



PACIFIC NW
28TH ANNUAL
SOFTWARE
QUALITY
CONFERENCE

OCTOBER 18TH – 19TH, 2010



ACHIEVING
QUALITY
IN A COMPLEX
ENVIRONMENT

*Conference Paper Excerpt
from the*
**CONFERENCE
PROCEEDINGS**

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Contextually-driven System Architecture Reviews

F. Michael Dedolph

Abstract:

When the World Trade Center collapsed, the switching systems in the basement correctly diagnosed which lines were still working, and continued to connect calls using backup power for several days. One factor contributing to this remarkable product reliability was the AT&T / Bell Labs practice of early systems architecture reviews.

With concerns over systems and software reliability increasing every time you read the paper, some kind of architecture review is a necessity for any organization that wants to minimize liability while producing innovative, high quality products and services.

This paper will:

- Provide a simple model for defining and categorizing systems architecture that is context driven and representationally independent.
- Describe how to conduct an architecture review, using methods based on Lucent/Bell Labs Systems Architecture Review Board (SARB) process.
- Summarize the direct and indirect benefits of SARB-type reviews.
- Discuss how the review method was incorporated into the company's culture.

SARB-style reviews provide an alternative approach to the SEI's Architecture Tradeoff Analysis Method (ATAM) method. Compared to ATAM, SARB-style architecture reviews can be easily and flexibly tailored based on the context. The context for the review is established by the problem statement. The flexibility of the method makes it suitable for many kinds of systems and problem domains.

Background information:

The SARB review process was developed over time with extensive consulting support from Jerry Weinberg. F. Michael Dedolph was a SARB review leader for 7 years at Lucent/ Bell Labs, conducting more than 60 reviews during that time frame. The paper describes the review method as practiced at the time. Since the method is always evolving, current practices may differ.

Biography:

F. Michael Dedolph is currently a technical lead for a software process improvement effort within CSC. His organization recently achieved a CMMI Level 3 rating, and is moving on to Level 4.

From 1997 to 2004, Michael was a systems architecture review leader at Bell Labs (Lucent); he also managed and taught Lucent's Systems Architecture Introduction class. In addition to leading architecture Review Teams, he facilitated numerous risk identification, problem solving, and project retrospective sessions.

Prior to 1997, Michael worked in the Risk and Process programs at the SEI. While at the SEI, he was the technical lead for the teams that developed the SCE and CBA-IPI appraisal methods, and was the team leader for several Risk Reviews.

He started his IT career by spending 10 years as an Air Force computer officer.

What is “Architecture”?

If you want at least three different opinions about what an architecture is, ask any two architects.

Before we can conduct an architecture review, we need to know what we are reviewing. Unfortunately, there is little consensus in the computer and systems engineering fields about what architecture means.

To test this, ask a group of engineers or managers what architecture is, and record the answers. You will probably find design and interfaces mentioned a lot, but beyond that, very little real agreement.

For this discussion, the definition of architecture is that a system architecture provides a **solution** to a **problem** for a **Client**.

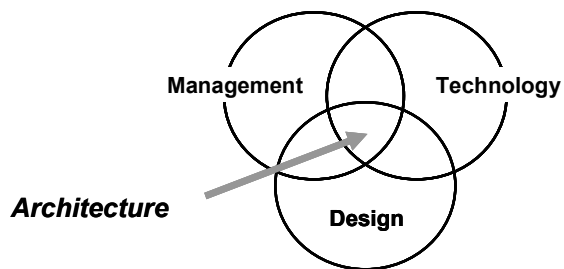
After construction, the architecture becomes the tangible solution you can see, along with a framework of supporting structures and system interfaces which may not be immediately visible. The constraints are part of the environment the system interacts with, and typically include operational performance and cost constraints.

Architecture has been characterized as “design with constraints”. Alternatively, you could say architecture *includes* a design that works *within* the constraints.

Before and during construction, the architecture is a *conceptual* or *potential* solution. During development, constraints also include cost and schedule.

Where can you find the system’s architecture?

Architecture exists at the intersection of management, technology, and design.



Management is concerned with cost, profitability, schedule, preserving legacy investments, customer willingness to pay, future liability . . .

Technology encompasses methods (process), materials, tools, approaches. Technology is constantly changing and evolving.

Design incorporates different views or models of the system for different users, includes interfaces, and, at the lowest level, must be “buildable”. Design concepts are often transferable across many domains and technologies. Because both architecture and design use multiple views, design is often mistaken for the architecture. For purposes of a review, different design views will be used to illuminate different aspects of the architecture, but design alone does not encompass the entire architecture.

Any given architecture is **not** the only solution, but some solutions are “better” than others.

How do we know how “good” an architecture is?

Taking a cue from the earlier question, ask the same group of engineers and managers what a “good” architecture is. This will likely produce even more varied answers, many of which boil down to “I’ll know it when I see it”. Other answers may focus on a particular aspect of the system, such as performance, reliability, or cost.

The key question for architecture reviews is how you can tell if an architecture is “good enough”. To decide if a solution (or proposed solution) is good enough, first someone has to define what the problem is that the architecture is supposed to solve.

Problem Statements

Fundamental to this type of review is the concept that a system architecture provides a **solution** to a **problem** for a **Client**.

The problem statement establishes the context for the review by encapsulating the essential domain information, management concerns, technological challenges, and design constraints from the Client’s point of view. The review method will examine the proposed or actual solution. However, to start the process, the Review Team needs a succinct problem statement that encompasses the problem.

A Framework for Architecture Problem Statements

The System Architecture Review Board (SARB) review method uses a specific framework for problem statements to address these aspects of the problem space:

- **Function** – what the system does, how “well” (fast, reliably, securely) it does it.
- **Form** – what it “looks like”; including the major environmental interfaces. Can be physical or logical form; for SW, includes things like protocols, languages, network environment, user interfaces, etc.
- **Economy** – cost and resource aspects, including development, maintenance, materials, system retirement costs, competitive pricing pressures, etc.
- **Time** – relationship to past, present, and future versions of the system, time to market, schedule.
- **Operational** – this is an extension to the original model. Operational aspects include things that pertain to development constraints and the operating environment, rather than the system itself.

The *function, form, economy, and time* (FFET) aspects have been widely used in civil architecture. Other frameworks exist⁽¹⁾; this one was borrowed from Pena⁽²⁾. The framework was extended to include the *operational* aspects, because these aspects can affect the architecture and its supporting systems.

The different aspects of the problem space are overlapping and interrelated. Many aspects of the system will affect more than one area, and there will always be architectural tradeoffs between the different aspects. One goal of the review process is to make these tradeoffs visible and explicit.

Measurable attributes are desirable, but this may not be possible or practical for early reviews. It is more important is that the criteria can be judged. For example, if system performance will be measured as throughput under peak load, it will not be possible to measure the it until the system is complete. Early on, a performance model will have to suffice.

A quick example of FFET/O from classical civil architecture is the famous Taj Majal. A problem statement for the Taj Majal might have read something like this:

- The **function** of Taj Mahal is to provide a tomb for the Emperor and a monument to the Emperor’s departed wife. It must be so beautiful that people will travel to it from around the world to honor her

memory, and know how much he cared for her. As a tomb for a Muslim Emperor and Empress, it must include a mosque on the grounds, and should provide a space for people to stay when they came to visit and marvel.

- The building **form** and materials will need to maintain symmetry, deal with the unprecedented weight of the structure, manage the shearing force of the nearby river on the foundations, avoid discoloration of the stone and fading of paint, and maintain visual perspective. A verse from the Koran will frame the main arch, and must appear to be written in the same size letters from the ground.
- **Economy** is unimportant. It will cost whatever it costs, and take as long as it takes. Up to 10,000 people can work on the Project, for as long as it takes.
- The structure is to last for all **time**. So the foundation must stand, and discoloration of the stone or fading of paint seen in previous palaces is unacceptable. Construction will take as long as it takes (work started around 1632 and was completed around 1653.)
- **Operationally**, there is insufficient expertise in the kingdom to build this. So the best Hindu and Persian architects will work together on the Project, and will hire anyone they need. Since some of the materials needed to meet the form constraints are heavy, a road will need to be built to transport them. Some valuable materials are needed that are only available in China, so reliable ordering and transportation must be arranged.

Against these criteria, the Taj Majal did pretty well. Construction was completed around 1653, and people still come to visit today, although they no longer stay on the premises. We might question the economic aspects of the problem statement—the eventual cost of the structure bankrupted the Kingdom and led to the overthrow of the Emperor.

Problem Statements for Reviews

The problem statement is the basis for the review—it provides a succinct summary of the critical success criteria. The problem statement includes major constraints, and covers the FFET aspects. Operational aspects may be included if needed.

A problem statement describing the necessary *function, form, economy, time* and *operational* (FFET/O) aspects of a system should:

- Be succinct.
- Provide the critical, discernible success criteria for the solution.
- Be expressed in sufficient detail to make judgments about the proposed solution, **but**,
- Avoid being unnecessarily proscriptive (“thou shalt not”) or prescriptive (“thou shalt”).
- Be client-centric.

Problem statements are **not** a requirements document, but may summarize the most critical high level requirements.

Problem statements also have unstated "always" criteria. These criteria depend on the Client and the type of system, and will include things like:

- It is possible to construct the system (operational).
- The Client can make money or will perceive value (economy).
- The solution won't result in harm (function, form).
- The solution is legal (function, form).

It is not necessary that the problem statement have separate sections for FFET/O, but it is often helpful. The important thing is that all of the aspects are covered in enough detail for the Review Team to use.

In addition to providing the criteria for the review, the problem statement establishes the specific expertise areas needed on the Review Team, and is used to establish the detailed review agenda.

The Architecture Review Problem

One of the things that makes the SARB-style review approach powerful is its flexibility. To a large degree, the flexibility comes from the problem statement—if you can write a problem statement, you can review against it. To illustrate the flexibility of the problem statement concept, here is a problem statement for an Architecture Review method.

=====

Architecture Review Problem Statement:

The review method should increase the probability of success for the Project (function) by addressing FFET/O aspects of the system being reviewed (form). Findings generated by the review need to be unbiased, independent, and relevant for both management and technical staff (function). The findings must be complete enough to alert management to high-risk issues (function), but also framed in a way that promotes project level “buy-in” for resolving the issues (operational, functional, form.)

The method should consider design, technology, and management constraints (form, operational), be representationally independent (form), and flexible enough to work in multiple domains (form, function).

The reviews must be cost effective (economy), and whenever possible, they should leverage existing expertise within the company (economy).

The review method should work at any point in the lifecycle, but, in particular, support early reviews (time, form). 90% or more of the reviews will be conducted in 3 days or less by a small, in-house team of 3-10 people (time, economy).

=====

As we discuss the SARB method, you can review it against this problem statement to see how well the method meets the need. Before proceeding, though, you might ask “Is there anything missing in this problem statement?”, and “Are there any aspects of the problem I don’t understand”? (These questions would typically be addressed in the pre-review meeting early in the process).

Fundamental Architecture Review Method

In a nutshell, the review method is:

- Define the problem the Client wants solved,
- Compare it to the architecture (proposed solution),
- Identify the gaps (or risks).

After the review, let the Project resolve the gaps.

The first activity is done by the Project and Client, possibly with consulting help from the Review Team. The next two activities are done by the Review Team. The project owns resolution of the findings.

Architecture Review Questions

Here are some key questions that need to be answered as the review unfolds.

Questions to answer **before** the review by the Client, project, and Review Team:

- Who decides what the critical success criteria are, and who sets priorities? (Know the Client)

- What problem are we trying to solve? Do both the Project and the Review Team understand the problem?
- Are key stakeholder interests represented?

Questions for the Review Team to answer **during** a review:

- How good is the proposed solution?
- What are the issues/ risks/ gaps in the solution?

Questions for the Project to answer **after** a review:

- Which things will we address?
- How will we address them?

How to Conduct an Architecture Review

Before listing the steps in the steps in the SARB review process, we'll briefly cover the roles in the process and the ground rules for a review

Review Roles

For the Review Team, essential roles in the process are the Review Leader, Angel, and Review Team Member. All reviewers are independent of the Project's reporting chain. At Lucent, the review **Angel** was a SARB Board member.

The **Angel** represents management interests for the company. In this role, the Angel co-facilitates the review activities, helps with recruiting team members, ensures follow up actions are appropriate, reviews the reports, and facilitates transfer of lessons learned to other projects. The **Review Leader** coordinates the review logistics, consults with the Project on the problem statement, recruits Review Team members based on the problem statement, facilitates the review activities, writes or edits the final reports, and coordinates follow up activities. Review Team **Members** bring the specific expertises to the table that are needed to cover all aspects of the problem statement.

For the Project, the **Client** is the person paying for the Project and sponsoring the review. Ultimately, the Client is responsible for the success of the product and project. In this capacity, the Client makes final decisions about what the critical success criteria for the product and project are, and decides how critical tradeoffs will be made if all the criteria can't be met. Typically the Client is at the VP level or higher. Using the Client as the decision maker is an implementation of the corporate "Golden Rule" – the person with the gold makes the rules.

The person designated as the **Architect** assists in preparing the problem statement, and is responsible for arranging the technical presentations used during the review. They typically present the top-level architectural view of the system. The architect usually has responsibility for following up on technical review findings. Architects may be part-time staff on small projects, or a part of an architecture group on a large project. For architecture groups, the lead Architect will coordinate review activities.

Project Management works with the Client and Architect to prepare the problem statement, and presents the problem statement and project overview to the team. They also assign some one to work the review meeting logistics, and are responsible for managing follow up actions after the review.

Other **project participants** include the presenters who have expertise in particular areas. Projects may choose to keep review involvement limited to the manager, architect, and presenters, or may use the review as an educational opportunity for staff members to get up to speed.

Review Ground Rules

Ground rules are covered in the pre-review, and again at the start of the review meeting. These ground rules are based on time-honored review principles established by Weinberg⁽³⁾ and others.

Reviews should be done without attribution—review products, processes, and ideas, not people. Frame issues and observations in terms of the product architecture and the problem it is intended to solve, not in terms of the presenter or architect

The Review Team is there to identify, not solve problems. (Asking engineers to follow this ground rule is a bit like asking cats not to shed on the couch. However, you can train engineers. Ask them to write their on-the-spot ideas as observations or suggestions, and record them on an observation/suggestion card.)

Review and project team members can ask questions at any time. The speaker may defer the answer, and the Review Leader may ask that the question be held until later. But, the basic rule is, ask it.

Review Team members should write any notes, questions or thoughts they have on cards. At the end of each presentation, Team members review their cards to make sure their questions got answered. If not, they should tag the card as an issue, or keep it as a question to follow up on later.

The Client requests and pays for the review (in the sense that the Project team is spending time on the review), but the Project owns the findings and responsibility for correcting them. There is one exception to this rule, the “management alert” (covered later).

In summary, as reviewers, the goal is to do no harm—to paraphrase the House of Blues slogan “help ever, hurt never”.

Review Phases and Steps

The review is broken up into three chronological phases: **Preparation**, **Review Meeting**, and **Follow-up**.

Preparation steps

1. Respond to initial contact/request.
2. Develop the problem statement.
3. Select team and develop agenda based on the problem statement.
4. Arrange logistics – travel, space, phones, connectivity, etc.
5. Hold pre-review call.

Review Meeting steps

1. Review the Project’s architecture presentations, with Q&A .
2. Hold Review Team Caucus.
3. Conduct Readout.
4. Hold Sponsor/Client meeting (optional).

Follow-up steps

1. Write and distribute the Review Report.
2. Present findings to lessons learned/review process governing groups (Board Report).
3. Review the Project’s action plans.
4. Close the review by meeting with the Project team and/or Client.

The discussion that follows covers the review activities, by phase. Not every step is discussed, but some steps or activities are called out in detail because they can be problematic, or because they represent something relatively unique to the method.

Preparation Considerations

Step 1 – Responding to the initial contact/request is essential for setting the right expectations for the review, especially for teams or organizations that have never been part of a SARB-style review.

Step 2 – Client and Project team develop a Problem Statement. The Problem Statement is the basis for the review. It seems like this would be easy, but developing it and getting agreement to it is often the hardest part.

In some cases, projects canceled a review because they didn't have the time to develop a problem statement. (In the author's experience, these projects were not often successful).

In many other cases, the initial review schedule was delayed while the Project and Client worked out the details of the problem statement.

One of the tough parts seems to be making it short. There seems to be great temptation to hand off a big honking requirements document in lieu of a succinct statement that reflects the truly critical success criteria. There is good reason for insisting on a short problem statement—the Project team members are likely to keep the short “vision” for the product in mind, but will only reference the requirements documentation if they need to work with the details.

Step 3 – Team selection is based on the problem statement. If the product has stringent performance, reliability, or security criteria, then you need that expertise on the team. Conversely, if the product doesn't have a critical data base component, don't bring in a data base expert.

To keep the team size small, senior people who can fill in multiple areas are desirable. Over time, the review leaders will become sufficiently rounded to cover (or be the backup) for most areas in the domain, and will recognize common pitfalls. In general, having more than one or two junior people on the team is detrimental.

Step 5 – The Pre-review call is a critical step to ensure that the Project and Review Team members are all in synch. The call serves as “just-in-time” training for new Review Team members, and for Project Team members that have not participated in a review.

During the call, the Project and Review Teams will go over the ground rules and review conduct.

Next, there will be a fairly detailed look at the problem statement. This is the Review Team's chance to ask questions and clarify the problem space, and to identify possible omissions.

- ***What do you mean by ...?***
- ***Did you consider ... ?***
- ***Is X a factor for your system? Should it be in the problem statement?***

For example, in the Architecture Review problem statement, the phrase “using a small, in-house team” might lead to questions like: (1) How small is small?, and (2) Does in-house mean the team can include Project Team members? The answers to these questions could lead to a modification of the problem statement, or to a better understanding of the meaning.

After the review of the problem statement, the team and project will verify that the agenda covers the problem statement topics, and that sufficient time has been allocated to cover the topics.

The last step is to request pre-reading materials, and to arrange for distribution to the team members. During a 3-day review, the Review Team will not get to see all of the details of a reliability model, but the reliability expert on the team can look at the model in advance to frame questions for the review.

Review Meeting Considerations

For SARB reviews, the review meetings were conducted face to face. In rare cases, an experienced reviewer was allowed to participate virtually; this is not as effective.

Virtual reviews are not as effective. This was (and is) increasingly problematic because of travel costs and extensive use of distributed teams.

Project Team member presentations can be done remotely, but a few Project Team representatives should be face to face with the Review Team. Typically the senior Architect and either the PM or deputy would attend the meeting. However, projects should keep in mind that a face-to-face review can have other benefits such as team building and team training that go well beyond the findings.

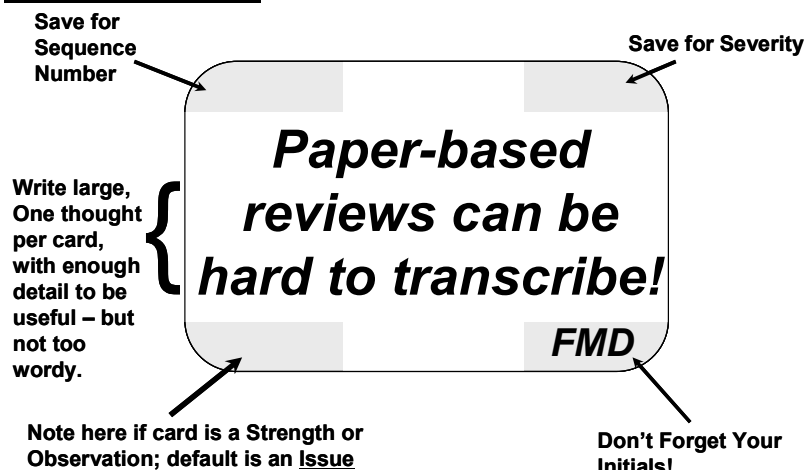
Step 1 – Review the Project’s Architecture Presentations, with Q&A.

Presentations are made by the Project team members in any format. PowerPoint can actually be detrimental and whiteboards can be good. As a rule, project should **not** develop totally new materials for the review, unless there was a major gap in planning (oops, no security model yet, better get a draft ready for the review). Usually existing presentations can be adapted for the review.

Questions can be asked by the Review or Project Team members at any time. The discussion is moderated by the Review Leader or Angel if needed.

During the presentations, strengths, issues, and (optionally) observations or suggestions are recorded. For SARB reviews, specialized index cards called “Snow Cards” are used to keep notes. Both review and project team members can write cards. The cards document findings in real time; editing and analysis is done during the Review Team Caucus.

Snow Card Example



Snow cards were a tradition at AT&T and Lucent, because they are ideal for the purposes of a review. A large index card or Post-It would also work. For some reviews, the paper based snow cards were supplemented by virtual “cards” collected in a data base. New tablet based media would also work.

However, in order to rapidly group, categorize, and prioritize the findings from the data base, the virtual cards had to be printed.

Like any technology or design feature, snow cards represent a trade-off. The visible **benefits** are that paper-based snow cards provide a real-time, unobtrusive way to capture issues and ideas (keyboards can be a major distraction). The cards can be posted on a wall, providing instant visibility into the findings. The media also supports being able to quickly edit, group, and prioritize the issues. Other, less visible benefits of using snow cards (versus laptops) include keeping the team focused on the review presentations rather than a pop-up e-mail, and keeping the findings relatively short. The **drawbacks** to using cards are that they need to be transcribed for the report, they can be overly terse, and they can be hard to read.

Step 2 – Hold Review Team Caucus.

The Review Team Caucus is led by the Review Leader and Angel. Only Review Team members participate in the caucus. During the caucus, the Review Team categorizes, summarizes, and prioritizes the findings. After analysis, the findings are then used to prepare the readout for the Project Team, and from there flow directly into the Review Report.

A “typical” review produces 80–350 observations, arranged into 10-25 topics. **Categorization** is done by the reviewers as a quick group exercise, using affinity grouping. Each affinity group is given a topic heading. The topics are created to provide a consistent “story” for the read-out. Most of the topics will map to the review agenda or to problem statement areas.

Strengths and issues may be categorized separately, or included in each topic area with the issues.

During readout prep, a **summary** of each topic area is written by the Review Team subject matter experts, working alone or in small groups. This can be done by hand on the back of a snow card, transcribed into PowerPoint for the readout, and then re-used for the review report.

Prioritization can be at the level of a topic area, a subset of snow cards within a topic area, or an individual card. Strengths and Observations/Suggestions are not prioritized.

Since reviews point out issues that often lead to architecture changes, it is important to point out the things that are going well. **Strengths** are used to document architectural or project features that should be retained going forward.

The review process de-emphasizes making suggestions, because it gets the Project team into the mode of problem solving. However, some suggestions are inevitable, and they can be useful. For example, a suggestion to “Contact Susan in the XYZ project about their approach to optimizing data base performance for open source data base systems” could be very handy when the Project addresses related issues.

The SARB review process uses these priority categories for issues: **management alert, critical, major, and minor**.

A **management alert** is sent to the Client and/or the Project’s upper management, and represents the one exception to the rule that the Project owns the findings. All other findings go to the Project—critical, major, and minor issues, along with strengths and observations.

Determining that an issue is a management alert requires team consensus. If the Review Team does not reach consensus, the issue is categorized as a critical issue. The criteria for deciding an issue is a management alert is found in the header used to present the issue: **“In the unanimous opinion of the Review Team, the Project will fail unless these issues are immediately addressed:”**

Unlike a management alert, the critical, major, and minor priorities are set by the team member who is the subject matter expert in the area, or by the Team Leader.

The criteria for deciding that an issue is critical are found in the critical issue header: “Unless these issues are addressed, the architecture will not meet all of its success criteria, resulting in significant rework and/or customer dissatisfaction. These issues are seen as complex, and/or will require significant effort to resolve.” Major and minor issues represent a partial failure to meet the Projects success criteria, but are typically seen as easier to fix.

Step 3 – Conduct Readout

The readout shows the Project what the most important issues are, so they can get started, and so there are no surprises in the Review Report. Upper management (e.g., the Client) is invited by the project, only if the Project wants to include them in the readout. Readouts use a PowerPoint presentation to summarize the snow cards, supplemented by the snow cards themselves.

The Review Team will **not** withdraw a finding or lower the priority. Since the Project owns the findings and responsibility for resolving them, if the Review Team over-reacted, the Project can adjust when they do action planning.

However, an issue's priority may be increased at project request. In a few cases, project teams explicitly asked that a critical issues be turned into management alert issues, because they had been trying to resolve the issue and needed upper management support.

In some cases, the Client/Sponsor will request a supplementary meeting to provide a private forum for discussion.

Follow Up Considerations

The readout and Client meeting are the last steps in the review meeting, but can also be thought of as the first steps in follow up. A copy of the readout is provided to the Project so they can immediately begin action planning to resolve the issues.

Management Alert and Critical Letters are sent within a week. Management alerts go to the Project's upper management and/or the Client (usually the Client is part of the Project's upper management chain). The Project's upper management is responsible for the action plan. Most management alerts are related to cost, schedule, and resource issues. Only rarely will a management alert be purely technical in nature.

The Critical Issue letter goes to the Project, and Project management is responsible for the action plan

The review report includes a transcription of all snow cards, along with summaries for all the issues. It provides the Project with a level of detail to support action planning, including the suggestions from the team. Reports should be completed and sent within 2 weeks.

If there is a neutral oversight organization, a separate Board Report will be generated to provide a corporate-level summary and to capture lessons learned.

For management alerts and critical issues, there is a specified time window for the responsible manager to prepare an action plan. Action plans are **not** required for major and minor issues. Although the Review Team has no ability to enforce the action plans, the plans are reviewed by the Review Team, and feedback is provided to the Project. **Closure meetings** with project and/or sponsor are used to provide feedback to the Project after the action plans are reviewed.

Project Ownership of the Findings

Project ownership of the findings is a key feature of this type of review. Project responses to a review range from putting the findings in a drawer and refusing to participate in action planning or any closure activities, to posting all of the findings on the Project's web page along with the action plan. (Guess which was the more successful approach).

Most projects do **not** address all of the findings – there simply isn't time, and the snapshot taken by the Review Team gives way to new challenges as the development activities unfold.

Given that addressing findings is at the discretion of the Project, what happens:

- When a project ignores a Management Alert or Critical issue?
- When a project fixes everything?
- When the Review Team identifies an issue that turns out to **not** be a problem ("false positives")?

Ignoring Issues

Experience shows that ignoring a management alert or critical issue is not a good path for the Project to follow. Without naming the projects, here are two examples.

One project tolerated an architecture review they didn't want, then used their prerogative to ignore several management alert and critical issues. The project later became a textbook example used in internal classes about what not to do, while losing the company a **lot** of money (not millions, think big).

A more typical and prosaic example saw a project neglecting a critical performance issue because it was hard to resolve, and the schedule was tight. The issue identified a mismatch in interface as a potential performance bottleneck. Ultimately, the system hardware met the specifications, but the software protocol mismatch caused the equipment to handle only 50% of what was required. This meant the customer had to buy twice as much equipment to achieve the performance they wanted, and/or the product price had to be discounted sharply. The result—major dissatisfaction for everyone.

Fixing Everything

As to fixing everything, the author never witnessed it. At least one proactive project had annual reviews for each major release, posted the findings on the Project web site along side the action plan, and worked off all of the Management Alert and Critical issues. The product continued to improve over time.

The stellar reliability and performance of the 5ESS switch was mentioned in the abstract. These results were due in large part to a culture of reviewing every major change, and resolving the issues that were identified. In addition to architecture reviews, extensive use of software peer reviews and several phases of testing contributed to the product's legendary reliability. As part of every architecture review, the problem statement addressed performance and reliability issues. By the way, this example shows that the utility of reviews is **not** limited to new products—in fact, it may be more difficult to act on review issues for new products may because of market time pressures.

Issues that Aren't Really Issues

Identifying issues that turn out **not** to be a problem is not really a disadvantage of the method, if you take a risk management perspective. A “false positive” forces the Project to do due diligence on the solution; the Project's action plan may simply state: “Here is the issue, we looked into it, and our original approach is the best available option at this time.”

Major Benefits of this Type of Review

SARB review sponsors included people who were clients for reviews—project managers, directors and VPs that had come to see the value of the process.

A survey sent to the SARB review sponsors ⁽⁴⁾, ⁽⁵⁾ found that reviews had the following effects—projects avoided problems, identified issues, and as a result, saved money.

Savings from risk and issue avoidance are hard to quantify, but it was estimated that the savings averaged more than \$1M per large project, especially if reviews are done early⁽⁴⁾. Similar estimates were derived from internal Monte Carlo simulations based on analyses of architecture review findings⁽⁴⁾.

In some cases, there were direct results attributable to a specific review. One sponsor told his team to re-engineer the product after a review. The resulting architectural changes trimmed 9 months off the development schedule and saved more than \$7 million.

Sponsors also noted less tangible benefits, beginning with the review preparation. Activities such as preparing the problem statement forced people to think through the problem, and communicate their thinking with other team members and management.

Sponsors also noted cross-pollination of techniques and practices across the larger organization. This was echoed by Review Team members, who enthusiastically cited how much they learned while participating.

Other benefits noted by some sponsors included synching up Project Team members with work others on the Project are doing—getting everyone “on the same page”.

Another, longer range benefit was the ability to focus and refine a set of key success factors for a product family over time, e.g., reliability improvements in the 5ESS switch.

Potential Shortfalls of this Type of Review

SARB-style reviews are a potent tool for improving systems and software quality. However, like any other technology or method, there are trade-offs that can represent drawbacks.

Risks of the Client Centered Approach

There is a fundamental risk inherent in the client-centered approach. The advantage is that responsibility for decision making is clear, and the funding source is plugged into resolving the issues, which minimizes politics. However, for a variety of reasons the Client can grow out of touch with the market, customer needs, and end user issues. Making sure that alternative points of view are heard by the Client can be a problem. If these issues are apparent, the Review Team (particularly the Angel) can raise the issues when the problem statement is being clarified.

In government contracting, many of the success criteria are negotiated into the contract, and represent political compromises rather than a specific, knowledgeable decision maker’s opinion. The government contract officer is the closet person to a “Client”, but typically is a contracting specialist rather than a knowledgeable system user or engineer. The contract officer may not be willing or able to make decisions that contradict the “black and white” of the contract.

Incomplete Findings and Level of Detail

The review findings are inherently incomplete. Any finding identified early is a plus for the Project, but the issues found in a review will always be missing something—hopefully something with a lower impact on project success. This is true of any method, but provides a cautionary note for using SARB-style reviews as the only mechanism for identifying issues.

Although the problem statement can be focused on particular, very detailed aspects of the system, the review technique is best used when the goal is to make sure the whole architecture hangs together. But, a high level review is not suited for all problem spaces, and 2 to 3 days will **not** flush out all of the “devils in the details”.

For example, a high level review may identify that the performance model for a subsystem is incomplete because it omits a major interface, but is unlikely to identify a specific problematic transaction for the system. The rework done by the Project to update the model at the detailed level is much more likely to catch the problem transaction, so the review can facilitate correcting the details, even if they are not caught in the review.

Resistance to Face to Face Meetings

There is growing resistance to corporate travel, so it is increasingly difficult to arrange the face to face environment where SARB-style reviews are most effective.

What's Different (or the Same) About this Kind of Review?

In common with other review methods such as the SEI’s ATAM⁽¹⁾, SARB-style reviews bring impartial, outside eyes into the Project by using a team that is external to the development effort. Like other walkthroughs and inspections, products, processes, and ideas are reviewed, not people. Also, the review is **not** a problem solving session or fact-finding audit.

SARB-style reviews differ from other review methods in the following ways:

- **The method emphasizes problem and solution congruence**, rather than adherence to a particular notational standard or concept of what an architecture should look like. For example, a system may be described as a Client-Server or Service-Oriented Architecture, but the system as defined may not solve the real problem.
- The **Client has ownership** of the decision making process.
- The method is **context and domain independent**—the problem statement is used to adjust the review to the domain.
- The method is also **notationally and representationally independent**, not model or standard based, although these aspects can be included in the problem statement, if needed. In other words, you don't have to do an object oriented design if a state machine or transaction model is a better fit.
- Follow up includes a **clear statement of potential failure areas**, with a prioritized hierarchy of findings.
- **The Project sees and owns the findings**. The Project can throw them away, or post their action plan to their web site. There is balanced but extensive **project participation** during the review, and the Project Team is trusted to be responsible stewards of problem resolution efforts afterwards. All of this helps to build buy-in.
- The role of **Review Angel** is unique, and helps the Review Team focus on providing an unbiased look at the technical and management aspects of the architecture without trying to deal with any political fallout.

In summary, SARB-style reviews balance a defined review process with extreme flexibility.

SARB-style Reviews and ATAM

In the Author's opinion, SARB-style reviews are more flexible because of the use of the problem statement, and the reliance of ATAM on particular architectural representations. The free-wheeling nature of SARB is better suited for early concept exploration, and SARB-style reviews work for various kinds of systems and domains—hardware and networks, as well as software.

SARB reviews take less time, and hence they probably cost less.

ATAM reviews provide more structured outputs that can be revisited systematically over the life of the Project. In particular, the ATAM focus on identifying explicit risk trade-off points for the software architecture provides a way of evaluating the impact of changes over time without re-reviewing the entire system.

There is some convergence—both SARB-style reviews and ATAM can be used for risk identification and mitigation⁽⁶⁾. Also, recent changes/additions to the SEI's Architecture methods have focused on defining architectural attributes that are explicitly related to business goals⁽⁷⁾, which is reminiscent of the concept of a problem statement. The SEI also recently extended the ATAM method to include systems as well as software⁽⁸⁾.

In fact there is room for both—they could be used as complimentary techniques. For example, maximum benefits might ensue if an early SARB-style review at the systems level was followed by an ATAM review of the software after the high-level design was complete.

Incorporating reviews into the organization's culture

To close, here are some notes on how the process became part of the corporate culture. These notes reflect the author's experience at Lucent, and probably do not reflect the current environment.

Architecture reviews were embraced enthusiastically by some parts of the company, and rejected by others. The same pattern held with companies acquired by Lucent—some embraced the reviews, others fled from them.

Initially, a particular Vice President who was close to retirement adapted the SARB review methodology, and promoted it as his legacy to the company. He was a relentless champion of the method and paid to establish a core group of review leaders, and, with the advice of a consultant, worked to establish the SARB Board.

The SARB Board provides oversight of the method. Essentially the SARB Board is a large, distributed group of review sponsors. Board members have multiple roles—they serve as review Angels, act as Clients for their own projects, promote the method, share lessons learned, and provide team members and travel funds to support reviews on a quid-pro-quo basis.

The initial funding model for SARB was central funding by one VP, which evolved into corporate level central funding by the Chief Technical Officer. As times got tight in the Telecomm industry, the funding model changed to funding by Sponsors. Sponsors set aside a portion of their budgets to pay for a certain number of reviews in their organization each year; this money was used to retain a core team. Excess capacity in the core team was used to conduct a few other reviews for high visibility projects. Other reviews were funded on an as-requested basis by the requester. Typical cost of a requested review was in the \$40-70 K range (cheap compared to the potential \$1M savings).

Project ownership of findings coupled with senior management insight into critical problem areas has proven to be a good basis for project's wanting to participate, providing a "bottoms up" push for reviews in some groups.

As a strategy for other organizations, the following steps may serve as a starting point:

1. Find a high-level champion.
2. Obtain central funding for an extended period of time – say 5 years.
3. Train a core team of leaders and angels to do reviews.
4. Charter a board; recruit advocates.
5. Perform the reviews; capture data and lessons learned to document the method's value.
6. Move to a sponsor-based funding model after the initial time period, when the method has proven itself to the Project teams and sponsors.
7. Retain the focus on client-centered problem statements, project ownership of findings, limited but pertinent management alerts, and non-attribution.

References:

1. CMU/SEI-2000-TR-004, "ATAM: Method for Architecture Evaluation"; Kazman, Klein, Clements, 2000
2. Problem Seeking: An Architectural Programming Primer, 4th Ed; Pena and Parshall, 2001, ISBN 0-913962-87-2
3. Handbook of Walkthroughs, Inspections, and Technical Reviews; Freedman and Weinberg, 1990, ISBN 0-932633-19-6
4. IEEE Software, March/April 2005, "Architecture Reviews: Practice and Experience"; Maranzano, Rozsypal, Zimmerman, Warnken, Wirth, and Weiss
5. STQE Jul/Aug 2002 (Vol. 4, Issue 4) Feature: Measurement & Analysis "A Blueprint for Success: Implementing an architectural review system"; Daniel Starr, Gus Zimmerman
6. CMU/SEI-2006-TR-012, "Risk Themes Discovered Through Architecture Evaluations"; Bass, Nord, Wood, Zubrow; 2006
7. CMU/SEI-2010-TN-018, "Relating Business Goals to Architecturally Significant Requirements for Software Systems"; Clements, Bass; 2010
8. CMU/SEI Webinar, "SoS Architecture Evaluation and Quality Attribute Specification (Webinar)"; Gagliardi; 2010