



Carnegie Mellon University
Software Engineering Institute

Experiences in Architecture Based Development

Dennis B. Smith,
Session Chair
March 5, 1999

**Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890**

Sponsored by the U.S. Department of Defense





Purpose

- Identify key issues that have emerged from experiences with architecture based development
- Discuss approaches for addressing these issues
- Develop a potential research agenda for next steps



Panel Members

Morris Brill, TRW

Kris Christianson, Joint National Test Facility (TRW)

James Hager, Penn State, SRI

John Ohlinger, National Reconnaissance Office

Jaap Schekkerman, Cap Gemini Netherlands



Template for Discussion

Describe the problem

- why it is important
- what are the implications

Current approaches to addressing the problem

Unanswered questions

Next steps



Candidate Issues: 1

Architectural beauty

Measure a good architecture (how often were components/connectors used)

What is a good architecture

Nobody wants go give up an asset (eg legacy code)

Concept of operations

Measurement

Organizational cooperation

Enterprise management

Integrate legacy with new code

Smart abstractions



Candidate Issues: 2

Use of CORBA/middleware

Use of standards

Use of COTS

Representation (immaturity, e.g. UML)

Management procurement issues

Overcoming stove pipes

Convince corporation to use assets

Test issues

How to get buy-in

Articulation of drivers of architecture (performance, maintenance, time, self-healing, availability, coupling/cohesion, role and function of actor,

Understandability



Candidate Issues:3

Architecture is not detailed design (de-emphasize systems engineering)

How to enforce architecture

Migration of legacy

How to achieve interoperability (mismatch issue)

What makes system architectural based

Trade-off analysis

Be realistic (intermediate milestones)

Level of detail

Representation

Necessity of hierarchy

Design integrity

Overall picture and view (but picture is not the architecture)



Issues Selected

1. What is a good architecture?

Or: What makes an architecture ugly?

2. Legacy code and architecture



What is a Good Architecture: Overall Situation

Architectural goodness depends on goals of system

Drivers need to reflect business goals

A good architecture will

- reduce complexity
- satisfy stated and unstated requirements
- reflect views of different stakeholders



Ugly Architectures: 1

Monolithic, tightly coupled systems

Closed and proprietary systems

Architectures driven by edicts

- Inappropriate use of COTS and middleware
- Use of standards such as TAFIM to simply check off a box
- Schedule pressures resulting in tossing out discipline

Poorly designed information architectures

Lack of a strong concept of operations can result in users building workarounds



Ugly Architectures: 2

Systems patterned according to organizational structure

Lack of a strong process resulting in a disconnect between architects and programmers

Decisions made solely in terms of functionality leading to a neglect of quality attributes

No architectural specification – work only from requirements

Lack of a big picture – everybody sees only the trees



Unmet Needs

Better capturing of views of different stakeholders

Narrowing the gap between requirements and architecture

- tie down very general requirements

Moving from top-level pictures which look nice to corresponding low-level specifications

Architecting is also about process enforcement: making sure developers follow architecture

COTS and their views and assumptions about the world

- need to overcome COTS mismatch



Promising Approaches

Architectural Tradeoff Analysis to identify tradeoffs between competing quality attributes

Win-win to quantify stakeholder win conditions

Templates for better concepts of operations

Stronger techniques of representation



Legacy Systems and Architectures

For a new mission, we can't displace the legacy system – need to continue working with it

Cost of maintaining legacy code can be higher than the cost for replacing it

In legacy systems there is often not a top level view – only detailed views

- lack of an understanding of how a system fits together

Technology changes (hardware, COTS, middleware) force rehosting and migration decisions

- when to rehost
- when to replace home-grown code
- what happens when new products don't match current API's



Current Approaches Within Domain

Satellites must deal with 1960s technology. Options include:

- Abstract the interfaces to those of new system
- Wait until satellite runs out of power
- Maintain old system with legacy coders (e.g., Jovial)
- Take proven pieces of code, wrap them and reuse them
- Complex filters and wrappers
- Rehost old system
- Run systems side by side and gradually replace old system components



Problems and Unanswered Questions Within Domain

How to not compromise quality of code that worked

How to make decisions on whether to port old code

Need for better total cost data

RFP's can mandate poor technical choices

RFPs may ask for a fixed price on systems that are unknowable

Decisions are made at wrong level, for wrong reasons



Problems and Unanswered Questions Within Domain

Multiple O&M contractors can get in each other's way

Purchase of COTS may be too risky and have too many unknowns

Contractors are often not permitted to touch legacy code

Architecture to support its legacy

Implications of legacy on software developers (e.g. old programming languages can be career killer; need for incentives?)



Legacy System Needs

Decision models

Total cost data (eg – cost of recruiting/keeping Jovial programmers)

Better empirical models of costs of legacy systems; they can be unbounded

Better data on remaining useful life cycle of systems

Document code as implemented and then keep up to date

- Assign a person make person accountable

Better ways of understanding systems

Cycle is getting shorter all the time



A Potential Step Forward

What are architectural principles of this domain that all legacy systems share

What are the canonical structures

Is there a reference architecture for ground systems