Designing a DSL for Information Systems Architecture

Eoin Woods
*UBS Investment Bank*
www.eoinwoods.info

Nick Rozanski
*Marks and Spencer*
www.rozanski.org.uk
Timetable

09:00 – 09:10  Introductions
09:10 – 09:25  Presentation: Architectural Description
09:25 – 09:40  Exercise 1: What Do We Need?
09:40 – 09:50  Collect outputs of exercise
09:50 – 10:10  Presentation: Architectural Notations
10:10 – 10:25  Exercise 2: Quivering at Arrows
11:00 – 11:20  Collect outputs of exercise
11:20 – 11:30  Summary and recap

Optional  Exercise 3: Testing Your Vision
Goals

- Existing description notations have proved to be weak in practice
- Architectural constructs lost as we move to implementation
- Could something better be done?
- We’ll explore this during the session
Timetable

09:00 – 09:10  Introductions
09:10 – 09:25  Presentation: Architectural Description
09:25 – 09:40  Exercise 1: What Do We Need?
09:40 – 09:50  Collect outputs of exercise
09:50 – 10:10  Presentation: Architectural Notations
10:10 – 10:25  Exercise 2: Quivering at Arrows
11:00 – 11:20  Collect outputs of exercise
11:20 – 11:30  Summary and recap
Optional  Exercise 3: Testing Your Vision
What is Software Architecture

The common definition:

- The software architecture of a program or computing system is the **structure or structures** of the system, which comprise software **elements** the externally visible **qualities** of those elements, and the **relationships** among them.

- Len Bass, Paul Clements and Rick Kazman
  Software Architecture in Practice, 2nd Edition
What is Software Architecture

- An alternative definition ...
  - The set of system design decisions that dictate the fundamental structure and properties of a system
  - Thus, the set of decisions that will cause the system to fail if made incorrectly
  - The set of design decisions which, if made wrongly, cause your project to be cancelled!
Architectural Views

- Decompose an architectural description
- Target one or more concerns
- Focus attention on one piece of the problem (one type of structure)
  - functional, deployment, information, ...
- Aid effective communication
  - appropriate representations for the view
Role of the Description

- Communicate the architecture
  - System overview (with selected detail)
- Ongoing reference documentation
  - For architects, developers, testers, support staff,...
- Analysis of the architecture
  - Performance, availability, evolution, ...
- Could it also be the basis of the implementation?
  - And so survive at runtime
Descriptive Difficulties

- An AD contains *architectural* elements
  - Middleware, hardware, component types, connectors, information flows, ...
- The content required varies by context
  - Varying type, precision, detail
- No link from AD to implementation
Possible Future Approach

Hand coded application components within configured “boxes”

Views

Encoding

Configuration

Runtime Platform

Note: encoding/simple transformation not code generation!

Note subtle difference from MDA/MDD – architectural description configures a runtime platform directly rather than trying to generate artefacts for a general purpose runtime environment like J2EE.
Timetable

09:00 – 09:10  Introductions
09:10 – 09:25  Presentation: Architectural Description
09:25 – 09:40  Exercise 1: What Do We Need?
09:40 – 09:50  Collect outputs of exercise
09:50 – 10:10  Presentation: Architectural Notations
10:10 – 10:25  Exercise 2: Quivering at Arrows
11:00 – 11:20  Collect outputs of exercise
11:20 – 11:30  Summary and recap
Optional       Exercise 3: Testing Your Vision
Exercise 1 – What Do We Need?

- Consider **what** needs to be described for the architecture of an information system
  - Technologies? Data Stores? Constraints?

- **How** you could use such a description?
  - Static documentation?
  - Analysis / simulation? (Of what? Why?)
  - Code generation?
  - Configuration of runtime environment?
Exercise 1 – What Do We Need?

- Collect Outputs
Timetable

09:00 – 09:10  Introductions
09:10 – 09:25  Presentation: Architectural Description
09:25 – 09:40  Exercise 1: What Do We Need?
09:40 – 09:50  Collect outputs of exercise
09:50 – 10:10  Presentation: Architectural Notations
10:10 – 10:25  Exercise 2: Quivering at Arrows
11:00 – 11:20  Collect outputs of exercise
11:20 – 11:30  Summary and recap
Optional      Exercise 3: Testing Your Vision
Notations – 3 Approaches

- **Formal textual languages**
  - Architecture Description Languages
    - ACME, Wright, xADL, ...
  - General purpose DSLs for the architectural domain

- **Specific graphical notations**
  - “Boxes and Lines” usually ad-hoc notations
  - Usually very specific to a particular situation

- **Tailored general purpose notations**
  - i.e. UML the de-facto standard
Notations - ADLs

- Many exist in the research domain
  - Wright, ACME, UniCon, xADL, ...
  - www.sei.cmu.edu/architecture/adl.html
- Few (none) have seen industrial use
  - Restrictive assumptions
  - Lack of multiple views
  - Lack of domain/technology specifics
  - Tools
  - Technology transfer
Notations - ADLs

A simple C/S System described in ACME (from CMU) ...

```plaintext
System simple_cs = {
    Component client = {
        Port send-request;
        Properties { Aasp-style : style-id = client-server;
            UniCom-style : style-id = cs;
            source-code : external = "CIDE-LIR/client.c" }}

    Component server = {
        Port receive-request;
        Properties { idempotence : boolean = true;
            max-concurrent-clients : integer = 1;
            source-code : external = "CIDE-LIR/server.c" }}

    Connector rpc = {
        Roles {caller, callee};
        Properties { synchronos : boolean = true;
            max-roles : integer = 2;
            protocol : Wright = "... " }}

    Attachments {
        client.send-request to rpc.caller;
        server.receive-request to rpc.callee }
}
```

http://www.cs.cmu.edu/~acme/
Notations - Boxes and Lines

- The most popular architectural notation
  - Flexible
  - Good tool support
  - Low learning curve

- Limitations
  - Ambiguity
  - Need to explain notation
  - Time to design notation
Notations - UML

- The de-facto “formal” notation
- General purpose software modeling language
  - Little specific architecture support
  - Needs abused or extended for architecture
- Widely understood, wide tool support
  - Although depth of understanding varies
The UML component model ... one of UML’s fairly useful architectural models
UML as an ADL

- UML is really an OOD notation
  - Grown over the years
  - Everything is a class
- Architectural constructs are basic
  - “Component”, interface, dependency
  - Node, link
- Architects lean heavily on extensions
  - Stereotypes, tagged values, notes(!)
- Yet it is the de-facto standard
An Ideal ADL

- What would our ideal notation look like?
- What element types would it contain?
- What could it be used for?
- Whose needs would it address?
- What would make it different from existing approaches?
A Proto-ADL

One possibility ... a simple evolution and specialisation of UML
A Proto-ADL

Another example, for stakeholders who need a more informal and “pictorial” style
<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 – 09:10</td>
<td>Introductions</td>
</tr>
<tr>
<td>09:10 – 09:25</td>
<td><strong>Presentation</strong>: Architectural Description</td>
</tr>
<tr>
<td>09:25 – 09:40</td>
<td><strong>Exercise 1</strong>: <em>What Do We Need?</em></td>
</tr>
<tr>
<td>09:40 – 09:50</td>
<td>Collect outputs of exercise</td>
</tr>
<tr>
<td>09:50 – 10:10</td>
<td><strong>Presentation</strong>: Architectural Notations</td>
</tr>
<tr>
<td>10:10 – 10:25</td>
<td><strong>Exercise 2</strong>: <em>Quivering at Arrows</em></td>
</tr>
<tr>
<td>11:00 – 11:20</td>
<td>Collect outputs of exercise</td>
</tr>
<tr>
<td>11:20 – 11:30</td>
<td>Summary and recap</td>
</tr>
<tr>
<td>Optional</td>
<td><strong>Exercise 3</strong>: Testing Your Vision</td>
</tr>
</tbody>
</table>
Exercise 2: Quivering at Arrows

- Attempt to design our own language for information systems architectural description
  - Pick a fairly narrow domain to keep the problem manageable
- Sketch a graphical ADL language considering
  - Component types you’ll need
  - Connector types needed to link components
  - How to define deployment to runtime nodes
  - Defining environmental constraints
  - Environment configuration
Exercise 2: Quivering at Arrows

- Try to define some of the following:
  - Language entities, relationships & semantics
  - Syntax (graphical and/or textual)
  - What it can be used for?
  - What tools would you need to provide?
  - Examples

- Focus on architectural constructs
  - Don’t worry about business logic
  - Assume manual coding of components
Presentations

- Each group to present their language
- Keep presentations to about 5 minutes
<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 – 09:10</td>
<td>Introductions</td>
</tr>
<tr>
<td>09:10 – 09:25</td>
<td><strong>Presentation</strong>: Architectural Description</td>
</tr>
<tr>
<td>09:25 – 09:40</td>
<td><strong>Exercise 1</strong>: <em>What Do We Need?</em></td>
</tr>
<tr>
<td>09:40 – 09:50</td>
<td>Collect outputs of exercise</td>
</tr>
<tr>
<td>09:50 – 10:10</td>
<td><strong>Presentation</strong>: Architectural Notations</td>
</tr>
<tr>
<td>10:10 – 10:25</td>
<td><strong>Exercise 2</strong>: <em>Quivering at Arrows</em></td>
</tr>
<tr>
<td>11:00 – 11:20</td>
<td>Collect outputs of exercise</td>
</tr>
<tr>
<td>11:20 – 11:30</td>
<td>Summary and recap</td>
</tr>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td><strong>Exercise 3</strong>: Testing Your Vision</td>
</tr>
</tbody>
</table>
Going back to our possible future architecture environment ...

What would the runtime platform need to provide?
=> Types of component, connector, declarative services, monitoring, reflection,...
An Architecture Runtime Platform

- An runtime platform would provide architecture constructs as first class elements
  - Component, interface, queue, message bus, node, information store, ...
- This would allow system architecture to be extracted from running systems
  - Reverse engineering
  - Monitoring and analysis
  - System management
  - Developer support (in IDEs, debuggers, ...)
Summary

- Today we lose most of our architectural constructs when we get to runtime
  - Current approaches don’t change this significantly
- DSLs (ADLs) may give us better architectural description techniques
  - More natural and effective descriptions than UML
- If we could create the matching runtime platform, the architectural constructs would live on at runtime
For Help With Today’s Realities ... 

*Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*

Nick Rozanski & Eoin Woods
Addison Wesley, 2005

http://www.viewpoints-and-perspectives.info
Thank you
Appendix
Exercise 3 (Optional)
Exercise 3: Testing Your Vision

- Given your DSL, what primitives would a supporting runtime platform need to provide?
  - Presumably the set of primitives in the DSL
  - Plus a set of services to support applications
- Define what your runtime would provide
- Try to represent a small system in your DSL
  - Would your system actually run on your platform?
  - What are you missing in your DSL or platform?
- List anything else needed that is out of scope
  - How would you provide these missing pieces?
Experience Reports

- Did your DSL / platform combination hang together and allow a system to be created?
- What were you missing that you needed to add?
- What was out of scope and how would you provide these aspects of the system definition?
Appendix

UML for Architectural Description
UML for Functional Structure

- **GUI Client**: tagged values used to indicate interface characteristics if needed
- **Statistics Accessor**: ClientActions {type=SOAP}
- **Statistics Calculator**: stereotype used to indicate external entity
- **Statistics Store**: element interface and dependent elements using it
- **StatsQuery**
- **StatsUpdate**
- **<<external>> Bulk Loader**
UML for Deployment Structure
UML for Concurrency Structure
But how about
• Entity life history?
• Data flow?
• Volumetrics?
• Ownership?