Architecture Reviews: Practice and Experience

Joseph F. Maranzano, *Millennium Services*
Sandra A. Rozsypal and Gus H. Zimmerman, *Lucent Technologies*
Guy W. Warnken and Patricia E. Wirth, *AT&T Labs*
David M. Weiss, *Avaya Labs*

Find out how to incorporate architecture reviews into your development methodology. They'll help you fix problems before they become costly, and they'll help management make better-informed decisions.

Large, complex software projects are notoriously late to market, often exhibit quality problems, and don't always deliver on promised functionality.[1] In our companies, we have observed too many projects that either fail or require significant rework late in their schedules. Line managers and engineers can't communicate problems consistently up the chain of command because they fear reprisal or they don't have the knowledge or expertise to detect a significant issue. Therefore, upper management often isn't sufficiently warned of serious problems. Architecture reviews have evolved over the past decade to become a critical part of our continuing efforts to improve this state of affairs. We use them to identify project problems before they become costly to fix and to provide timely information to upper management so that they can make better-informed decisions. The reviews also help identify best practices to projects and socialize such practices across the organization, thereby improving the organization's quality and operations.

An architecture review is a mechanism for increasing the likelihood that a system architecture will be complete and consistent. Good software architecture is critically important for successful software development. It's the key framework for all technical decisions. It provides the foundation for reuse, using commercially available software, and getting to the marketplace fast. In addition, software architecture has a profound influence on project organizations' functioning and structure. Poor architecture can reflect poorly defined organizations with inherent project inefficiencies, poor communication, and poor decision-making.[2,3]

Having evolved from a common ancestor, the review processes in our four companies are all similar, embodying the same principles but varying in implementation. In addition to review meetings, they include meeting preparation and a mechanism for reporting on, tracking, and closing important issues. The architecture review we describe here is based on the processes our companies use today. The process is rooted in AT&T's *Best Current Practices: Software Architecture Validation*.[4,5]

0740-7459/05/$20.00 © 2005 IEEE

Architecture review value proposition

Architecture reviews have consistently produced value for our corporations. We've preserved the process even in the most difficult economic and cost-cutting times. Reviews are valuable because they

- Find design problems early in development when they're less expensive to fix
- Leverage experienced people by using their expertise and experience to help other projects in the company
- Let the companies better manage software components suppliers
- Provide management with better visibility into technical and project management issues
- Generate good problem descriptions by having the review team critique them for consistency and completeness
- Rapidly identify knowledge gaps and establish training in areas where errors frequently occur (for example, creating a company-wide performance course when many reviews indicated performance issues)
- Promote cross-product knowledge and learning
- Keep experts engaged
- Spread knowledge of proven practices in the company by using the review teams to capture these practices across projects

Since 1988, we've conducted more than 700 project reviews. Starting with AT&T, architecture reviews have grown across all our companies. We estimate that projects of 100,000 noncommentary source lines of code have saved an average of US$1 million each by identifying and resolving problems early.

Our five principles

Five principles form the basis of our architecture reviews.

1. A clearly defined problem statement drives the system architecture. The review looks at the project as a reasonable solution to the problem statement. Problem statements contain key considerations and constraints that must be met to have a solution. One useful way to evaluate a statement is to see if it has considerations or constraints in the areas of *function*, what the system must do; *form*, how the system fits into its environment; *economy*, the cost of developing, deploying, or operating the system; and *time*, the release dates and the system's evolvability. There must be a clear problem statement before an effective review can take place.[6]

2. Product line and business application projects require a system architect at all phases. The system architect, whether a person or small team, is an explicitly identified role. The architect must explain the reasons for the technical decisions that define the system, including alternatives considered, and must document choices in a concise, coherent architecture specification.[7]

3. Independent experts conduct reviews. The chosen reviewers are recognized experts in their fields, generally with considerable experience. To assure impartiality, they're also independent of the particular project under review.[8,9]

4. Reviews are open processes. A review isn't a project's audit or evaluation against some arbitrary standard, nor is it a tutorial or evaluation of the architects' performance. Thus, the review team conducts the review openly and invites project members to attend. Everyone involved in the review process can see the architecture's issues and strengths and note them (often on 5 × 8 cards, called *snow cards*), displaying the cards in the meeting room. An open approach develops the trust between reviewers and project members that's necessary for a successful review. The project members don't expect the review team to solve the issues but rather to apply its expertise to uncover issues.

5. Companies conduct reviews for the project's benefit. All reaction to the issues discovered, such as a project's architectural changes, replanning, or even cancellation, is the project team and management's responsibility. The review process focuses on providing feedback to the project so that it can improve the way it solves the problem. Accordingly, the project must decide how to respond to the issues that come up during the review. Furthermore, the architects provide guidance to the review team about aspects of the problem solution that particularly concern them.

Our process

Effective architecture reviews aren't difficult to implement with the right process.

Projects of 100,000 noncommentary source lines of code saved an average of US$1 million each by identifying and resolving problems early.

Three primary organizations are involved in an architecture review: the project team, the review team, and the architecture review board.

Participants

Three primary organizations are involved in an architecture review: the project team, review team, and architecture review board. Each organization has a variety of stakeholders in the system that play different roles in the review process.

The review client pays for the system development or is the requested architecture review's sponsor. A review's primary *customers* are the project team members and management. Therefore, the project team can ask the review team to pay particular attention to specific architectural concerns.[11]

The project members who created, contributed to, and will use the architecture constitute the project team. They present the problem and the proposed solution and will ultimately act on the issues. One project team member is often selected to be the project contact to manage the review logistics and provide all the required artifacts to the review team.

The project management consists of all the managers responsible for the project's success. Their responsibilities include nominating the project for review, supporting the project team through all phases of the review process, attending the review read-out presentation and the management briefing, and ensuring that an action plan is prepared and followed. High-level managers are generally involved most in the follow-up part of the process when serious issues are found.

The review team consists of subject matter experts, assembled for the review on the basis of their expertise, their independence from the specific project, and their ability to conduct themselves appropriately in a potentially difficult interpersonal situation. One or more members of the architecture review board (explained later) select reviewers from a companywide database that lists potential reviewers' area of expertise, their domain knowledge, and the number of reviews they've participated in over the past year. External subject matter experts occasionally supplement the review team. The review team prepares for the review by studying the problem statement and architecture specification, listens to the project team's presentation, asks questions, and identifies possible problems with the architecture.

The architecture review board is a standing board that oversees the review process and its effect on the organization. The ARB consists of respected members of the organization who have managerial experience and have demonstrated an ability to effectively address difficult technical and organizational issues. It helps determine whether a project would benefit from a review, selects a review "angel" (explained later), follows up, and changes the review process where necessary to maintain its value and currency. The ARB is also responsible for improving the quality of the organization's architectures, maintaining a review results archive, and identifying opportunities for cross-organizational improvements.

Without such a board or similar organization—whether formally or informally constituted—architecture reviews become uneven in quality, and the knowledge of how to conduct them can disappear. In other words, the ARB is the keeper of architecture reviews' practice and tradition.

The review angel is chosen by the ARB from among its members with managerial experience. The angel works with the project team to address any organizational or political issues that arise.

The ARB chair must be a strong architecture review process advocate and a highly respected upper-level manager. The chair is responsible for ensuring that the reviews remain effective and securing continuing support for them.

Implementation

Our process is simple to implement in companies with experienced architects and domain experts. We select reviewers from the company's architect pool and teach them the review process. We focus their technical expertise and domain knowledge where they're needed and occasionally supplement the internal review team with external subject matter experts.

Table 1 shows the differences in the architecture review processes across our companies. For example, Lucent uses a formal ARB comprising a cross-organizational team that actively supports and encourages using architecture reviews. We could view the ARB as the

Table 1

Variation in architecture review processes

Process	AT&T	Avaya	Lucent	Millennium Services
Architecture review board	Core team	Core team	Active, cross-organizational team	Core team
Review team	Core team augmented with a virtual team from internal and external subject matter experts	Core team members augmented with internal and external subject matter experts	Internal and external subject matter experts with an architecture review board member	Core team augmented with external subject matter experts
Project nomination	Self-nominating	Self-nominating	Self-nominating	Self-nominating
Review preparation time	4–6 weeks	2–4 weeks	4–5 weeks	2–4 weeks
Reviews	Face-to-face reviews supplemented by confidential interviews	Various, ranging from conference calls to face-to-face reviews supplemented by interviews	Face-to-face reviews	Face-to-face reviews
Typical duration	3–5 days	1–5 days	2–3 days	3–5 days
Follow-up required	Depends on customer needs and by client request	Recommended but depends on client	Only for management alerts and critical issues	Follow-up action plan generated with project team

keeper and sustaining advocate for Lucent's standard architecture review process inside the company. In the other companies, a team in a particular organizational unit, often called a *core team*, plays a similar role, often as part of a larger mission such as process improvement.

Many projects have more than one review. Typically, once the team adequately defines the problem, it holds a brief initial review. Longer, more detailed reviews focusing on the proposed solution come later. A review might look at a development project that produces a product or service, an outside vendor, a third-party product, or a project intended to produce reusable assets and platforms.

The review process always follows the same four phases, although their implementation may vary.

Phase 1. Screening. The project indicates its interest in having a review by filing a Project Profile Form. The project staff and ARB review this profile to determine whether a review would benefit the project. If they recommend a review, the ARB selects a review angel to oversee the project's review process.

Phase 2. Preparation. The ARB or core team selects a review team, including a review leader, and works with the project to determine the number of reviews and the initial review's date and agenda. The staff, project team, review leader, and review angel verify that the project has an adequately clear problem statement and appropriate documentation.

Phase 3. Review meeting. Review meetings vary in length—a review could be as short as half a day (a straightforward project's initial review) or as long as a week (a very complex, groundbreaking project's final review). During the review, the project team presents its problem statement and outlines how the proposed architecture solves it. The reviewers ask questions and record issues they believe could make the project an incomplete or inadequate solution to the problem. The reviewers' domain knowledge and a standard review checklist guide the questions, which are often augmented for the particular review (see Figure 1).

At the end of the review meeting, the review team meets privately to organize and synthesize the important issues they identified into affinity groups. They determine the issues' severity, sorting them into the following categories:

- A *management alert* could cause the project to fail if it isn't addressed immediately.
- A *critical* issue might not cause the project to fail but should be a candidate for a follow-up review because the architecture will require rework. The solution is usually complex and not obvious.
- A *major* issue could potentially cause major customer dissatisfaction or significant rework.

Figure 1. Excerpts from the architecture review checklist.

Problem Statement

1. What is the problem you are trying to solve? What are the top value-added features/services for the product? Who are the target customers? What is the market-window for the product?
2. What are the functional requirements for the product? Have the key functional requirements been summarized in a clear, concise priority order? Have the requirements been reviewed with customers?
3. What are the environments that the system/product will operate in? What are the constraints on the system? What are the major external interfaces?

...

Error Recovery

1. What is the error rate expected in parsing data input from other systems and data feeds? Is manual handling required to correct such errors?
2. Is there an error recovery code to clean up situations when an error is detected? How do you know that the error recovery code is invoked? Are there external processes that can be used to perform error cleanup?
3. If the customer has requested fault tolerance, what are the customer's expectations? Will your system provide both hardware and software fault tolerance? What techniques are you using to ensure software fault tolerance?
4. Is a sparing strategy employed for processors, other hardware components, and/or network components to reduce effects of hardware errors? If so, how long does it take to recover the system?
5. Can diagnostics be run online while the system is running? Is there a consistent error log to capture information from diagnostics?
6. What is the back-out strategy if errors occur during installation?

...

Performance

1. Are the application requirements for performance written down? What are the characteristics of these performance requirements?
2. Are there processes in the system through which the application is single-threaded? What is the implication of this on performance?
3. How much headroom has been built into the system?
4. What performance characteristics of database management are required?
5. If your system runs as a distributed architecture, what criteria have been used for distributing processing and/or data?
6. How much memory is needed for buffers, program space, and operating system functions? What is the maximum load that can be run under this memory allocation?
7. Is there sufficient bandwidth in the I/O subsystems to meet the application requirements?

...

- A *minor* issue probably won't affect overall project success but the project team still should investigate it.

The review team then delivers a read-out presentation to the project team and its management, summarizing significant issues and the project's condition.

Phase 4. Follow-up. The review team delivers a report with the review's findings to the project team within 15 days of the review. Figure 2 shows an example output from a review. If management alerts exist, the review angel, staff, and ARB chair prepare a management alert letter for the managers responsible for the project's success. The project team and management must respond within two weeks. The review angel can opt to hold a management briefing to give feedback on the project's strengths and action plans to resolve issues the review raised.[10]

1 Introduction

An architecture review was conducted of the XYZ project on 5/1/2004. This report summarizes the finding of that review.

2 Summary

There are serious deficiencies in the current architecture separate and apart from short-term systems failures and performance problems. Unless critical improvements to the Usage Handling subsystem and Service Delivery subsystem are implemented, anticipated volumes and business needs are problematic. In addition, to meet the anticipated workload, the system will have to be migrated to new hardware. As of our review meeting, we did not see a definite schedule for this activity.

There are three management alerts resulting from this review: Structural deficiencies in the current architecture, performance deficiencies, and lack of adequate error-handling designs.

...

3 Performance Issues

3.1 Failure to provide performance engineering (management alert)

There was no evidence of proactive performance engineering in the XYZ development process. The approach used is the classic "measure and tune" strategy that may work for a small-scale, low-volume application but is inadequate for the validation of large-scale production software. There does exist a substantial amount of benchmark and production data, but the model that enables that data to influence future development decisions is missing. The result is that there is no systematic way of evaluating design options, a situation that obviously could lead to the choice of less-than-optimal development directions.

...

3.2 System resources to support workload have not been quantified (critical)

XYZ described several improvements to the Service Delivery subsystem including elimination of full table scans and polling that are projected to deliver 3x and 2x increases, respectively, in subsystem throughput. Since XYZ was unable to quantify what the current subsystem architecture can deliver, it is unknown whether these improvements will be sufficient to meet customers' projected load.

...

3.3 Scalability (major)

Supporting an order entry volume of 5,000 orders per day will require an estimated 1,000 active agents in the busy hours by September. XYZ did not envision problems in supporting this number of agents but did not present proof that it could, in fact, be supported.

...

4 Operations, Administration, and Maintenance

4.1 Database purging and archiving not systematically addressed (major)

Given that the database is expected to grow to 1.7TB, it is critical that the database be kept to the minimal size required by purging and archiving unneeded data. Failure to do so will increase operational expenses and have a negative impact on both online and batch performance. The ability to perform these functions must be included in the basic architecture. Planning for this is a major effort.

...

5 Error Recovery

5.1 Choke points and single points of failure (critical)

The current design in which all components of the application exist on a single server presents multiple problems. First, a failure of that server results in an outage for all components of the application. Second, a performance issue (e.g. – runaway task, daemon hang, etc.) in one component has the potential of impacting the performance of all components. Third, scalability of the overall application may be constrained by the scalability of any given component as well as the capacity of the server on which the application is placed.

...

Figure 2. Excerpts from an architecture review report.

Between 1989 and 2000, our architecture reviews found more than 1,000 issues.

Artifacts

The review team relies on both documents and interaction with architects to understand and review the architecture. To ensure compliance with the architecture review process, the team uses several supporting documents.

Architecture review checklist. The checklist contains questions that the architects should consider in preparing for the review and that the reviewers consider during the review. The questions are a guide that often leads to further architecture exploration. The architects can augment the checklist for a particular review to help focus the reviewers' attention on key architecture areas. The checklist evolves over time according to the most serious and prevalent issues the reviews uncover. One might think of the checklist as an accumulated institutional knowledge repository about issues that arise in creating architectures. Figure 1 shows an excerpt from an architecture review checklist.

Input to the review. The project team assembles a set of artifacts and delivers it to the review team before the review. In addition to the problem statement, it comprises system requirements, functional requirements, an architectural specification, and other informational documents.

- System requirements define the behavior that the system must meet. They include such items as reliability and availability requirements, capacity and performance requirements, and operational and maintenance requirements.
- Functional requirements defines the features that the system must support. These are often too voluminous for the review team to study in depth. For most reviews, the project team provides the review team with a sample of the functional requirements, including a table of all the features and their key system attributes, sample reports and screens, sizing assumptions, and a sample of the interfaces to other systems and hardware.
- An architecture specification specifies how the system will meet the system requirements and what its structures are.
- Informational documents are project artifacts that would help the reviewers better understand the system's function. These include a product overview, a project plan, and organizational charts defining the project team's structure.[12,13]

Output from the review. An architecture review has three primary outputs: a set of issues (often in the form of snow cards), a review report, and an optional management alert letter. When the review team puts the issues into a priority-ordered list as described earlier, they make recommendations to support the project team in developing an action plan. They also list architecture and project strengths. The ARB report (see Figure 2) is a written report that presents the review findings and is delivered to the project team.

Our results

Between 1989 and 2000, our architecture reviews found more than 1,000 issues. We organized them into six categories:

- *Problem definition*. The problem isn't completely or clearly defined.
- *Product architecture and design*. The proposed solution doesn't adequately solve the problem.
- *Technology*. The languages, tools, and components being used to build the system are inadequate.
- *Domain knowledge*. The team doesn't have adequate knowledge or experience to solve the problem.
- *Process*. The process for stating the problem and creating the solution isn't systematic, complete, or designed to make development manageable.
- *Management controls*. The necessary management monitoring capabilities, staffing, controls, and decision-making mechanisms are inadequate.

Table 2 shows what proportion of issues belongs to each category. Proportions are shown as a range, indicating how they've varied over time. For example, problem definitions constituted as little as 10 percent and as much as 18 percent of review issues in different years.

Between 12 and 15 percent of the findings were management alerts, with about another 25 percent classified as critical. Over the years, we've reduced the incidence of problem defini-

tion issues as a direct result of early architecture review findings. We've put processes in place to develop and document effective problem definitions. On the other hand, management controls issues have increased recently as market requirements change with increasing rapidity and companies reduce staffing.

Product architecture and design issues are often the most expensive problems to fix because they generally impact many components and much code. The design issues we find most often are in performance engineering, error handling and recovery, reliability and availability, operations administration and maintenance, and system evolvability. Figure 2 is a sample of issues excerpted (after suitable sanitization) from an architecture review report.

Lessons learned

When starting an architecture review process in your company, select both projects that are in relatively good shape and those that are problematic. The architecture review process, like any other new process, needs to gain acceptance and be viewed as a helpful mechanism rather than a management "stick." It's therefore vital that the reviews provide objective findings. Place the primary focus on helping projects succeed.

Establishing a review board with executive support helps others accept the process. Select board members from across the organization and encourage them to solicit projects for review from their organizational entity. Have the board monitor the reviews' effectiveness and make necessary improvements. The board should also monitor trends that appear across reviews and fix systemic issues (for example, by establishing a training program or investing in a commercially available training program).

Commit to keeping the review results confidential to the project, its management, and the board unless stakeholders are willing to distribute them more widely. Establish a reviewer database that contains reviewers' names, organization, and technical expertise. Use board members to recruit available reviewers. Conduct some early test reviews where you can tune the process to work best for your company and business environment. Alternatively, hire a credible external review company to conduct some early reviews and then establish your own process.

Table 2

Categories of architecture review issues

Category	Percent of issues
Problem definition	10–18%
Product architecture and design	29–49%
Technology	3–14%
Domain knowledge	2–5%
Process	4–19%
Management controls	14–26%

A few don'ts

Here are additional practical tips for setting up an architecture review process:

- Don't sacrifice the reviews' integrity for political expediency; don't change the review team's results because a high-level manager doesn't like them.
- Don't punish the managers of projects whose reviews reveal significant problems; instead, educate them.
- Don't apply different criteria to different projects or people.
- Don't forbid the review team to ask any questions.
- Don't constrain the process too strictly; keep it flexible. Make sure you address what's useful to your clients and customers.

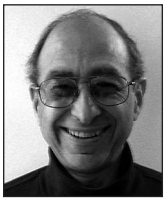
Benefits

Although an architecture review's primary purpose is to improve the products reviewed, numerous beneficial side effects exist for the organizations that conduct them:

Cross-organizational learning is enhanced. By reinforcing a project's successes and strengths, engineers and architects can transfer lessons learned and practices that they found to be the best to subsequent projects they undertake. As a result, the organization's output quality will improve. The review process also enables them—albeit informally—to identify and share corporate assets and reusable components.

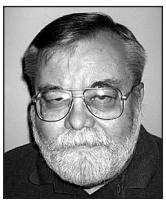
Architecture reviews get management attention without personal retribution. Because independent experts conduct reviews and the organization's management sanctions them, they're a safe way to make a project's most serious issues

About the Authors

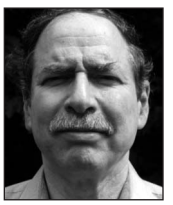


Joseph F. Maranzano is vice president of engineering at Millennium Services. His research interests include software architecture and project management. He received his MS in electrical engineering from New York University. He's a member of the IEEE and a Bell Labs fellow. Contact him at 63 Pomeroy Rd., Madison, NJ 07940; jfm@att.net.

Sandra A. Rozsypal is retired from Lucent Technologies, where she worked on financial and product management and was a member of its Systems Architecture Review Board. Her research interests include creating, developing, and realizing telecommunications products. She received her MA in marketing from Benedictine University. Contact her at 792 Lake Holiday Dr., Sandwich, IL 60548; srozsypal@earthlink.net.



Guy W. Warnken leads the Technical Assessments Group of AT&T Global Networking Technology Services. His research interests include systems architecture and effective software development teams. He received his BS in plant science from Rutgers University. Contact him at AT&T Labs, Room D3-3C01, 200 Laurel Ave., Middletown, NJ 07748-4801; warnken@att.com.



David M. Weiss is the head of the Software Technology Research Dept. at Avaya Labs. His research interests focus on making software development more effective. He received his PhD in computer science from the University of Maryland. He's a senior member of the IEEE, a member of the ACM, and associate editor in chief of *IEEE Transactions on Software Engineering*. Contact him at Software Technology Research, Avaya Labs, 233 Mt. Airy Rd., Basking Ridge, NJ 07920; weiss@avaya.com.

Patricia E. Wirth recently retired as the director of AT&T Labs' Network Design and Performance Analysis Department. Her research interests include improving technical organizations' effectiveness. She received her DSc in systems science and mathematics from Washington University in St. Louis. She's an AT&T fellow. Contact her at 1281 Eatontown Blvd., Oceanport, NJ 07757; pwirth@att.net.

Gus H. Zimmerman is the director of Lucent Technologies' Systems Architecture Review Board. His primary research interest is telecommunications products' systems architecture. He received his PhD in physics from Harvard University. He's a senior member of the IEEE and a member of the ACM and the American Society for Quality (ASQ). Contact him at Bell Laboratories, Lucent Technologies, Room 9F-345, 1960 Lucent Ln., Naperville, IL 60566; ghzimmerman@lucent.com.

known to the right management levels. Project members often provide information and messages to the review team that they haven't felt comfortable taking to their line management.

Information sharing becomes possible. The architecture reviews provide an opportunity for the key project team members, including the business leaders, to hear a presentation on the architecture and reasons for key decisions. Because all project team members receive the review team read-out at once, it provides an excellent learning opportunity for the entire team.

Architecture reviews assist organizational change. Architecture reviews provide an intervention opportunity to an organizational-change effort. By observing both what projects are doing and how they're doing it, interactions among the various process participants can introduce change into individual projects and the larger organization.

Greater opportunities exist to find different defects in integration and system tests. When you find product design defects in architecture reviews, you can use integration and system testing to perform other testing functions (such as integrating several products into a service).

Architecture reviews have proven to be an invaluable part of our companies' quality improvement process by finding complex and expensive system problems early in the development life cycle.

Companywide quality improvement and training programs, such as architecture workshops and performance engineering courses, have resulted from the major issues the architecture reviews found. We've defined, refined, and institutionalized problem definition processes to make sure projects could concisely document the business problems they needed to solve. In addition, the architecture review process has helped train people to become better architects and helped establish a consistent view across our companies both of what architecture is and what good architecture's characteristics are.

Over the years, we've refined and improved our architecture review process as we better understood how to staff reviews, generate valuable architecture checklists, and create effective architecture specification content. Just the process of preparing for an architecture review became a forcing function to improve our architecture creation process. The review process became a standard practice in our companies and is still evolving today. Most projects in our companies now wouldn't consider developing a complex software system without conducting an architecture review.

Acknowledgments

Many people contributed significantly to developing our process. They include Barbara Alleman, Larry Bernstein, James Cochrane, Joe Colson, Phil Fuhrer, Lou Haskell, Bin Ho, James Holtman, Mo Iwama, Nick Landsberg, Deborah Maracich, Bob Martin, Kurt Neubek, Jan Norton, Lynette Parker, Steve Parshall, William Peña, Clark Ryan, Daniel Starr, Eric Sumner Sr., Eric Sumner Jr., Frank Webb, Jerry Weinberg, and Mary Zajac. We're particularly grateful for the advice, support, and encouragement of current and past members of Lucent's Systems Architecture Review Board and its core staff. Neil Harrison helped introduce the architecture review process into Avaya and helped extract and analyze Avaya architecture review data. Mark Clay and Jeanne Eisele, AT&T Labs Technical Assessment Group team members, were valuable contributors on the current process and results in AT&T.

References

1. C. Jones, *Assessment and Control of Software Risks*, Prentice Hall, 1994.
2. P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley, 2002.
3. J. Bosch, *Design and Use of Software Architectures*, Addison-Wesley, 2000.
4. *Best Current Practices: Software Architecture Validation*, AT&T Bell Labs, 1990.
5. D. Starr and G. Zimmerman, "A Blueprint for Success: Implementing an Architectural Review System," *STQE Magazine*, Jul./Aug. 2002, pp. 50–55.
6. W.M. Peña and S. Parshall, *Problem Seeking: An Architectural Programming Primer*, 4th ed., John Wiley & Sons, 2001.
7. P. Clements et al., *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, 2002.
8. D.P. Freeman and G.M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews*, 3rd ed., Dorset House, 1990.
9. M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, vol. 15, no. 3, 1976, pp. 182–211.
10. R. Kazman and L. Bass, "Making Architecture Reviews Work in the Real World," *IEEE Software*, vol. 19, no. 1, 2002, pp. 67–73.
11. D.L. Parnas and D.M. Weiss, "Active Design Reviews: Principles and Practices," *Software Fundamentals: Collected Papers of David L. Parnas*, D.M. Hoffman and D.M. Weiss, eds., Addison-Wesley, 2001, pp. 339–352.
12. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed., Addison-Wesley, 2003.
13. *IEEE/EIA Std. 12207, Information Technology—Software Life Cycle Processes—Life Cycle Data, Software Architecture Description*, IEEE, 2001.