Model Driven Business Architecture

Pete Rivett
CTO, Adaptive
pete.rivett@adaptive.com
Outline

• What is business architecture?
• User needs
• Information needs (metamodels)
• Use of standard metamodels
• Suitability of UML
• Future
• Summary
Adaptive Experience

- 10+ blue-chip Enterprise Architecture projects
- Addressing real ‘business pain’
  - usually across the ‘IT-business’ gap
- Implemented using Adaptive’s repository-based software product set
- Time-boxed short iterations
- Common models evolved from projects
- Provide sound starting point
  - (And a product)
Why Business Level Architecture?

• Separation of concerns
• Outside vs inside
• Longer-lived structures
• Reference models and patterns
• Manage complexity, change and danger
• Esp. for B2B, outsourcing, mergers
• Applies to processes, people, objectives, business relationships/contracts…
• “Component Based Organizations”
Why Link Business and Software Architectures?

- Traceability to business goals
- Prioritisation
- Impact analysis
- Dependencies
- Basis for configurations
- Two-way communication
- Cut redundancy/duplication
- Regulatory governance
Business Architecture Scope

What

- Objectives
- Capabilities
- Benefits & Measurements
- Change Proposals (Gap Analysis)
- Business & IT Architectures (Current)
- Business & IT Architectures (Target)
- High-level Dependency Plan
- Assessments

With

- Governance
- Programmes
- IS Architectures
- Process Architectures
- Programmes

How (Change Proposals)

- We need to do this ....
- Which means that we need the ability to do these things ....
- These are the detailed things impacted ...
- It would look like this ....
- This is how it can be achieved ....

Quantify it .......

Should we do it? / what priority? / max benefits .........

Make it so and keep me posted ....

Copyright Adaptive Ltd. 2001

13/02/2002
### ENTERPRISE ARCHITECTURE - A FRAMEWORK ™

<table>
<thead>
<tr>
<th>DATA</th>
<th>FUNCTION</th>
<th>NETWORK</th>
<th>PEOPLE</th>
<th>TIME</th>
<th>MOTIVATION</th>
<th>SCOPE (CONTEXTUAL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Things Important to the Business</td>
<td>List of Processes the Business Performs</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/Strats</td>
<td></td>
</tr>
</tbody>
</table>

**Planner**

- **ENTITY = Class of Business Thing**
- **FUNCTION = Class of Business Process**
- **Node = Major Business Location**
- **People = Major Organizations**
- **Time = Major Business Event**
- **Ends/Means = Major Bus. Goal/Critical Success Factor**

**ENTERPRISE MODEL (CONCEPTUAL)**

- e.g. Semantic Model
- e.g. Business Process Model
- e.g. Business Logistics System
- e.g. Work Flow Model
- e.g. Master Schedule
- e.g. Business Plan

**Owner**

- **Ent = Business Entity**
- **Proc. = Business Process**
- **Node = Business Location**
- **People = Organization Unit**
- **Time = Business Event Cycle**
- **End = Business Objective Means = Business Strategy**

**SYSTEM MODEL (LOGICAL)**

- e.g. Logical Data Model
- e.g. Application Architecture
- e.g. Distributed System Architecture
- e.g. Human Interface Architecture
- e.g. Processing Structure
- e.g. Business Rule Model

**Designer**

- **Ent = Data Entity**
- **Reln = Data Relationship**
- **Proc. = Application Function**
- **I/O = User Views**
- **Node = I/S Function**
- **People = Role**
- **Time = System Event Cycle = Processing cycle**
- **End = Structural Assertion Means = Action Assertion**

**TECHNOLOGY MODEL (PHYSICAL)**

- e.g. Physical Data Model
- e.g. System Design
- e.g. Technology Architecture
- e.g. Presentation Architecture
- e.g. Control Structure
- e.g. Rule Design

**Builder**

- **Ent = Segment/Table/etc.**
- **Reln = Pointer/Key/etc.**
- **Proc. = Computer Function**
- **I/O = Data Elements/Sets**
- **Node = Hard ware/System Software**
- **People = User**
- **Time = Execute Cycle = Component Cycle**
- **End = Condition Means = Action**

**DETAILED REPRESENTATIONS (OUT-OF-CONTEXT)**

- e.g. Data Definition
- e.g. Program
- e.g. Network Architecture
- e.g. Security Architecture
- e.g. Timing Definition
- e.g. Rule Specification

**Sub-Contractor**

- **Ent = Field**
- **Reln = Address**
- **Proc. = Language Strt**
- **I/O = Control Block**
- **Node = Addresses**
- **People = Identity**
- **Time = Interrupt Cycle = Machine Cycle**
- **End = Sub-condition Means = Step**

**FUNCTIONING ENTERPRISE**

- e.g. DATA
- e.g. FUNCTION
- e.g. NETWORK
- e.g. ORGANIZATION
- e.g. SCHEDULE
- e.g. STRATEGY

John A. Zachman, Zachman International (810) 231-0531
Users

• Wide variety of roles
  – Many not technical
  – Many with only casual use
• Focus on relationships
• Different levels of abstraction
• Need for visualization
• Legacy of representation style and layout
Simple Business Context Diagram

- Company X
- Prospects
- Customers
- Suppliers
- Government
- Marketing
- Sales
- Finance
- Manufacturing
- Suppliers
- Competitive Products
- Competitive
- Products
- Advertising
- Selling
- Invoice
- Money
- Order
- Order
- Product
- Product Spec
- Pricing
- Executive
- Direction
- Budget
- Regulations
- Government
- Materials
- Executive
- Company X
- Budget
- Regulations
- Tax
“A model can tell a thousand pictures”
• Tools perceived as:
  – Too technical
  – Hard to learn
  – Expensive (usually)

• Metamodel perceived as too complex and abstract

• Profiles give neither simplicity of model nor fitness for purpose
Approach

• Complex metamodels (e.g. UML) for technical areas where existing data and tool integrations exist
• Simpler metamodels for new areas
• Views and navigations to simplify complex models
  – User/role specific
• Generic Visio-template mapping technology
Business Architecture Metamodels

…linked with relevant technical metamodels e.g. UML, CWM
Standards - MOF

• Pros
  – Essential for metamodel-driven approach
  – Up to the job, simple and flexible
  – Federation and combining metamodels good
  – XMI ‘out of the box’

• Cons
  – Need extra properties on the UML
  – Unclear implications of some choices
  – 1-way navigations a pain
  – Physical considerations get in the way
  – Missing some basics like versioning, views, queries
Standards - EDOC

- Choice of profile or metamodel (with notation)
- Starts from (business view of) systems not ‘real’ business
- Only just adopted so little experience
- CCA part quite mature
  - Commonality with EAI, ebXML
  - Used successfully for inter-application architecture
• Choice of profile or metamodel
• Extends EDOC with detail for inter-application integration, e.g.
  – Events
  – Flows
  – Adapters/Connectors
  – Messages (detailed format)
• Just adopted (in final throes)
• Needs clearer positioning with EDOC, CWM
• For business architecture EDOC probably OK
• Mature and at second revision
• To support warehousing covers many areas including information resources, transformations etc
• Modular and extensible
• Good basis for:
  – Information architecture
  – Glossary/nomenclature
  – Software deployment (needs extending)
• Visio shapes mapped to metamodel
  – Custom appearance
    • 2d and 3d shapes
    • Connectors
  – Custom behavior
    • drill-down
    • editing form
  – Help/process
  – Some degree of checking (must be programmed)
• UML Profiles just have a shape and a set of tags (displayed in a generic form). This cannot compete for business users.
UML Suitability (1)

- Class diagrams
  - OK

- Instance diagrams
  - Vital but shamefully neglected in common tools

- Use case diagrams
  - OK as far as they go
  - Diagrams say too little – all devolved to the documents
  - Useful if extended to show links to data
UML Suitability (2)

- Activity diagrams
  - OKish: fit with common process notations
  - UML tool support not
  - Metamodel a complete nightmare
- Collaboration diagrams
  - Useful at system level
  - Generally need more richness as in EDOC/CCA
• Sequence diagrams
  – Too low level in general
  – In some cases useful for processes (cf activity diagrams with swim lanes)
  – Useful for establishing system dependencies
• State diagrams
  – Not of interest in general
• Deployment diagrams
  – Far too limited
Other UML Issues

• Mapping notation to metamodel
• Too many inherited features
• Cannot create views
• Cannot support refinement
  – e.g. an analysis model – copy as the start of a design model and keep 2-way traceability
• Poor package management
• No global object identity
• No diagram interchange
• No control over rigor (when to check)
UML2 Vision

• Core common to MOF2 and UML2
  – Build ‘families’ of languages
  – So can use UML tool as a default with less hassle
  – So can use purpose-specific tools on same data
  – Selectable constraint checking

• Formal mapping to diagrams
  – Could drive more generic drawing
The Future?

• Will UML2 remain an analysis and design language?
• Or a panacea for everything (Universal Modeling Language)?
• Will it have specifiable notations?
• Most talk about applying it to business does not understand the issues
Summary

• Business architecture has a different set of users and needs
• Puts modeling at the start of the process
• Requires integrating standard technical and business metamodels
• OMG has several relevant technical standards in addition to UML
• UML has limited applicability for business aspects
• UML2/MOF2 should improve this depending on notation flexibility