Rich Software Interconnections

Nikunj Mehta

USC-Center for Software Engineering
GSAW 2000
State of the Practice

• Creating modern software
  – Identify hardware platform
  – Identify middleware platform
  – Create conforming components
  – Lack of heterogeneity within a system
  – Lock in into vendor specific solutions
• Lack of determinism of system properties
• Limited architectural primitives
• Different standards from different vendor groups
State of Our Understanding

- Emphasis on formal representation
- Implementation level design decisions not considered in architectures
- Components and interfaces determine system properties
- Insufficient understanding of foundational elements of software architecture
## Architectural Elements

<table>
<thead>
<tr>
<th>Components</th>
<th>Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory and Computation</td>
<td>Interactions and protocols</td>
</tr>
<tr>
<td>Domain dependent</td>
<td>Domain independent</td>
</tr>
<tr>
<td>Realize functional requirements</td>
<td>Meet extra-functional properties</td>
</tr>
<tr>
<td>Fairly well understood</td>
<td>Mostly not well understood</td>
</tr>
<tr>
<td>Original <em>inhabitants</em></td>
<td><em>Second</em>-class citizens</td>
</tr>
<tr>
<td>Bugs</td>
<td>System dynamism</td>
</tr>
<tr>
<td>Logic, optimization</td>
<td>Middleware</td>
</tr>
</tbody>
</table>

*Architectural Elements*

2/24/2000 GSAW 2000
Software Connectors

• Role of connectors
  – To mediate interactions among components
  – To provide auxiliary mechanisms for interaction

• Not all connectors are simple
  – Complex software involves complex interactions

• Simple connectors
  – Procedure calls, module dependencies, pipes

• Complex connectors
  – DNS, remote procedure calls, repository access, semaphores
A *Disconnected* Understanding

- **Connector modeling**
  - ADL constructs, graphical notations, module dependencies describe well understood connectors
- **Large scale development does not adequately address issues of interaction**
- **Herd*ing* mentality – one size fits all!**
- **Focus on connectors**
  - component logic is essentially frozen early
  - connectors evolve to improve levels of service
Classification Framework

• Atomic elements
  – Ducts, data transfer and control transfer

• Service categories
  – Communication
  – Coordination
  – Conversion
  – Facilitation

• Connector types, dimensions
  – Finitely many values
Applications of the Taxonomy

- **Architecture is not just a dependency graph**
- Linux architecture – Bowman et al
  - A fully connected graph of components
  - Dependencies as connectors
- Rich connectors simplify architecture representation and improve understanding
- Connectors have internal architecture and contain simpler components and connectors internally
Family Architectures

• Need for bounded ambiguity
  – Limited vagueness to support variations
  – Component standardization is a long drawn process
  – Connectors are easily parameterized
  – *Don’t standardize irons or calculators, standardize the power supply*

• Describe component interactions using generic connectors
  – Product architectures need precise dimension and values
Family Designs

• Architecture refinement is a difficult problem
  – Consistency, conformance and automation
• Architecture provides specialized system views for analyzing specific properties
• Design supports realization of the system
• Family designs use design patterns
  – Family designs realize the building blocks in the family architecture
  – Family designs for connectors readily identifiable
Future Work

• The taxonomy is comprehensive – not complete
• Dynamic architectures
  – Evolution of connector dimensions and values
  – Composition of arbitrarily complex connectors
  – Architectural gauges and connector instrumentation
  – Dynamic architecture simulation tools
  – Enhancing the C2 infrastructure for experimentation
  – OTS middleware for realizing rich connectors
References

• Towards a Taxonomy of Software Connectors
  • Mehta, Medvidovic and Phadke
  • CSE Technical Report USC-CSE-99-529

• Software Connectors and Refinement in Family Architectures
  • Egyed, Mehta and Medvidovic
  • To Appear in IW - SAPF ‘3