Assessing the State of Software in a Large Enterprise: A Twelve Year Retrospective

Randy Hackbarth*[1], Audris Mockus*, John Palframan*, David Weiss[2]**

ABSTRACT

To be relevant to the goals of an enterprise, an industrial software engineering research organization must identify problems of interest to, and find solutions that have an impact on, software development within the company. Using a systematic measurement program both to identify the problems and assess the impact of solutions is important to satisfying this need. Avaya has had such a program in place for about 12 years. Every year we produce an annual report known as the State of Software in Avaya that describes software development trends throughout the company and that contains prioritized recommendations for improving Avaya's software development capabilities. The process of identifying trends and recommending actions for improvement starts with identifying the goals of the enterprise and uses the goal-question-metric (GQM) approach to identify relevant measures [1,4,5, 15]. There are three primary results:

- Insight into the enterprise's problems in software development,
- Recommendations for improving the development process, and
- Identification of problems that require research to solve.

This chapter focuses on the first two.

Our process for collecting software development data and analyzing it has undergone considerable evolution over time, and has had continuing impact, from both internal and external viewpoints. We use both qualitative measures (interviews, surveys) and quantitative measures (financial, organizational, code repository, defect repository, quality) to assess our impact. Our purpose in this chapter is two-fold. It provides a model for assessment, based on twelve years of Avaya experience, which others may emulate. Such assessment leads to continuing improvement and substantial impact. In addition it spotlights analyses and conclusions that we feel are common to software development today. Note that there are other organizations that conduct such assessments, but with few exceptions, such as [13,26], they usually don't publish details about their results or methods

We illustrate our process with examples from the Avaya Resource Center for Software Technology in Avaya Labs, whose purpose is to improve the state of software development and know it. "Know it" means that improvement should be subjectively evident and objectively quantifiable. "Know it" also means that one must be skilled at identifying the data sources, performing the appropriate analyses to answer the questions of interest, and validating that the data are accurate and appropriate for the purpose. We will use several examples to illustrate our results including (1)how and why we developed a measure of software quality that appeals to customers, (2)how and why we are studying the effectiveness of distributed software development, and (3)how and why we are helping development organizations to identify the riskier portions of their code base. We will discuss how we keep the company apprised of the current strengths and weaknesses of software development in Avaya through the publication of the annual State of Software in Avaya Report.

We also discuss the aspects of the assessment process that have changed and those that have remained the same over time, and explain how and why the evolution occurred.

[1] * Avaya Labs, randyh@avaya.com, audris@avaya.com, palframan@avaya.com
[2] ** Iowa State University, weiss@iastate.edu

15.1 Introduction

How does a company evaluate and improve its software development capabilities? This chapter describes an annual software assessment process developed and used by Avaya. Avaya is a large telecommunications company that started as a spin-off from Lucent Technologies. It has evolved today to be a provider of open mobile collaborative platforms. Software has always been central to its success, and its software development and sustainment capabilities have evolved with the company. Its software assessment process has been in place for 12 years and likewise continues to evolve.

Avaya's R&D organization, which numbers over 2000, is continually called upon to improve the quality of its software, to decrease its time-to-market, and to decrease the cost of development and maintenance of its software. Under these pressures it is critical to identify changes in development processes, environments, cultures, and tools that maximize improvement, that can be accomplished with existing resources, and that produce measurable results.

Avaya's assessment process was originated by and is carried out by the Avaya Resource Center for Software Technology (ARC), part of Avaya Research, a separate organization within Avaya R&D. A primary goal of the ARC is to improve the state of software in Avaya and to know it. "Knowing it" means that improvement should be subjectively evident and objectively quantifiable. Every year the ARC produces an annual report known as the State of Software in Avaya. The report describes software development trends throughout the company and contains prioritized recommendations for improvement. Priorities are assigned to the recommendations based on their expected impact and on the estimated capability of the software development organizations to implement them. Accordingly, part of the report is devoted to showing year over year changes in Avaya's development capabilities, with attention paid to the impact of previous recommendations. The report provides a feedback mechanism to help direct the evolution of Avaya's software development and sustainment capabilities so that the company may meet its goals. As the company's software capabilities have evolved, the report has evolved as well. The ARC is now at a point where it can look back and trace the evolution of the report, and assess its impact since its inception. Its methods may be a model for others to use.

In this chapter we use examples taken from the annual reports to illustrate the methods used in and the lessons learned from them. We show why and how the scope of the report and the methods used evolved over time, how the report became a basis for software improvement in the company, what the impact of the report was and how we estimate that impact, both financially and subjectively. We also provide some suggestions for how others may initiate a corresponding effort. Section 15.2 describes the evolution of the approach used to create the report and section 15.3 summarizes its impact. Section 15.4 provides more detail on the approach used, what aspects remained constant over time and what aspects changed. Section 15.5 describes our data sources, how we identified them, and how we validated the data and assured its accuracy. Section 15.6 gives examples of the different types of analyses performed over time, and how they evolved, focused primarily on software qualities, and section 15.7 does the same with software practices within the company. Section 15.8 illustrates the types of recommendations provided in the report and how the recommendations are deployed. Section 15.9 provides examples of how we assess the impact of the report and its recommendations. Section 15.10 summarizes what the ARC has learned from producing the reports, how the report continues to evolve, how we expect it to evolve in the future, and the applicability to other organizations of the practices the ARC uses in crafting the report.

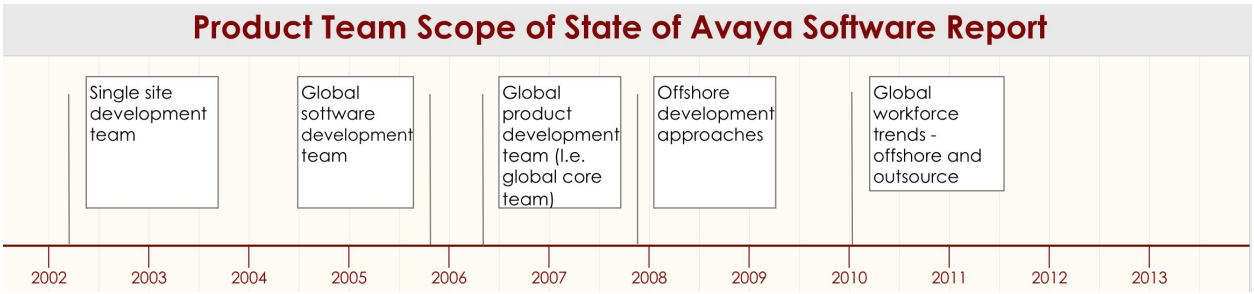
15.2 Evolution of the process and the assessment

Our primary project focus areas are derived from company goals, from the data available, and from what we think is feasible for the ARC to do. In 2002, most projects were single site projects since Avaya had limited offshore and outsourced development at that time and our initial focus was, therefore, on assessment of individual projects. Following is a summary of our focus areas in 2002.

- ***Project characteristics***: project descriptions, project goals, and number of releases approaching general availability (GA)
- ***Technologies in use in Avaya* projects:** the types of target and development platforms (e.g. VxWorks, LINUX), technologies (e.g. J2EE, .Net), protocols, development methodologies, and tools used across projects.
- ***People skills***: a snapshot of the domain expertise, roles, and experience of the Avaya R&D community, and changes in these areas
- ***Software quality***: the quality goals of development projects, and customer perception of quality.
- ***Project completion intervals (time it took to complete a project)***: an analysis of project intervals, including typical project intervals, and differences between forecast and actual intervals in Avaya projects.

As Avaya has evolved, the nature of project teams has changed. Many Avaya projects have become multi-site, moved offshore, and incorporate outsourced teams. As a result the scope of the report has also evolved to assess the performance of teams of the current nature (see Figure 15.1). This led us to analyze levels of experience, knowledge transfer techniques, multi-site project coordination, and communication mechanisms, among other distributed development factors.

Figure 15.1: Evolution of Product Team Scope in Avaya State of Software Assessments



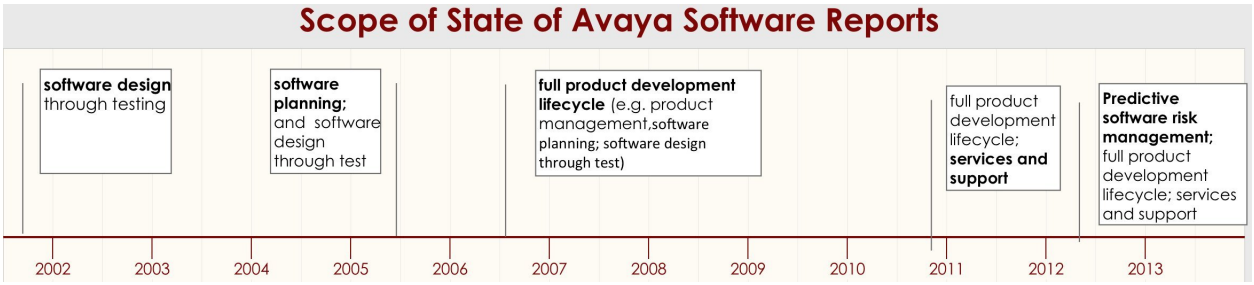
Our initial assessment scope was primarily R&D development activities, including architecture, design, implementation, and functional and system test. These activities remain a focus of each report, but Avaya's business has changed, and the company has evolved from the goal of being a leader in providing primarily voice-based enterprise-based telecommunications, to a goal of being the preferred provider of open mobile enterprise collaboration platforms.

Because of this business change, Avaya products are primarily software-based and need to interoperate effectively, which requires carefully coordinated cross-product planning, design and testing. As a result, we have expanded the scope of the report in two ways. First is to include an assessment of full lifecycle activities (see Figure 15.2). Second is to include cross-project development activities (see Figure 15.3). As shown in the figures, we now define the scope from both viewpoints, as follows.

Full life-cycle activities

- Front end planning activities, such as requirements specification, requirements review and project estimation.
- Full product development lifecycle, including product management and program management of functional teams (e.g. global market introduction, documentation, services, marketing, and sales).
- Software services and support functions, such as time to resolve customer service requests and assessment of the completeness and ease of use of product information available to service staff.

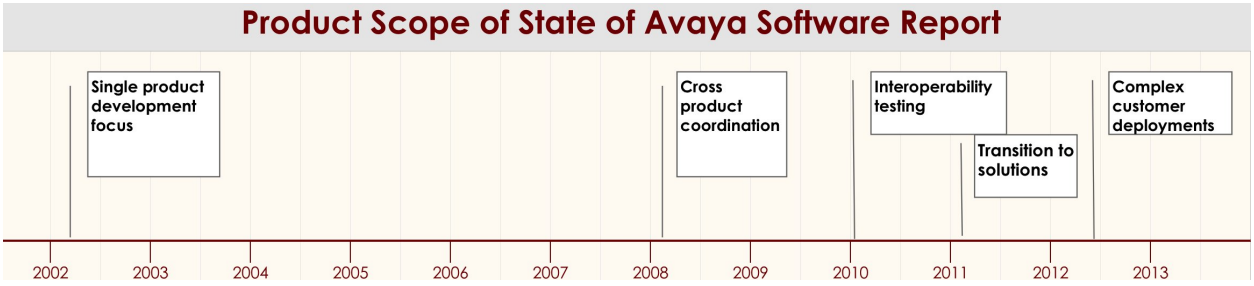
Figure 15.2: Evolution of the Scope of State of Avaya Software Reports



Cross-project development activities

- Cross-project interoperability management, including interoperability specification, commitments, and testing.
- The creation, testing, and deployments of multi-product solutions.
- The capability to predict and mitigate issues in complex customer environments.

Figure 15.3: Evolution of the Focus of State of Avaya Software Reports from Single Product to include Cross-Product Interoperability Management and Development and Deployment of Solutions



The most recent step in the evolution of the reports is to include also an assessment of software risk management practices. For example, the focus of the 2013 report (excerpted from the report) is as follows (See Figure 15.2).

> *Our emphasis in 2013 has been on software risk management practices focused on customer driven quality consistent with Avaya's mission to be the preferred provider of open mobile enterprise collaboration platforms.*

Note that as the assessments evolve, some activities are transient because of evolution of goals, and some are long-term, because they are associated with the company's long-term survival and success. We have taken our current focus, in partnership with Avaya leadership, in recognition of the transition of Avaya to

a software-based company and the resulting importance to the business of effective software risk management practices.

Each report also analyzes a special topic of interest for the year. For example, in 2008 we analyzed the deployment of architecture guided iterative development in Avaya. We identify the focus area in cooperation with the company leadership, leading to greater interest in and relevance of the report. Each focus area analysis is based on a study of some set of input data. The data are carefully cleaned and validated throughout the process. Table 15.1 shows the annual focus areas of State of Avaya Software reports from 2009 through 2013. About 50% of each report is devoted to the focus area.

Year	Focus	Requested by
2009	Quality in Avaya	Avaya Product Quality VP
2010	Improving Quality and Operational Efficiency	Avaya Research leadership
2011	State of Testing	Avaya CEO
2012	Critical Software Risk Management Approaches in Avaya	Avaya General Manager
2013	Software Risk Management: Customer Driven Quality	Avaya CTO

Table 15.1: Focus areas of State of Avaya Software Reports (2009-2013)

15.3 Impact Summary of the State of Avaya Software Report

Because of changes in technology, market conditions, ownership, and other factors, over the past 12 years, as the company has evolved, Avaya has become a more software focused company and the report has provided insight and guidance to R&D leaders and business leaders. New R&D leaders have typically provided feedback to Avaya Labs on the value of the report in providing a clear objective view of the strengths and areas for improvement in Avaya's software competencies.

The report is widely distributed to the entire R&D community, and the authors receive consistent feedback that it is read and appreciated by R&D staff.

> *"It [the report] really helped me hit the ground running."* – new corporate quality leader

> *"The report has helped me focus on the right areas for quality improvement."* – new R&D leader

We evaluate the impact of the assessments in the following ways.

- Usage and effectiveness of targeted areas, i.e. once a set of related software practices, such as build management, has been targeted for improvement, we monitor how widely used and how effective it is in later years. As an example, automated build management was widely used but not very effective in 2002, but by 2008 it had become very widely used and very effective (Section 15.9.1 discusses this example in more detail.)
- The extent of deployment of key software risk management practices, such as risky file management.
- Improvements in customers' view of Avaya quality as measured by the customer quality metric[3] [20].

Avaya leadership has had a strong focus on improving quality for the past three years, and our estimate of the financial benefit of the report and related quality-focused initiatives over this period of time is that

[3] The customer quality metric is a measure of customers' view of a product's quality based on customer found product defects normalized by the number of installations of the product [20]

operational costs per year have been reduced by at least $60M, which is many times more than the investment made to achieve the savings (Section 15.9).

15.4 Assessment Approach and Mechanisms

We do not attempt to characterize software development with a single number, such as with the CMMI approach [33]. Rather, we use the goal-question-metric approach [4, 5, 15]. We first establish the goals of software development in Avaya based on Avaya's business goals and on the goals established by the development organizations in order for the business goals to be achieved. As noted by Rifkin [2], different organizations adopt different product development styles depending on their approaches to their businesses. Companies that most highly prize innovation tend to have a different style than companies that most highly prize quality. We believe that it is a mistake to evaluate the different styles with the same set of measures. We use the goals to identify questions of interest and then define measures that we use to answer those questions. Section 15.6 includes some examples of questions and measures.

In addition, comparing our data and results with other enterprises is problematic because of the difficulty in assuring comparability among data. Validating our data internally is a difficult task, as we discuss in section 15.5.1 and as the reader may infer from our discussions of the types of data that we analyze in our examples in section 15.6. As an example, simply trying to determine how much code Avaya has in its code bases is complicated by a number of factors, including, but not limited to, the following.

1. What source code languages to consider. Avaya's products use C, C++, C#, Java, and many others, in very unequal parts.
2. Whether to include third party commercial or open source code or not, and how to identify it in code bases in either case.
3. How to identify duplicated code that occurs when a new configuration control system goes into use in a product and existing code is copied into it.

Comparing code size with other enterprises would require consistent answers to these questions from all enterprises involved in the comparison.

Furthermore, we do not have a way of validating and verifying non-Avaya enterprise data. Even assuming that we could be assured of comparability in the data, we would then have to take into consideration whether or not the enterprises had the same goals. For example, comparing product development time for a company whose goal is rapid time-to-market with a company whose goal is innovation or high quality may lead to misleading results and actions, particularly if one is not aware of the difference in goals. Put another way, we do not benchmark our software measures against other enterprises, either in an attempt to do better or to relax with the thought that we are better. Rather, we compare against our own goals and try to improve with respect to them.

In planning our approach, we have been particularly influenced by Walston and Felix's early work [27], by work on the Experience Factory by Basili and others [6], and by the foundational work on industrial software measurement by Grady and Caswell [13]. Like Quality Function Design (QFD)[24], we analyze quantitative data as well as seek out and prioritize spoken and unspoken customer (Avaya R&D) needs, using a list of questions. The product we generate, in the form of the State of Software in Avaya report (SOSA), contains a series of recommendations with deliverable actions that we expect will improve Avaya's software development practices. Unlike QFD, we do this on a global scale, and quality is only one of the aspects we seek to optimize in R&D.

Since Avaya produces a number of different products for different markets and market segments, the business goals of the individual product development organizations differ, although all strive to meet certain common goals. As an example, in recent years improvement in customer satisfaction ratings has

been a prominent goal. All divisions are striving to meet this goal. On the other hand, in certain markets, low cost and ease of installation and use are primary goals, whereas in others very high reliability is the dominant goal. Both common and individual goals color our assessments and the recommendations we make to different divisions. On presenting the report to a particular division we emphasize the results particular to that organization and make recommendations based on that emphasis. In doing so, we highlight their goals and present an analysis of their data, which is generally a subset of all the data we analyze for the year.

Data sources for the report include both quantitative and qualitative analysis of data. For example, in 2008 quantitative data came from sources such as distributions of defects found by customers, modifications made by software developers and described in modification reports (MRs), code in Avaya's software repositories, and demographic data. Qualitative data results from sources such as interviews with software developers, software managers, software product managers, and others[4], from impressions and data gained from participating in architecture and other reviews, and from specialized assessments of particular issues, such as how well Avaya development organizations are applying iterative development techniques. The Introduction to the 2008 State of Software in Avaya Report describes the sources of information used in producing the report. It includes the following description (slightly edited to preserve confidentiality).

> *The State of Software in Avaya report series is published by the Avaya Resource Center for Software Technology (ARC) to provide periodic snapshots of Avaya's software production capability. Our goal is to give Avaya R&D organizations a picture of where they need to focus resources to make improvements to achieve operational excellence. Our intent is to give the reader insight into how well Avaya is using its software production capacity, including what resources are available for software development, how effective those resources are, where they are located, and whether they are sufficient.*
>
> *The reports draw on learning from the services and analyses that the ARC provides to Avaya projects[5]. This year's report draws on the following sources.*
>
> - *Our learning from our work with the divisions on a software improvement initiative based on last year's report.*
> - *Our 2008 assessment of iterative development deployment in Avaya R&D*
> - *Our coordination of the 2007 Avaya Software Symposium and the 2008 Avaya Test Forum.*
> - *Our participation in a variety of software architecture reviews, and other services.*
> - *Our conduct of individual and small group input sessions with more than 120 members of the R&D community, and with program and product management from all divisions, distributed across Avaya's worldwide R&D locations. Our findings have been reviewed with and adapted based on feedback from these individuals.*
> - *Our quantitative analyses of data such as demographic data obtained from SAP and other sources, customer found defects, code in Avaya software repositories, project data, and MR data reported in the data warehouse and various configuration management systems used in Avaya.*

[4] In 2008 we interviewed 120 people from R&D and Product Management specifically for the report.
[5] A project is undertaken to develop one or more releases of a product. In some cases the development of a release is organized into multiple projects.

The Appendix contains a list of example questions used in the interviews to which the report Introduction refers.

15.4.1 Evolution of the Approach Over Time

The report initially gained credibility within R&D, but has now spread far outside R&D to the highest executives. This has affected how each year's theme is chosen and how follow-up is done. Twelve years ago the theme was chosen by Avaya Research, but for the last several years it has been chosen by corporate executives. The report evolves primarily based on the changing goals of the corporation.

Initially we conducted follow-up sessions on the report with R&D management. Now we also meet with personnel from product management, quality, and services, as well as with Avaya business leaders. A version of the report is tailored for each organization based on data that focuses on that organization and recommendations that are the most relevant for it. For each recommendation we now suggest a role responsible for implementing the recommendation. Because of resource constraints in development we focus on the top few recommendations appropriate for an organization in order to increase the probability that action will be taken by that organization.

When Avaya was formed there was a heavy emphasis on time to market. Since Avaya products can be used in mission critical situations corporate goals more recently place a heavier emphasis on product and solution quality [24]. There has also been increased emphasis on making R&D more efficient. This has affected what data are gathered and what analysis is performed.

The number of acquisitions has increased over time, bringing new issues, cultures, practices, market areas, and repositories. This has affected both the data gathered and the analyses performed.

Avaya products are increasingly interdependent, and the company is transitioning from a product management model to a "solution" development model. Along with this comes new data, such as interoperability matrices, that we use in our analysis.
The next two sections contrast what has changed with what has remained the same.

15.4.1.1 What Has Changed?

The mechanisms have evolved over time to accommodate the change in approach as well as the change in data available, as follows.

- More data sources are available, allowing more in-depth quantitative analysis. See section 15.5 for details.
- The quality of the data collection practices has improved over time.
- Data are easier to access. Many of the software projects moved to Avaya Forge or started using central resources for issue tracking. See section 15.5 for details on Avaya Forge.
- We regularly conduct interviews outside of R&D, including field services personnel, corporate quality personnel, managers in outsourced organizations doing Avaya product development, and Avaya executives. The number of interviews conducted has generally increased over time. Interview questions associated with the focus for a given year are specific to that year.
- Originally we conducted web based surveys to get an overall view of R&D concerns and practices. Because we have built up significant relationships with the development community and because the response rate had become too low to be significant we discontinued surveys.

- We no longer track software development practice trends, so we do not gather data in this area. Instead we do an in-depth analysis of selected practices, such as testing or use of static analysis, and make recommendations focused on them.
- In our first year we tried doing the report semi-annually. Because of the amount of work involved we quickly moved to an annual report. Since 2003 we have published the report for the preceding year during the first part of January.

15.4.1.2 What Has Remained The Same?

We still rely on both quantitative and qualitative data. Many of the types of underlying data sources we use remain the same, even if the specifics have changed, e.g. code repositories are accessed, though the technologies have changed. We still rely on partnerships with Avaya product teams, assessment of good software practice deployments in Avaya and industry, and data from internal software conferences.

We continue to do interviews to gather data as well as draw on engagements with projects and business units that occur throughout the year. Tailored questions for each interview are established in advance to give the interviewee time to think about the interview, though we emphasize that no preparation is required, given the busy schedules of the interviewees. In many cases the interview goes off in an unexpected direction, providing us more insight into development concerns. In this case the concern may be factored into questions for subsequent interviews, and we typically add interviewees to our list to make sure the concern is properly understood. We document each interview for later analysis. As a result, we now maintain a large repository of interviews covering a 12 year period, which is very valuable when analyzing multi-year trends.

We review a draft of the report, including recommendations, with those from whom we obtained data. Such reviews are part of validating the report prior to completion and distribution.

We continue to meet with management after the report is published to get their view on recommendations and we track to see what recommendations get implemented.

15.5 Data Sources

We access many types of data from a variety of sources, including the following.

- Code Repositories (Lines of Code, Commit info, branching info, etc.)
- Defect Tracking Systems (defects by stage, field escalations, service requests, etc.)
- Demographics (distribution, experience, churn, distribution by roles, etc.)
- Development data (code/document/design review information, metrics on code coverage, static analysis, performance, longevity, reliability, build metrics, etc.)
- Document Repositories (requirements, designs, project plans, test plans, etc.)
- Project WIKIs (Processes and practices, project status)
- Quality data (Interoperability, in-process metrics, customer quality, customer satisfaction, quality improvement plans, etc.)
- Use Cases
- Sales information (distribution of products by release, product and solution configurations, upgrades)

- Services information (escalations, customer found defects, trends, etc.)

There were several reasons for major changes in data sources as our assessment process evolved over time.

- First, the trend to an increasingly centralized (cloud-based) administration of software development support tools such as version control, problem tracking, and related systems continued to accelerate. A large fraction of projects moved to Avaya Forge and to a single instance of ClearQuest (an issue tracker). Avaya Forge is a corporate-source cloud-based tool similar to SourceForge that provides a suite of tools integrated by Atlassian, such as JIRA, Crucible, FindBugs, Subversion, Git, and Confluence. This trend has concentrated data from many of the hundreds of projects into a single location, making it simpler to access and use. In addition, this centralization has unified the identification of individuals. The typical projects analyzed in early state of software reports had their development support and issue tracking systems administered by individual projects and the same person often had multiple IDs associated with each system.

- Second, major new acquisitions brought in an entirely different set of systems and practices. To avoid fragmentation of developer support tools, the issue tracking systems for the projects in this acquisition were migrated to a single (ClearQuest or JIRA) platform.
- Third, projects continued to move to more advanced tools. In the early period projects were moving from Sablime to ClearCase for version control. Later, Subversion became the standard tool with many projects moving from ClearCase to Subversion. Over the last few years another large migration to Git VCS has started. The employee directory system has changed to a new platform, but continued to provide similar types of data and continued to serve as a means to capture statistics for the entire enterprise.
  One of the biggest changes in the use of advanced tools was a move to Siebel of the customer relationship management (CRM) system that was used to track and resolve customer issues.
- Fourth, as the business value of central data collection became more obvious, a data warehouse with information related to sales, field support, licensing, and other types of data was established. In addition, more aspects of software development were supported by tools providing additional data sources.

On the positive side, the collection of data became easier with the centralized tools, identifying individuals became simpler, and a wider set of tools, e.g., Atlassian Crucible for inspection, JIRA, and others were introduced to provide additional opportunities to understand and improve software development. However, these migrations have substantially complicated historic analysis, because not all the past data were migrated, newer tools had different types of attributes, and the ways in which the tools were used changed substantially. Many of the no-longer-used tools were decommissioned, and the associated historic data were lost. This has validated our approach to store clones of the systems such as Sablime, Subversion, Git, or ClearQuest or to store snapshots for tools that do not keep track of state changes, for example, information from code coverage tools.

Because of the number and variety of Avaya products, the difficulty of accurately combining the data from the actively used and no longer used tools, and the transfer of reporting responsibilities to the business units, we no longer provide context data for the code size, the productivity of developers, and the quality for all Avaya projects. Instead, we have integrated a variety of measures and tools into a toolset for identifying and reducing risk in software projects. The measures we use allow answers to a number of specific practical questions, as described in section 15.6.

Development still uses a variety of defect management tools (ClearQuest, JIRA, Rally) and Source Code Management (ClearCase, SVN, GIT, Sablime). With the increasing scope of the data warehouse there has been standardization of the semantics of the fields used by these tools. Also, as quality councils were established, the need for common reporting forced the standardization of semantics.

The rest of this section lists some of the new sources of data, some of the data sources that changed, and a few examples of the data and our analyses. Over time, many new sources of data became available, such as the following examples.

- Customer driven requirements and prioritization information are stored in a single repository covering all products.
- Coding data such as static analysis repositories, code coverage data, automated test coverage, code inspection data, build frequency and breakage, interoperability, and technical debt are maintained for most Avaya products.
- Project management data as provided by Agile management systems like Rally or Green-hopper are available to gather data on project backlog trends, velocity, and quality.
- A comprehensive program management website was established that makes available planned and actual release data as well as other project-related data for each Avaya program.
- System test and developer test data such as test plans, test coverage, test pass and fail rates, and test efficiency are stored in a common repository for a large set of products.
- Open source data in the form of a repository of open source code described in [31] (that now includes over 200 million unique versions) and a Black Duck repository identifying open source use by projects have been created. As a result, we can filter out open source code from code growth trends.
- Product quality trends based on the "Customer Quality Metric" [20] are tracked and used to report on product field quality
- Quality data are maintained based on policies that Avaya developed for minor releases and patches, namely feature packs and service packs.
- Good software practices are tracked by Avaya's R&D quality council. This data is used to encourage improvement in software practices.
- Avaya now uses a Siebel customer relationships management tool that provides information on entitlements and service reports.
- SalesForce, which contains information about customers and their account managers, is used across the company.
- Billing, licensing, and download information is used to estimate the number of users of Avaya's software-based systems and the extent of usage. Downloads may be associated with licenses but also obtained from third parties, such as mobile application stores.

It is worth noting some of the difficulties encountered as projects migrated to new systems. For example, to determine which MRs are related to customer issues, distinct systems have to be linked: a CRM system Siebel, and an issue tracking system for development, such as JIRA or ClearQuest. Because CRM is used by a variety of the service personnel in almost all countries of the world, and MR's are tracked by different software development projects, it takes time to propagate uniform practices and definitions to this very large and diverse population. Active efforts from business units and quality organizations has, over time, led to more uniform and more accurate data entry and resulted in better quality data.[6]

Surprisingly, almost all data sources have changed over the considered period because of evolution of technology, because of the move to cloud-based systems, or other reasons listed above. Only one large project continues to use a custom set of tools built in the late 1980s. This toolset integrates many of the

[6] As usually happens, better quality data leads to more precise, more insightful, quicker analysis, which leads to better recommendations for improvements, which leads to faster improvement.

development process stages including inspections, testing, build, change control, and issue tracking. It is interesting to observe that only recently have the integrated suites, such as the one provided by Atlassian, started to approximate the functionality of the legacy tools mentioned earlier.

The main message from our experience of collecting data, however, is that over longer periods of time it is reasonable to expect to see migrations to different development support tools. This suggests that one has to be ready to adapt to these long-term changes to stay relevant, and that studies done over the long term may require special techniques to adjust for the tool migration and the ways it affects collected data and analyses. For example, while some data were migrated in the projects we have been investigating, the decommissioned systems were not retained. Second, the data migrated from earlier systems were typically different from new data collected in the course of use of a new system. Third, the practices associated with using the new systems often are substantially different from practices previously employed. These three differences make it difficult to conduct historic analysis that crosses the migration boundary.

15.5.1 Data Accuracy

In addition to new data migration challenges noted previously, assuring and estimating the accuracy of our data is a major concern. We deal with large data sets that are distributed across a variety of repositories, and for which some of the data are entered manually, often by people far removed from the analysis of the data. For example, MR data are stored in different configuration control repositories and the descriptions of the changes in the MRs are entered by software developers or by support teams that may not be aware that their entries become part of a company-wide analysis. Human error or even just vagueness in such descriptions can be a major source of inaccuracy, as discussed and shown in [5]. The problem is compounded when performing analyses that include several different sources or types of data. We believe it is incumbent on us to estimate the error in the analyses that we do, and we continue to seek good methods for doing so. This is a research topic that the measurement community should address. In the following sections, where we present some examples of our analyses, we will provide error estimates where we can reasonably make them.

15.5.2 Types of Data Analyzed

As previously noted, our quantitative data sources tend to focus on the process by which the requirements are obtained and tracked, the process by which software is developed, the demographics surrounding development, and the quality as seen by the developers and by the customers. Each area has a rich set of tools that can be used to quantify the history of events in the area and how they relate to, for example, customer-perceived quality downstream. Code commits and MRs are used to extract information about a variety of issues, including types of errors that occur and practices that are used. For example, by examining when an error is found we can usually determine the detection technique used, such as a code review or a system test. It is difficult to spot corporate-wide trends by looking at code measures since languages, platforms, and development tools vary across the company; we leave such analysis to the individual projects. However, as projects have deployed various static analysis tools, such as FindBugs and Coverity, and inspection tools such as Atlassian's Crucible, richer sources of data have become available.

15.6 Examples of Analyses

The initial focus of our reports was analysis of demographic trends, domain expertise, code trends, software quality trends, productivity, predictability, transfer of work, and software practice trends. These trends continue to be important to Avaya, but the nature of our analyses has changed.

- We no longer perform annual analyses of some trends because Avaya product groups now track them on a regular basis as part of business operations. Two examples are regular tracking of the customer quality metric [20], which represents the probability that a customer will observe a failure within a certain interval after software release, and quarterly analyses of schedule performance, with a comparison of predicted schedule to actual schedule (Section 15.6.2). In both cases mitigation steps are identified by the business operations team and taken as needed.
- As Avaya development has become a global endeavor, our demographic analyses have increased their focus on multi-project, multi-location, off shoring and outsourcing trends.
- We have discovered that it is even more important to help projects implement recommendations than to provide recommendations. In particular, we found that providing detailed recommendations with procedures and tools to help implement them was embraced the most readily and widely. Therefore we have started to provide integrated analyses of multiple data sources and tools and procedures to aid in the action steps for individual projects. For example, risky file management (section15.6.3) relates several sources of data, such as defect counts, file churn, author churn, and file size to identify the potentially most risky files in the code base. In addition to the risky file analysis, we provide procedures and tools to assist a product team in mitigating the riskiest files as described in section 15.6.3

Our demographic analyses of the R&D community (section 15.6.1) remain highly anticipated. For example, they help R&D and business leaders examine staffing, expertise, offshoring, outsourcing and other R&D demographic trends. These trends are typically available for individual projects, but the report provides the trends at an organization and Avaya-wide basis and helps R&D and business leaders assess the capacity of their organization, its training needs, its ability to deliver products on time and with quality and similar factors.

Our analyses of other trends, such as domain expertise, code trends, software quality trends, transfer of work and productivity trends were performed independently in earlier reports [24]. In some cases, we compared two factors whose relationship we thought might be significant, such as product team productivity and lines of code in the product as shown in Figure 15.4. To measure developer productivity we followed the approach introduced in [22]. We first select a subset of developers who together contributed 80% of the changes each year. We refer to them as the core group. The number of changes to the source code by this core group is then divided by the group's size to get the productivity measured by number of changes per developer per year. We consider only changes made to the source code files. Lines of code are calculated based on the contents of the project’s code repository.

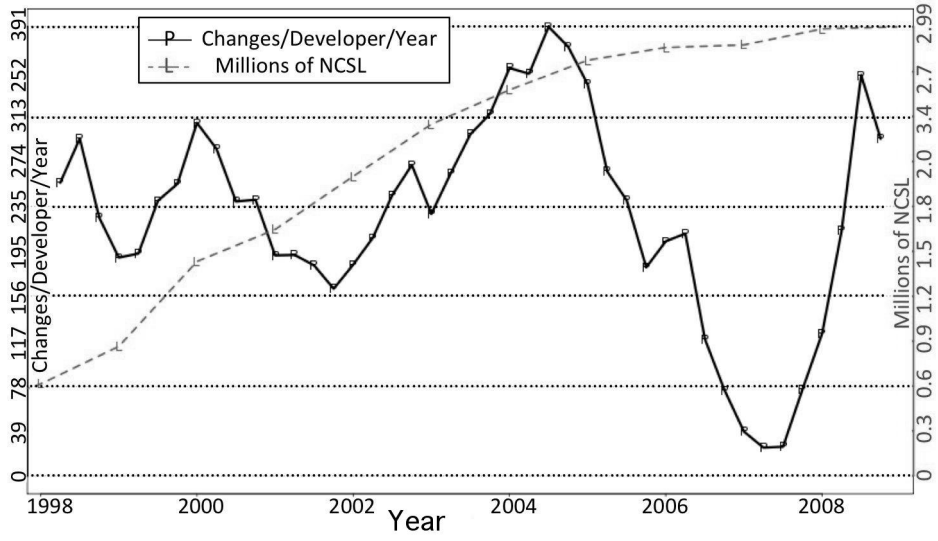


Figure 15.4: Productivity Trend and Lines of Code Trend for a Sample Product

As previously noted, part of the validation process is showing the analysis results to people in a variety of roles in R&D and other organizations and noting their objections and confirmations. We then make a final pass through the data and analyses, making adjustments as necessary. The analyses that follow are each intended to answer some specific question(s), which are noted at the beginning of the section.

15.6.1 Demographic Analyses

Questions: How is staff distributed across locations and divisions?
How much staff churn is there?

The reports look at the distribution of the current R&D population by location, titles, experience, division, company of origin, and by type of staff (employee, contractor, outsource, or offshore). Understanding the distribution and composition of the R&D organizations helps us to understand problems R&D is facing and to spot trends. For example, Figure 15.5 shows that many divisions in Avaya were spread across many locations, prompting us to study issues that projects have with distributed development. Figure 15.6 shows a major buildup in the use of contractors and in outsourcing, and so we are drawn to address issues such as knowledge transfer and how work is partitioned.

Both Figures are based on an analysis of Avaya's corporate personnel directory, which includes information on the role and location of every employee. In addition, some data on outsourced teams is obtained by interviewing Avaya staff that is tasked with managing outsourced teams.

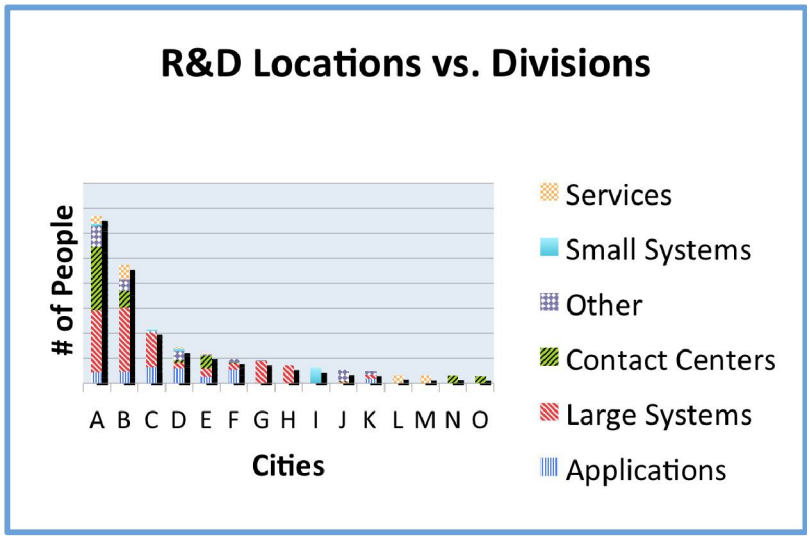


Figure 15.5: Distribution of Divisions Across Locations

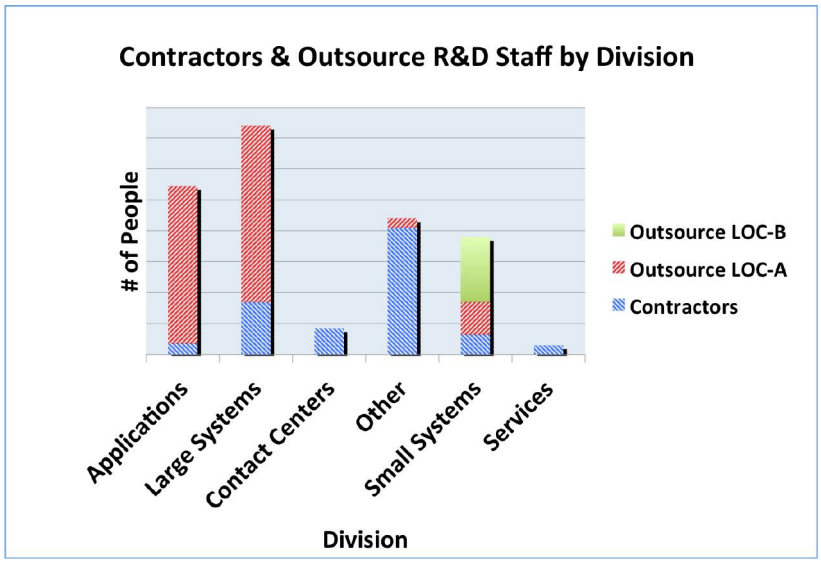


Figure 15.6: Outsourcing by Division

Analyzing the origins of the company's current employees provides insight into products, technologies, and markets with which they are familiar. This has impact on software development methodologies and on the locations of experts in various disciplines. A more detailed view (not shown here) reveals for each year hired how many employees remain and how many of the original set have left the company. This can be used to examine retention policies and morale issues[7].

The demographic trends can be combined with other trends, such as code growth, to show where support may need to be rebalanced.

[7] This may only be possible because Avaya is a relatively new company and we can access records back to its origin in 2000.

Another analysis displays staff churn trends, i.e., the turnover rate at different locations, which allows us to estimate the rate of loss (or gain) of domain expertise. Because real-time telecommunications products are quite complex, it can take 6 to12 months or more before a developer new to the product or company can be effective. In more complex projects we observed that it can take 3 years for a developer to become effective [32]. The churn trend can be used in analyzing productivity losses caused by staff loss or by training rates for new staff. Figure 15.7 shows long term trends in Avaya product experience, and, along with more detailed charts, such as Figure 15.8, it is used to recommend what an appropriate balance of new to experienced staff should be based on historical data. The acquisitions demonstrate where infusion of developers new to Avaya occurs. Figure 15.8 shows a comparison of the geographic distribution for two specific points in time. Variations of this chart for individual organizations or functions highlight where the experience is out of balance and corrective action is needed. We do not have experience history from all acquired companies so the experience in the worst case could be understated by as much as 11%, but is more likely under 5%.

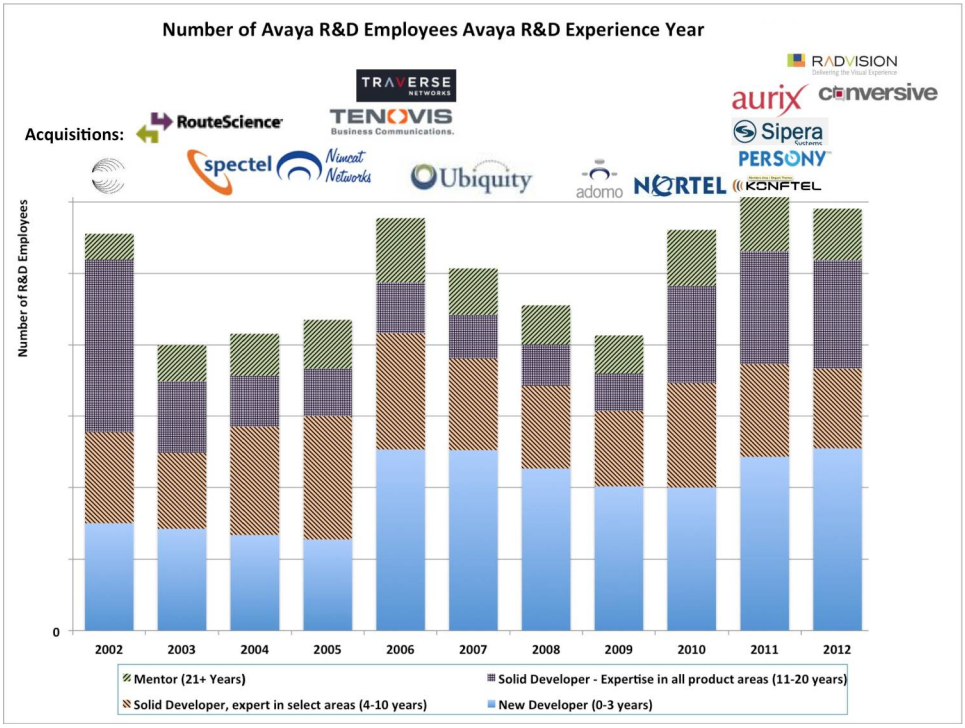


Figure 15.7: Long Term Avaya Product Experience Trends

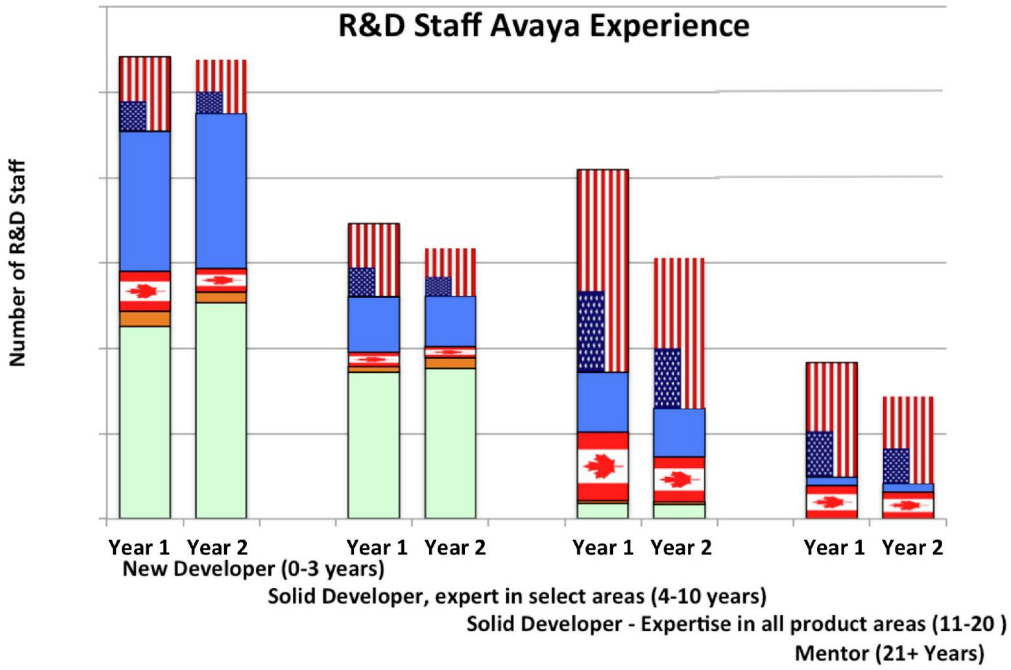


Figure 15.8: R&D Product Experience Comparisons by Location

Our demographic analyses are based on our corporate personnel directory. We create a snapshot of the personnel directory every month and use this data to analyze trends.

In summary, the demographic analyses provide a background and context that assist us in understanding other significant trends in the company

15.6.2 Analysis of Predictability

Question: How well do Avaya projects predict their completion dates?

Avaya employs a business process with gated reviews for all projects to move projects forward and to synchronize such functions as development, training, documentation, services, installation, and customization. Depending on project characteristics, Avaya uses a variety of processes within the development phase, including iterative and traditional waterfall processes. Predictability of development is important to make sure all functional areas allocate resources appropriately, including areas such as design, system test, interoperability test, documentation, globalization, localization, services for alpha trials, beta trials, general introduction, and support. Data are provided to and tracked by per-product Product Management Teams.

We are able to extract planned and actual dates from a number of sources and compare them for consistency. Figure 15.9 shows how well projects predict their completion dates at different gates in the development cycle. The Y-axis shows how late or early a project completion estimate is relative to its actual completion. The horizontal line indicates on-time completion. The distribution of estimates at different gates in the development cycle is shown as shaded ovals. For example, at the OK-to-Plan gate, which marks the end of the feasibility phase, the shaded oval shows the distribution of estimates. The median estimate is indicated by a small black horizontal oval within the larger oval. One can see that

very few projects fulfill their estimates. Furthermore, even when development has advanced to a later gate, there is only a small improvement.

Figure 15.9 was created by analyzing the predicted and actual project business gate dates for all Avaya projects over an 18 month period in 2004 and 2005. Schedule prediction is now closely tracked by Avaya's operations team and schedule performance has substantially improved

Figure 15.10 shows a relation between three factors and projects' ability to predict their schedule. For example, one factor is the degree of cross-project and cross-division cooperation required. We found that projects involving cooperation among divisions were worse at predicting completion dates than projects that were located within a single division but had development staff at several different sites, which were worse than projects that were located at a single site. Without quantitative data an experienced project manager might guess the same result, but would not know the magnitude of the difference. We also look at predictability by phase and predictability by variables such as project complexity or size, number of sites, or development methodology. We track the predictability trend to ensure that practices are being put in place to improve predictability. At the time the charts in Figure 15.9 and Figure 15.10 were generated we observed that Avaya was not good at prediction and that Avaya's ability did not improve as the development cycle progressed. Focusing attention on this issue through quantitative analysis led to the use of better prediction techniques and close tracking by Avaya's operations teams. Subsequent analysis showed improvements. This is a good example of an improvement resulting from (quantitatively) highlighting a problem.

Figure 15.10 shows factors that appear to be correlated with predictability. It was created by analyzing the predicted and actual project business gate dates for all Avaya projects over an 18 month period in 2004 and 2005. We identified project size and organizational complexity from project staffing profiles, and determined the product complexity from requirements and design documentation.

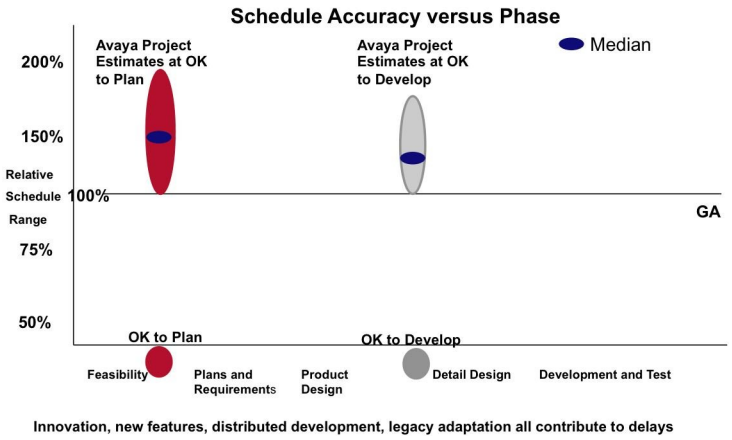


Figure 15.9: Schedule Predictions

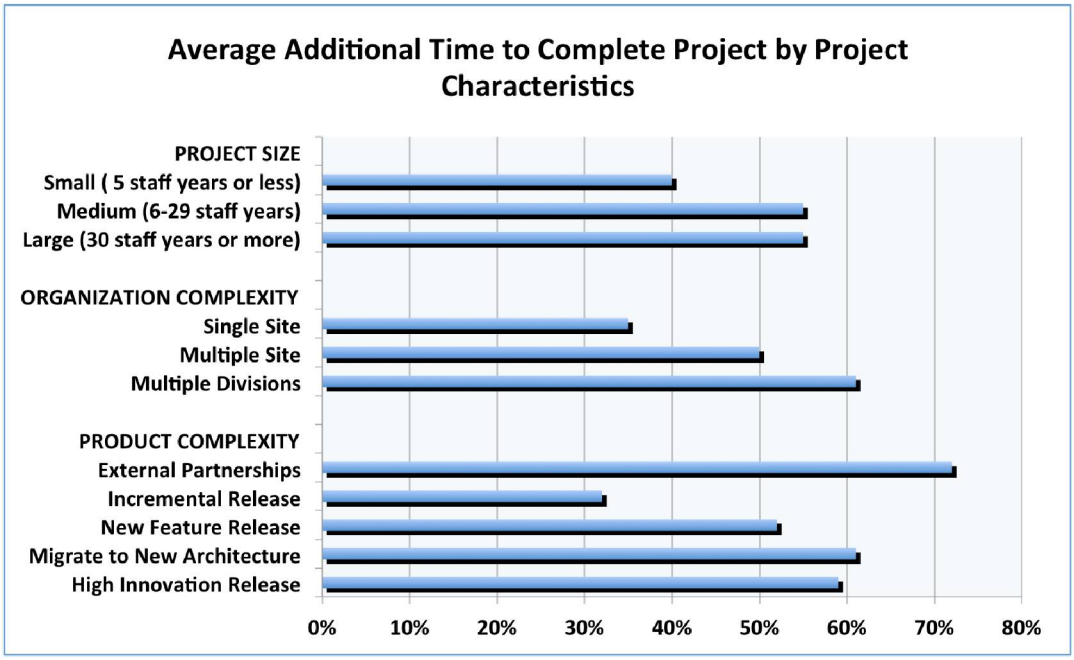


Figure 15.10: Some Factors Correlated With Predictability (circa 2005)

Our primary data sources for predictability analyses are the committed and actual dates that are tracked in Avaya's program management repository. These data are updated and tracked on a monthly basis and we estimate the data is about 95% accurate because it is carefully reviewed by Avaya's program managers.

15.6.3 Risky File Management

We observed, based on empirical analysis, that 1% of project source code files contain changes for more than 60% of the customer reported defects in most Avaya products. The ability to identify these "1%" files helps prioritize risk mitigation activities.

Risky File Management (RFM) [23] helps to prioritize and determine the most appropriate risk mitigation actions. We have refined this approach while working with over 50 Avaya projects. Briefly, it involves annotating the source code at the file and module level with the historic information needed to identify risk, and proposes the most suitable actions to mitigate the particular types of risk. It is somewhat similar to approaches used to produce highly reliable software for critical situations, such as described in [30]. However, in contrast to work described in [30], the purpose of risky file management is to prioritize risk remediation by selecting risky areas and the remediation approaches that are likely to be most cost-effective. For example, to advocate static analysis techniques described in [30], we tried to find (but did not) instances of a defect flagged by static analysis techniques that caused a failure reported by a customer. In contrast, several projects reported a non-negligible fraction of fixes to static analysis warnings that introduced new defects. As a result, we prioritize fixes only for certain classes of static analysis defects and only for selected areas of the code.

The data we use comes from version control and MR tracking systems. Software projects use a version control system (VCS) to keep track of versions of the source code. Each revision of a source code file a developer makes is "committed" to the VCS. Information about the commit that can be retrieved includes

author, date, affected file and its content, and a commit message that typically contains related Modification Request IDs. Examples of VCS include Subversion, ClearCase, Git, Mercurial, CVS, Bazaar, and SCCS. Developers create branches of a project source code to work on a particular release. VCS typically support creation of a branch so that changes made to one branch can be tracked separately from changes made to another branch. Most projects have at least two branches: a development branch with the latest features, and the release branch (for code that is in the product released to customers), which is unchanged except for important bug fixes. MR/Issue tracking systems are used to track the resolution of MRs (each commit is typically associated with an MR). The MR can be thought of as a software development task. The task may be to implement a new feature or to fix a defect. Among attributes associated with an MR is one that can be used to determine if the MR was customer-reported, so we can tell which defects were discovered by customers. Common issue tracking systems include Bugzilla, JIRA, ClearQuest, and Sablime.

The following list the main steps of the RFM approach.

1. Data collection and analysis

We gather data about each version of every file in a set of source code repositories for the majority of Avaya projects, and then prioritize the files and identify a candidate set of riskiest files (about 1% of all files) using a weighted algorithm based on the empirical results. While we create the risk profiles based on all files in all Avaya source code repositories, each project is presented with the most risky files that are in repositories related to that project. We obtain the following information for each revision of every file in each source code repository:

1.1. Path name, which uniquely identifies the file in the repository. Pathnames will be different for each branch in some of the version control systems, for example in SVN.

1.2. Author, date, commit message, and content. We process the content of the file to obtain an Abstract syntax tree (AST), and size (in lines of code). We also process the commit message to identify MR identifiers, if present.

1.3. To determine equivalence classes for a file, all versions of all files obtained in step 1.2 are examined as follows:

1.3.1. If some version v1 of file f1 matches some version v2 of file f2 (matches means that they have identical content or that they have an identical abstract syntax tree (AST)) the files f1 and f2 are considered to be "related". Typically the same file may be modified in multiple branches or even different repositories (when the same code is used in multiple projects).

1.3.2. The "related" relationship is transitively closed, i.e., if f1 is related to f2 and f2 is related to f3, then we declare that f1 is related to f3 even if f1 and f3 may not have a single version with the same content or AST.

1.3.3. Open Source Software (OSS) files are identified by matching each version of each file for each equivalence class of the related files identified in step 1.3.2 to the large repository of open source code with more than 200M unique versions mentioned earlier. If a match is found, then the equivalence class is considered to be an OSS file.

1.4. The corporate personnel directory is accessed to determine for each author obtained in step 1.2 if the author is an active employee. If the author is an active employee the employee's name, phone number and email address are obtained. Any other information available in the corporate directory could also be obtained.

1.5. We identify Customer Found Defects (CFDs) and other MR attributes by matching MR identifiers obtained in step 1.2 to data in the MR tracking system.

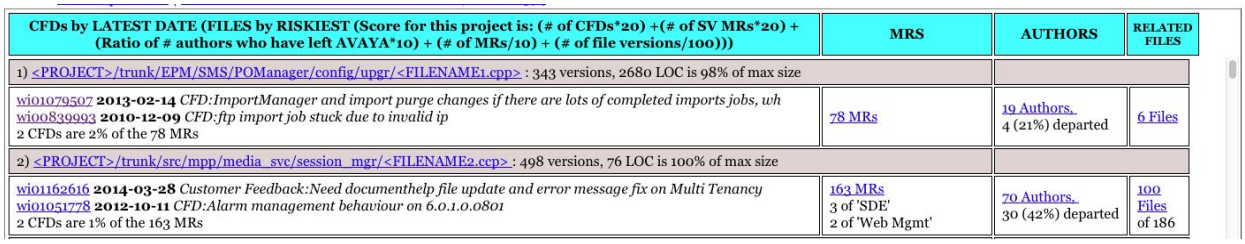
1.6. For each equivalence class of related files, aggregate the data over all related files to obtain the number of commits, number of authors, number of authors who left Avaya, and the number of CFDs.

1.7. Obtain an empirical relationship between properties of the file and CFDs using statistical models using all commits to the project's version control system for a period of time (typically a three year period) and fitting a logistic regression model with an observation representing an equivalence class of related files, with the response being whether or not any of the files in the equivalence class had a CFD, and the predictors including those described in step 1.6. For example, in several projects the most important predictors of future CFDs are the number of past changes, the number of SV MRs, and the number of authors who left Avaya.

The fitted model coefficients are then used to prioritize the list of candidate riskiest files.

2. **Presentation of the analysis results to relevant stakeholders.**

The approach provides a subject matter expert exploration view, an online dynamic table, and a downloadable spreadsheet. This is the critical and important part of the approach as it provides information tailored to the decision-making needs of different stakeholders. Figure 15.11 shows an example of the subject matter expert exploratory view. The candidate risky files and two most recent CFDs are identified in the first column. The second column contains the number of MRs requiring changes to the file or a related file and a hyperlink to a list and description of each MR. The third column contains the number of authors who have changed the file or a related file and a hyperlink to the list of authors. The third column also contains the number and percentage of authors who are no longer in the company. The fourth column contains the number of related files and a hyperlink to the list of related files



CFDs by LATEST DATE (FILES by RISKIEST (Score for this project is: (# of CFDs*20) +(# of SV MRs*20) + (Ratio of # authors who have left AVAYA*10) + (# of MRs/10) + (# of file versions/100)))	MRS	AUTHORS	RELATED FILES
1) <PROJECT>/trunk/EPM/SMS/POManager/config/upgr/<FILENAME1.cpp> : 343 versions, 2680 LOC is 98% of max size			
wi01079507 **2013-02-14** *CFD:ImportManager and import purge changes if there are lots of completed imports jobs, wh* 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 (21%) departed	6 Files
2) <PROJECT>/trunk/src/mpp/media_svc/session_mgr/<FILENAME2.ccp> : 498 versions, 76 LOC is 100% of max size			
wi01162616 **2014-03-28** *Customer Feedback:Need documenthelp file update and error message fix on Multi Tenancy* wi01051778 **2012-10-11** *CFD:Alarm management behaviour on 6.0.1.0.0801* 2 CFDs are 1% of the 163 MRs	163 MRs 3 of 'SDE' 2 of 'Web Mgmt'	70 Authors, 30 (42%) departed	100 Files of 186

Figure 15.11: Extract of Exploratory View provided by Risky File Management

Results from different tables or different portions of the table can be tailored to the needs, interests, and skills of a particular stakeholder. The following are a few examples.

- A project manager sees the priority of the risk and an estimate of resources or time to remediate the risk.
- A subject matter expert sees more technical details of what the type of risk is, what underlying technical details led to the file or module being classified as risky, and so forth.
- A development manager sees who made changes to the file and can use that information to identify potential owners of the code or to identify reviewers for design or code reviews.

Table 15.2 shows key data needed by subject matter experts to reach the most appropriate risk remediation action.

Table 15.2: Information Presented to Subject Matter Experts

TYPE OF DATA	DESCRIPTION OF DATA

TYPE OF DATA	DESCRIPTION OF DATA
List of CFDs	A link to the CFD, the date, and an abstract to aid the subject matter expert in understanding the defect to which the file contributed.
List of Related Files	The name of each related file, last commit date, first commit date, number of commit, and last author to make the commit. The list is sorted by the date of last commit.
List of File authors	The name, email address, phone number, number of deltas made by the author, and total number of deltas made by the author to all related files. In addition the first and last date that the author made commits can be provided. The list is sorted by relative contribution (number of deltas) made by the author.
List of all MRs	A link to the MR, the date of the MR, an MR abstract, and an indication of whether the MR is a CFD is provided for each MR against the file. The list is sorted by most recent date.
Lines of Code	The size of the file in lines of code (LOC) is provided as well as the percentage of the current size of the file compared to its maximum size at any point in the past[8].

3. **Remediation Actions.** A checklist of heuristics based on experience and empirical data to help the expert take the most appropriate action: e.g., no action, control program, or reengineering. For each candidate or indicated risky file, the subject matter expert can analyze the file or module and any associated data. The system can provide an optional guideline, based on empirical data of previous actions taken to remediate risks.
   3.1. No action may be required, if, for example, development is complete for this file; the candidate file will not be used in the near future; the candidate file is changed with a risky file, but is not itself risky.
   3.2. The subject matter expert can recommend a control program involving additional review and testing of all changes to the file, or creating additional documentation to make clearer the design and implementation issues considered in producing the file. Such a control program mitigates risk from changes to the file. For example, if the file has many authors or other reasons warrant, the file owner can create a 1-page design guidance document that is available for anyone who changes the file. The same design guidance document may also apply to a set of files that all contribute to the same component or feature.
   3.3. Finally, the subject matter expert can recommend that the file be reengineered if development is active and the file is deemed to be fragile and difficult to change without introducing more risk.

   Typically, the subject matter expert works with the development manager and project manager to schedule any recommended changes to the file. Figure 15.12 shows the distribution of actions taken in response to identification of risky files by an example project.

[8] The calculation comparing current size of a file with its maximum size is important, because if the percentage is significantly less than 100%, it is likely that the file has been refactored at some point.

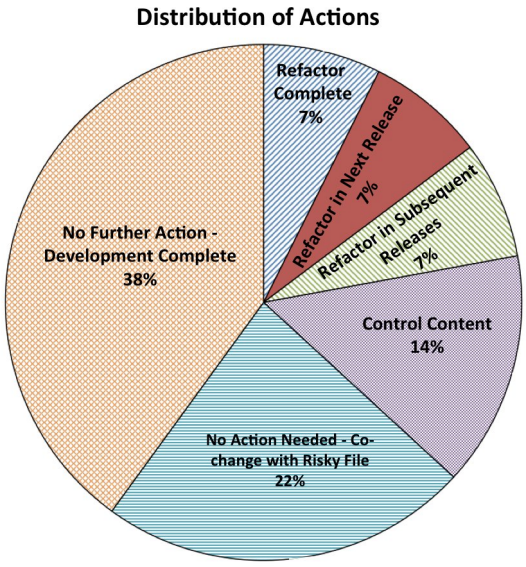


Figure 15.12: Distribution of Risky File Actions for an Example Project

15.7 Software Practices

Question: What software development practices are most often and most effectively used?

We originally assessed the extent of usage and the effectiveness of seven key software practice areas and individual practices that make up these areas (Section 15.7.1). These were based on the practices needed to meet stated Avaya goals, which changed over time, and our observation of industry practices. We have generated in-depth reports on the use of selected practices across most Avaya R&D projects, for example Metrics, Agile Practices, Estimation, Continuous Integration, Software Builds, Combinatorial Array Testing, Dynamic Analysis, and Risky File Management. These analyses and others have been used to share best practices across the R&D community.

Currently the R&D quality council tracks four practices as representative of the larger set of development and test practices (Section 15.7.2). These, along with tracking of schedule adherence and product/solution quality, allow Avaya business leaders to gauge the health of a project's development processes without getting lost in the details.

Section 15.7.3 summarizes our assessment of an example practice area (Design Quality In) that focuses on 11 individual practices. Section 15.7.4 summarizes our assessment of static analysis, a key individual practice in Avaya R&D.

15.7.1 Original Seven Key Software Areas

The following seven practice areas were analyzed until 2011.

- Customer focused development: Practices that emphasize customer input and feedback throughout the development lifecycle, e.g. Root Cause Analysis, Customer Feedback, Front End Planning, Increase Customer Understanding, Empower Product Owner.
- Design Quality In: Practices that provide a focus on quality early in the development lifecycle, e.g. Adherence to Internal Technology Standards, Architecture, Baseline requirements, Build Management, Code Inspections, Collaborative Product Team, Cross Division Architecture reviews, Design Reviews, Interface Specifications, Management of 3rd Party Deliverables, Refactoring, Reuse.

- Improve Testing Practices: Practices that improve the automation and comprehensiveness of developer and system tests, e.g. Code Coverage, Memory Leak Detection, System Test Automation, Stress and Load Testing, Unit Test Automation, Test Focused Development.
- Software Project Management: Practices to support planning, monitoring and controlling an individual software project, e.g. Cross-Division Cooperation, Knowledge Transfer, Measurement, Predictability, Product Management and R&D Learning, R&D Skills, Release Tracking.
- Multisite/offshore development: Practices that support the effective development of software across geographic boundaries, especially those involving offshore teams, e.g. Cultural Training, Decouple Work, Ease of Communication, Empowered Teams, Knowledge Transfer, Multi-Site Development Environment, Trust.
- Architecture-guided iterative development: Practices drawn from agile and traditional methodologies that are organized into a family of iterative processes, where development is guided by a well-defined architecture, e.g. Agile: Collaborative Product Team, Customer Feedback, Daily Stand-up, Document Just Enough, Ease of Communication, Empower Product Owner, Empowered Team Lead, Iteration Retrospective, Prioritized Feature List, Refactor, Test Automation, Test Focused Design, Time Boxed Iterations, Track Iteration; Traditional Architecture, Baseline Requirements, Build Management, Code Inspections, Manage 3rd Party Deliverables, Track Releases.
- Cross-project cooperation and coordination: Practices that support cooperation and coordination across individual project boundaries. Such cooperation is required for platform-based development and solution-based development.

Our assessment for each practice was based on:

- Criteria for effective deployment of the practice based on accepted industry practices, such as those published in the International Conference on Software Engineering or IEEE Software, and Avaya good practices.
- The extent of usage based on input sessions and our partnerships with Avaya projects.
- Self-assessment of the effectiveness of the practices by a project and the reasons for that assessment.

Each practice was plotted on a grid showing effectiveness and extent of deployment (see Figure 15.13).

The areas and practices were kept relatively consistent until 2010 so that we could analyze trends and make recommendation based on those trends. In addition, several practices within the practice areas were selected as indicators of projects that focus on quality, as discussed in the next section.

15.7.2 Four Practices Tracked As Representative

Beginning in 2012, the following four practices are tracked by the Avaya quality council as necessary, but not sufficient, for achieving good quality while meeting schedules[9].

- Static Analysis, using a commercial or open source tool.
- Code Review and Inspections.
- Automated Regression Testing, to prevent breakage and provide acceptance criteria for code being delivered to test or other projects.

[9] We found that projects that do well on the assessment of these practices usually have embraced a host of other good development practices, and have established a culture of quality in the project.

• Code Coverage, to identify areas of the code that had not been adequately tested.

The report originally started tracking these as a group in 2011, contributing to the Avaya Quality Council's efforts to standardize and focus on a small set of representative good practices. Objective targets for these practices were established so that it was clear what actions were expected of projects. The scores of these four practices are averaged for each project to provide a development process measure that is easy to discuss with the quality council and with R&D leaders.

A project's practices are assessed at each product development business gate, including prior to the start of development, when implementation is complete, e.g. prior to beta testing, and prior to product launch. We have found these measures to be predictive of the quality of the end product [14].

15.7.3 Example Practice Area – Design Quality In

The following eleven individual practices are representative individual practices associated with the "Design Quality In" practice area.

- Baseline requirements and place them under change control.
- Specify and update an architecture sufficient to guide development.
- Deploy internal technical standards.
- Create well-defined interface specifications [10, 11].
- Perform cross-division architecture reviews [18].
- Establish a component integration and reuse program.
- Review requirements, interface specifications, design artifacts, and test scripts.
- Perform inspections of new or changed code.
- Perform automated build management (at least daily) with automated sanity tests[12].
- Carefully manage 3rd party deliverables and dependencies on other projects.
- Refactor a criteria-based selection of modules.

Each of these practices may be further partitioned into sub-practices. For example, in section 15.8.1.1, we identify six key sub-practices of the automated build management practice.

We have defined criteria for effective deployment of each of these practices. For example, the criteria for effective deployment of "manage 3rd party deliverables" are the following.

- All 3rd party deliverables are identified.
- Quality policies are in place and communicated to 3rd party owners.
- Policies are in place on conditions for accepting updates.
- An acceptance test program is in place for each 3rd party deliverable.
- A development team member is identified as "local owner" of each 3rd party deliverable.
- Schedule and quality impact of 3rd party deliverables is assessed.

An assessment of the individual practices for "Design Quality In" from several years ago is shown in Figure 15.13. The X-axis represents how widely the activity is deployed (Usage), and the Y-axis represents how effectively the process has been deployed (Effectiveness) based on criteria for each practice. The evaluation of effectiveness and usage for a practice is a judgment based on our discussions with R&D project members and our analysis of quantitative data. In Figure 15.13 the judgments are represented by the positions of the black circles. For example, at the time of the assessment automated build management was widely deployed and Avaya R&D projects were highly effective in performing automated build management. On the other hand, there was limited usage of structured refactoring

techniques [12] and where deployed the practice was judged to have medium effectiveness. The Overall circle, in blue, represents our judgment of the usage and effectiveness of the total set of "Design Quality In" practices, which was medium to high usage and medium effectiveness.

These charts provide guidance by identifying software practices to target for improvement. Avaya provides guidance on best practices to projects, but leaves the implementation of each specific practice to each project.

Figure 15.14 shows our overall assessment of design quality in from 2002 to 2008. This practice area shows initial improvement, then decline followed by slow improvement in usage and minimal improvement in effectiveness. Avaya has inherited a long tradition from Bell Labs of quality products and quality-focused development processes (see [8], for example), and quality remains a strong emphasis for Avaya R&D leaders and staff.

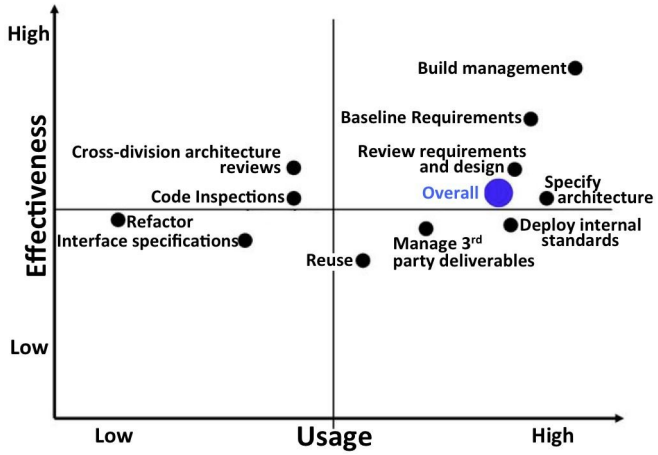


Figure 15.13: "Design Quality In" Practice Assessment

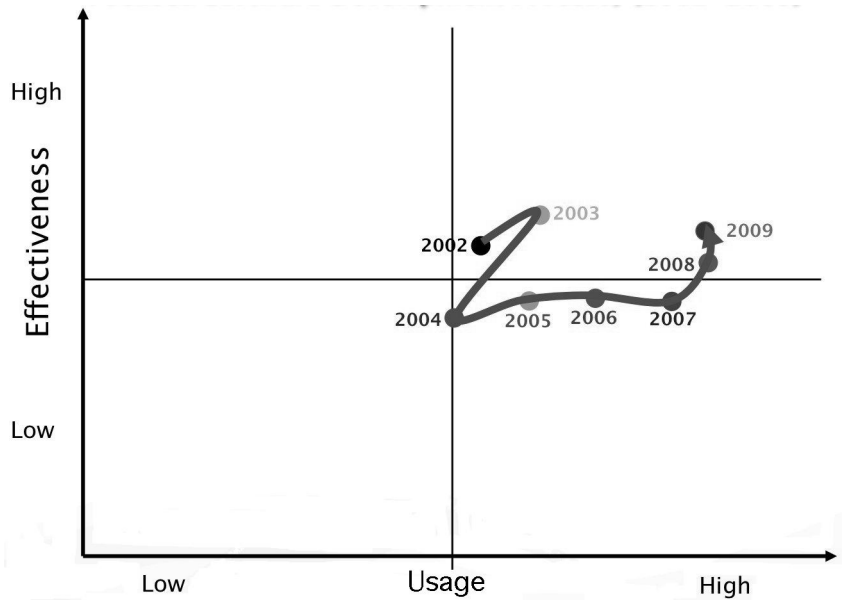


Figure 15.14: Multi Year Trend for Design Quality In (2002-2008)

15.7.4 Example Individual Practice – Static Analysis

When a few products were not meeting quality goals many business units within Avaya made the use of static analysis mandatory. Projects had a choice of tools - either a centrally funded commercial tool or open source tools such as FindBugs for Java. Quality councils tracked how frequently static analysis was used and how aggressively projects fixed real violations. This proactive effort to remove defects complemented code inspections and automated testing.

We analyzed the strategies that projects were using to remove violations, tracked trends across projects that had used static analysis for multiple releases[10], and summarized good project policies in addressing static analysis defects.

For example, depending on product quality, project stage, staff expertise and other characteristics, one or more of the following static analysis policies are typically used by Avaya development teams.

- No new or high impact violations allowed in any build
- Target critical violations (potentially high impact outliers)
- Decrease or eradicate medium impact violations
- Cover fixed code by automated tests
- Fix when a file is being changed for other reasons, especially for legacy systems.

When fixing violations in legacy code where there was limited or no experience remaining with the file, projects took two approaches to minimize breakage:

1. If a file was being changed, and the developer had some familiarity with the file, the project took the opportunity to correct high and medium impacting violations
2. A project addressed all violations of a particular type in legacy code all at once, regardless of whether a file was being changed for some other reason

We found that a few types of violations accounted for the majority of violations. We also found that some violations had a more critical impact on the code than might be suggested by the vendor. We

[10] Many business critical systems in Avaya have infrequent releases both because of the complexity and interaction of features, and because business customers cannot absorb frequent releases.

gave guidance on what type of violations to fix first. Figure 15.15 shows the distribution of the top violations across a sample of 32 Avaya projects. For these 32 projects, over 26% of all static analysis violations are "pass by value" or "uninitialized constructor" violations.

Similar charts are created for individual projects.

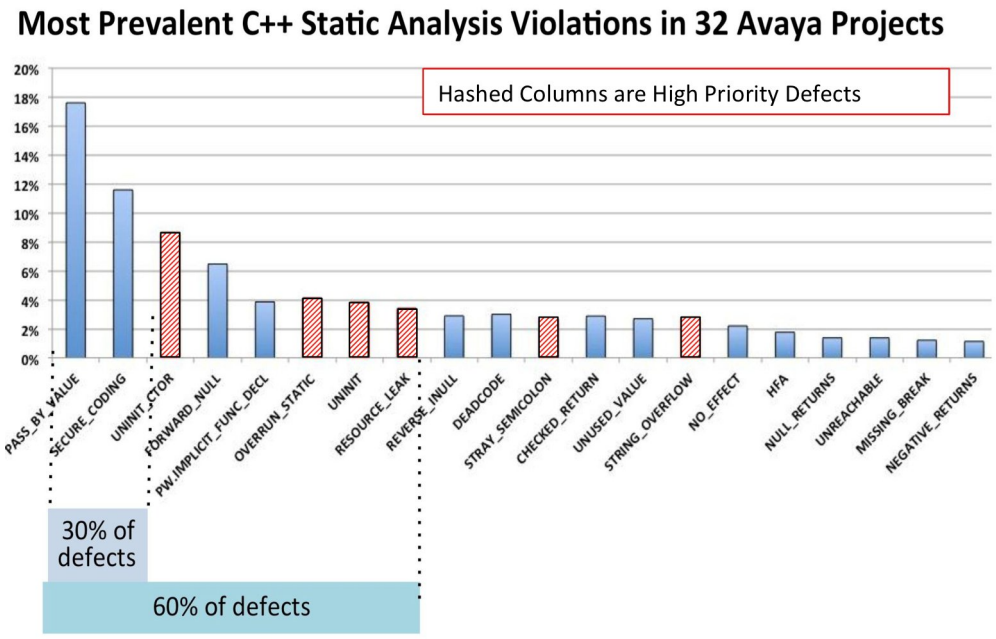


Figure 15.15: Distribution of Static Analysis Violations

We track to determine if product teams are improving in their deployment of static analysis practices. Figure 15.16 shows an example for one Avaya R&D organization. The average static analysis score for the organization's product teams as determined by the R&D quality council increased year over year for most years leading up to 2013. In addition the number of product teams with a static analysis score of 4[11] (the largest score possible) increased each year.

[11] A score of 4 means that the product team performs static analysis with every build, and the team addresses all serious defects. 2 means static analysis is run occasionally on new/changed code, there is no systematic approach to defect correction, but output is monitored and the most serious defects are corrected. 0 means no static analysis.

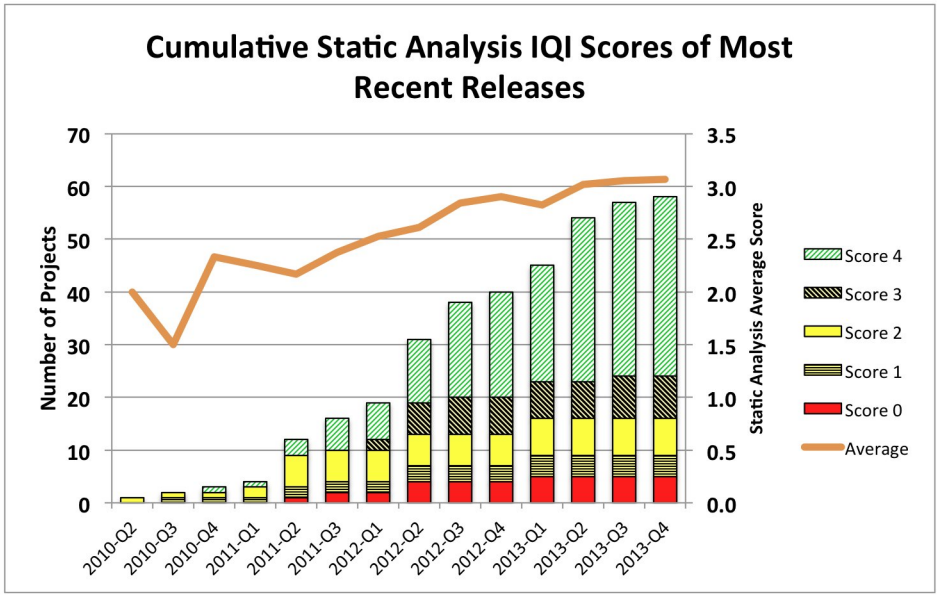


Figure 15.16: Trend in Static Analysis Scores

15.8 Assessment Follow-up: Recommendations and Impact

We create a set of five to seven prioritized recommendations each year based on the assessment findings and identify a suggested owner for each recommendation. Suggested owners typically include the following roles.

- The project team leader if the recommendation is within the scope of an individual project.
- The R&D leader for an organization if the recommendation is within the scope of an individual R&D organization.
- The product management leader within a division if the recommendation requires the leadership of the product management team.
- The division general manager if the recommendation requires cooperation with a division across the entire functional areas of the business.
- Avaya's operations leader if the recommendation requires cross-division cooperation.

No individual role is assigned more than two or three areas of responsibility in any given year.

The report provides an abstraction of each recommendation in a summary table similar to Table 15.3, and a detailed description in the body of the report.

The first column of the summary table contains the recommendation abstract. The second column identifies the software practice addressed by the recommendation, and the third column identifies the suggested owner.

Recommendation	Software Practice Area Addressed	Deployment Responsibility

Recommendation	Software Practice Area Addressed	Deployment Responsibility
1. **Recommendation One:** Continue the software improvement program with a concentration on customer-focused development Monitor the impact of the program	• Customer Focused Development	• Division Product Management leader
2. **Recommendation Two:** Improve build management practices as a lever to improve quality. Monitor the impact.	• Design Quality In	• Project team lead for each project
3. **Recommendation Three:** Improve the program for transfer … Monitor the impact.	• ….	• R&D Vice President for each division program

Table 15.3: Example Recommendations Summary Table (Excerpt)

Two example recommendations are described in section 15.8.1. Section 15.8.2 describes how the recommendations are deployed.

15.8.1 Example Recommendations

Recommendations made between 2002 and 2013 cover a variety of topics, such as the following.

- Front end planning,
- Deployment of architecture-guided iterative and agile practices,
- Software project management,
- Multi-site and offshore development practices, and
- Domain expertise of the R&D staff.

The following sections describe two example recommendations. The automated build recommendation is an early recommendation (section 15.8.1.1), initially made in 2003. The risky file management recommendation (section 15.8.1.2) is more recent, initially made in 2012.

15.8.1.1 Automated Build Management Recommendation

We classify automated build management as a practice in the "design quality in" practice area (see section 15.7.3).

Avaya had several challenges in build management when the corporation was initially formed. While some Avaya projects had very sophisticated build management practices that supported distributed development and automated sanity testing[12], the practices were not standard across the corporation. Build practices in many projects were inefficient, not fully automated and limited in their testing and reporting capabilities. In many projects, build management practices were more of a hindrance than a help to developers in rapidly integrating and testing their code.

[12] Sanity testing is the practice of running a few tests after a build to ensure that the product was built correctly and that basic functionality works.

Our recommendation was to improve build management practices as a lever to improve quality, including the following set of industry-proven good practices [3, 17].

- Good Practice 1. Treat load check-in as a quality gate
- Good Practice 2. Automatically track versions of all components of a load
- Good Practice 3. Establish a fully automated build and sanity test process
- Good Practice 4. Perform daily builds and sanity test
- Good Practice 5. Automatically link change tracking and version control
- Good Practice 6. Define and track load building metrics

We provided details on each practice in a separate report and identified example projects that could serve as mentors to other projects.

15.8.1.2 Risky File Management: Target the Riskiest Files Causing Field Defects for Improvement

We identified risky file management as a key risk management technique to focus resources on the most critical files in a project's code repository (section 15.6.3).

Our recommended action in the 2012 report was for product teams with field quality issues or a large customer base, to establish a standard practice of reviewing the project's riskiest files and determine an appropriate step, such as content control or refactor, to reduce the risk associated with changes to these files.

In 2013, Avaya Labs created a dashboard of candidate risky files for a large cross-section of Avaya products. The dashboard helps projects identify their riskiest files and is automatically updated weekly.

15.8.2 Deployment of Recommendations

Recommendations are deployed in partnership with business and R&D leaders. We conduct follow-up sessions with individual business leaders, R&D leaders, and product management leaders of each organization. The sessions are typically structured based on the following list of questions for each recommendation.

- Is the recommendation relevant to your organization/team?
- If relevant, is it already deployed in your organization/team?
- If relevant and not deployed, what are the barriers to deployment?
- What actions are you prepared to take?
- How can our team help in deployment?

Note that as a result of these sessions, a recommendation may not be deployed in a particular organization either because it is not relevant to that organization or the barriers for deployment are too large. In addition, the recommendation may be adapted to make it more deployable in a particular organization. When appropriate actions for an organization are identified for a recommendation, action plans are then defined, deployed and tracked. Results are incorporated into the next year's assessment report.

15.9 Impact of the Assessments

As noted in section 15.3, we evaluate the impact based on usage and effectiveness of the practices (see example in section 15.9.1), on the extent and impact of deployment (section 15.9.2), and on the improvements in the customers' view of quality as measured by the customer quality metric [20] (section 15.9.3). Following are a few examples.

15.9.1 Example: Automated build management

Our assessment in 2002 and 2003 was that Avaya projects had medium to high usage of automated build management practices with low to medium effectiveness[13] as shown in Figure 15.17. In 2003 and 2004, we made specific improvement recommendations as described in section 15.8. These recommendations helped create a focus in Avaya R&D in 2004 and 2005 to improve build management practices. The good practices in build management described in section 15.8.1.1 are now widely deployed and build management is an effective, efficient practice for most Avaya projects (see Figure 15.17) for 2002 through 2008).

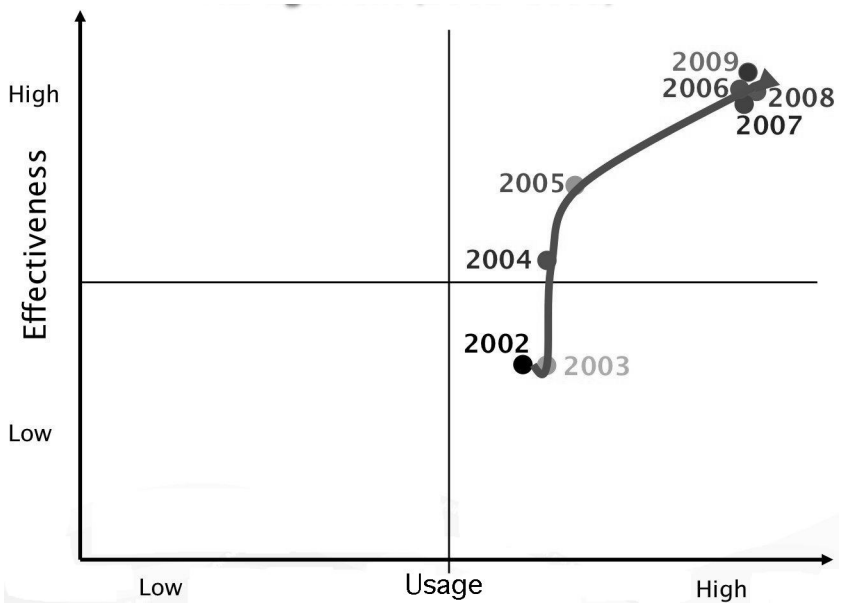


Figure 15.17: Multi-year Trend of Usage and Effectiveness of Automated Build Management Practices (2002-2008)

Automated build management continues to be a strength of Avaya product teams in 2014 with widespread and effective deployment. In fact, our assessment focus is no longer on automated build management, but on continuous build and integration practices.

15.9.2 Example: Deployment of Risky file management

We provided briefings on risky file management to about 40% of the active projects in the first half of 2013 (i.e. the first several months after the 2012 State of Avaya Software report was published).

- 50% of those projects' products deployed risky file management as a part of their development process.
- 25% of the projects are considering deployment.

[13] The scale for Usage and Effectiveness are defined in the same way as used in section 15.7.

- 25% of the projects are not deploying, typically because the project is no longer doing significant new development – i.e. they are now doing sustaining engineering.

We do not yet have post-release quality data for most of the projects that deployed risky file management, because they have not yet been deployed long enough to obtain reliable defect data. Two projects with sufficient defect data that used risky file management as part of their quality improvement program had significant improvements in quality.

- The customer quality metric (CQM) [20] for an Avaya contact center product improved by 30%.
- The CQM for an Avaya endpoints product improved by well over 50%.

15.9.3 Improvement in Customer Quality Metric (CQM)

The operational cost of fixing a customer found defect includes the cost to diagnose the defect, to implement and test the fix to the defect, and to deploy the corrected fix.

We estimate the impact of the assessments based on the operational savings as a result of fewer customer found defects (CFDs).

The operational benefit for any product release is calculated as follows. First we calculate or obtain the following data.

- Expected number of CFDs against the product release; the expected number is based on CQM (which is based on in-service time per customer) prior to improvement and the number of product release installations.
- Actual number of CFDs against the product release.
- Operational cost of fixing a CFD (cost provided by Avaya corporate quality team).

Then,
Operational benefit = Operational cost of fixing a CFD * (Expected number of CFDs – Actual number of CFDs)

Table 15.4 shows the operational savings calculations for a sample of product releases over the past three years.

Product Release	Operational Benefit
Product A Release 6.1	$328K
Product A Release 6.2	$406K
Product A Release 6.3	$27K
Product B Release 6.1	$450K
Product B Release 6.2	$773K
Product B Release 6.3	$932K
Product C Release 6.2	$5,974K
Product D Release 6.0	$1,686K
Product D Release 7.0	-$124K
Product D Release 8.0	$4,073K
Product D Release 8.1	$3,620K

Table 15.4: Operational Savings Calculations for Product Releases of Four Avaya Products

The prior release of Product C (Release 6.1) did not deploy practices recommended in the reports and had numerous quality issues resulting in many millions of dollars in additional operational costs.

Using the approach described in this section, we estimate the annual operational savings based on the deployment of quality focused improvements for the past three years to be at least $60M per year based only on reduced cost of dealing with customer reported issues, and not including benefits resulting from increased customer satisfaction. The investment in the quality improvement effort was likely to be much lower. Approximately 20% of the effort of the ARC team goes into producing the report and obtaining relevant measures. The investments made by development, testing, and quality organizations are harder to quantify, but are highly likely not to have exceeded the savings.

15.10 Conclusions

15.10.1 Impact of the Assessment Process

As noted in earlier sections, the process of assessing software development and sustainment in Avaya has had significant impact, on the way that software is developed, on the practices and processes targeted for improvement, on achieving company goals, and on the cost and value of software technology within the company.

The impact has been made possible by focusing on the goals of the company and on obtaining data that can be used to analyze how well software development within the company is contributing to meeting those goals, following the GQM approach to empirical studies. The idea has been to improve the way the company does software development and to know it.

The impact of the process is evident in several ways. The annual analysis, as described in the report The State of Software in Avaya, shows to all who are interested what improvements have occurred and what areas need improvement, often prompting allocation of resources to making improvements and monitoring those areas. It helps the company set goals and understand better whether or not it is achieving those goals. For example, improving software quality by reducing defects found by customers frees software developers' time to work on adding new features and capabilities to products rather than correcting bugs in those products. With appropriate focus on data collection, we are able to estimate how much the savings from improved quality is, providing more substance to the idea of knowing what the improvement is.

The assessment process can also be viewed as providing a competitive advantage to the company. It both allows the company to show customers the quality improvements and allows more rapid development of new features or customization to customers' needs. As a result, the company both retains existing customers and attracts new customers, instilling confidence in customers that the company has a systematic process in place to improve quality.

15.10.2 Factors Contributing to Success

The assessment process was started in 2002 and is a continuing process, producing the State of Software in Avaya report at the end of every year. Based on its longevity and continuing impact, we believe the assessment process has been a success. Factors contributing to that success include the following.

1. A well-defined measurement methodology, founded on the goal-question-metric approach [4, 5, 7].
2. The ability to identify organizational goals.
3. The availability of the necessary data and the willingness of the producers of the data to share it.
4. Cooperation by people at all levels of technical and management positions, from software developers to the company's executives.

5. Confidence by all involved that the conclusions drawn from analysis of the data, both quantitative and qualitative, are valid, with such confidence instilled by continuing discussion with development organizations, and retrospective views of the results.
6. Acknowledgement from the software development organizations that the annual recommendations are useful and lead to (needed) improvements.
7. Evidence that the assessment process is having a positive financial impact on the company, e.g., that considerable time and effort is saved through improvements in quality of the type and by the means suggested in the state of software reports.
8. An incremental approach to implementing the assessment program, starting with relatively modest goals and an in-house focus, using a subset of the types of data that are currently analyzed, and proceeding over the years to a broader focus incorporating customer-focused data into the analysis.

In summary the assessment process started with clear goals, a repeatable, systematic measurement process, gained the confidence of those who provided the data by providing them with useful analyses and recommendations, and then expanded the focus of the analysis and the breadth of the data analyzed. During the 12 years, so far, of the assessment process, the company has undergone significant changes in management and personnel, but the utility of the assessment process has remained clear to all, and it has continued to receive strong support from the software development organizations, from company executives, and from Avaya Research.

15.10.3 Organizational Attributes

The success of the process, of course, is also greatly owed to the skills and commitment of the ARC who, from the beginning, formed close relationships with the development organizations who both provided the data and were the subjects of the recommendations. The ARC was (and is) principally staffed by two or three people with considerable development experience and experience in working with development organizations to foster improvements. The ARC was established by a research director and was a part of a research department, and received direction from the director, and not from someone in the direct management line of the development organizations. Accordingly, it was perceived as being objective and non-competitive. It was also able to call on others in the research department for support in discussing and implementing different forms of data analysis. Furthermore, the ARC has remained a stable organization, with two of the same members that it had at its inception, and with continuing support from the same researchers.

15.10.4 Selling the Assessment Process

As previously noted, there are a number of factors contributing to the success of the assessment process and the corresponding annual report. Perhaps most important to the success of the process was showing that the findings of the reports could have significant impact on the company by improving software development and related practices, and by making stakeholders out of a wide variety of people in roles throughout the company, ranging from typical software developers through the company CEO. Without continuing support from the stakeholders, the assessment process and report would have had no way to demonstrate value. Early in the process, we worked with small groups who had confidence in the ARC based on earlier dealings with its members. Based on the early results, as the process continued we broadened the stakeholder base, and always made sure we were focusing on issues of interest to stakeholders, often by showing them preliminary results and asking for their validation. As the process evolved, we could show the results of the recommendations from earlier analyses, thereby helping to validate the value of the process, and establishing a virtuous cycle. We think that such an approach helps assure success and would advise others who are interested in starting an assessment program to use a similar strategy.

15.10.5 Next Steps

We will continue to prepare an annual state of software report for Avaya and expect to focus on the following areas in the coming years.

- Our demographic assessments focus on the R&D development and testing community and, because of the role of software across Avaya's business, we intend to include product management, services and support as well as other areas in future demographic assessments.
- As noted in section 15.6, we have discovered that it is even more important to help projects implement recommendations than to provide them. As a result, we will continue to provide integrated analyses of multiple data sources and tools and procedures to aid in the action steps for individual projects. Risky file management (section 15.6.3) is an example of this approach.
- As we utilize new data repositories (section 15.5), we will improve our ability to estimate the accuracy of our data and analyses.

We are also interested in benchmarking with other companies who are performing similar analyses and with whom we could assure data comparability.

It is an open question as to how the various factors analyzed interact with each other, and is a subject of future research.

The techniques we use in producing the report could be applied in any organization that can define its goals, that is willing to collect the data needed to measure progress towards those goals, and that is willing to make an investment in improving its development practices. Although we typically think of such organizations as companies, or business units within companies, the techniques could be applied to any entity that develops software. For example, a worthwhile goal might be to produce a report on the state of software in the U.S. Certainly, the challenges would be large, but the payoffs in terms of knowing where one should invest to improve the way software is developed in the country would be large as well.

15.11 Acknowledgements

We thank the 300+ people who have contributed data, participated in interviews, validated results, and helped with the analysis of data for the State of Software in Avaya reports.

We especially thank David Bennett, Neil Harrison, Joe Maranzano, and John Payseur for their contributions to early assessments, Trung Dinh-Trong for his contributions to reports, especially in the areas of test automation and code coverage, Zhou Minghui for her work in Avaya on developer productivity, Evelyn Moritz for her analyses and feedback to the report as well as co-authorship of several reports, Pierre Osborne for his leadership in establishing and supporting Avaya's R&D data warehouse, and Jon Bentley for his careful and helpful reviews of the reports. Thanks to Lee Laskin, Dan Kovacs, Saied Seghatoleslami, and Patrick Peisker for their contributions around product quality processes and data. Thanks to Ravi Sethi, Brett Shockley, and Venky Krishnaswamy for their executive sponsorship and support of the ARC program and Christian Von Reventlow for his enthusiastic assistance in establishing deployment programs in partnership with business and R&D leaders. Thanks to Ashwin Kallingal Joshy for careful readings of earlier versions.

References

1. Weiss, D., Bennett, D. Payseur, J., Zhang, P., Tendick, P.; "Goal-Oriented Software Assessment, International Conference on Software Engineering", May 2002
2. Rifkin, S.; "Why Software Process Innovations Are Not Adopted", IEEE Software, July/August 2001
3. Appleton, B., Berczuk, S.P., Software Configuration Management Patterns, Addison Wesley, 2003.
4. Basili, V., and D. Weiss; "Evaluating Software Development By Analysis of Changes: Some Data From The Software Engineering Laboratory", IEEE Trans. on Software Engineering, February, 1985
5. Basili, V., and Weiss, D.; "A Methodology For Collecting Valid Software Engineering Data", IEEE Trans. on Software Engineering, November, 1984
6. Basili, V., Caldiera, G., McGarry, F., Pajeraki. R., Page, G., Waligora, S.;"The Software Engineering Laboratory—An Operational Software Experience Factory, International Conference on Software Engineering", 1992, ISBN: 0-8791-504-6.
7. Basili, V., Caldiera, G., and Rombach, H.D.; "Goal Question Metric Approach," Encyclopedia of Software Engineering, pp. 528-532, John Wiley & Sons, Inc., 1994.
8. Bell Labs Technical Journal, Special Issue Software, Volume 5, Issue 2 (April/June 2000), available at http://www3.interscience.wiley.com/journal/97518354/issue.
9. Brown., M, Goldenson, D.; "Measurement and Analysis: What Can and Does Go Wrong?, Proceedings of the 10th International Symposium on Software Metrics (METRICS'04)", Septtember, 2004,
10. Clements, P., Britton, K., Parnas, D. L., Weiss, D. "Interface Specifications for the SCR (A-7E) Extended Computer Module", NRL Report 4843, Naval Research Laboratory, Washington, D.C., January 1983
11. Britton, K. H., Parker, R.A., Parnas, D.L.; A Procedure For Designing Abstract Interfaces for Device Interface Modules, Proc. 5th Int. Conf. Software Eng., San Diego, CA,1981, pp. 195-204
12. Fowler, M., Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.
13. Grady, R., Caswell, E.; Software Metrics, Prentice-Hall, 1987
14. Hackbarth, R., Mockus, A., Palframan, J., Sethi, R., "Customer Quality Improvement of Software Systems", to be published
15. van Solingen, R. and Berghout, E., The Goal/Question/Metric Method: A practical guide for quality improvement of software development, McGraw-Hill Publishers, 1999.
16. Chang, H-F., and Mockus,A., "Evaluation of source code copy detection methods on freebsd", In Mining Software Repositories. ACM Press, May 10-11,2008
17. McConnel, S., Rapid Development, Microsoft Press, 1996
18. Maranzano, J., Rozsypal, S., Warnken, G., Weiss, D., Wirth, P., Zimmerman, A.; "Architecture Reviews: Practice and Experience", IEEE Software, March/April 2005
19. Mockus, A, "Large-scale code reuse in open source software". In ICSE'07 Intl. Workshop on Emerging Trends in FLOSS Research and Development, Minneapolis, Minnesota, May 21 2007
20. Mockus, A. and Weiss, D., "Interval quality: Relating Customer-perceived Quality to Process Quality", In 2008 International Conference on Software Engineering, pages 733-740, Leipzig, Germany, May 10-18 2008. ACM Press.
21. Mockus, A. and Weiss. D., "Globalization by Chunking: A Quantitative Approach," IEEE Software, vol. 18, no. 2, pp. 30-37, Mar./Apr. 2001.
22. Mockus, A., Fielding, R.T., and Herbsleb, J.,"Two Case Studies of Open Source Software Development: Apache and Mozilla." ACM Transactions on Software Engineering and Methodology, 11(3):1-38, July 2002
23. Mockus, A., Hackbarth, R., Palframan, J., "Risky Files: An Approach to Focus Quality Improvement Efforts", June 3, 2013, CID 162981, ALR-2013-23, and presented at Foundations of Software Engineering Conference/2013, August 18, 2013

24. Hackbarth, R. , Mockus, A., Palframan, J, Weiss, D., "Assessing the State of Software in a Large Organization", ALR-2009-004) Journal of Empirical Software Engineering, 2009.

25. QFD Institute: The Official Source for QFD, http://www.qfdi.org/

26. Rotella, P, Chuylani, S Implementing Quality Metrics and Goals at the Corporate Level, Working Conference on Mining Software Repositories, 2011

27. Stokes, D.; Pasteur's Quadrant: Basic Science and Technological Innovation, Brookings Institution Press, 1997

28. Walston, CE., and Felix, C.P., "A Method of Programming Measurement and Estimation," IBM System Journal, Vol. 16, No. 1, 1977

29. Zhou, Minghui, "Learning Legacy Products: How Product Structure Shapes Organization", ICSE workshop on Socio-technical Congruence. May 19, 2009.

30. Holzmann, G; Mars Code, Comm. ACM, Vol 57, No. 2, 2014

31. Mockus, Audris. Amassing and indexing a large sample of version control systems: towards the census of public source code history. In 6th IEEE Working Conference on Mining Software Repositories, May 16-17 2009

32. Minghui Zhou and Audris Mockus. Developer fluency: Achieving true mastery in software projects. In ACM SIGSOFT / FSE, pages 137-146, Santa Fe, New Mexico, November 7-11 2010

33. CMMI Institute; http://whatis.cmmiinstitute.com

15.12 Appendix

Example Questions Used For Input Sessions

As one source of input we use individual and small group input sessions with R&D and product management. These sessions cover all divisions, major R&D sites, various levels of technical staff and management, and a variety of R&D roles, including architects, developers, project and program managers, testers, documentation, and support. Our findings are reviewed with them and adapted based on their feedback. Each session is based on two sets of questions and a set of software practices that we tailor based on the role of those in the input session. Common questions include:

- What are the 3 primary software-related areas that your project is doing particularly well?
- What are the 3 primary software-related issues facing your project?
- What is your assessment of the software quality for your project and why?
- What is your assessment of the productivity for your project and why?
- What is your assessment of the domain expertise of your staff? Why?
- What software development approach(es) are used in your project?
- If you could change one software related area in Avaya, what would you change?

The nature of these sessions and the questions that we ask has evolved over time. For early reports we had less data on trends and on a variety of factors such as productivity, cost, and quality. Accordingly, we spent more time in individual discussion sessions. Currently, we are able to show session participants charts and graphs showing trends and, as necessary, data about their organizations, and use those to focus the session. As a result, sessions are more to the point and briefer.

Randy Hackbath

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, 320 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.

Audris Mockus:

Audris Mockus received a B.S. in Applied Mathematics from Moscow Institute of Physics and Technology and a Ph.D. in Statistics from Carnegie Mellon University. He teaches at the University of Tennessee and works part-time at Avaya Labs Research. Contact him at audris@utk.edu.

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 communications software and software development tools. He has an M.Math and a B.Math (co-op) from the University of Waterloo. Contact him at palframan@avaya.com .

David Weiss:

David M. Weiss received the B.S. degree in Mathematics in 1964 from Union College, and the M.S. in Computer Science in 1974 and the Ph.D. in Computer Science in 1981 from the University of Maryland. He is currently professor of software engineering at Iowa State University. Previously, he was the Director of the Software Technology Research Department at Avaya Laboratories, where he worked on the problem of how to improve the effectiveness of software development in general and of Avaya's software development processes in particular. To focus on the latter problem, he formed and led the Avaya Resource Center for Software Technology. Dr. Weiss's principal research interests are in the area of software engineering, particularly in software development processes and methodologies, software design, software measurement, and globally-distributed software development. His best known work is the goal-question-metric approach to software

measurement, his work on the modular structure of software systems, and his work in software product-line engineering.