephone surveys. Appendix 3 contains an extract from our interview guide.

Results

We have seen positive results in overall customer satisfaction as well as in the usability-related items addressed by our testing.

To clearly point out areas for improvement, we use a gap analysis approach that highlights the difference between customer expectations and our performance. It is not enough simply to improve our performance level; we must improve relative to what our customers expect from us.

Our goal is to achieve high ratings from customers in the areas that are most important to them. In the customer satisfaction results graphed below (Figures 1-3), the Expectations data reflect the percentage of customers rating an item's importance as "above average" or "essential." The Performance data show the percentage of customers rating our performance as "good" or excellent." Smaller gap numbers mean higher customer satisfaction where expectations exceed our performance.

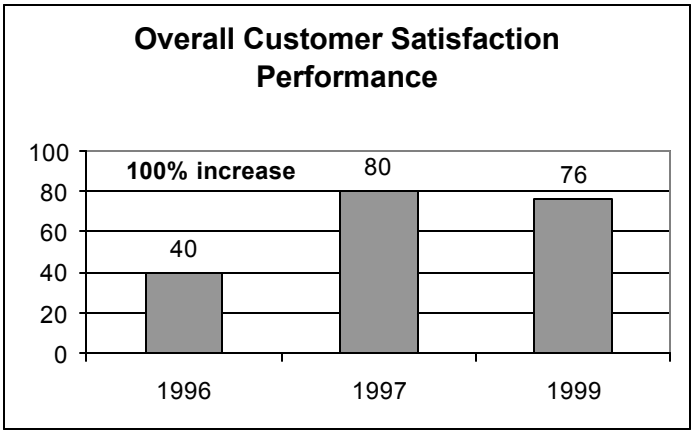
Overall customer satisfaction

After the baseline survey in 1996, the senior executives in each business group chose specific improvement goals. The achievement of these goals was measured and was linked to individual, group, and corporate bonuses.

As Figure 1 shows, overall customer satisfaction increased 100% in the first year as a result of this effort. The reason? "What You Measure Is What You Get!" (WYMIWYG) The WYMIWYG principle is an important success factor in all of our Software Quality Team improvement work: when people are measured and rewarded for their achievements, they will work in new ways.

Figure 1: Overall customer satisfaction increase of 100% in the first year

This figure shows the percentage of customers who rated our performance "good" or "excellent" on those items that were "above average" or "essential" to them in importance.



After the initial 100% jump, performance stabilized at around 80%. We believe this shows we have already obtained the easiest, most immediate, and most visible results.

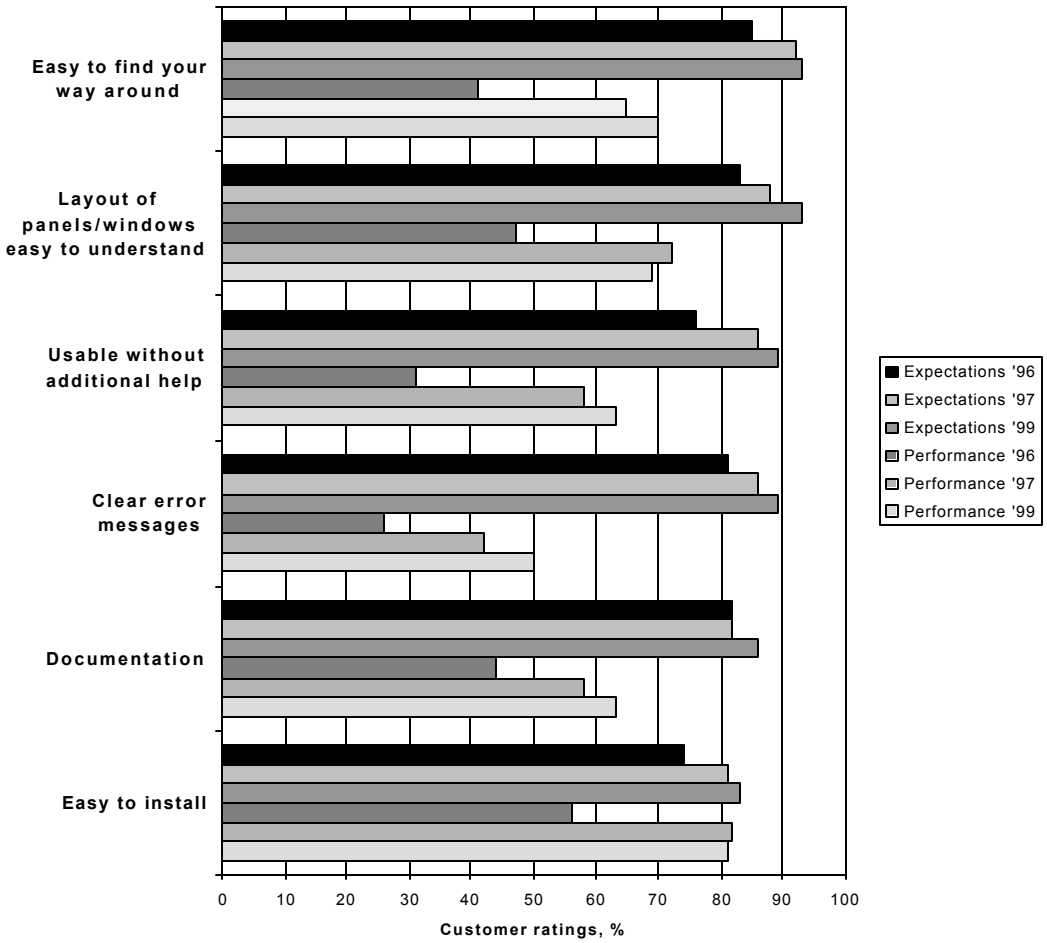
Further, we believe these results imply that we may have reached a limit to the gains we can achieve through our usability acceptance testing program. To further increase customer satisfaction, we will need to increase our Software Quality Team involvement earlier in the software development life cycle.

Usability item trends

In the baseline survey of 1996, six of the top ten customer concerns involved software usability. We have tracked these concerns in each subsequent survey.

The three-year trends in Figure 2 show that customer expectations for each key usability item are rising, but that our performance is more than keeping up.

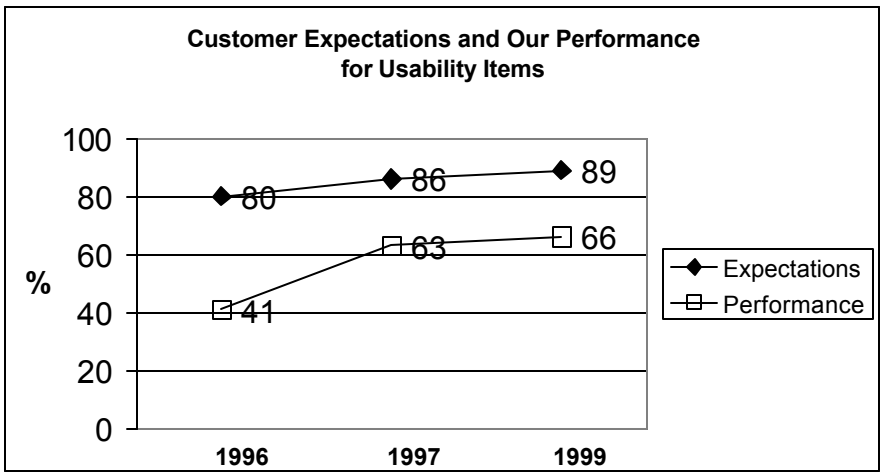
Figure 2: Trends in customer usability expectations and our performance



We believe that these trends reflect the success of our usability testing process. Some satisfaction gaps, such as installation, have closed significantly as a result of testing.

Figure 3 below displays the expectation and performance scores for the key usability items viewed together as a group. It clearly shows that we are closing the gap between customer expectations and our performance.

Figure 3: Gaps for usability performance items narrow as expectations rise



Success factors

Our customer satisfaction survey process has succeeded because we:

- Partner with an expert supplier of customer satisfaction data
- Express the results in "gap analysis" language which is familiar to the business, and which highlights action areas
- Use prior experience with the initial software study to focus the interview questions on key customer concerns
- Ensure that the number of data points is large enough to generalize with reasonable confidence
- Use the same core set of questions in each survey to enable trend analysis
- Interview end users of the software, rather than easier-to-reach company contacts
- Compare results with the benchmark survey taken when the Quality Policy was first issued. This enables us to show the progress we are making.

Customer-Reported Problems

What we did

To capture field problems, we set up a tracking system at our customer software support center. Support employees log customer trouble calls, as well as comments from software registration cards. Our usability problem database includes contact information, problem detail and resolution, date, software name, and product release version. The data are categorized into groups that reflect our usability testing areas.

Our goals in tracking customer-reported problems are to:

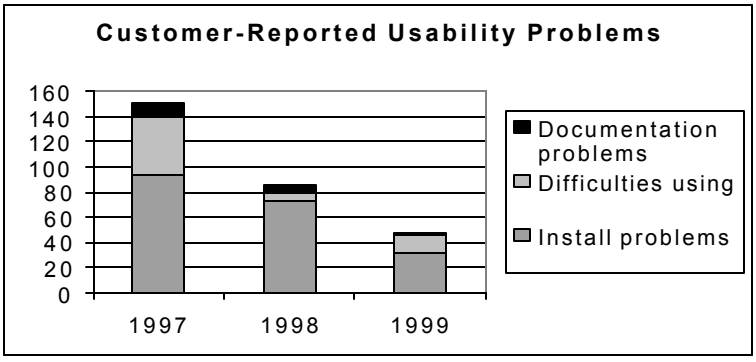
- Assess results of usability testing
- Track key customer concerns to guide future action

Results

We have customer-reported problem data for 1997, 1998, and 1999. During these three years, customers have reported a total of about 300 software usability problems to us. Each year, we ship 4000-5000 orders.

Figure 4 below shows that the number of usability problem calls received is declining at about 50% yearly.

Figure 4: Count of customer-reported usability problems 97-98-99



We believe the number of usability problems is dropping as a direct result of our testing, because we specifically test for the items grouped in the installation, documentation, and "difficulties using" categories shown in Figure 4.

But we realize that many customers who experience problems with user-friendliness will never report them to us. Thus, we do not use the self-reported data as a reliable indicator of

perceptions throughout our customer base. Instead, we rely upon the customer satisfaction survey data as a more accurate measure of our perceived software usability.

Success factors

The items below are central to our customer-reported problems program:

- Maintaining a database history of phone calls and registration cards
- Aligning data categories with testing areas to evaluate usability testing effectiveness

Usability Risk Assessment Pilot Program

We initiated a risk assessment process along with the in-house usability testing, because we realized that not all "A" grades indicate equal levels of user-friendliness. As described in the first section of this paper, our software is extensively tested for technical accuracy before it comes to the Software Quality Team for usability testing. Thus, our risk assessment program focuses solely on potential problems relating to the software user interface.

What we did

Set objectives

We began by establishing our goal: to identify and address the sources of usability problems that may still remain in the software after testing and acceptance. We intend to use the usability risk assessment process to:

- Understand the likelihood that users could experience a particular software application as "hard to use"
- Guide process improvements to address causes and continue to make EPRI software easier to use

Established a usability risk assessment scale

While we do not measure test coverage for our usability testing, we know there is much of the code that we do not test for user-friendliness: our testing focuses principally on the developer-supplied solved examples in the user documentation.

Our received software tends to fall into two groups: one group passes through usability testing in a straightforward manner and receives an "A" grade on the first try. The second group contains usability problems that prevent achieving an "A" the first time through testing. Examples of these types of problems appear in the Grading Criteria (Appendix 1).

As a starting point in evaluating the usability risk that remains in the software after the "A" grade, our Software Quality group has decided to test the following hypothesis: important

usability errors seen during testing mean a higher likelihood that customers will experience other usability errors we did not detect.

The usability risk scale summarized in Table 4 below is based upon how much re-testing it has taken to release a particular application, and on what types of changes were required to improve user-friendliness before final release. Appendix 4 contains the full usability risk assessment scale.

Table 4: Summary of Usability Risk Assessment Scale

Usability Risk Level	Description
5	**High** – Software had several attempts at final test; user-friendliness is not up to our usual level at release
4	**Above Normal**– Several attempts at final test; important user documentation fixes required; non-minor software changes made after "A" grade
3	**Normal**– One final test attempt; few documentation fixes required; only minor software changes made after "A" grade
2	**Below Normal** – One final test attempt; software is easy to use, with only minor user interface problems; no required changes with "A" grade; software is not changed after testing
1	**Low** – Software is very user-friendly the first time tested; no documentation or software changes required or made

In Table 4, a level of 3 describes the normally-expected chance that a customer could experience usability difficulties after release. Levels 2 and 1 indicate a below-normal likelihood of difficulties, while levels 4 and 5 indicate a greater chance that a customer could have difficulties using an application. We plan to measure customer satisfaction and reported problems against the assigned usability risk levels to evaluate the accuracy of this scale.

We intend to align our testing and improvement activities with customer problems. If we encounter many field problems with risk level 4 software, we will set up special requirements for follow-on releases of level 4 applications. For example, we could require developers to submit proof of their own usability testing results, and more solved example problems.

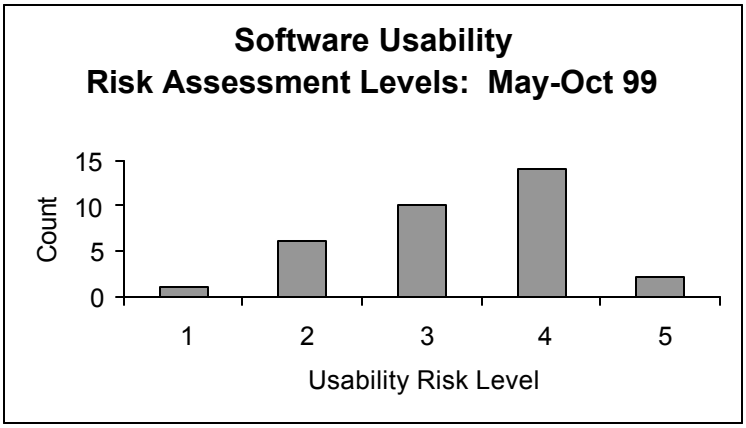
Results

This pilot program has existed for only a few months, so there are no results yet. We do have some preliminary measures in place, however.

During the first 6 months, 33 applications were assessed for usability risk. Roughly half achieved a level 3 or lower. The other half came in at a level of 4 or higher. The higher-usability-risk applications required multiple passes through final test, and we believe they could contain other usability problems that we did not detect during testing.

Figure 5 shows the usability risk levels assigned during the first 6 months of the pilot program. We anticipate that this program will take another 2-3 years to evaluate.

Figure 5: Distribution of usability risk levels for first 6 months of pilot program



Success factors

Our usability risk assessment pilot program will be a success if we demonstrate:

- High correlation between the assigned usability risk factor for an application, and customer-reported usability problems/satisfaction survey results
- Effective intervention in the development process to improve user-friendliness for follow-on releases of software initially assessed at higher risk levels

Summary

Beyond usability testing: next steps

We realize that you can't test user-friendliness, or other quality attributes, into a product at the end of development. Based upon the plateau in customer satisfaction at around 80% shown in Figure 1, we believe the time is right to increase our Software Quality Team involvement earlier in the software development life cycle.

A particularly important opportunity lies in addressing the quality of requirements. The highest customer expectation in each of the three customer satisfaction surveys--and consistently one of

the highest satisfaction gap items--was *"Meets my needs and solves my problems."* Clearly, a great deal of value lies in addressing requirements to make sure we are delivering the right product. Reviewing requirements early in the development process could have an important positive influence on this metric.

However, we must still continue usability acceptance testing in order to ensure we deliver consistently user-friendly software. Given our outsourcing approach and the need to generate improvements without direct control over the many different development processes used, we must work to increase customer satisfaction by selectively adding upstream checkpoints to a continuing program of final usability acceptance testing.

Key Lessons Learned

As we move ahead with usability testing, customer satisfaction surveys, tracking of customer-reported problems, and usability risk assessment, these important principles continue to guide us:

- Make strategic choices and set priorities: What will we do and what will we not do?
- Implement measures, even coarse ones, and compare results against a baseline to prove the worth of improvement work
- Start simple and quick, show immediate results

We have also made some interesting discoveries along the way:

- Customers continue to raise the bar on what they expect (Figure 2)
- You can successfully implement selected traditional quality improvement steps in a non-traditional software development environment

Through the efforts of many at EPRI, we have greatly improved the user-friendliness of our software over the past three years. We are pleased that our customers have recognized our efforts, and we intend to continue consistently delivering software that meets or exceeds their ever-increasing expectations.

Acknowledgments

John Houk, SatisfactionWorks, San Francisco, CA
Ken Doran, Test Director, S. Levy, Inc., Santa Clara, CA
Bob Lara, Software Quality Manager, EPRI, Palo Alto, CA
Jocelyn Ding, IT Director, EPRI, Palo Alto, CA.
Ramtin Mahini, Software Development Quality Manager, EPRI, Palo Alto, CA.
Mary McKenna, Software Test Manager, EPRI, Palo Alto, CA.

Useful References

Kaplan, Craig, Secrets of Software Quality, McGraw-Hill, New York, 1995.
Gitlow, Howard , The Deming Guide to Quality and Competitive Position, Prentice-Hall, Englewood Cliffs, N.J., 1987.
EPRI Web site: http://www.epri.com/eprisoftware/roadmap/homepage.html

Appendices

Appendix 1: Usability Acceptance Testing Grading Criteria
Appendix 2: Example of Tester's Checklist – User Interface (GUI) Checklist
Appendix 3: Extract from Customer Satisfaction Phone Survey Interview Guide
Appendix 4: Usability Risk Assessment Scale

Appendix 1: Usability Acceptance Testing Grading Criteria

EPRI's policy is to distribute only "A" grade software. The important usability problems noted for the non-"A" grades below must be addressed before release.

GRADE	CAUSE	ACTION
A	No *major* problems identified (see below for examples of major problems). *Minor* problem examples: mislabeled toolbar button or plot axis; inconsistent screen shot in manual; or cannot print from toolbar but can using menu.	Fix (or prepare a plan to fix) minor problems, review with EPRI Software Quality Manager, then distribute.

B	Any one of the following:
	• Installation problems (some typical examples):
	  • Insufficient or incorrect installation procedure.
	  • Application files placed into wrong directories.
	  • Software modifies the configuration or system files without warning or backing up originals.
	  • No program icon or group created (if applicable)
	  • Cannot install on a non-C: drive (e.g., D:).
	• 1 or 2 occurrences of General Protection Faults (GPFs) or infinite loops.
	• Virus detected on the distributed disks.
	• Frequent abnormal program exits through multiple event sequences (e.g., memory limits, bad input).
	• Non-standard, illogical Windows GUIs (if Win app).
	• Non-functioning icons or menu items (e.g., PRINT does not work; inactive options are not ghosted).
	• Insufficient/inaccurate printed documentation (e.g., no input/output views or descriptions of menu options).
	• Insufficient/non-functioning on-line help, without backup printed documentation.
	• Test cases:
	  • Lack of test cases, tutorial, or sample cases.
	  • Test cases abort or documented results not reproducible.
	• Combination of several problems with user interface, documentation, or on-line help.
		Fix major problems, prepare a plan for fixing minor problems, review with EPRI Software Quality Manager (SQM), then distribute or re-test. SQM will decide if re-test, or partial re-test, is required.

C	Any one of the following: • Installation failure. • 3 or more GPFs or infinite loops • Frequently encountered abnormal program exits through 3 or more sequences of events. • Combination of 3 bullet items classified under B grade. • Combination of 2 GPFs and either a test case abort or a non-	Fix problems, then resubmit for testing.

	reproducible documented test case result.	
F	Combination of two or more of bullet items classified under C grade.	Fix problems, then resubmit for testing.

Appendix 2: Example Tester's Checklist—User Interface (GUI) Checklist

Testers use this checklist to help them detect user interface errors. It is one of six lists that document the usability testing process. Numbers in parentheses refer to explanatory notes (not included).

Completed	Actions to check first time through.
	Do the windows fit on the screen?
	When opening another window does it appear on top?
	Do the words fit the window?
	Do the window titles match the window's function?
	Can the windows be resized?
	What effect does the previous action have on the code? Good or bad?
	Do all buttons function as expected?
	Does a minimized or closed window leave a blanked section on other windows that it once overlapped?
	How much can be entered into a field before it reacts?
	Does the GUI work? Or does it freeze up?
	Does the screen match the screen shots in the manual?
	Does the "x" (close button) in the upper right corner work?
	Does the "_" (minimize button) work?
	Does the GUI freeze up when a window is minimized and then restored?
	When switching from window to window by the pressing of a button, does the active window appear on top of all others, or does it appear behind the others?
	Can the user enter keyboard equivalents in lieu of pull-down menus?
	Do the windows display correctly in either large or small fonts mode?
	Do all menu commands work?
	Are all words in dialog boxes, windows, etc. spelled correctly?
	Are all the non-functional buttons grayed out? (9)
	Do all keyboard shortcuts listed next to the menu commands work? (10)
	Are all read-only data fields locked? (11)
	Do windows or dialogs completely fit on the screen at low display resolutions? (3)
	Does the tab key access the input fields and/or buttons in a logical order? (7)
	Will the program run when there is no printer or printer driver attached? (laptop as an example)

Appendix 3: Extract from Customer Satisfaction Phone Survey Interview Guide

Hello, my name is _____. I'm a consultant for EPRI and I've been asked to interview you on the subject of EPRI Distribution Software.

Could you spare me 15 minutes? _Yes _No (Reschedule or terminate)Reschedule date: _____

1. To begin with let's focus on (Product) / (Version). Do you use it? ___Yes __No

If "No" to Question 1:

1-A Why don't you use ___(Product)_____?

1-B What specific improvement(s) would encourage you to use it?

If "Yes" to Question 1:

1-C Which version do you use? _____

1-D How often do you use it? __Frequently __Occasionally _Seldom __Never

1-E How would you rate the Overall performance of _____(Product) ?
__Excellent ___Good ___Just OK ___Poor ___Very Poor

2. How important are the following criteria to you when evaluating software? Please rate them on a 5 point scale where 5 is Extremely Important, 3 is Average Importance and 1 is Not Important.

	Extremely Important	Above Average Importance	Average Importance	Below Average Importance	Not Important	Don't Know
Ease of Ordering						
A. Understanding content/features before ordering it	❑	❑	❑	❑	❑	❍
Timeliness						
B. Available when you needed it	❑	❑	❑	❑	❑	❍
C. Timely notification of availability	❑	❑	❑	❑	❑	❍
User Friendliness						
D. Easy to install	❑	❑	❑	❑	❑	❍
E. Program samples/test cases	❑	❑	❑	❑	❑	❍

F. Layout of panels/windows easy to understand	❑	❑	❑	❑	❑	❍
G. Easy to find your way around	❑	❑	❑	❑	❑	❍
H. Clear error messages	❑	❑	❑	❑	❑	❍
I. Usable without additional help	❑	❑	❑	❑	❑	❍
J. Usefulness of "help" menu	❑	❑	❑	❑	❑	❍
Overall Flexibility						
K. Easy to interface with other EPRI software	❑	❑	❑	❑	❑	❍
L. Easy to interface with commercial software	❑	❑	❑	❑	❑	❍
M. Ability to customize to your needs	❑	❑	❑	❑	❑	❍
Technical Content						
N. Adequacy of function (meets your needs and problems)	❑	❑	❑	❑	❑	❍
O. Better than commercially available alternatives	❑	❑	❑	❑	❑	❍
P. Timely content upgrades	❑	❑	❑	❑	❑	❍
Support						
Q. Phone support	❑	❑	❑	❑	❑	❍
R. Documentation	❑	❑	❑	❑	❑	❍
S. Timely correction of problems	❑	❑	❑	❑	❑	❍

3. Now, using another 5 point scale where 5 is Excellent, 3 is Just OK and 1 is Very Poor, please rate ___(product)__'s performance on the following criteria:
(Here, question 2 is repeated to gather data on performance)

4. Looking at your answers, there appear to be significant gaps between EPRI's performance and your expectations on items _____ and _____. Which of these is most important to you? Item: _____ Why? _____

5. What does EPRI need to do to close this gap?

Let's talk about … *(several questions follow on desired platforms, delivery)*

13. Is there anyone else in your company that we should be speaking to about this product?

_____

14. Looking back over the last few years, about how many pieces of EPRI software have you received? _____Pieces

15. On balance, how valuable have they been to you? _____Why? _____

16. Have you seen any noticeable difference in EPRI's software in the last year or so?
__Yes __No ___Don't know

17. If yes, what? _____

Yes	**No**	
❑	❑	Is it OK if we share your comments with EPRI's Senior Management?
❑	❑	Would you like an immediate personal response to your comments?

THANK YOU FOR YOUR TIME!

Appendix 4: Usability Risk Assessment Scale

We are using this scale in a pilot program to estimate the comparative user-friendliness of our software. We will need to gather 2-3 years of data to evaluate whether the scale is accurate.

5	**High likelihood of usability problems:** - Software has needed several attempts at final usability test - Software has had a poor start (e.g. change of developers, persistent usability errors) and continues to be user-unfriendly despite many fixes - Important usability problems remain when the software is at the final release decision point (e.g. difficult installation, abnormal exits, user help incomplete, difficulties with printing) - The likelihood of fixes causing more errors is seen as high - Software may need to be released with some important user-friendliness issues remaining
4	**Above-normal likelihood of usability problems:** - Software has needed more than one attempt at final usability test - Software has had a poor start (e.g. change of developers, persistent usability errors) but it appears to be user-friendly after required fixes - Software fixes and/or extensive documentation fixes are required as part of "A" grade - Developer voluntarily makes non-minor changes after "A" grade
3	**Normal likelihood of usability problems:** - Software had one prescreen and one final usability test - No potentially important user-friendliness issues remain - "A" grade is awarded with moderate documentation fixes and/or very minor software fixes required - Developer voluntarily makes minor software fixes after "A" grade
2	**Below-normal likelihood of usability problems** - Software is easy to use, with only minor GUI problems - Software has one prescreen and one final usability test - Only minor documentation fixes required (or none) as part of "A" grade; no software fixes required - Software is not changed by developer after testing
1	**Low likelihood of usability problems** - Software appears very user-friendly the first time it arrives at usability testing - Software goes directly to final usability test when received and is not returned for fixes (prescreen step is not needed) - "A" grade is issued with no required documentation or software fixes - Software is not changed by developer after testing

Marilyn Valentino

Marilyn Valentino is a software quality assurance engineer at EPRI, a producer of scientific software for small user communities. She has over 10 years' experience in quality assurance engineering. She is an experienced presenter and published author on front-end product definition processes, and has written and presented several technical conference papers on this subject.

Marilyn holds an M.S. degree in engineering (design) from Stanford University and undergraduate degrees in mechanical engineering, product design engineering, and communication----also from Stanford.

Conferences, Ms. Valentino has presented at include NEPCON (National Electronic Producers Conference), ASME (American Society of Mechanical Engineers), and The Symposium on Quality Function Deployment.