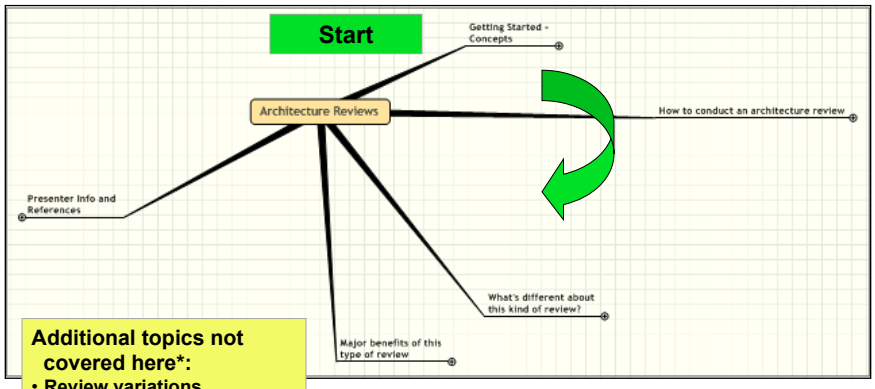
Contextually Driven Architecture Reviews -

Ensuring Quality Before Construction

for SATURN 2009
F. Michael Dedolph
fmdedolph@netscape.net

May 7, 2009

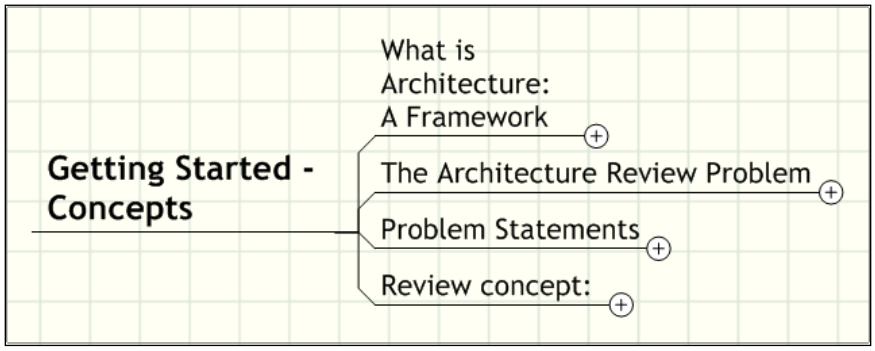
Architecture Reviews



Additional topics not covered here*:
- **Review variations**
- **How reviews can be institutionalized**

*** See me if you are interested in discussing them**

Getting Started - Concepts



What is “Architecture”?

> Before we can review it, we should know what “it” is.

"A system architecture . . ."

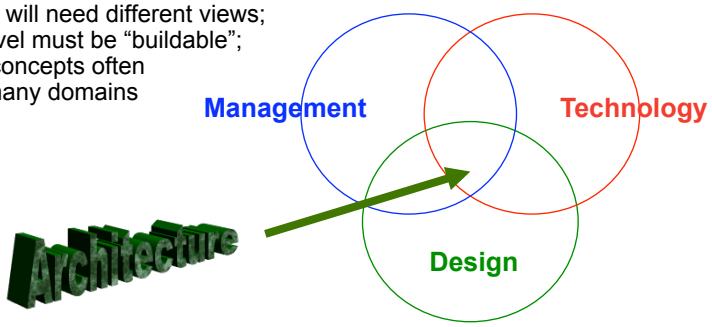
- **Provides a <u>solution</u> to a <u>problem</u> for a <u>client.</u>**
  - For early reviews, it is a *conceptual* or *potential* solution.

- Architecture has been characterized as "design with constraints"
  - Alternatively, you could say architecture *includes* a design that works *within* the constraints
  - **Constraints include cost and schedule!**

> ***Any given architecture is NOT the ONLY solution, but some solutions are "better" than others.***

Where Architecture Exists

- Architecture exists at the intersection of management, technology, and design.
  - **<u>Management</u>** – cost, profitability, schedule, preserving legacy investments, . . .
  - **<u>Technology</u>** – methods (process), materials, tools, approaches.
  - **<u>Design</u>** – will need different views; lowest-level must be "buildable"; designs concepts often work in many domains



Classical Architecture -

A Framework for Architecture Problem Statements

- Architecture is the solution to a problem for a client. The problem has different aspects. These aspects are borrowed from civil architecture:
  - **Function** – what it does, how well it does it.
  - **Form** – what it "looks like"; includes major environmental interfaces. Can be physical or logical form; for SW, includes things like protocols, languages, . . .
  - **Economy** – cost aspects, including development, maintenance cost, material cost, system retirement, etc.
  - **Time** – relationship to past, present, and future.
  - **Operational/ Developmental** – Things that pertain to development constraints and system operating environment, rather than the system itself

- FFET/O should
  - Provide the critical, discernible success criteria
  - Be expressed in sufficient detail to be used to make judgments about the proposed solution, BUT,
  - Avoid being unnecessarily proscriptive (thou shalt not) or prescriptive (thou shalt)

The Architecture Review Problem

Develop and deploy a review method that:

- **Produces relevant findings** for both management and technical staff (function)
- **Increases P (success)** for the project (function)
- **Addresses** design, technology, and management **constraints** (form, operational)
- **Representationally independent** (form)
- **Flexible** enough to work in multiple domains (form)
- **Cost effective** (economy)
- **Leverages existing expertise** (economy)
- Works at any point in the lifecycle, but, in particular, **supports early reviews** (time, form)
- Can be **conducted in 3 days or less** (time, economy)
- **Promotes "buy-in"** for resolving issues (operational, functional)

Architecture Review Questions

> *Question to answer **before** a review:*
- *Who decides? (Know the client)*
- *What problem are we trying to solve?*
- *Are key stakeholder interests represented?*

> *Questions to answer **during** a review:*
- *How good is the proposed solution?*
- *What are the issues/ risks/ gaps in the solution?*

> *Questions to answer **after** a review:*
- *Which things will we address?*
- *How will we address them?*

Problem Statements

Are the basis for the review.

Provide a succinct summary of **critical success criteria**
- Short: 2 pages is good
- Includes major constraints
- Client-centric
- Cover Function, Form, Economy, Time, and (optionally) Operational

Problem Statements Are NOT
- A requirements document, but may summarize the most critical high level requirements

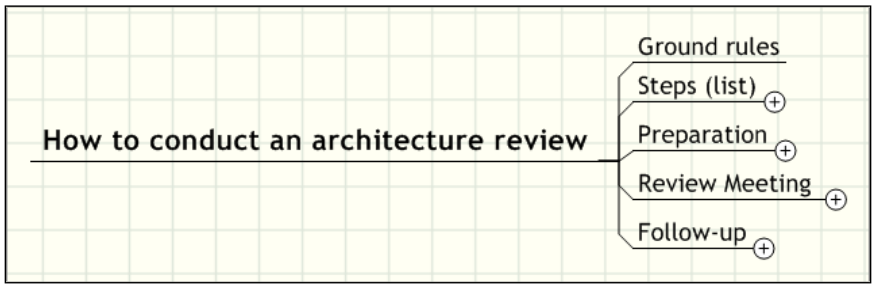
Problem statements also have unstated "always" criteria, e.g.,:
- Possible to construct
- Can make money/ will provide value
- Solution won't result in harm
- Legal

Fundamental Architecture Review Method

- **Define** the **problem** the client wants solved
- **Compare** it to the **architecture** (proposed solution)
- **Identify** the **gaps** (or risks)

After:
- Let the *project* **resolve** the **gaps**.

How to Conduct an Architecture Review



Review Ground Rules

- **Ground rules are covered in the pre-review, and again at the start of the review meeting.**
  - No 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 the presenter or architect
    - **Do no harm: "Help ever, hurt never"**
  - Team is there to identify, not solve problems
    - If you must, write an observation/suggestion card
  - Client requests (and pays) for the review, **but**, the **project** owns the findings and responsibility for correcting them
    - **One exception, the "management alert", covered later**
  - Ask questions at any time. (The speaker may defer the answer.)
  - Write any notes, questions or thoughts you have on index cards.
  - At the end of the presentation, review the index cards to make sure your questions got answered.
    - If not, make sure to record it as an issue.

Review Steps (list)

- **Preparation**
  1. Respond to **initial contact/request**
  2. Develop **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**
  6. Review **Project presentations**, with Q&A
  7. Hold Team **Caucus**
  8. Conduct **Readout**
  9. Hold **Sponsor/client meeting** (optional)
- **Follow-up**
  10. Write **Report**
  11. Present findings to lessons learned/review process governing groups (**Board report**)
  12. Review **action plans** by review team angel, lead, and members
  13. Review **closed** by meeting with project and/or sponsor

Preparation Notes

Step 2 – Client and project team develop a Problem Statement

- The Problem Statement is the basis for the review; developing it is often the hardest part
  - I've seen projects cancel a review because they didn't have time to develop a problem statement,
  - In many cases, the initial review schedule was delayed while the project and client worked out the problem statement.

Step 3 – Team Selection and Roles

- Depends on the problem statement. Key roles are Team Leader (critical), "Angel" (critical).
  - Team members (SMEs)

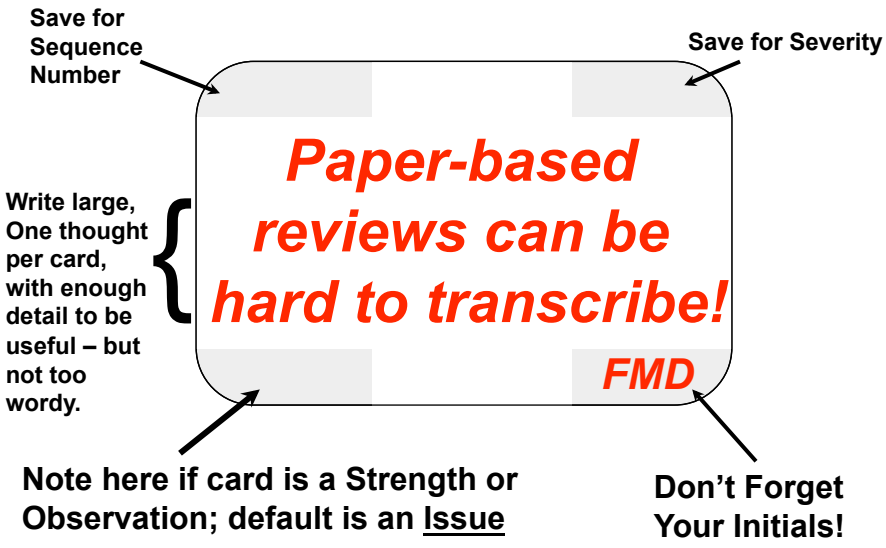
Step 5 – Pre-review call

- Discuss Review plan: agenda, logistics (travel, space, phones, connectivity, etc), schedule, outputs
  - Discuss Review ground rules and conduct
  - **Discuss Problem Statement - in detail**
    - ***What do you mean by ...?***
  - Verify that agenda covers problem statement, check times
  - Assign Pre-reading

Review Meeting - Overview

- **Presentations –** by the project team, any format.
  - PowerPoint can actually be detrimental; whiteboards can be good.
  - As a rule, project should NOT develop new materials.
- **Questions** by the review team, at any time, moderated by the review Leader or Angel if needed.
- **Strengths, Issues, Observations (optional) recorded**
  - Index Cards ("Snow Cards") are used to keep notes
  - Both review team and project team members can write cards
- **Caucus**
  - Categorize, Summarize, and Prioritize the findings
  - Prepare the readout
- **Readout –** management typically invited, uses the Snow Cards and/or PowerPoint.
- **(Optional) Sponsor/ Client meeting** used, if requested to provide a private forum for discussion.

Snow Card Example - 1



Review Meeting - Caucus

- **Led by review Leader and Angel.**
- **Categorization** is initially done by reviewers, topics are created to provide a consistent "story" for the read-out.
  - Use affinity grouping; many topics will map to initial agenda or problem statement topics.
  - Strengths may be kept separate.
  - **A "typical" review produces 80–350 observations, arranged into 10-25 topics**
- **Prioritization**: "Management Alert", Critical, Major, Minor, Strengths, (optionally, Suggestions/ Observations)
- **Summary** of each topic area written by team subject matter experts

Prioritization

- **Prioritization** can be at the level of a topic area, a set of snow cards, or an individual card.
  - A "**Management Alert**" goes to client/ upper management
  - **Critical**, **Major**, **Minor**, **Strengths**, and (optional) **Suggestions/ Observations** go to the project
- **Management Alert header:**
  - **"In the unanimous opinion of the Review Team, the project will fail unless these issues are immediately addressed:"**
- **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:"
- **Readout** shows the project what the most important issues are, so they can get started, and so there are no surprises in the report
  - The review team will **NOT** withdraw a finding or lower the priority
    - Project owns the findings and responsibility for resolving them; if the review team over-reacted, the project can adjust when they do action planning
    - Priority may be increased at project request

Follow-up

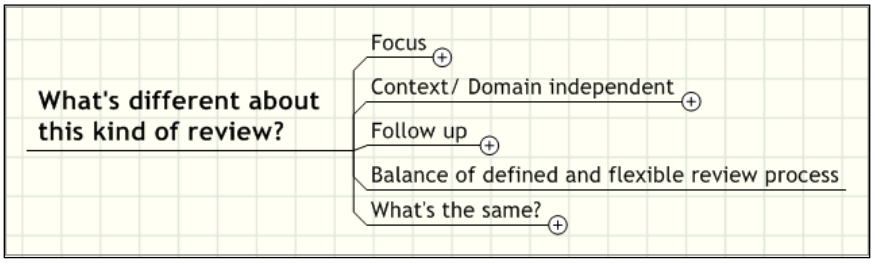
- **Readout and (Optional) Sponsor/ Client meeting** are the last steps in the Review meeting and/or first steps in follow up.
- **Management Alert and Critical Letters within a week**
  - **Management Alerts go to upper management:**
    - **Upper management is responsible for the action plan**
    - **Most management alerts are related to cost, schedule, and resource issues, NOT purely technical issues.**
  - **Critical Issue letter goes to the Project**
    - **Project management is responsible for the action plan**

- **Report** – includes a transcription of all snow cards, and the summaries for all MA, Critical, major, and Minor issues.
- **Board Report –** corporate level summary to capture lessons (if there is a neutral oversight organization)
- **Review of action plans** by review team angel, lead, and members
- **Closure meetings** with project and/or sponsor

What Happens When?

- . . . a project ignores a Management Alert?
- . . . a project ignores a Critical issue?
- . . . a project fixes everything?
- . . . the review team identifies an issue/risk that turns out to not be a problem (“false positives”)?

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



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

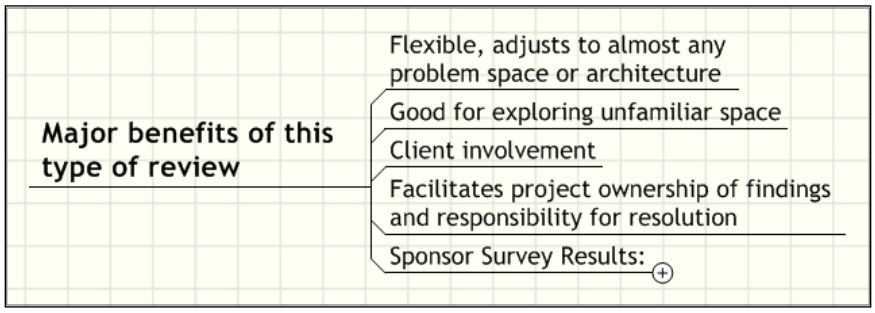
What's **different** about this kind of review?

- **Focus**: problem/solution congruence, Client ownership, participation of project team
- **Context/ Domain independent:** Notation and representation independent, not model or standard based (although this can be included in the problem statement, if needed)
- **Follow up:** Clear statement of potential failure areas, hierarchy of findings (client sees only essential findings, even sometimes none.
  - Project sees and owns findings – can throw them away, or post their action plan on their web site.
  - Role of "Angel"
- **Balances a defined review process with extreme flexibility.**

What's the **same** about this kind of review?

- Outside eyes—external team; Products, processes, ideas are reviewed, not people; Not a problem solving session

Major benefits of this type of review



Sponsor Survey - Benefits

Avoided problems, **identified issues, and as a result:**

> **Saved money**. Estimated at > $1M per large project, especially if reviews are done early; this estimate comes from the IEEE article referenced at the end of the presentation.

> Similar estimates were derived from internal Monte Carlo simulations based on analyses of architecture findings.

Sponsors also noted less tangible benefits including:

- **Preparation** (especially preparing the problem statement) forces people to think through the problem
- **Cross-pollination** of techniques across the larger organization
- **Learning** as a review team member
- **Synching up** project team members with work others on the project are doing; getting everyone "on the same page"

Another long range benefit was the ability to focus and refine a set of key success factors for a product family – e.g., reliability.

SARB-like Reviews and ATAM

Author's opinion:

- **SARB-like reviews are more flexible**
  - **Better suited for early concept exploration**
  - **Less dependence on putting the architecture into a particular format**
  - **Works for various kinds of systems/domains—HW as well as SW**
- **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**
- **Room for both – complimentary techniques**
  - **An early SARB-like review followed by an ATAM review of the software after the high-level design was complete would provide maximum benefits**

Questions???

Presenter Info

F. Michael Dedolph
301-429-7953
fmdedolph@netscape.net

BIO:

F. Michael Dedolph is currently the technical lead for a CMMI-based software process improvement effort within CSC.

From 1997 to 2004, Michael was a SARB review leader at Lucent, and he also managed and taught Lucent's Systems Architecture Introduction class.

Michael worked at the SEI for almost 5 years in the Risk and Process programs. 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.

Abstract

When the World Trade Center collapsed, 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 Bell Labs practice of early architecture reviews.

The SEI's Architecture Tradeoff Analysis Method (ATAM) provides a standardized technique for evaluating software architecture. The review method presented here provides an alternative (and complementary) approach to architecture reviews that can be flexibly tailored based on the context for many kinds of systems, and used in any problem domain.

In this session, the instructor will:
• provide a simple model for defining and categorizing systems architecture that is representationally independent
• describe how to conduct an architecture review, using methods based on Bell Labs Systems Architecture Review Board (SARB) process
• discuss how the SARB methods can complement and supplement ATAM reviews.

Background information:
The Bell Labs SARB 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, and managed and taught Lucent's Systems Architecture Introduction class.

References:

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