

# Making Architecture Reviews Work in the Real World

**Rick Kazman and Len Bass**, *Software Engineering Institute, Carnegie Mellon University*

**A** software architecture is more than just a technical blueprint of a complex software-intensive system. In addition to its technical functions, a software architecture has important social, organizational, managerial, and business implications.<sup>1</sup> This same observation holds of architecture reviews. We can't simply regard them as technical reviews and ignore their other implications.

Architecture reviews differ from other technical reviews because of their close relationship to a system's business goals. Consequently, they should be approached differently, with an eye for nontechnical issues. Here, the authors explore the social, psychological, and managerial issues of formal architecture reviews.

Over the past five years, we have participated in over 25 evaluations of software and system architectures in many different application domains, using first the Software Architecture Analysis Method (SAAM)<sup>2</sup> and later the Architecture Trade-off Analysis Method (ATAM).<sup>3</sup> In discussing these methods in the past, we reported almost solely on their technical aspects. However, developing and refining these methods exposed us to a wide variety of systems, organizations, organizational goals, management styles, and individual personalities. These experiences have forged our opinions—in particular, we are now convinced of the need to explicitly teach and manage the nontechnical aspects of running an architecture review.

We thus decided to record our observations and findings on the management, psychology, and sociology of performing architecture evaluations. Getting these factors wrong can doom the best technical effort.

This observation is not particularly an artifact of software development; it holds true of any complex engineering effort:

*When Brunel and Robert Stephenson were building railways in the 1830s and 1840s, they were expected to involve themselves with raising capital, appearing before Parliamentary Committees, conciliating influential people who might oppose the necessary bill in Parliament, negotiating with landlords over whose land the tracks were to be laid, managing huge gangs of labourers, and dealing with subcontractors. Railway engineers had to be expert in finance, in politics, in real estate, in labour management, and in procurement. Why should we be surprised if software engineers may need to draw on expertise in mathematics, financial analysis, business, production quality control, sociology, and law, as well as in each application area they deal with?<sup>4</sup>*

The point is that these kinds of issues are relevant for anyone who has a business inter-

**The difference between an architecture review and a code review is revealed in the question the review is designed to answer.**

est in a system's success—primarily the architects but also the managers, customers, and architecture reviewers. In particular, as architecture reviewers, we continually run into social, psychological, and managerial issues and must be prepared to deal with them.

### **Architecture reviews**

Since at least 1976,<sup>5</sup> reviews have been recognized as an efficient means for detecting defects in code or other artifacts. Why then, is an architecture review worthy of special consideration?

In one of our engagements, during a discussion of business goals, one stakeholder turned to another and said, "I told you when you originally brought this up, and I will tell you again—you are a small portion of my market, and I cannot adjust [the topic under discussion] to satisfy you."

The difference between an architecture review and a code review is revealed in the question the review is designed to answer. The code review is designed to answer, "Does this code successfully implement its specification?" The architecture review is designed to answer, "Will the computer system to be built from this architecture satisfy its business goals?"

Asking whether code meets specifications assumes the specifications are clear. Often a precondition for a code inspection is a prior inspection of the specifications. Asking whether an architecture satisfies business goals cannot simply assume clarity of the business goals. A system's business goals will vary depending on the stakeholders' perspectives and how much time has passed since the system was conceived. Time to market, cost, quality, and function priorities can all change based on events that occur during the initial architecture design. If the business goals aren't clear, then one (important) portion of the architecture review process is to operationalize the business goals.

### **Who participates in the review?**

In a code review, only the reviewers (usually three to five) are present, and they focus on the task at hand. In an architecture review, not only are three to five reviewers present but also the architect and, possibly, the architecture team. In addition, because business goals are discussed, a variety of stakeholders must also participate. At one

review, we had 30 to 40 stakeholders in the room. We were reviewing a large system for which the government had contracted, and many government agencies and contractors were developing parts of the hardware and software. As we will later discuss, setting the business goals and having different stakeholders involved causes many logistical problems.

Furthermore, in a code review, most of the participants are development professionals who are familiar with the review process. Those who are not can be instructed to study a description of the code inspection process prior to the review. For many stakeholders, the architecture review is their first experience with a review that attempts to match business goals with an architecture, and many are busy managers and professionals who can't be expected to study a process description. Consequently, we must spend valuable review time explaining the process. Also, having stakeholders who cross organizational boundaries inherently raises the review's visibility. Managers from multiple groups interested in the system's success (or failure) are aware of the review and interested in its outcome.

### **What does "success" mean?**

A client (whose system was behind schedule and over budget) once complained that his architecture had already been reviewed multiple times. "I'm tired of being reviewed by amateurs," he said. "They simply repeat what they were told and have no value added."

Code reviews have as their output a collection of discovered defects. There is ample evidence that discovering defects during a code inspection is more cost-effective than discovering the defects after deployment. Furthermore, there is usually no controversy about whether a defect has been discovered. Someone proposes a sequence of events under which incorrect results will occur to convince the inspection team of the defect. However, the outputs from an architecture review are much more varied—some architectural documentation, a set of scenarios of concern, and list of risks, sensitivity points, and tradeoff points. If the stakeholders do not agree on the form or value of the review's outputs, then success will be difficult to achieve (or even define).

## Reviewing the risks

ATAM-based architecture inspections have as their major output a collection of risks, sensitivity points, and tradeoff points. A risk is an alternative that might cause problems with respect to meeting a business goal. Notice the vagueness in this definition: an alternative that “might” cause problems. Because software systems are expensive to construct, it is nearly impossible to gather evidence about development paths not taken.

There are four broad categories of outputs that can emerge from an architecture review:

- *Technical risks*, which emerge from a SAAM or an ATAM. For example, “Given the current characterization of peak load and the existence of only two Web servers, we might not meet the architecture’s latency goals.” We can mitigate such risks through more analysis, simulation, or prototyping.
- *Information risks*, involving areas of the architecture lacking information. There are times during a review when the response to a reviewer’s questions is simply, “We haven’t thought about that.”
- *Economic risks (cost, benefit, and schedule)*, which are not directly technical in nature but are about dollars and deliverables. “Can we deliver this functionality to our customers by August? Will we lose market share if our performance isn’t quite as good as our competitor’s? Can we build this for less than \$80 per unit?” Architectural decisions all profoundly affect these questions. We can mitigate these risks using architecture analysis techniques that focus on these issues (for example, the Cost–Benefit Analysis Method<sup>6</sup>).
- *Managerial risks*, which involve having an architecture improperly aligned with the organization’s business goals.<sup>7</sup> Such misalignment can be risky for an organization—regardless of its product’s technical superiority—and might require a costly realignment process. Examples of other kinds of managerial risks include misaligning the architecture’s structure and the development organization’s structure or depending on suppliers whose reliability is unknown or suspect. Each case requires a kind of realignment

between the architecture and management or business strategy.

## Planned versus unplanned

Code reviews are usually included in the normal development plan as are architecture reviews. However, unplanned architecture reviews also sometimes occur, either because a stakeholder (typically an architect or manager) becomes interested in the possibility of doing such a review to validate and improve an existing project or because an upper-level manager wants a review to scrutinize a project that is perceived as being in trouble. Unplanned reviews for troubled projects have a certain amount of inherent tension. Everyone involved understands that the stakes are high and the project could be cancelled as a result of the review. Thus, such reviews often result in finger pointing and blaming but unfortunately offer little hope of rescuing the architecture or project.

Unplanned reviews also require a certain amount of selling. “What are the reviewers reviewing? What will be the outcomes? How much time will it cost us [the client] in dollars and days?” These are the kinds of questions that the stakeholders ask. They must be convinced that an architecture review will help and not hinder the project or their decision making prior to proceeding.

## Review preparation

A code inspection meeting typically inspects about 300 lines of code. Although not all of a system’s code is necessarily inspected, the decision of which code to inspect is made outside of the inspection. Furthermore, a system’s code should be available prior to the review—otherwise, there’s nothing to review.

In an architecture review, because its goal is to decide how well the architecture supports the business goals, reviewers can inspect any portion of the system. Furthermore, because the business goals are often ambiguous, we need to dedicate a portion of the review to determining which parts of the system are involved in meeting those business goals. This means that the reviewers must be aware of the entire system’s architecture. Also, the architecture is not always adequately documented—in contrast to code inspections, adequate architectural documentation might not have been prepared. For example, one of our reviews was

**ATAM-based architecture inspections have as their major output a collection of risks, sensitivity points, and tradeoff points.**

**The review team must have members who are experienced, architecturally savvy, and quick on their feet.**

unplanned in nature, and although we were committed to a particular schedule, the architecture documentation was not forthcoming. Finally, two days before the review took place, we received a vast collection of class descriptions said to constitute the “architecture documentation.” That was the basis under which we were forced to conduct the review. As might be expected, the review did not follow the normal procedure and was somewhat ad hoc in nature.

### **Managing the review process**

Let’s apply these observations about architecture reviews to motivate a set of concerns that a reviewer will face when working through the review process. Different architecture-review techniques have different processes. For this reason, we restrict ourselves to discussing the three broad stages of activity in any review: *prework*, which involves negotiating and preparing for the review; *work*, which involves scrutinizing the architectural artifacts; and *post-work*, which is when you report the review’s results and determine what actions to take.

#### **Pework**

It is easy to pay less attention to the activities that precede a review, because they don’t involve any technical analyses. However, because of an architecture review’s unique nature, if these activities are ignored, the review’s results might be meaningless. If the customer is completely unprepared to be reviewed, it wastes the time of both the reviewers and the team under review. Pework usually involves three main tasks, and although it isn’t always exciting, it ensures that the later work is exciting and meaningful.

First, pework involves selling the review. Because reviews tend to originate from different sources within an organization, it is often the case that some of the participants are unhappy with the review. This is particularly the case when the review activity is not part of the normal software process. In such a case, it is often viewed as an intrusion on the stakeholders’ time. So, it is crucial, both before and during the review, to sell management, the architects, and the other stakeholders on the review’s value—that a reviewed system will meet the stakeholder’s goals better than an unreviewed system.

Second, it is also important to set expecta-

tations. All of the stakeholders should understand the method’s capabilities. In a typical ATAM, you meet with a client for just three days. You need to use this limited time to evaluate the architecture’s ability to achieve its business goals—you will not do detailed technical analyses in any dimension.

Finally, you need to decide who should be there for what stages. Some participants will perceive any review, no matter how crucial, as an intrusion on their precious time. Hence, it is important to identify and forewarn the stakeholders and ensure that each of them knows which steps of the process require their presence, and why. In addition, it is often useful to strictly limit attendance in many of the steps to the minimal set of stakeholders needed, because this typically makes for a more efficient and productive use of time. As mentioned earlier, there are times when 30 or 40 stakeholders will want to attend, perhaps to impress their bosses or sponsors, further their own agendas, or bill for extra time. Whatever the reason, their agendas should not dilute the review’s effectiveness.

As part of the standard set of ATAM materials, we have prepared a presentation designed to support these three goals. The presentation explains the evaluation’s goals and process as well as its costs and benefits. This presentation is useful both for those who are going to participate in the evaluation and those whose concurrence is needed for the evaluation to proceed.

#### **Work**

During an architecture review, we not only perform the steps of the review process but also act as facilitators. This is where the reviewer’s personal qualities (mentioned earlier) are important. The review team must have members who are experienced, architecturally savvy, and quick on their feet.

#### *Technical rationale*

During an architecture review, we should be able to establish business goals, rely on the architect, and identify risks.

An effective architecture review process presents the system’s business goals to the reviewers and stakeholders and identifies and prioritizes a set of scenarios that represent concrete specifications of these goals. For example, if one business goal is for the system to be long-lasting, then modification

scenarios become a high priority. These high-priority scenarios determine the evaluation's focus.

Because the architect identifies how the architecture will satisfy such scenarios, he or she is critical to making the architecture understandable. Each scenario begins with a stimulus—an event arriving at the system, a fault occurring, or a change request being given to the development team. The architect walks through the architecture, explaining what is affected by the stimulus—for example, how the system will process an event or detect a fault, or what components will be affected by a change. The architect will also review how a response to the stimulus will be made—the recovery mechanisms needed and which modules will change when a change request is addressed.

Each scenario reflects specific business goals. To ensure that these business goals are adequately met, part of the evaluators' due diligence is to understand the architect's explanations of how the architecture will respond to the stimulus and how any identified risks will be dealt with. Risks are identified as a result of three possible conditions: the explanation of how the architecture responds to the stimulus is not convincing, business goals other than the ones reflected in the scenario are violated, or fundamental decisions have not yet been made. Usually, when an architecture review is held, not all architectural decisions have been made. However, this becomes a risk when too many decisions have yet to be made or when the indecision threatens business goals.

#### *Social behavior*

To best use the stakeholders' time during the review, the team needs to

1. control the crowd,
2. involve the key stakeholders,
3. engage all participants,
4. maintain authority,
5. control the pace, and
6. get concurrence and feedback.

Crowd control is critical. At the start of the review meeting, establish how and when people may interact with each other. For example, it is important to avoid disruptions, so side conversations, cell phones, and pagers should be banned. Establish at the

outset whether people can come and go, and when they absolutely must be present. Latecomers should not expect the proceedings to stop so they can be updated. Holding the review meeting away from the home site of the team being reviewed is an effective way of minimizing interruptions.

Involving key stakeholders is also important, because any activity that involves the entire group interacting also helps achieve buy-in. As each point important for the design or analysis emerges, record it visibly and verify that the recording is correct. Participants then feel they are helping in the process; it isn't just the reviewers reviewing the architecture—it's the entire group discovering aspects of the architecture. For example, in one review, the architecture team (and the lead architect, in particular) were skeptical of the review's value and hence initially participated only grudgingly. Throughout the review, and as insights into the system emerged, we convinced the architect that this activity was for everyone's benefit and that we were not there to point fingers but rather to improve the architecture and hence the resulting system. By the end of the review, the architect was an enthusiastic participant.

In addition to involving the key stakeholders, it is important to engage all participants. It is not uncommon, in any kind of meeting, to have people who dominate the airwaves or people who are shy about participating. For example, some people might be reluctant to speak frankly in front of their bosses or their subordinates. For these people, it is important to provide a forum in which they are either alone or only among peers. It is also important to provide a mix of free-for-all participation and periods where each person has a dedicated "safe" time to speak.

Of course, despite your best efforts, there might be times when an evaluation gets out of control: people will have side conversations, try to steal the agenda, or resist providing information. The review team needs to know who has ultimate authority if people are being disruptive. Is it the review team leader, the customer, a specific manager, or the architect? The review team facilitator should have a strong (but not dogmatic) personality and should be able to manage the group dynamics through a combination of humor, appeal, and authority.

**Because the architect identifies how the architecture will satisfy such scenarios, he or she is critical to making the architecture understandable.**

**When the review is complete, there must be some agreement on how to communicate the outputs back to the stakeholders.**

Similarly, the team architecture review facilitator must be able to control the pace. Any meeting with a diverse group of (likely highly opinionated) stakeholders will range out of control from time to time, as mentioned. But sometimes these conversations are revealing—hidden agendas, worries, future requirements, past problems, and a myriad of other issues come to light. The facilitator must be aware of these digressions and know both when to squelch them and when to let a conversation continue. A portion of the time allocated is reserved for the reviewers to prepare their briefing. This same time can be used for offline meetings among the stakeholders.

This kind of facilitation can be exhausting: assimilating huge amounts of information, looking for problems, and managing all of the political and personal issues simultaneously. Thus, we have found that it is useful to have two facilitators and to switch between them periodically to give each a mental break.

Finally, be sure to obtain concurrence and feedback. The activities and information generated in a review are not really directed at the review team, even though the review team is frequently the focus of the conversation and the source of many of the probing questions. The review's outputs are really for the stakeholders—the review team members are just there to act as catalysts, experts, and facilitators. Because of this mismatch between the producers and consumers of the information and the way that the information is elicited (through the facilitation of the review team), extra care must be paid to ensure that all stakeholders concur with whatever is recorded. In our reviews, for example, we typically record information on flip-charts in real time as well as on a computer—for later presentation or out-briefing and for the final report. When we put information up on a flip-chart, we have a golden opportunity to ensure that the information recorded is correct. Thus, we endeavor to get concurrence as items are posted and ensure that we keep flip-charts visible around the room so that they are always available for consultation, correction, and refinement. Architects are human and welcome encouragement and positive feedback. A review is mostly concerned with finding problematic decisions, so it is useful for the reviewers to occasionally

make positive comments about particular architectural decisions. This helps alleviate the generally negative questions and sets a more positive tone.

### **Postwork**

Reviews of all kinds are part of a mature organization's software process, but the time and trouble involved are wasted unless there is a predefined output and a determination of who will act on the results. Otherwise, the review's outputs will end up buried on someone's desk and become a low-priority item. Thus, postwork activities must report the outputs and follow up on the review's results.

When the review is complete, there must be some agreement on how to communicate the outputs back to the stakeholders—in particular, to management and the architecture team. In the ATAM, we give slide presentations to the stakeholders immediately after the review and then, some weeks later, we deliver a formal and more extensive written report. Regardless of the result's form, the review team must know who gets told what and when. For example, can the architects respond to the review report before it goes to management or other stakeholders?

A slide presentation or report from a review can have many possible destinies. In our experience, the review's outputs (lists of scenarios, architectural styles, risks, nonrisks, sensitivities, and trade offs) validate existing project practices, change existing project practices (such as architecture design and documentation), argue for and obtain additional funding from management, and plan for future architectural evolution. In one case, a participant, the day after the review, used the outbrief to convince management that he needed more resources for the project—an unsuccessful appeal prior to the review. It is thus important to consider the report's goal and to plan accordingly.

**T**he time is ripe for the widespread adoption of architecture reviews as a standard part of the software engineering lifecycle. Materials to teach and support the practice of architecture reviews have been increasing in both quantity and

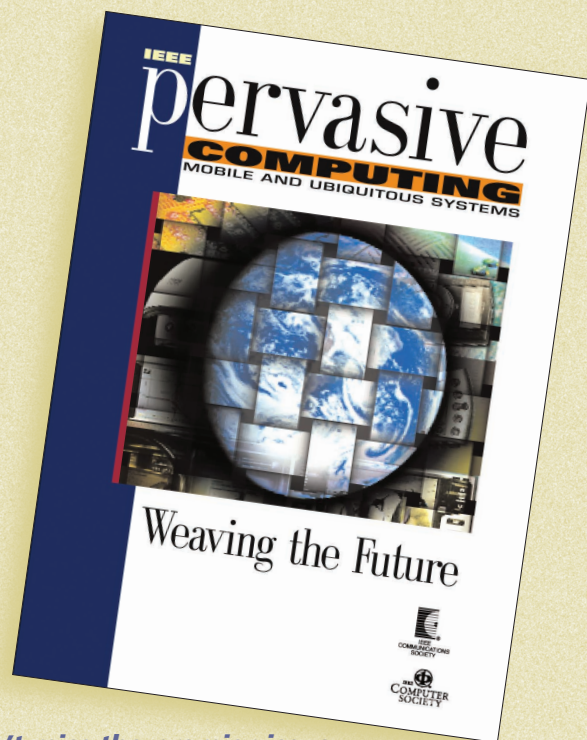
# NEW FOR 2002

The exploding popularity of mobile Internet access, third-generation wireless communication, and wearable and handheld devices have made pervasive computing a reality. New mobile computing architectures, algorithms, environments, support services, hardware, and applications are coming online faster than ever. To help you keep pace, the IEEE Computer Society and Communications Society are proud to announce *IEEE Pervasive Computing*.

This new quarterly magazine aims to advance pervasive computing by bringing together its various disciplines, including

- **Hardware technologies**
- **Software infrastructure**
- **Real-world sensing and interaction**
- **Human-computer interaction**
- **Security, scalability, and privacy**

Led by Editor in Chief M. Satyanarayanan, the founding editorial board features leading experts from UC Berkeley, Stanford, Sun Microsystems, and Intel.



## VISIT

IEEE Distributed Systems Online,  
*Pervasive Computing's* online resource.


DS Online offers expert-moderated information on topics including mobile & wireless computing, distributed agents, and operating systems.

[computer.org/dsonline](http://computer.org/dsonline)

Don't miss the premier issue  
**SUBSCRIBE NOW!**

<http://computer.org/pervasive>

quality, including a book devoted entirely to architecture reviews.<sup>3</sup> Many large corporations are now adopting architecture reviews as part of their standard software engineering development practice, and some are even including these reviews as part of their contracting language when dealing with subcontractors.

In addition, the scope of these reviews is growing to include far more than just the technical issues. As we have stressed, when dealing with an architecture, business issues are the driving factors in design. By considering the relations between business goals and architecture, and considering them early in the software development (or redevelopment) process, they might be dealt with in a way that provides the greatest benefit for the system's many stakeholders. 

## References

1. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison-Wesley, Reading, Mass., 1998.
2. R. Kazman et al., "SAAM: A Method for Analyzing the Properties of Software Architectures," *Proc. 16th Int'l Conf. Software Eng.*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 81-90.
3. P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley, Reading, Mass., 2001.
4. M. Jackson, *Software Requirements and Specifications*, Addison-Wesley, Reading, Mass., 1995.
5. M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, vol. 15, no. 3, 1976, pp. 182-211.
6. R. Kazman, J. Asundi, and M. Klein, "Quantifying the Costs and Benefits of Architectural Decisions," *Proc. 23rd Int'l Conf. Software Eng.*, IEEE CS Press, Los Alamitos, Calif., 2001, pp. 297-306.
7. J. Henderson and N. Venkatraman, "Strategic Alignment: Leveraging Information Technology for Transforming Organizations," *IBM Systems J.*, vol. 32, no. 1, 1993, pp. 4-16.

For more information on this or any other computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.

## About the Authors



**Rick Kazman** is a senior researcher at the Software Engineering Institute of Carnegie Mellon University and an adjunct professor at the Universities of Waterloo and Toronto. His primary research interests are software engineering (software architecture, design tools, and software visualization), human-computer interaction (particularly interaction with 3D environments), and computational linguistics (information retrieval). He received a BA and M.Math from the University of Waterloo, an MA from York University, and a PhD from Carnegie Mellon University. His book *Software Architecture in Practice* (written with Len Bass and Paul Clements, Addison-Wesley, 1998) received *Software Development Magazine's* Productivity Award. His most recent book is *Evaluating Software Architectures: Methods and Case Studies* (written with Paul Clements and Mark Klein, Addison-Wesley, 2001). Contact him at [kazman@sei.cmu.edu](mailto:kazman@sei.cmu.edu).

**Len Bass** is a senior researcher at the Software Engineering Institute of Carnegie Mellon University. He has written or edited six books and numerous papers in a wide variety of areas of computer science, including software engineering, human-computer interaction, databases, operating systems, and theory of computation. He received his PhD in computer science from Purdue University. Contact him at [ljb@sei.cmu.edu](mailto:ljb@sei.cmu.edu).

