Architecture Technology That's Worth Your Investment

"don't settle for fluff"

Richard N. Taylor
Institute for Software Research
University of California, Irvine

taylor@uci.edu
www.ics.uci.edu/~taylor

Architecture-Based Software Engineering

"An approach to software systems development and evolution which uses as its primary abstraction a model of the system's components, connectors, and interconnection topology"

- Domain engineering and requirements definition *may* precede
- OOD *may* supplement; coding follows
- But the architecture remains the primary focus for all system evolution. It is technical. It is critical.

So What's Different?

- The degree to which architectures are made explicit, and how they are described
- Using the description as the basis for implementation & evolution (static and dynamic)
  - it is not a throw-away model
- The degree to which communication is separated from computation
- The prominence given to connectors

“Free Investment Advice”

- If your architectural model can’t support you through implementation and system evolution, it is not worth your investment.
- If you are only using your architectural model for “communication”, you are at best playing at engineering, and may be lying to yourself, your management, and your customers.

References

- UC Irvine's architecture technologies
  - ArchStudio (engineering environment)
    http://www.isr.uci.edu/projects/archstudio/
  - xADL (extensible XML-based ADL)
    http://www.isr.uci.edu/projects/xarchuci/

What's Needed to Support It?

- Appropriate descriptive formalisms
  - Architecture description languages (ADLs)
- Tools
  - to carry out analyses, support editing, generate code, support run-time dynamism, etc.
- Processes
  - to create or recover architectures
  - to integrate architecture-based development into the broader development and organizational contexts

Architecture-Based Dynamism

- Component and connector addition, removal, replacement, and reconnection
- Requires consistency maintenance
  - some guaranteed by *a priori* analysis
  - on-the-fly constraint enforcement
  - faithful architectural model
  - mapping changes to the implementation
- Requires minimal disruption
  - state preservation, replication, migration
  - near-continuous service availability
  - reverting to reliable configurations

Key Facilitator: Connectors

- Traditionally used in system modeling
  - explicit in design, indiscrete in implementation
- Provide a critical abstraction for dynamism
  - should remain discrete, flexible entities in the implementation
  - mediate communication between components
  - specify communication mechanism independent of component behavior
  - encapsulate change application policy
  - boundaries for confining change scope
- Communication using asynchronous messages
  - reduces component communication dependencies
  - how stateful is an event?

Architecture-Based Engineering Environment

- Functional areas:
  - Architecture development and analysis
  - From model to implementation
  - Evolution: static and dynamic
    - Rationale capture
  - Multi-model support

Multi-model Support: xADL

- XML-based representation of ADLs
  - run-time and design-time elements of a system;
  - support for architectural types;
  - configuration management concepts such as versions, options, and variants;
  - product family architectures;
  - architecture "diff"ing

Summary

- Architecture-based software engineering addresses the core issue of system design
  - takes some old and mainstream ideas and makes them better
    - design notations, domain knowledge rep.
  - leverages event-based interaction paradigm
  - leverages results in middleware
  - provides a viable basis for software reuse?
  - provides a viable basis for dynamic adaptation