

Software Architecture

- The highest level organization of the components of a system
- What factors might go into choosing the components?

Software Architecture

- The highest level organization of the components of a system
- What factors might go into choosing the components?
 - Components in the analysis model, existing reusable components, physical machine architecture, expertise of staff, structure of development organization (Conway's Law), development technology (e.g. WebSphere), projected evolution trajectories, non-functional requirements

KWIC Exercise

Software Architecture

(Informal Definition)

- The organization of a system into component subsystems or modules
- *Box and Arrow* diagrams
- Iteratively refined
- Often makes use of stereotypical architectural styles

Architecture (USP Definition)

The set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization: these elements and their interfaces, their collaborations, and their composition. Software architecture is concerned not only with structure and behavior but with usage, functionality, performance, resilience, reuse, comprehensibility, economics and technology constraints and trade-offs, and aesthetic concerns.

Representing Architectures

- *Component*: computational or data element plus interface (ports) to the rest of the system
 - *Requires*: what the component needs from the rest of the system
 - *Provides*: what the component supplies to the rest of the system
- *Connector*: communication protocols among components
- *Configuration*: components hooked up with connectors

Architectural Styles

- High-level, generic abstractions summarizing a related set of design decisions
- Example: *client-server*
 - What are the decisions?
- Benefits
 - Encoded experience
 - Standards: validation; selection criteria
 - Reuse
 - Support of process

Catalog Of Architectural Styles

- *Master control*: hierarchical call and return
- *Pipe and filter*: one-way, sequential, stdin
-stdout, ASCII streams
- *Batch sequential*: validate, edit, update cycle
- *Implicit invocation*: callbacks, registration
-broadcast
- *Table-driven interpreter*: parse and dispatch

More Styles

- *Client-server*: transaction processing, multi-tiers
- *Process control*: with feedback loop
- *Blackboard*: repository, opportunistic control, cooperating agents
- *Shared memory*: with locks for synchronization
- *Production systems*: rule base, conditional firing

And More

- *Coroutines*: inverted implementation
- *Abstract data type*: ADT, hide representation
- *Layered architecture*: abstract machines, limited visibility
- *State-transition systems*: reactive, real-time
- *Object-oriented*: asynchronous message passing, independent threads of control
- *Peer-to-peer*: equal partners sharing responsibility

Style Issues

- Some systems have more than one style
 - *Heterogeneous*
- Domain-specific software architecture (DSSA)
 - Sometimes called *reference architecture*
- Semantics
- Architecture Description Languages (ADLS)
 - Acme, Wright, Rapide, ArTek, Demeter, CODE, Modechart, PSDL/CAPS, Resolve, UniCon

Architectural Evaluation

- Systematic assessment of architecture properties
- Software Architecture Assessment Method (SAAM), Architecture Tradeoff Analysis Method (ATAM)
- Uses
 - Impact assessment
 - Architecture review board

SAAM

