From cradle to grave: An architecture substrate for software lifecycles

Nicolas Rouquette
Principal Member of Technical Staff
Jet Propulsion Laboratory
California Institute of Technology
The waterflow lifecycle revisited

- Is there a consensus that architecture is a pervasive concern?
  - Issue: Architecture as a document vs. an engineering model

- Is there a continuity of architecture throughout?
  - Issue: Traceability back and forth among phases

- Do we need architecture everywhere?
  - Issue: Representation & semantics of architecture in each phase

- Is Architectural change propagation cost-effective?
  - Issue: Transforming from one phase to the next (manual, assisted, ...)

Phases

- Analysis
- Design
- Coding
- Compile
- Link
- Execution

Architecture

- Style 1
- Style 2
- Style 3
- Style 4
- Style 5
- Style 6
The process methodology is flexible...
... as long as State Analysis is applied throughout

Subtle difference between:
- Explicit architecture representation everywhere
- Explicit architecture awareness
- The former is “nice to have”
- The latter is a pragmatic tradeoff for size & fit
<table>
<thead>
<tr>
<th>Similar Dimensions of Concerns</th>
<th>Code level</th>
<th>Component Architecture</th>
<th>System Architecture</th>
</tr>
</thead>
</table>
| Type information              | - procedural code  
- functions  
- classes (OO) | - methods  
- interfaces (sets of methods)  
- ports (interface signature)  
- components (sets of ports)  
- connectors (sets of ports)  
- hierarchical composition | - types  
(from state analysis)  
State variables  
Achievers, etc...  
- domain-specific types  
Units, etc... |
| Instance information          | - variables  
- events  
- objects | - component instances  
- connector instances  
- links (pairs of port bindings)  
- hierarchical composition | - instances  
(from state analysis) |
| Composition Mechanisms (how is it built?) | - function calls  
- symbolic references/linkages  
- shared variables  
- ad-hoc runtime mechanisms | - prescription languages  
(requires type database)  
Base schema: xADL instances  
Extensible via XML schemas  
Compressible via transformations | - subject to lower-level mechanisms |
| Description Mechanisms (what’s inside?) | - ad-hoc runtime mechanisms | - description languages  
(requires instance database)  
Base schema: xADL instances  
Extensible via XML schemas  
Compressible via transformations | - subject to lower-level mechanisms |
Architecture Composition in MDS

- xADL extension for component/connector implementation inheritance
- Separation of structure (defined in xADL) & implementation

- Architecture profiling for optimizing transformations of xADL to code

- Packaging of architecture elements into shared objects
- Dynamic registration of architecture elements at shared object init/fini

- Extensible prescription protocols support connector optimizations
- Architecture evolution includes types & instances
  - Type reconfiguration via dynamic object loading/unloading
  - Instance reconfiguration via prescription changes
The architecture workflow

Key underlying principles
- Easy to do the right thing
- Eliminate semantic replication
- Plan for technology evolution
- Low buy-in cost of adoption

Model-based transformation
- user-assisted process
- more than just “code generation”
- rigorous mapping of input/output theories

Evolution of the State Database

State analysis
Software architecture
Analysis ➔ Design ➔ Coding ➔ Compile ➔ Link ➔ Execution

Style 1 ➔ Style 2 ➔ Style 3 ➔ Style 4 ➔ Style 5 ➔ Style 6

Evolution of Scenarios

System Configuration
Target Configuration
Algorithms
State analysis
Results

Analysis
Compiling
Linking
Execution

[Diagram showing the workflow with various stages and styles]