Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Architecture Evolution Working Session

SATURN 2006 Working Session

Software Engineering Institute

DRAFT – Work in Progress

This summary is meant to convey preliminary ideas for the purpose of getting feedback. It does not necessarily represent the consensus of the members of the session.

Sponsored by the U.S. Department of Defense

Carnegie Mellon
Software Engineering Institute

Goal

Provide an effective, integrated, widely applicable, and tailorable set of life-cycle architectural practices and tools to help an architect keep a software architecture in line with its goals as the system evolves.

Challenges

- understand where we are - readily determine the current state of the architectural design and evolve from that position.
- find the appropriate practice or tool that can take us where we need to go - this may involve the need to flexibly tailor and integrate practices with each other and to understand the appropriate fit with other architectural processes and technologies.

Carnegie Mellon
Software Engineering Institute

Objectives of the Working Session

Elicit situations from the participants where they have evolved the architecture.

Identify existing practices and tools that are used to solve the problem.

Identify gaps in existing practices and tools that are used to solve the problem.

Review opportunities for further work.

Carnegie Mellon
Software Engineering Institute

Architecture Evolution

Where are you?

- what is the current state of the architecture and the forces that impact the architecture
- explore evaluation, reconstruction, and documentation techniques to understand the architecture

Where do you want to go?

- what new business opportunities do you want to fulfill, what risks need to be mitigated
- explore real options and utility techniques to understand alternatives and to characterize the benefits of qualities

How do you get there?

- how do we change the system, what architectural strategies provide the most benefit given the cost
- explore design methods to transform the architecture and cost benefit analysis to choose alternatives

Carnegie Mellon
Software Engineering Institute

Anticipated vs. Unanticipated Evolution

It seems that architecture evolution is not a problem if evolution was anticipated and architecture was prepared for it.

Evolution becomes a “simple” change

Carnegie Mellon
Software Engineering Institute

Unanticipated Evolution

Hard problem is having to evolve an architecture to accommodate unanticipated changes.

How was it discovered?

- ATAM
- Integration
- Obvious (new business strategy, new technology, new feature, putting in the field)

Why did it happen?

- Short term thinking (Let's work for the next version)
- Expertise (Where not able to anticipate likely changes)
- Run out of time/money

Carnegie Mellon
Software Engineering Institute

Avoid Evolution?

So, always think about everything that can happen and prepare the architecture?
Would that make the evolution problem disappear?

Would be nice, but:

- Time and cost limitations
- Risk of over-engineering – introduction of too much overhead or “analysis paralysis”
- Reality – the things that happen that nobody thought could happen

Carnegie Mellon
Software Engineering Institute

What to do if evolution occurs

Understand the reasons why it was not anticipated

- Expertise?
- Technology?
- Time?

Understand the scope

- Micro vs. Macro – one product involved or multiple products
- Use of scenarios to reason about a change
- Have a process for decisions, such as a Change Control Board (CCB) or Architecture Review Board (ARB) – changing the architecture is costly, don't make ad hoc decisions!

Carnegie Mellon
Software Engineering Institute

What to do if evolution occurs

Decide what to do

- Architectural – redesign
  - Have strategy for what to do – not just go and fix it! Define how to do it.
  - Make sure everything else is still as expected ("Architecture Regression Test" using the scenarios)
- Context in which the architecture lives
  - Increase expertise (training / new hires) - Convincing people to do something different can be a challenge
  - Infrastructure in place for evolution – can the organization deal with the consequences of an architectural change
- Time frame – strong opinion for incremental change. Don't try to do everything in one step. Pick your battle!

Carnegie Mellon
Software Engineering Institute

Take Away

Do as much as appropriate to avoid unanticipated change

Use architecture evaluations, such as ATAM®, to discover discrepancies

Use Quality Attribute Scenarios to guide the evolution