Customer Quality Improvement of Software Systems

Randy Hackbarth
Avaya Labs Research
4125 Johnson Avenue, Western Springs, IL 60558
randyh@avaya.com

Audris Mockus
Avaya Labs Research
211 Mt. Airy Road, Basking Ridge, NJ 07920
audris@avaya.com

John Palframan
Avaya Labs Research
211 Mt. Airy Road, Basking Ridge, NJ 07920
palframan@avaya.com

Ravi Sethi
Computer Science Department
University of Arizona, Tucson, AZ 85721
rsethi@email.arizona.edu

Ravi Sethi was with Avaya Labs Research when this work was done.

POSSIBLE TWEETS

- What's worse: ten software systems with one defect or one system with ten defects?
- Customer Quality: the chance that a customer system will encounter a software defect
- Top 1% of source code files have over 60% of fixes
- Focus quality assurance resources on the top 1%
- Avaya case study: improve customer quality of software systems and know it

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

Customer Quality Improvement of Software Systems

Randy Hackbarth, Audris Mockus, John Palframan, and Ravi Sethi

Avaya Labs Research

The proposed software quality improvement method is data driven and has three elements: (a) a downstream metric that quantifies quality as perceived by customers; (b) an upstream implementation quality index that measures the effectiveness of error removal practices during development; and (c) prioritization tools and techniques for focusing limited development resources. The downstream customer quality metric measures the impact on customers of serious defects; it is based on data collected after systems are deployed. The upstream implementation quality index serves as a predictor of future customer quality; it has a positive correlation with the customer quality metric. The prioritization techniques are used to focus limited resources on the riskiest files in the code. This paper is based on a multi-year program to improve the quality of delivered systems at Avaya, a global provider of business communication and collaboration systems. Governance for the Avaya program was provided by regular reviews with an R&D quality council.

Index Terms: Software quality method, customer perceived quality, data-driven software process improvement, software risk mitigation, case study

INTRODUCTION

We focus on quality as perceived by customers – *customer quality* in short – in terms of the impact on customers of serious defects. As Watts Humphrey notes, "the cost and time spent in removing software defects currently consumes such a large proportion of our efforts that it overwhelms everything else." [1] Other aspects of quality, such as whether a product does what customers expected, are outside the scope of this paper.

Let us refer to an installation of a product at a customer site as a *system*. We reserve the term customer found defect (*CFD*) for the relatively few customer service requests that survive thoroughly vetting by customer support and development personnel.

The *customer quality method* in this paper has the following elements:

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

- A customer quality metric based on the proportion of systems reporting CFDs. Ten systems reporting one CFD – even the same CFD – is worse than one system reporting ten CFDs.
- An index of error-removal practices during implementation that is a predictor of future post-install customer quality.
- The metrics are accompanied by prioritization techniques and tools for focusing limited resources on the riskiest files in the project's code repository.

Other metrics may be added – say, for testing practices – as long as they correlate with improved customer quality downstream. Further, the method allows other prioritization or risk-management techniques to be added.

Such a customer quality improvement method has been adopted company-wide at Avaya, a global provider of business communication and collaboration systems. Avaya already had a strong commitment to quality when the company faced quality issues with some of its products in 2011. With strong executive focus and governance provided by an R&D quality council, the method contributed to continuing 30+% year-over-year improvements in key customer quality metrics.

CUSTOMER FOUND DEFECTS

Not all defects are equal. Most defects are found and fixed during development or testing, before a product is delivered. Once the product is in use, customers have to observe an issue and care enough to report it, for the issue to reach a services organization. The issue must then survive various screening levels to be escalated to the development group. The development team does its own screening before identifying the issue as a software defect. At Avaya, less than 1% of customer service requests materialize as CFDs; see Figure 1.

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

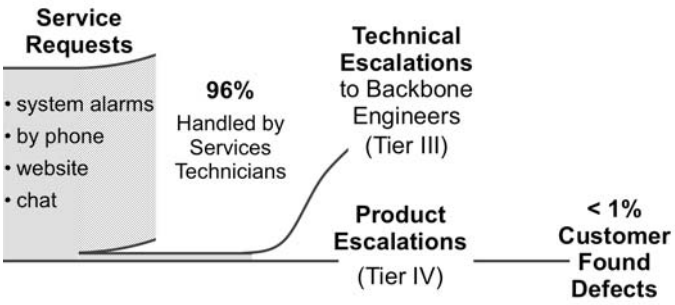

Figure 1. At Avaya, less than 1% of Customer Service Requests are classified as Customer Found Defects (CFDs).

FIELD QUALITY: CUSTOMER QUALITY METRIC (CQM)

The metric for customer quality is based on the fraction of systems affected by defects. It represents the probability that a randomly chosen customer will be affected. Lower probability is better. We say that a system is affected by a defect even if the same defect has been previously reported. A few systems reporting many defects is better than many customer sites affected by a few defects.

The fraction of affected systems better reflects the impact on customers than traditional product quality metrics [2] like defect density and the number of CFDs. Defect density is a property of the code rather than customer experience. The experience across Avaya products is that the number of CFDs measures the breadth of product deployment rather than the probability that a customer system will be affected. Figure 2 shows data for releases of the Avaya Aura® Communication Manager. Mature quality practices and a wealth of data make it a good candidate for illustrating trends. Based on 272,000 system installs and upgrades since 2002, the number of CFDs increases linearly with the number of installed systems.

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

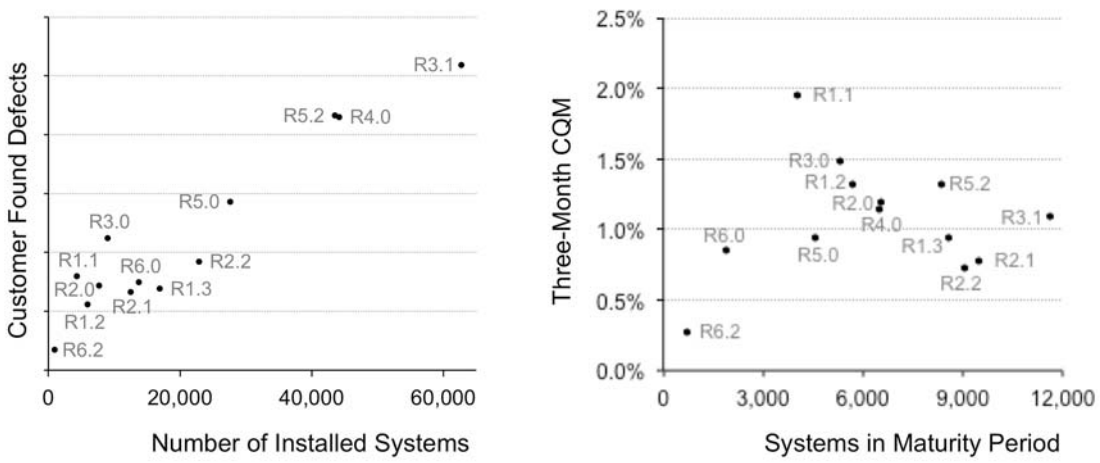

Figure 2. (a) Number of CFDs measures installs. (b) Three-Month CQM (lower is better).

To facilitate quality comparisons across products and releases, we use two parameters:

- *Product maturity period*, *m* (shaded in Figure 3). Quality improves as a product matures, since early defects get fixed and later installs go more smoothly. The product maturity period is the first *m* months after release. At Avaya, *m* is usually 7 months.

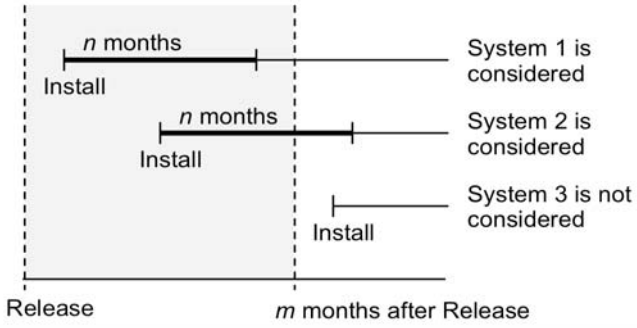

Figure 3. To be considered for the Customer Quality Metric, a system must be installed within the m-month product maturity period (shaded), and report a CFD within an n-post-install interval.

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

- *Post-install interval*, *n* (solid lines in Figure 3). While a longer interval captures more issues, thus providing a more accurate indication of customer experience, it requires waiting longer after release for the data. At Avaya, post-release quality is estimated using one, three, and six month intervals.

Definition. The *n-month Customer Quality Metric (CQM)* [3] is the fraction of systems installed within the first *m* months after release that have a trouble ticket leading to a CFD within their *n*-month post-install interval.

Lower CQM is better, since lower implies proportionately fewer defects. Within Avaya, the current three-month CQM standard for new releases is 2%. The three-month CQM values for releases 1.1 through 6.2 of Communication Manager in Figure 2(b) are all below the 2% corporate standard. CQM values have been dropping in Avaya: 77% of tracked projects met this standard in 2013, up from 30% in 2011.

ERROR REMOVAL: IMPLEMENTATION QUALITY INDEX (IQI)

What development practices does a project need to improve today, in anticipation of improved customer quality in the future? The scoring mechanism for IQI (defined below) provides development teams with guidance on where to invest proactively in error-removal practices.

Definition. The *Implementation Quality Index (IQI)* for a development project is a measure of the effectiveness with which the project engages in four error-removal practices:

- Static analysis, using industry standard tools.
- Code coverage, e.g., developing unit "white-box" tests coincident with writing code and assuring adequate coverage via execution of a code coverage tool
- Code reviews and inspections.
- Automated regression testing, primarily "black-box" tests.

Each practice is assigned a score on a scale of 0-4; higher is better. Scoring is based on criteria specific to the practice, with 4 for "done well," 2 for "done partially," 0 for "done poorly or not at all."

IQI is the average of the scores for the individual practices.

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

The Avaya standard for IQI is 3.0; see Figure 4. Although diminishing returns start to set in, individual teams perceive enough benefit that they are increasingly setting higher targets than 3.0. An IQI score below 2.0 reflects poor practices.

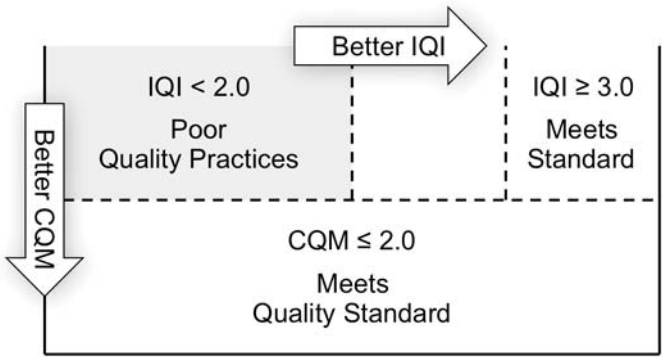

Figure 4. The Implementation Quality Index (IQI) is scored on a scale of 0-4.

The IQI practices themselves are standard industry practices. Their combination is known to be effective for error removal: from the benchmarking data provided by Capers Jones [2], the combination of reviews, static analysis, and testing is 85%-99% effective in error-removal. The IQI practices relate to several CMMI level 3 process areas, such as Technical Solution (TS), Verification (VER), and Validation (VAL) as well as the level 2 process area, Measurement and Analysis (MA).

Scoring for IQI. At Avaya, scoring for IQI is done in two stages. The initial scoring is done by the projects themselves. For the initial scoring, projects are provided with a standard template and detailed guidelines, specific to each practice, on what would be considered a top score (4), a moderate score (2) or a poor score (0). See Table I for a summary of the scoring guidelines.

An R&D Quality Council often adjusts the initial score during a review with probing questions; e.g.,

- How consistent is the scoring, compared to other projects?
- How effectively is the team engaging in the practices?
- Are they acting on the defects uncovered?

The resulting IQI score is accompanied by supporting comments that capture any concerns in plain English.

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

Table I. Guidelines for scoring practices for the Implementation Quality Index (IQI).

PRACTICE		GREEN (4)	YELLOW (2)	RED (0)
Static Analysis				
	Run Regularly?	Part of build process	Occasionally	Sparingly or not at all
	Defects Tracked?	Yes	No	No
	High Impact Defects Corrected?	All	Most	Few
Code Coverage				
	Percentage of code	≥ 75%	≥ 50%	< 20%
Code Reviews				
	Extent of Reviews	All new/changed code	Most code	Ad-hoc or no reviews
Automated Regression Testing				
	Percentage of Tests	≥ 70%	≥ 40%	< 20%
	Investment	Ongoing	Much manual testing	Lacking

Correlation with Customer Quality. Based on experience with over 50 major projects, we have observed a positive correlation between improved development practices (higher IQI) and improved field quality (lower CQM); see Figure 4. This empirical relationship justifies the time and effort spent in improving IQI.

RISK MITIGATION

Simply providing information about the risk (high CQM) and suggested process improvements (through IQI scoring) was not enough: the projects needed help with focusing their improvement efforts. We therefore developed risk-prediction techniques and tools for prioritizing remediation actions.

Risk Factors. The intuition that some parts of the source code are riskier than others is not new.

- Anecdotal Evidence. Grady and Caswell recommend focus on "the most complex modules. [4]" At IBM in the 1980s, Humphrey recalls a case where "86 percent of the [1600] modules had had no defects in three years. So 14 percent of the modules had all the defects, and 3 percent had half of them. [5]"
- Defect Prediction. A number of studies have shown that prior changes are a good predictor of post-release defects [6,7].

Practical applications of such predictions have lagged, however [8]. Risk prediction needs to be focused and accompanied with tool support. For example,

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

"20% of the files" does not provide enough guidance for the deployment of limited resources.

Based on regression analysis of historical defect data, the weighted sum of the following factors (over the trailing three years) was a good predictor of *risky* files—files likely to have future CFDs:

- Number of past CFDs fixed in the file × 20
- Number of file authors who have left × 10
- Number of modification requests (MRs) × 0.1
- Number of unique versions × 0.01

The weights, 0.01 for unique versions and 20 for past CFDs, take into account the fact that there can be orders of magnitude more versions than CFDs; see the examples in Figure 5.

CFDs by LATEST DATE (FILES by RISKIEST)	MRs	AUTHORS	RELATED FILES
1) <PROJECT>/trunk/EPM/SMS/POManager/config/upgr/<FILE1.cpp>: 343 versions			
wi01079507 **2013-02-14** *CFD:ImportManager and import purge changes if there are lots of completed import jobs …* wi00839993 **2010-12-09** *CFD:ftp import job stuck due to invalid ip* 2 CFDs are 2% of the 78 MRs	78 MRs	19 Authors 4 departed (21%)	6 Files
2) <PROJECT> /trunk/src/mpp/media_svc/session_mgr/<FILE2.ccp>: 498 versions			
wi01162616 **2014-03-28** *Customer Feedback: Need document help file update and error message fix on Multi-Tenancy* wi01051778 **2012-10-11** *CFD: Alarm management behavior on 6.0.1.0.0.0801* 2 CFDs are 1% of the 163 MRs	163 MRs 3 of 'SDE' 2 of 'Web Mgmt'	70 Authors 30 departed (42%)	100 Files of 180

Figure 5. A rendering of a risk-mitigation tool that links analysis with code, developer, and organizational data.

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

For Avaya projects, our experience is that the top 1% of files identified by this heuristic contribute to fixes to 60+% of the CFDs. Mockus et al. describe an earlier version of the risk predictor [9].

Tool Support. The interactive risk-mitigation tool illustrated in Figure 5 supports both problem discovery and problem resolution. It satisfies the diverse needs of developers tasked with fixing the code and product managers tasked with budgeting and scheduling risk-reduction efforts.

The tool links risky file analysis with code, developer, and organizational data. Code data includes the source code of individual files, modification requests (MRs), related files (files that were identical in the past to a candidate risky file), and other data from version control systems. Organizational data includes historical data from corporate directories, to identify authors who have left and when they left the organization. From code data, we can infer the expertise of each author with this and other files. MRs and CFDs provide helpful context to those who are evaluating what actions to take with each risky file. The risk-mitigation tool builds on the expertise browser [10].

The tool accommodates the variety of defect amelioration approaches that were in use. The nature of the risk and future development plans led to policies like the following:

- Place high-risk areas into a "control" program where changes are discouraged or, when necessary, require better inspections and testing.
- Assign owners for areas that are risky because of lost expertise. Give owners sole responsibility for making most of the changes and overseeing others working in the area. The intent was to build expertise and increase accountability.
- Consider refactoring or reengineering the riskiest areas that are expected to see much new development.

For one typical project, 16 files were candidates for control programs and 1 file was identified for reengineering. Often projects spread risk reduction work over several releases, starting with the easiest-to-implement steps, such as control programs and ownership/governance policies.

DISCUSSION

The methods in this paper have contributed to dramatic improvements in customer quality across Avaya. Between 2012 and 2014,

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

- The average customer quality metric (CQM) dropped from 2.9% to well below 1%.
- The average implementation quality index (IQI) improved by 50%.
- Avaya's experience is that customer perceptions of product quality are a key contributor to customer satisfaction, measured by net promoter score (NPS) [11]. NPS has increased by 60%.

Others seeking to apply the customer quality improvement method to their own organizations might proceed as follows:

- If needed, establish data collection about customer deployments, service alarms and requests, version control, code change information, and related organizational data.
- Measure quality from a customer perspective, using a metric like CQM, which reflects the fraction of customer systems affected by defects, rather than the number of defects. In analogy with testing, use a black box customer quality metric rather than a white box source code metric. As a reference point, the initial Avaya standard for CQM was 2%.
- Once a customer quality metric is in place, use an in-process scoring mechanism like IQI, so development teams can improve their practices today, in anticipation of improved customer quality in the future, after the project is complete and systems are deployed at customer sites.
- Empirically, risk is concentrated in a small fraction of the files, so use a heuristic like the one for risky files to focus improvement efforts. At Avaya, the teams benefited greatly from concentrating on the top 1% of risky files.

Any quality improvement program needs governance and strong executive support. The customer quality improvement method in this paper builds on an active research program to improve the state of software in Avaya, in partnership with the business groups. Measurable improvements in the state of software at Avaya accelerated after CQM and IQI became part of the corporate quality metrics.

ACKNOWLEDGEMENTS

David Weiss built up and led the software technology research program while he was with Avaya Labs. Evelyn Moritz created the IQI metric and championed its use along with quality council leaders Sarah Kiefhaber and Mike Jubenville. Jerry

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

Glembocki, Saied Seghatoleslami, Dan Kovacs, and Lee Laskin were influential in establishing the corporate metric program for CQM and IQI.

Jon Bentley provided a careful and helpful review of this paper.

We also thank Avaya R&D leaders and team members who have embraced quality-focused software development and the IQI and CQM metrics in particular as a means of tracking progress.

REFERENCES

1. W. S. Humphrey. 2004. Defective software works. Software Engineering Institute, Carnegie Mellon University. (January 2004). Retrieved January 16, 2014, http://www.sei.cmu.edu/library/abstracts/news-at-sei/wattsnew20041.cfm
2. C. Jones. 2013. Software quality in 2013: a survey of the state of the art. Retrieved January 17, 2014 from http:// http://namcookanalytics.com/software-quality-survey-state-art/
3. A. Mockus and D. M. Weiss. 2008. Interval quality: relating customer-perceived quality to process quality. In *International Conference on Software Engineering (ICSE '08)*. ACM Press, New York, NY, 723–732.
4. R. B. Grady and D. L. Caswell. 1987. *Software Metrics: Establishing a Company-Wide Program*. Prentice-Hall, Englewood Cliffs, NJ.
5. W. S. Humphrey. 2009. Oral history of Watts Humphrey. Interviewed by Grady Booch: June 17-22, 2009. Computer History Museum, Reference No. X5584.2010.
6. T. J. Yu, V. Y. Shen, and H. E. Dunsmore. 1988. An analysis of several soft ware defect models. *IEEE Transactions on Software Engineering* 14,9 (September 1988) 1261–1270.
7. T. J. Ostrand, E. J. Weyuker, and R. E. Bell. Where the bugs are. *International Symposium on Software Testing and Analysis (ISSTA '04)* ACM Press, New York, NY, 86–96.
8. E. Shihab, A. Mockus, Y. Kamei, B. Adams, and A. E. Hassan. 2011. High-impact defects: a study of breakage and surprise defects. In *European Conference on Software Engineering and ACM SIGSOFT Symposium on Foundations of Software Engineering (ECSE/FSE '11)*. ACM Press, New York, NY, 300–310.
9. A. Mockus, R. Hackbarth, and J. D. Palframan. 2013. Risky files: an approach to focus quality improvement effort. In *European Conference on Software Engineering and ACM SIGSOFT Symposium on Foundations of Software Engineering (ECSE/FSE '13)*. ACM Press, New York, NY, 691–694.
10. A. Mockus and J. Herbsleb. 2002. Expertise browser: a quantitative approach to identifying expertise. In International Conference on Software Engineering (ICSE '02). ACM Press, New York, NY, 503–512.
11. F. Reichheld and R. Markey, 2011. *The Ultimate Question 2.0: How Net Promoter Companies Thrive in a Customer-Driven World*, Harvard Business Review Press, (September 20, 2011).

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

ABOUT THE AUTHORS

320

Randy Hackbarth:

325 Randy coordinates a team dedicated to improving the state of the practice of software development in Avaya . Before joining Avaya, Randy worked for 20 years at Bell Labs, with a focus on establishing, coordinating and contributing to business unit - research partnership projects. He has an MS in Computer Science and an MA in Mathematics, both from the University of Wisconsin-Madison. Randy is a member of IEEE and ACM. Contact him at randyh@avaya.com.

330

Audris Mockus:

335 Audris Mockus is affiliated with the University of Tennessee and Avaya Labs Research. He received a B.S. in Applied Mathematics from Moscow Institute of Physics and Technology and a Ph.D. in Statistics from Carnegie Mellon University. Contact him at audris@avaya.com.

340 **John Palframan:**

John is a research scientist with Avaya Labs. He works with Randy on improving software practices in Avaya. Before joining Avaya, John was a technical manager in Bell Labs responsible for developing
345 communications software and software development tools. He has an M.Math and a B.Math (co-op) from the University of Waterloo. John is a member of the IEEE. Contact him at palframan@avaya.com .

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE

Ravi Sethi

Ravi Sethi launched the research organization in Avaya and was president of Avaya Labs before joining the University of Arizona. He is a co-author of the popular “dragon book” on compilers. Ravi has a B.Tech. from IIT Kanpur and a Ph.D. from Princeton. He is an ACM Fellow and a member of IEEE. Contact him at rsethi@email.arizona.edu .

Digital Object Indentifier 10.1109/MS.2015.76 0740-7459/$26.00 2015 IEEE