Understanding Architectural Assets

Peter Eeles
peter.eeles@uk.ibm.com
Agenda

Introduction
- Sources of architecture
- Types of architectural asset
- Characterizing architectural assets
- Automating asset reuse
- Conclusion
Inputs into this Presentation

- Working IEEE/IFIP Conference on Software Architecture (WICSA) 2008
  - 18 – 22 February 2008, Vancouver, BC, Canada
  - Working session: Architectural Knowledge
- IBM Asset Architecture Board
- Reusable Asset Specification
- Rational Asset Manager
- RUP for Asset-based Development

© 2006 Tourism Vancouver
Agenda

- Introduction
  - Sources of architecture
    - Types of architectural asset
    - Characterizing architectural assets
    - Automating asset reuse
  - Conclusion
Sources of Architecture

- Theft
  - From a previous system or from technical literature

- Method
  - An approach to deriving the architecture from the requirements

- Intuition
  - The experience of the architect

From “Mommy, Where Do Software Architectures Come From?”, Philippe Kruchten
1st International Workshop on Architectures for Software Systems, Seattle, 1995
Agenda

- Introduction
- Sources of architecture
- Types of architectural asset
  - Characterizing architectural assets
  - Automating asset reuse
  - Conclusion
### What Types of Architectural Asset are there?

<table>
<thead>
<tr>
<th>Reference Architecture</th>
<th>Design Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy Application</td>
<td>Architectural Mechanism</td>
</tr>
<tr>
<td>Pattern Language</td>
<td>Packaged Application</td>
</tr>
<tr>
<td>Development Method</td>
<td>Reference Model</td>
</tr>
<tr>
<td>Architectural Decision</td>
<td>Programming Pattern</td>
</tr>
<tr>
<td>Pattern</td>
<td>Component Library</td>
</tr>
<tr>
<td>Component</td>
<td>Architectural Pattern</td>
</tr>
<tr>
<td>Architectural Style</td>
<td>Application Framework</td>
</tr>
</tbody>
</table>
Development Method

- Best practices
- Guidance (techniques)
- Work product templates (e.g. architecture description template)
- Work product examples
- …

Rational Unified Process
Pattern

- [A pattern is] a common solution to a common problem in a given context. [UML User Guide]

- Pattern types
  - Architectural Patterns
    - Distribution patterns
    - Security Patterns
    - ...
  - Design Patterns
  - Programming Patterns
  - Requirements Patterns
  - Testing Patterns
  - Project Management Patterns
  - Process Patterns
  - Organizational Patterns
  - ...
Architectural Pattern

- An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them. [Buschmann]

- Example:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>A system that requires decomposition</td>
</tr>
<tr>
<td>Problem</td>
<td>High-level elements rely on lower-level elements and the following forces must be balanced:</td>
</tr>
<tr>
<td></td>
<td>- Interfaces should be stable</td>
</tr>
<tr>
<td></td>
<td>- Parts of the system should be exchangeable</td>
</tr>
<tr>
<td></td>
<td>- Source code changes should not ripple through the system</td>
</tr>
<tr>
<td>Solution</td>
<td>Structure the system into layers</td>
</tr>
</tbody>
</table>
ISO OSI 7-Layer Model

- **Layer 7**: Application
  - Provides application facilities
- **Layer 6**: Presentation
  - Structures information as required
- **Layer 5**: Session
  - Manages the connection
- **Layer 4**: Transport
  - Creates packets of data
- **Layer 3**: Network
  - Routes packets of data
- **Layer 2**: Data Link
  - Detects and corrects errors
- **Layer 1**: Physical
  - Transmits bits

**Personal Organizer**

- **Application-Specific**: Personal Organizer
- **Business-Specific**: Address Book, Calculator
- **Base**: Filestore Management, Memory Management, Math
Design Pattern

- A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context. [Gamma]

Observer Pattern

![Observer Pattern Diagram]
Programming Pattern

- An idiom is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language. [Buschmann]

```
// Swap the values of 2 variables
temp = a;
a = b;
b = temp;
```
Architectural Style

- An architectural style defines a family of systems in terms of a pattern of structural organization. More specifically, an architectural style defines a vocabulary of components and connector types, and a set of constraints on how they can be combined. [Shaw]

- Client-server
  - Supports the physical separation of client-side processing (such as a browser) and server-side processing (such as an application server that accesses a database)

- Event-based
  - Promotes a publish-subscribe way of working, applied strategically across large areas of the architecture

- Pipes-and-filters
  - A series of filters that provide data transformation, and pipes that connect the filters. Examples include compilers, signal processing, Straight Through Processing (STP) and trading of electricity, oil and gas
Pattern Language

- A pattern language defines a collection of patterns and the rules to combine them. Pattern languages are often used to describe a family of systems.

IBM Patterns for e-Business

- A set of architectural patterns that describe various web-based applications
- Includes a pattern selection process that drives:
  - Selection of a business, integration or composite pattern
  - Selection of application patterns
  - Selection of runtime patterns
  - Identification of product mappings

*See http://www.ibm.com/developerworks/patterns
Reference Architecture

- A reference architecture is an architecture representation of a particular domain of interest. It typically includes many different architectural patterns, applied in different areas of its structure.

- Examples include J2EE and .NET.
Reference Model

- A reference model is an abstract representation of entities, their relationships and behavior, in a given domain of interest, and which typically forms the conceptual basis for the development of more concrete elements.
- Examples include a business model, an information model and a glossary of terms.

IBM Information FrameWork (IFW)

- IFW Information Models (Banking Data Warehouse)
  - IFW Foundation Models
    - Financial Services Data Model
    - Business Solution Templates
    - Banking Data Warehouse Model
- IFW Process & Integration Models
  - Financial Services Function Model
  - Financial Services Workflow Model
  - Business Object Model
  - Business Process Model
  - Interface Design Model
Application Framework

- An application framework represents the partial implementation of a specific area of an application
- Most widely-known frameworks are those supporting user interfaces
  - Java Server Pages
  - ASP.NET
Architectural Mechanism

- Architectural mechanisms represent common concrete solutions to frequently encountered problems. They may be patterns of structure, patterns of behavior, or both. [RUP]

- Often characterized as
  - “the mechanism for achieving X”
  - “this element is underpinned by mechanism Y”

- Examples
  - Persistency mechanism
  - Error logging mechanism
  - Communication mechanism
  - Shopping cart
Packaged Application

- A packaged application is a large-grained Commercial-Off-The-Shelf (COTS) product that provides a significant amount of capability (and reuse)

- Examples
  - Customer Relationship Management (CRM) application (e.g. Siebel)
  - Enterprise Resource Planning (ERP) application (e.g. SAP)

- The amount of custom development required is greatly reduced

- Primary focus is on configuring the application
Component & component library

- Component examples
  - GUI widget (such as a table)
  - Service

- Component library examples
  - Class libraries (e.g. Java class library)
  - Procedure libraries
Legacy Application

- A legacy application is a system that continues to be used because the owning organization cannot replace or redesign it
- Tends to be a focus on integration rather than new development
- Often results in a focus on enterprise application integration (EAI)
Architectural Decision

- [Architectural decisions are] conscious design decisions concerning a software system as a whole, or one or more of its core components. These decisions determine the non-functional characteristics and quality factors of the system. [Zimmermann]

- Decision rationale may come from experience, method or some other asset
Agenda

- Introduction
- Sources of architecture
- Types of architectural asset
- Characterizing architectural assets
  - Automating asset reuse
  - Conclusion
An Architectural Asset Metamodel
Attributes of an architectural asset

Granularity

- Specification
- Articulation
- Implementation

Large-Grained
- Reference Architecture
- Reference Model
- Pattern Language
- Architectural Style
- Architectural Pattern
- Design Pattern
- Programming Pattern
- Architectural Decision

Fine-Grained
- Packaged Application
- Application Framework
- Architectural Mechanism
- Component Library
- Component
- Legacy Application
Asset Attributes

- **General attributes**
  - Contained artifacts
  - Name
  - Related assets
  - Usage instructions
  - Version

- **Process-related attributes**
  - Author
  - Feedback
  - Rating
  - Reviewer
  - State

- **Architecture-related attributes**
  - Application type (e.g. custom app.)
  - Articulation (e.g. specification)
  - Asset type (e.g. design pattern)
  - Business domain (e.g. telecoms)
  - Development discipline (e.g. testing)
  - Development process (e.g. RUP)
  - Granularity (e.g. fine-grained)
  - Level of abstraction (e.g. logical)
  - Lifecycle phase (e.g. inception)
  - Non-functional properties (e.g. cost)
  - Scope (e.g. systems engineering)
  - Technical domain (e.g. embedded)
  - Variability (e.g. limited)
  - Visibility (e.g. public scope)
Asset Categories

- A category is a search mechanism
- It provides a match on assets with specific attribute values
  - E.g. All assets associated with the telecoms business domain
  - E.g. All assets whose cost is less than $100
Agenda

- Introduction
- Sources of architecture
- Types of architectural asset
- Characterizing architectural assets
- Automating asset reuse
- Conclusion
The Reusable Asset Specification (RAS)

- An OMG standard
- Defines a standard way to describe and package assets
- Defines the interface to a RAS repository
- RAS is used to package many kinds of assets including components, services, patterns, and so on

<table>
<thead>
<tr>
<th>Asset</th>
<th>Name Desc State Ver Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Descriptors: Name/Value pairs</td>
</tr>
<tr>
<td>Context</td>
<td>Domain, Development, Test, Deployment, and so on…</td>
</tr>
<tr>
<td>Solution</td>
<td>Asset Overview Requirements</td>
</tr>
<tr>
<td>Artifacts</td>
<td>Models, Code, Tests Documents …</td>
</tr>
<tr>
<td>Usage</td>
<td>Usage Instructions &amp; Activities Filling Variability Points</td>
</tr>
<tr>
<td>Related Assets</td>
<td>Association, Aggregation, Dependency, Parent</td>
</tr>
</tbody>
</table>
Rational Asset Manager (RAM)

- Manages assets across their lifecycle from design/creation to consumption/change
- Leverages an extensive library of process best practices for asset creation & reuse
RAM - Configuration

Diagram:

- **User**
  - Many-to-Many with **Community**
  - Many-to-One with **Role**
  - Many-to-One with **Review Process**

- **Community**
  - One-to-One with **Role**
  - One-to-One with **Review Process**

- **Role**
  - Many-to-One with **User**
  - Many-to-One with **Asset Type**

- **Review Process**
  - One-to-One with **Asset Type**
  - Many-to-One with **Asset Attribute**
  - Many-to-One with **Category Schema**

**Repository Administration**

- Communities
- Asset Types
- Category Schemas
- Relationship Types
- Asset Attributes

- Tools
- Configuration
- Repository Administrators
## RAM – Asset types, category schema

### Asset Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Framework</td>
<td>An application framework represents the partial implementation of a specific area of an application.</td>
</tr>
<tr>
<td>Architectural Decision</td>
<td>TBD.</td>
</tr>
<tr>
<td>Architectural Mechanism</td>
<td>Architectural mechanisms represent common concrete solutions to frequently encountered patterns of behavior, or both.</td>
</tr>
<tr>
<td>Architectural Pattern</td>
<td>An architectural pattern expresses a fundamental structural organization schema for a set of subsystems, specifies their responsibilities, and includes rules and guidelines for organizing that schema.</td>
</tr>
<tr>
<td>Architectural Style</td>
<td>An architectural style defines a family of systems in terms of a pattern of structural organization. It defines a vocabulary of components and connector types, and a set of constraints on the interactions among them.</td>
</tr>
<tr>
<td>Architecture</td>
<td>An architectural description to which service interfaces, implementation, db designs, and other architectural requirements are referred.</td>
</tr>
<tr>
<td>Asset Case Study</td>
<td>Assets containing lessons learned and success stories using certain assets.</td>
</tr>
<tr>
<td>Case for Change</td>
<td>Assets describing the business and technical case for creating or changing assets.</td>
</tr>
<tr>
<td>Component</td>
<td>A reusable component.</td>
</tr>
<tr>
<td>Component Library</td>
<td>TBD.</td>
</tr>
<tr>
<td>DB Design</td>
<td>Design of a database.</td>
</tr>
<tr>
<td>Design Pattern</td>
<td>A design pattern provides a scheme for refining the subsystems or components of a software system. It describes a commonly recurring structure of communicating components that solve a particular problem.</td>
</tr>
<tr>
<td>Development Method</td>
<td>TBD.</td>
</tr>
<tr>
<td>Dev Time Policy</td>
<td>Assets describing the configuration of the repository. Many of these assets come from the specifications, workflow specifications, descriptions of category schemas, as well as the organizational structure.</td>
</tr>
<tr>
<td>Legacy Application</td>
<td>TBD.</td>
</tr>
<tr>
<td>Minutes</td>
<td>Minutes for a meeting.</td>
</tr>
<tr>
<td>Packaged Application</td>
<td>A packaged application is a large-grained Commercial-Off-The-Shelf (COTS) product that is reused.</td>
</tr>
<tr>
<td>Presentation</td>
<td>Typically a PowerPoint, PDF, HTML, and such.</td>
</tr>
<tr>
<td>Programming Pattern</td>
<td>A programming pattern is a low-level pattern specific to a programming language. Also the particular aspects of components or the relationships between them using the features of a programming language.</td>
</tr>
<tr>
<td>Reference Architecture</td>
<td>A reference architecture is an architecture that has already been created for a particular family of architectural patterns, applied in different areas of its structure.</td>
</tr>
<tr>
<td>Reference Model</td>
<td>TBD.</td>
</tr>
<tr>
<td>Requirement</td>
<td>Many kinds of requirements, functional as well as non-functional.</td>
</tr>
<tr>
<td>Service Design</td>
<td>Design of service interface or service implementation.</td>
</tr>
<tr>
<td>Service Impl</td>
<td>Implementation of a service.</td>
</tr>
</tbody>
</table>

### Category Schema

Define a hierarchy of categories for classifying assets.

**Names:**
- Development Process
- Business Modeling
- Architecture
- Detailed Design
- Implementation
- Testing
- Deployment
- Project Management
- Configuration and Change Management
- Environment

**Description:**
- A definition of the roles, tasks, work products and lifecycle required to support software development.

### Development Process

<table>
<thead>
<tr>
<th>Phase</th>
<th>Est</th>
<th>Insert</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed Design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration and Change Management</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Lifecycle Phase

<table>
<thead>
<tr>
<th>Phase</th>
<th>Est</th>
<th>Insert</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elaboration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transition</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RAM – Communities, review processes

### Communities
Select a community to manage or create new communities.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Architecture</td>
<td>Architects that focus on enterprise solution. Includes consideration of architecture, governance and transition planning.</td>
</tr>
<tr>
<td>Repository Configuration</td>
<td>Assets describing the policies and decisions for various configurations, such as for SOA, or for System Engineering, or for a specific project. Analytic focusing on business process models.</td>
</tr>
<tr>
<td>Service Development</td>
<td>The development resources creating and using services.</td>
</tr>
<tr>
<td>Service Testing</td>
<td></td>
</tr>
<tr>
<td>Software Architecture</td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td></td>
</tr>
<tr>
<td>Software Development</td>
<td></td>
</tr>
<tr>
<td>Systems Architecture</td>
<td></td>
</tr>
</tbody>
</table>

### Community: Software Architecture
Assign roles to all users (including those that are not signed in), all signed-in users, or individual users in the repository.

Filter users: [ ]
Filter by roles: [ ]

### Community: Software Architecture
Manage the processes that this community uses to review and approve assets.

#### Review Processes
Order review processes in decreasing priority, where the first review process to match the minimum set of criteria.

<table>
<thead>
<tr>
<th>Process</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review Process</td>
<td>Conditions</td>
</tr>
<tr>
<td>Application Framework Review Process</td>
<td>Asset type = Application Framework</td>
</tr>
<tr>
<td>Architectural Decision Review Process</td>
<td>Asset type = Architectural Decision</td>
</tr>
<tr>
<td>Architectural Mechanics Review Process</td>
<td>Asset type = Architectural Mechanics</td>
</tr>
<tr>
<td>Architectural Pattern Review Process</td>
<td>Asset type = Architectural Pattern</td>
</tr>
<tr>
<td>Architectural Style Review Process</td>
<td>Asset type = Architectural Style</td>
</tr>
<tr>
<td>Legacy Application Review Process</td>
<td>Asset type = Legacy Application</td>
</tr>
<tr>
<td>Packaged Application Review Process</td>
<td>Asset type = Packaged Application</td>
</tr>
<tr>
<td>Reference Architecture Review Process</td>
<td>Asset type = Reference Architecture</td>
</tr>
<tr>
<td>Reference Model Review Process</td>
<td>Asset type = Reference Model</td>
</tr>
</tbody>
</table>
RAM – Asset Lifecycle
RAM - Architecture

Software Development Platform
- Analyst: RSM, WBM
- Architect: RSA, RSD, RSM
- Developer: RAD, WID
- Tester: RFT, RPT
- Deployment Manager: Tivoli

Browser
- Web client
- Publish, search, browse, retrieve
- Submit and query change requests

Rational Asset Manager Server
- WAS
- Metadata
- Storage
  - LDAP*
  - DB2
  - Oracle
  - SQL Server
  - Filesystem
  - CVS*
  - ClearCase*

Web Services
- Export asset metrics

Measurement and Control Platform
- WebSphere Studio Asset Analyzer
- Identify & harvest candidate assets
- Create, modify, use & review assets
- Measure perf

Asset Management Platform
- Asset Governance & Asset-Based Development Process

* Optional
**RAM and WSRR**

**Development-time**

- **Rational Asset Manager**
  - ClearQuest
  - ClearCase
  - Software Architect

  - Asset based development change, notification & review process
  - Service traceability to versioned assets and referenced artifacts
  - Ensures services are developed consistently & in compliance with architecture

**Deployed Run-time**

- **WebSphere Service Registry and Repository**
  - Publish
  - Find
  - Enrich
  - Manage
  - Govern

  - Any CICS Web services provider program publish & read capability
  - Web services client can publish & search
  - Mediations based on WSRR Lookup for dynamic endpoint selection & binding

Manages information that is useful for developing, re-using and managing all types of **reusable assets**

Manages information that is useful for the **runtime operation, management and development** use of **services**
Conclusion

- Many types of architectural asset are at the disposal of the architect
- Understanding their characteristics and value can help the architect in their decision-making
- Application of appropriate automation is critical in ensuring the success of strategic reuse
References

- [Alexander]
  - Notes on the Synthesis of Form, Christopher Alexander, Harvard University Press, 1964
- [Alexander2]
  - A Pattern Language, Christopher Alexander, Sara Ishikawa and Murray Silverstein, Oxford University Press, 1977
- [Alexander3]
- [Brown]
- [Buschmann]
- [Fowler]
- [Gamma]
  - Design Patterns – Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Addison Wesley, ISBN 0-201-63361-2, 1996
- [Kruchten]
- [Kruchten2]
- [RAS]
  - Reusable Asset Specification, Object Management Group Inc., Document number 04-06-06, June 2004
- [Rozanski]
- [RUP]
- [Shaw]
- [UML User Guide]
- [Zimmermann]
  - Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method, Olaf Zimmermann et al, Proceedings of WICSA 2008
Thank You