How to Represent the Architecture of Your Enterprise Application Using UML 2.0 and More

Paulo Merson
Software Engineering Institute
www.sei.cmu.edu/architecture

Session TS-4619
We Can Do Better Than This!

I created this diagram eight years ago.

The design may be OK.

But the design description is bad and in one hour I’ll have told you why!
Goals of This Talk

• Tell you what information about an architecture should be captured
• Show you UML 2.0 and other notations and guidelines for architecture representation
• Give you a break from reading code :^)
Agenda

Introduction

Multi-View Architecture
  Module Views
  Runtime Views
  Deployment Views
  Data Model

Template for an Architecture Document

Outroduction
Agenda

Introduction

Multi-View Architecture
  Module Views
  Runtime Views
  Deployment Views
  Data Model

Template for an Architecture Document

Outroduction
Motivation for Software Architecture (1)

The Problem

Requirements

???

Implementation
Motivation for Software Architecture

The Role of Software Architecture

- Requirements
- Architecture
- Implementation
What Is Software Architecture?


Why Document the Architecture?

• In the software lifecycle we:
  • Create an architecture
    • Using architectural patterns, design patterns, experience
  • Evaluate the architecture
    • Using ATAM® for example
  • Refine, update and refactor the architecture along the way
  • Use the architecture to guide implementation
  • (Try to) enforce the architecture during implementation

Software Architecture Documentation Is the Key Artifact in All These Activities
Agenda

Introduction

**Multi-View Architecture**
- Module Views
- Runtime Views
- Deployment Views
- Data Model

Template for an Architecture Document

Introduction
Views

- Systems are composed of many structures
  - Code units, their decomposition and dependencies
  - Processes and how they interact
  - How software is deployed on hardware
  - Many others

A View Is a Representation of a Structure, That Is, a Representation of a Set of System Elements and the Relations Associated With Them
What Views Are Available?

• An architect can consider the system in at least four ways:

  1. How is it structured as a set of code units?
     
     *Module Views*

  3. How is it structured as a set of elements that have runtime presence?
     
     *Runtime Views*

  5. How are artifacts organized in the file system and how is the system deployed to hardware?
     
     *Deployment Views*

  7. What is the structure of the data repository?
     
     *Data Model*
Agenda

Introduction

Multi-View Architecture

Module Views

Runtime Views

Deployment Views

Data Model

Template for an Architecture Document

Outroduction
Module Views

• Show structure of the system in terms of units of implementation

• **Elements:** modules, i.e., code units that implement some functionality

• **Relations:**
  • A is part of B: part-whole relation among modules
  • A depends on B: dependency relation among modules
  • A is a B: specialization/generalization relation among modules, or interface realization
UML Relations Between Modules

“Is part of”
Package contains subpackages or classes

“Depends on”
Dependency can be <<use>>, <<refine>>, <<instantiate>>, etc.

“Is a”
Generalization and interface realization

UML Has Other Standard Relations, and You Can Specialize Any of Them With Stereotypes
High-Level Module View—Duke’s Bank(1)

Is This a Layered Design?

Notation: UML 2.0
High-Level Module View—Duke’s Bank\(^{(2)}\)

```
web

com.sun.ebank.web

com.sun.ebank.appclient

com.sun.ebank.ejb

com.sun.ebank.util

JSPs and static web artifacts (not Java sources)

Util classes use exception and EJB home classes defined in the ejb package.

Notation: UML 2.0
```

Showing Module Usage Decomposition

Reconstructed from Duke’s Bank Application—Sun J2EE 1.3 tutorial
What Are Module Views Good For?

- Construction—they are the blueprints for the code
- Budgeting, work assignment, tracking
- Education of new developers
- Modifiability and impact analysis
Agenda

Introduction

Multi-View Architecture
  Module Views
  **Runtime Views**
  Deployment Views
  Data Model

Template for an Architecture Document

Introduction
Runtime Views

• Show structure of the system when it’s executing

• **Elements:**
  • Components that have runtime presence (e.g., processes, threads, EJB™ components, servlets, DLLs, objects)
  • Data stores

• **Relations:**
  • Interaction mechanisms vary based on technology
  • Architect should differentiate:
    • Local from remote calls
    • Synchronous from asynchronous invocation
Runtime View—Duke’s Bank

Reconstructed from Duke’s Bank Application—Sun J2EE 1.3 tutorial
UML 2.0 for Runtime Views

- Component (as subtype of class)
- Port (which can have multiple instances)
- Required and provided interface (optionally connected through ports)
- Assembly connector
- Internal structure for classes
- Delegation connector

1 to 5 instances of this port (suggesting it can take up to 5 requests at a time)

Only 1 instance of this port

Provided interface

Online Services[1..5]

Port

Search

Admin Services[1]

ItemEntry

Search Engine

Catalog Mgr

Catalog

Data Cache

Dava Validation

Delegation connector

Assembly connector

Delegation connector

Required interface

Assembly connector

Authorized User

DataAccess
Runtime View—Duke’s Bank (UML)

Reconstructed from Duke’s Bank Application—Sun J2EE 1.3 tutorial
Runtime View—Duke’s Bank—Not So Good Example

Original diagram of Duke’s Bank Application from Sun J2EE 1.3 tutorial (with our amendment)
http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Ebank2.html
Runtime View—Adventure Builder

SOA Example

Reconstructed from Adventure Builder Application
http://java.sun.com/developer/releases/adventure/
What Are Runtime Views Good For?

• Explaining:
  • How components interact at runtime
  • What components are replicated
  • What components access data stores

• Education—starting point to show how the system works

• Analysis of runtime properties
  • Performance
  • Security
  • Reliability
Agenda

Introduction
Multi-View Architecture
  Module Views
  Runtime Views
  Deployment Views
Data Model
Template for an Architecture Document
Outroduction
Deployment Views

• Show at least two distinct but related structures:
  1. **Hardware** infrastructure of the production environment
  2. Structure of **directories and files** of deployed system

• **Elements:**
  1. Processing and communication nodes (e.g., server machine, router)
  2. Files, directories

• **Relations:**
  1. Interaction mechanisms between two elements are usually communication channels
  2. Containment: a directory/file contains other directories and files
Deployment View—Hardware Infrastructure—Duke’s Bank\(^{(1)}\)
Deployment View—Hardware Infrastructure—Duke’s Bank$^{(2)}$

<<deploy>> dependency shows how artifacts are deployed onto nodes.

<<executionEnvironment>> is a node that offers an environment to run specific types of components.

<<artifact>> denotes a file of any kind.

<<artifact>> app-client.jar

<<artifact>> DukesBankApp.ear
Deployment View—Deployment Files—Duke’s Bank\(^{(1)}\)

![Diagram showing deployment view and deployment files for Duke’s Bank.](image-url)
Deployment View—Deployment Files—Duke’s Bank

Notation: UML 2.0

<<manifest>> indicates the component is inside the artifact

<<manifest>> may also indicate that an artifact is inside another

These are the two top-level artifacts:

- DukesBankApp.ear
- app-client.jar
- web-client.war
- AdminMessages.properties
- Dispatcher
- WebMessages.properties
- struts.jar

- account-ejb.jar
- customer-ejb.jar
- tx-ejb.jar
- AccountControllerEJB
- CustomerControllerEJB
- TxControllerEJB
- AccountEJB
- CustomerEJB
- LoginServlet
- jsp
- *.jsp
- *.ltxd
- *.gif
- *.html

Shorthand for all JSP files
What Are Deployment Views Good For?

• Defining deployment and operation procedures
• Auditing runtime failures
• Planning purchasing options
• Analyzing:
  • Availability
  • Performance (e.g., throughput, bandwidth utilization)
  • Security
  • Reliability
• Education and stakeholder communication
Deployment View—Hardware Infrastructure—Real-World Example
Agenda

Introduction

Multi-View Architecture
  Module Views
  Runtime Views
  Deployment Views

Data Model

Template for an Architecture Document

Outroduction
Data Model

- Shows structure of the data repository
- **Elements**: entities (persisted domain elements)
- **Relations**:
  - 1:1, 1:n and n:n relationships
  - Generalization/specialization
  - Aggregation
Data Model—Duke’s Bank

**account**
- **PK**: account_id
- **VARCHAR(8)**
  - type
  - description
  - balance
  - credit_line
  - begin_balance
  - begin_balance_time_stamp

**tx**
- **PK**: tx_id
- **VARCHAR(8)**
- **FK1**
  - account_id
  - time_stamp
  - amount
  - balance
  - description
  - VARCHAR(8)
  - DATETIME
  - NUMERIC(10,2)
  - NUMERIC(10,2)
  - VARCHAR(30)

**customer_account_xref**
- **PK,FK1**
- **PK,FK2**
- account_id
- VARCHAR(8)
  - customer_id
  - VARCHAR(8)

**customer**
- **PK**: customer_id
- **VARCHAR(8)**
  - last_name
  - first_name
  - middle_initial
  - street
  - city
  - state
  - zip
  - phone
  - email
  - VARCHAR(30)
  - VARCHAR(30)
  - VARCHAR(1)
  - VARCHAR(40)
  - VARCHAR(40)
  - VARCHAR(2)
  - VARCHAR(5)
  - VARCHAR(16)
  - VARCHAR(30)

**Key:**
- entity
- PK=primary key
- FK=foreign key
- column name
- data type

- Each A has 0 or more Bs, each B is related to exactly one A
- Each A has 0 or more Bs, each B is related to exactly one A, A’s PK is needed as part of B’s PK
Data Model—Duke’s Bank

account
- account_id : varchar(8)
- type : varchar(24)
- description : varchar(30)
- balance : numeric(10,2)
- credit_line : numeric(10,2)
- begin_balance : numeric(10,2)
- begin_balance_time_stamp : datetime

0..*

customer_account_xref
- account_id : varchar(8)
- customer_id : varchar(8)

* customer
- customer_id : varchar(8)
- last_name : varchar(30)
- first_name : varchar(30)
- middle_initial : varchar(1)
- street : varchar(40)
- city : varchar(40)
- state : varchar(2)
- zip : varchar(5)
- phone : varchar(16)
- email : varchar(30)

next_id
- beanName : varchar(30)
- id : numeric(10,2)

Notation: UML 2.0
What Is the Data Model Good For?

• Blueprint for physical database
• Reference to all programs that manipulate data items
• Database tuning (e.g., via normalization):
  • For better performance and modifiability
  • To avoid redundancy
  • To enforce consistency
Agenda

Introduction

Multi-View Architecture
  Module Views
  Runtime Views
  Deployment Views
  Data Model

Template for an Architecture Document

Outroduction
Software Architecture Documentation

• How do we document a view?

• How do we document everything else beyond the views?
Documenting a View\(^{(1)}\)

1. Primary presentation
   - Is usually graphical
   - Shows elements and relations among them
   - Should include a key that explains the notation
     - Give meaning of each symbol; don’t forget the lines!

Many Times, the Primary Presentation Is All You Get. It’s Not Enough!
Documenting a View(2)

2. Element catalog

- Explains elements depicted in primary presentation
- Is usually a table with element name and textual description

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFConfigLoader</td>
<td>This class is able to read and parse the reasoning framework configuration file and provide its users the information required in the structure required.</td>
</tr>
<tr>
<td>vo</td>
<td>This package has classes that follow the value object design pattern [Fowler]. These classes hold the data being manipulated in the system. They are used by many other modules. The facts in memory are classes in this package.</td>
</tr>
<tr>
<td>corebridge</td>
<td>This package contains the functionality to ArchE core façade. One or more classes will be created with public methods that correspond to the command facts in ArchE core. These classes use the Jess Java API.</td>
</tr>
<tr>
<td>Jess Java API</td>
<td>This package has the Java API to manipulate facts/rules in the Jess rule engine. It is an external library that is not part of the studio development.</td>
</tr>
<tr>
<td>ExportDesign</td>
<td>This interface defines the general contract for exporting design to a specific tool. Because of this interface, the specific adapters have the same method signature and can be interchanged in the application. See the adapter design pattern in [GoF].</td>
</tr>
</tbody>
</table>
3. Variability guide

- Identify points where system can be configured
  - Number of instances in a pool
  - Optional inclusion of components (plug-ins, add-ons)
  - Selection among different implementations of a component or connector
  - Parameterized values set at build, deploy or run-time
Documenting a View\(^{(4)}\)

4. Architecture background

- Rationale for design decisions (including relevant rejected alternatives)
- Results of analysis, prototypes, and experiments
- Assumptions and constraints affecting this view

5. Related views

- Pointer to parent view or children views
Outline of a Documented View

Template for a View

1. Primary Presentation

2. Element Catalog

3. Variability Guide

4. Architecture Background
   4.1 Design Rationale
   4.2 Results of Analysis and Experiments
   4.3 Assumptions and Constraints

5. Related Views
Software Architecture Document\(^{(1)}\)

1. Documentation roadmap
   • Shows how documentation is organized
   • Has reference to template used
   • Includes scenarios for using the documentation

2. System overview
   • Description of the system and its purpose
   • Context diagram to show scope
   • May point to overview elsewhere in the overall system documentation
3. Requirements

• May point to separate requirements document

• Three kinds of requirements are relevant to the architecture:
  • Functional requirements (usually captured as use cases)
  • Quality attribute requirements (performance, availability, etc.)
  • Design constraints; example: “the system shall use Hibernate for persistence”
Software Architecture Document

4. Views

- 4.1
- 4.2
- 4.3

...
Software Architecture Document\(^{(4)}\)

5. Mapping between views

- Tables showing how elements in one view map to elements in another view

<table>
<thead>
<tr>
<th>Element in Runtime View X</th>
<th>Element in Module View Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAdmin</td>
<td>com.sun.ebank.appclient</td>
</tr>
<tr>
<td></td>
<td>com.sun.ebank.util</td>
</tr>
<tr>
<td></td>
<td>stubs from com.sun.ebank.ejb</td>
</tr>
<tr>
<td>Web browser</td>
<td>—</td>
</tr>
<tr>
<td>WebUI</td>
<td>web</td>
</tr>
<tr>
<td></td>
<td>com.sun.ebank.web</td>
</tr>
<tr>
<td></td>
<td>com.sun.ebank.util</td>
</tr>
<tr>
<td></td>
<td>stubs from com.sun.ebank.ejb</td>
</tr>
<tr>
<td>AccountControllerEJB</td>
<td>com.sun.ebank.ejb</td>
</tr>
<tr>
<td></td>
<td>com.sun.ebank.util</td>
</tr>
<tr>
<td>AccountEJB</td>
<td>com.sun.ebank.ejb</td>
</tr>
<tr>
<td></td>
<td>com.sun.ebank.util</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Only Relevant Mappings Are Documented
Software Architecture Document

6. Architecture analysis and rationale
   • Rationale for cross-view design decisions (including rejected alternatives)
   • Results of architecture evaluation (e.g., ATAM report)

7. Mapping requirements to architecture
   • Shows how each requirement is satisfied by one or more elements of the architecture, or an architectural approach

8. Glossary and acronym list
   • May point to a larger glossary elsewhere
Outline of Software Architecture Document

1. Documentation Roadmap
2. System Overview
3. Requirements
4. Views
   4.1 View X
   4.2 View Y
   etc.
5. Mapping Between Views
   5.1 Mapping View X to View Y
   5.2 Mapping View Y to View Z
   etc.
6. Architecture Analysis and Rationale
7. Mapping Architecture to Requirements
8. Glossary and Acronym List
Agenda

Introduction

Multi-View Architecture
  Module Views
  Runtime Views
  Deployment Views
  Data Model

Template for an Architecture Document

Outroduction
What Else Is Important?

• Document behavior using, for example:
  • Sequence diagrams for interesting* traces
  • Statecharts for things that have interesting* states
• Document interfaces beyond syntax
• Indicate what patterns you’re using
• Select views to document based on stakeholders’ needs

*Interesting = important and/or complex
Summary

• Important takeaways:
  • Describe the architecture in multiple views
    • Module views
    • Runtime views
    • Deployment views
    • Data Model
  • Documentation should not appeal to reader’s intuition
    • Always use a key or indicate the diagram notation
  • Follow a template
  • UML is not always the best notation
For More Information

- [www.sei.cmu.edu/architecture](http://www.sei.cmu.edu/architecture)
- “Documenting Software Architectures: Views and Beyond” by Paul Clements, et al.
- UML 2.0 Superstructure Specification ([www.uml.org](http://www.uml.org))
- [http://la.sei.cmu.edu/sad-wiki](http://la.sei.cmu.edu/sad-wiki) (for an example)

Or talk to This guy
Q&A

Paulo Merson
pfm@sei.cmu.edu

Suggestions:

• What’s the difference between architecture and detailed design?
• I use an Agile process. Should I care about this stuff?
• What if I follow RUP? What about the 4+1 views?
• Hey SEI guy, aren’t you gonna talk about CMM?
• Where’s the recipe for the Brazilian drink “caipirinha”? 
How to Represent the Architecture of Your Enterprise Application Using UML 2.0 and More

Paulo Merson

Software Engineering Institute
www.sei.cmu.edu/architecture

Session TS-4619