UML as an Architectural Modeling Language Option

Challenges and adjustments
The challenge mapping Enterprise Architectures with UML

- “UML is a general modeling language and does not provide all concepts that are important to architecture description.” (p. 1, Roh, Kim and Jeon, 2004)

- “Architectural Description Languages (ADL) describe the structure of a software system at a level of abstraction that is more closest to the intuition of a system designer.” (Arvind W. Kiwelekar, 2010)
The challenge mapping Enterprise Architectures with UML

- “UML is a general modeling language and does not provide all concepts that are important to architecture description.” (p. 1, Roh, Kim and Jeon, 2004)

- “Architectural Description Languages (ADL) describe the structure of a software system at a level of abstraction that is more closest to the intuition of a system designer.” (Arvind W. Kiwelekarkar, 2010)

So, what is the problem?

- ADLs designed to model software systems
- UML designed to model software systems
- EAs model?
ADLs and UML

- Why bother?
  - ADLs developed to have specific representations of the constructs in architectures, but not standardized and not generalized nor familiar.
  - UML is becoming ubiquitous if not ‘Universal’

- Common ADL constructs
  - Components
    - Either data store or unit of computation
  - Connectors
  - Systems
    - (configurations of components and connectors)
  - Ports
    - (points of interaction with a component)
  - Representations - for modeling hierarchical composition
  - Rep-maps,
    - mapping composite component’s or connector’s internal architecture of the external interface

- Semantic gaps exist between architecture and UML representations
  - Some UML diagrams favor physical objects/components and not logical constructs
  - There are no direct mappings ADL to UML
### Zachman Framework

Different descriptions for different purposes

<table>
<thead>
<tr>
<th>DATA</th>
<th>FUNCTION</th>
<th>NETWORK</th>
<th>PEOPLE</th>
<th>TIME</th>
<th>MOTIVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective/Scope</strong>&lt;br&gt;<strong>Contextual</strong>&lt;br&gt;Role: Planner</td>
<td>List of Things Important in the Business</td>
<td>List of Core Business Processes</td>
<td>List of Business Locations</td>
<td>List of Important Organizations</td>
<td>List of Events</td>
</tr>
<tr>
<td><strong>Enterprise Model</strong>&lt;br&gt;<strong>Conceptual</strong>&lt;br&gt;Role: Owner</td>
<td>Conceptual Data/Object Model</td>
<td>Business Process Model</td>
<td>Business Logistics System</td>
<td>Work Flow Model</td>
<td>Master Schedule</td>
</tr>
<tr>
<td><strong>System Model</strong>&lt;br&gt;<strong>Logical</strong>&lt;br&gt;Role: Designer</td>
<td>Logical Data Model</td>
<td>System Architecture Model</td>
<td>Distributed Systems Architecture</td>
<td>Human Interface Architecture</td>
<td>Processing Structure</td>
</tr>
<tr>
<td><strong>Technology Model</strong>&lt;br&gt;<strong>Physical</strong>&lt;br&gt;Role: Builder</td>
<td>Physical Data/Class Model</td>
<td>Technology Design Model</td>
<td>Technology Architecture</td>
<td>Presentation Architecture</td>
<td>Control Structure</td>
</tr>
<tr>
<td><strong>Functioning Enterprise</strong>&lt;br&gt;Role: User</td>
<td>Usable Data</td>
<td>Working Function</td>
<td>Usable Network</td>
<td>Functioning Organization</td>
<td>Implemented Schedule</td>
</tr>
</tbody>
</table>

The Zachman framework (Sowa and Zachman, 1992)

Cells are model types
Limits to using Zachman for EA guidance

- Lack of a unified methodology for the whole framework
- Lack of repositories supporting the integrity rules
- No established modeling notation(s)

(from Fatolahi & Shams, 2006)
Architecture is... for Fatolahi & Shams, 2006

- “architecture is a collection of models to be viewed from the viewpoint of different stakeholders with an enterprise." (p. 141)

- Must enforce consistency guidelines

- Must address integrity criteria

- No order to columns, clear basic models that are consistent and aggregate to or flow from the abstractions in the set associated with the row above or the row below.
UML for Rows two & three (Atrolahi and Shams, 2006)

- Function Column
  - Use cases and activity diagrams

- Network Column
  - Nothing explicit in UML
  - Back fit the organizational Unit stereotype extension and adapt

- People (the ‘Who’ column)
  - Organizational Unit stereotype to model ‘roles’

- Time Column
  - UML sequence diagrams

- Motivation Column
  - UML OCL rule modeler
### Table 6: UML models for the Zachman framework

<table>
<thead>
<tr>
<th>Perspectives</th>
<th>Data</th>
<th>Function</th>
<th>Network</th>
<th>People</th>
<th>Time</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planners</td>
<td>List of things important to the business</td>
<td>List of processes the business perform</td>
<td>List of locations in which the business operates</td>
<td>List of organizations important to the business</td>
<td>List of events significant to the business</td>
<td>List of business goals/strategies</td>
</tr>
<tr>
<td>Owners</td>
<td>Class model business entities</td>
<td>Business use case diagram and attached activity diagrams using send, receive and object flows</td>
<td>Organizational units within location stereotypes of packages associated using dependency relationships</td>
<td>Organization chart using organizational units and their included complementary business actors for business use case diagrams</td>
<td>Sequence diagrams using business actors and UML notes for periods</td>
<td>Activity diagrams with actions stereotyped as ends and flows stereotyped as means</td>
</tr>
<tr>
<td>Designers</td>
<td>Class model using entities</td>
<td>Use case diagrams and their attached activity diagrams within system/subsystem packages</td>
<td>Deployment diagram using location stereotype of packages</td>
<td>Completed use case diagrams with actors</td>
<td>Sequence diagrams with actors</td>
<td>Object constraint language</td>
</tr>
<tr>
<td>Builders</td>
<td>All models within the last row mapped onto specific technologies such as J2EE, Oracle, ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Sub-Contractors User | | | | | | Components
| User         | | | | | | Real Working Enterprise-Wide IT Environment |
Elements of a software architecture

“Architectural Description Languages (ADL) describe the structure of a software system at a level of abstraction that is more closest to the intuition of a system designer.” (Arvind W. Kiwelekar, 2010)

- Components
- Connectors
- Systems
- Architectural styles
- Application Oriented Properties
Components

- represent the primary computational elements and data stores of a system.
- Typical examples of component include such things as clients, servers, filters, objects, blackboards and databases.
- Components may have multiple interfaces, each interface defining a point of interaction between a component and its environment.

<table>
<thead>
<tr>
<th>Type of a Component</th>
<th>Interaction Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Procedure Call, Data Sharing</td>
</tr>
<tr>
<td>Object</td>
<td>Method Invocation</td>
</tr>
<tr>
<td>Filter</td>
<td>Data Flow</td>
</tr>
<tr>
<td>Data File</td>
<td>Read, Write</td>
</tr>
<tr>
<td>Database</td>
<td>Schema, Query Language</td>
</tr>
</tbody>
</table>

Table 1.1: Types of Components and Interactions Supported by them
Connectors

- Connectors represent interaction among components.
  - They provide the glue for architectural designs.
  - From the run time perspective, connectors mediate the communication and coordination activities among components.
    - Examples include simple forms of interaction, such as pipes, procedure call, and event broadcast.

- Connectors may also represent complex interactions, such as client-server protocol, or a SQL Link between a database and an application.

- Connectors have interfaces that define the roles played by the participants in the interaction.
Systems

- System represents graphs of components and connectors.
  - A particular arrangement of components and connectors are defined as a system configuration.
  - In general, systems may be hierarchical.
  - Components and connectors may represent subsystems that have their own internal architecture.
Architectural Styles

- Architectural styles describe the families of system that use the same types of components, types of interactions, structural constraints, and analysis.

- Systems built within a single style can be expected to be more compatible than those that mix styles:
  - it may be easier to make them interoperate, and
  - it may be easier to reuse parts within the family.
Application Oriented Properties

- These properties describe the states of a data structure that are of significance to the processing elements manipulating that structure.

- They can be used for such things as controlling the order of processing, helping to define the effects of a processing element on a data structure and even helping to define operations needed by the processing elements to achieve those effects.