

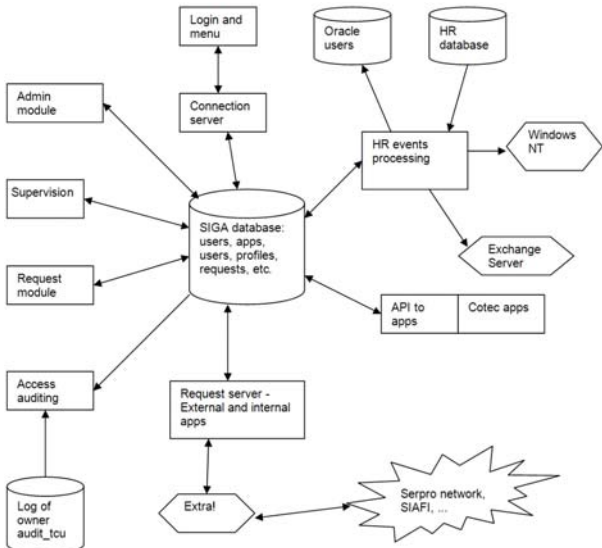
How to Represent the Architecture of Your Enterprise Application Using UML 2.0 and More

Paulo Merson

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

Session 4619


We Can Do Better Than This!



I created this diagram 8 years ago

The design may be OK

But the design description is bad and in one hour I'll have told you why!


 2006 JavaOneSM Conference | Session TS-4619 | 2 | java.sun.com/javaone/sf



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

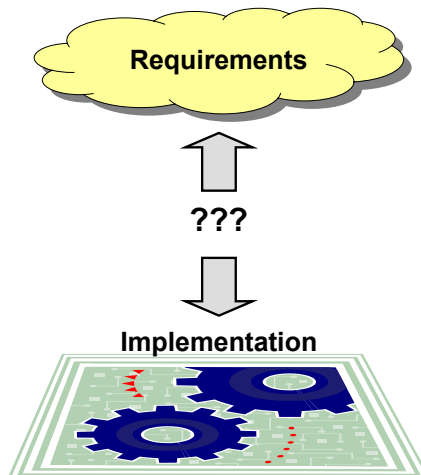
Data Model

Template for an Architecture Document

Outroduction

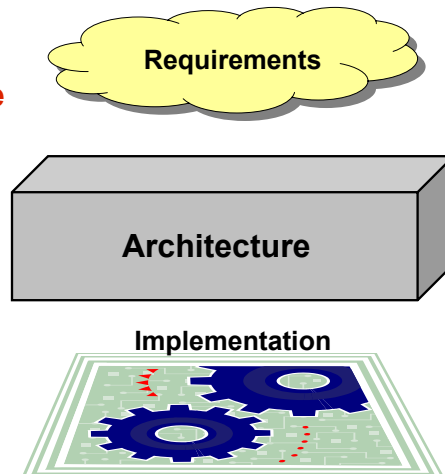
Motivation for Software Architecture (1)

The problem



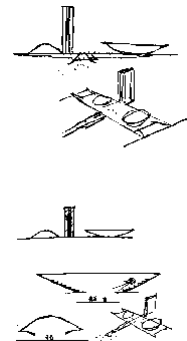
Motivation for Software Architecture (2)

The role of software architecture



What is Software Architecture?

A software architecture for a system is the structures of the system, which comprise elements, relations among them, and the externally visible properties of those elements and relations¹



¹ Adapted from Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice, 2nd Edition*. Addison-Wesley, 2003.



Why Document the Architecture?

- In the software life cycle 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

Outroduction



Views

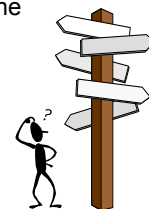
- Systems are composed of many structures
 - Code units, their decomposition and dependencies
 - Processes and how they interact
 - How software is deployed on hardware
 - Many others

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 4 ways:
 1. How is it structured as a set of code units?
Module Views
 2. How is it structured as a set of elements that have runtime presence?
Runtime Views
 3. How are artifacts organized in the file system and how is the system deployed to hardware?
Deployment Views
 4. 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



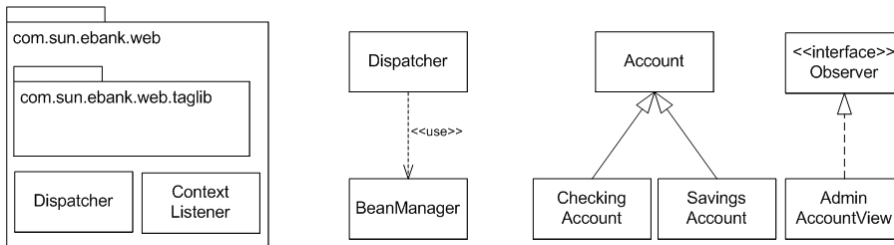
Module Views

- 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.

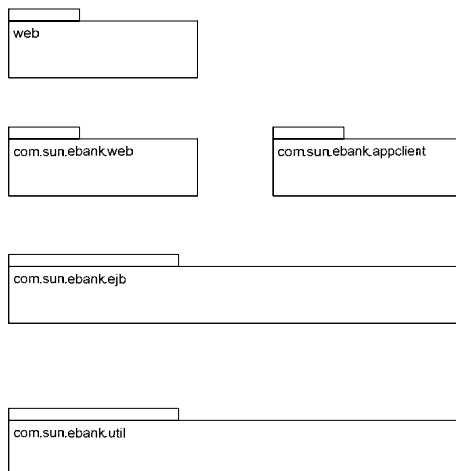
"Is a"

Generalization and interface realization

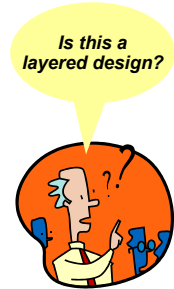
UML has other standard relations, and you can specialize any of them with stereotypes



High-Level Module View— Duke's Bank (1)



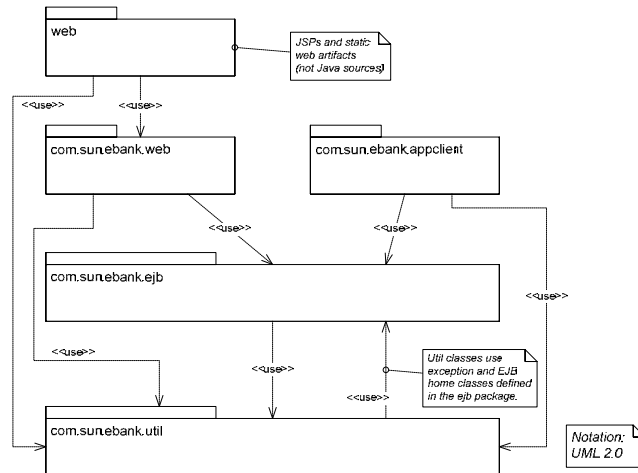
Showing only module decomposition



Notation: UML 2.0



High-Level Module View— Duke's Bank (2)



Showing module usage dependency

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

Outroduction



Runtime Views

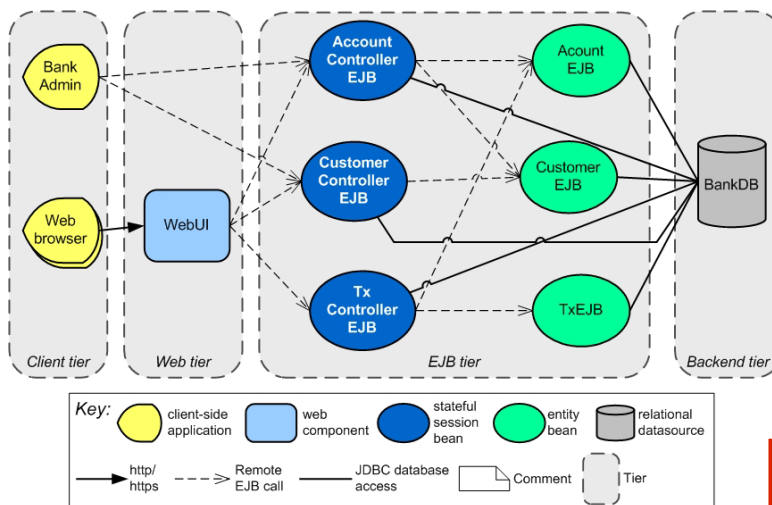


- Show structure of the system when it's executing
- **Elements:**
 - Components that have runtime presence (e.g., processes, threads, EJBTM components, Servlets, DLLs, objects)
 - Data stores
- **Relations:**
 - Interaction mechanisms vary based on technology
 - Architect should differentiate:
 - Local from remote calls
 - Synchronous from asynchronous invocation





Runtime View—Duke's Bank



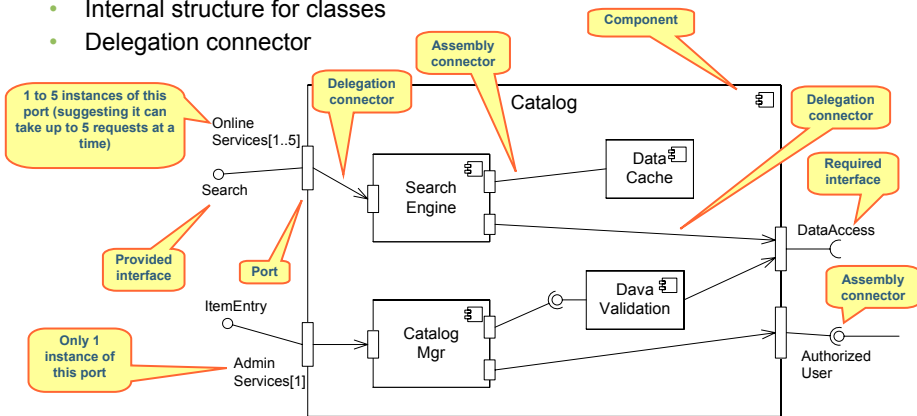
Informal notation

Reconstructed from Duke's Bank Application—Sun J2EE 1.3 tutorial



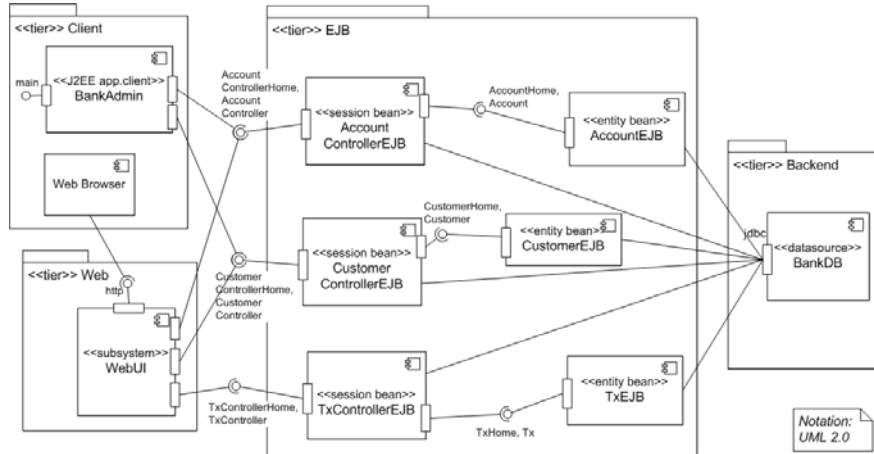
UML 2.0 For Runtime Views

- 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





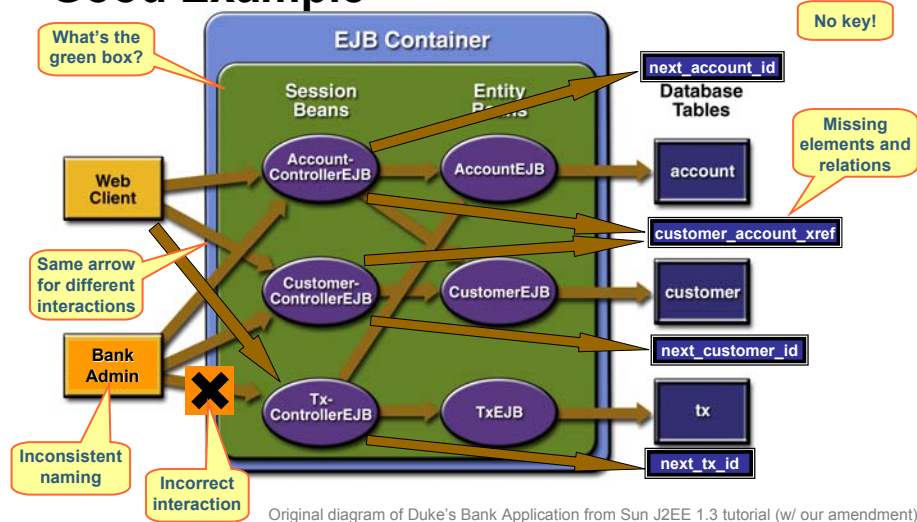
Runtime View—Duke's Bank (UML)



Reconstructed from Duke's Bank Application—Sun J2EE 1.3 tutorial



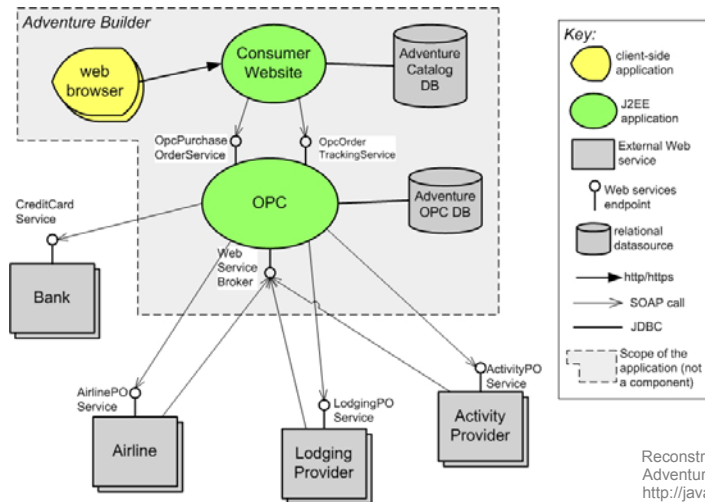
Runtime View—Duke's Bank—Not So Good Example



Original diagram of Duke's Bank Application from Sun J2EE 1.3 tutorial (w/ our amendment) http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Ebank2.html



Runtime View—Adventure Builder

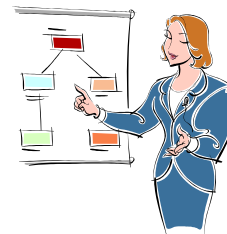


SOA example

Reconstructed & adapted 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

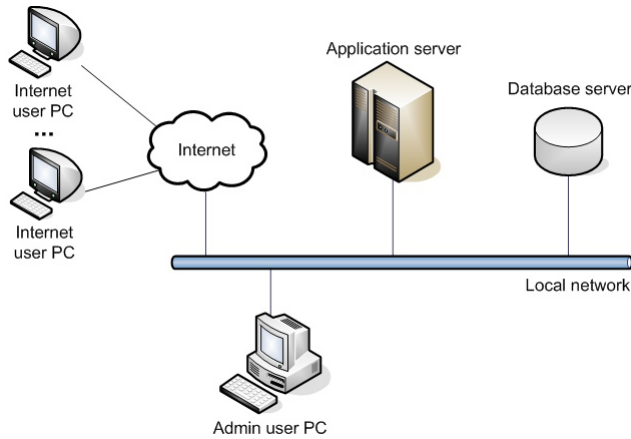
- Show at least two distinct but related structures:
 1. **Hardware** infrastructure of the production environment
 2. Structure of **directories and files** of deployed system
- **Elements:**
 1. Processing and communication nodes (e.g., server machine, router)
 2. Files, directories
- **Relations:**
 1. Interaction mechanisms between 2 elements are usually communication channels
 2. Containment: a directory/file contains other directories and files



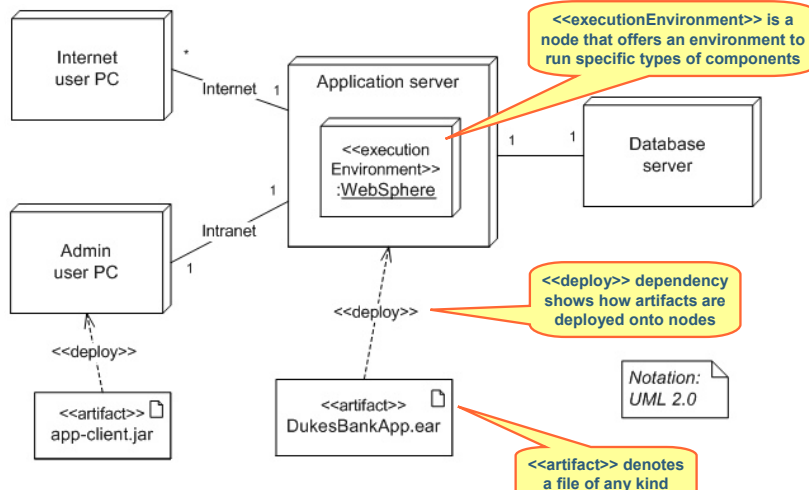


Deployment View—Hardware Infrastructure—Duke’s Bank (1)

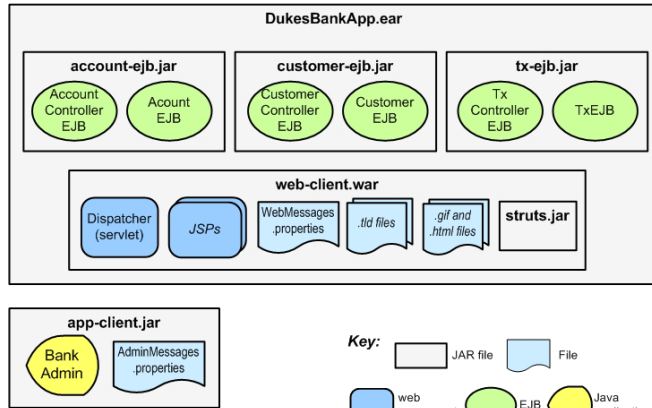
Informal notation



Deployment View—Hardware Infrastructure—Duke’s Bank (2)

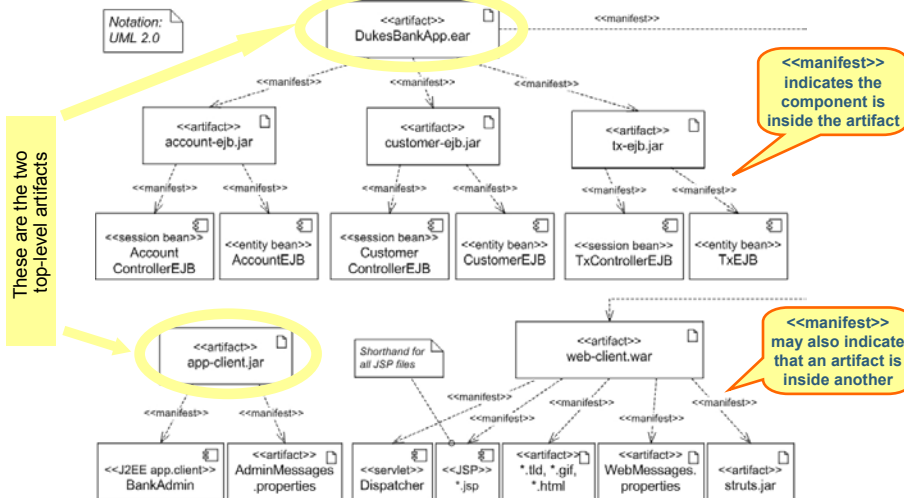


Deployment View—Deployment Files—Duke's Bank (1)



Informal notation

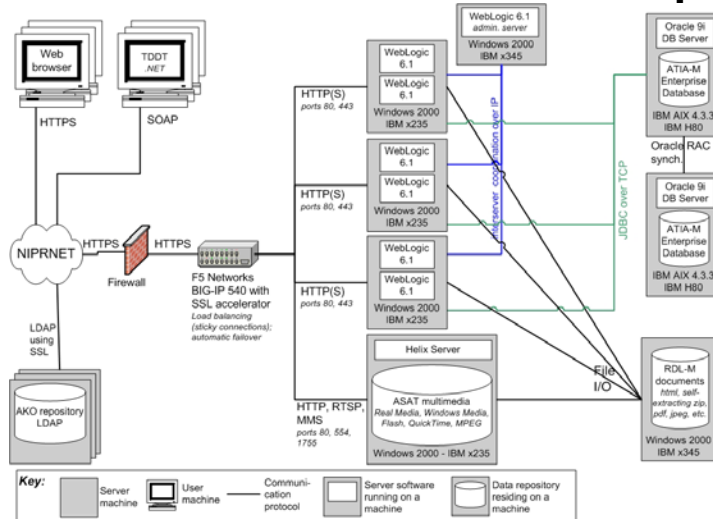
Deployment View—Deployment Files—Duke's Bank (2)



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



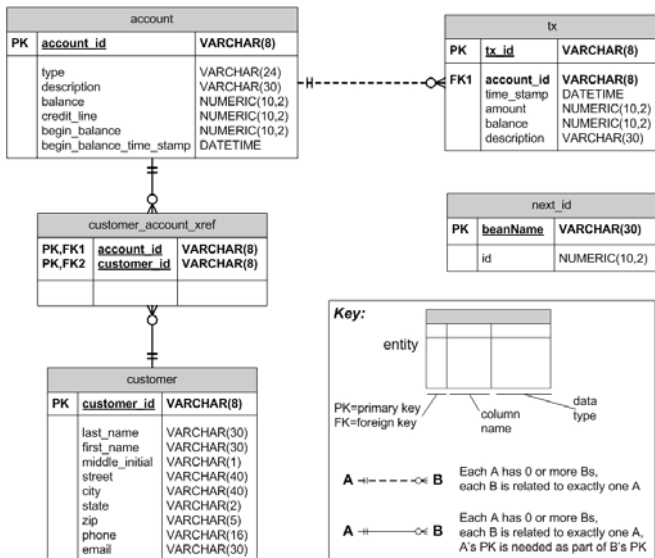
Data Model

- Shows structure of the data repository
- **Elements**: entities (persisted domain elements)
- **Relations**:
 - 1:1, 1:n and n:n relationships
 - Generalization/specialization
 - Aggregation





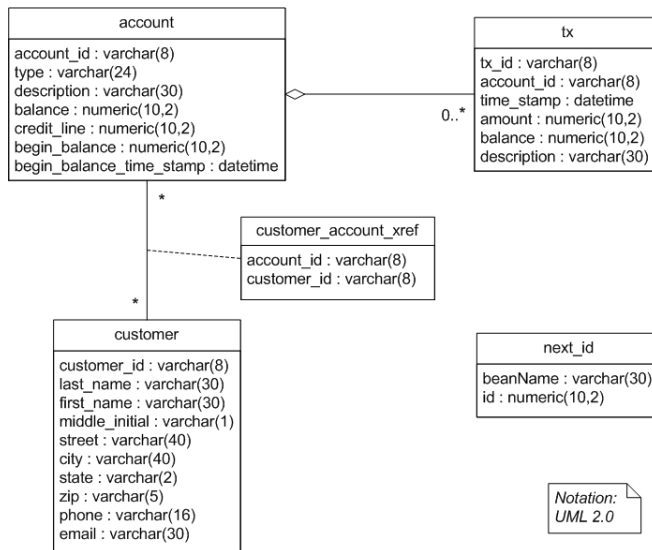
Data Model—Duke's Bank (1)



ERD notation



Data Model—Duke's Bank (2)





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

Element	Description
RFConfigLoader	This class is able to read and parse the reasoning framework configuration file and provide its users the information required in the structure required.
vo	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.
corebridge	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.
Jess Java API	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.
ExportDesign	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].

Documenting a View (3)

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



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



Software Architecture Document (2)



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 (3)

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

Only relevant mappings are documented

Element in Runtime View X	Element in Module View Y
BankAdmin	com.sun.ebank.appclient com.sun.ebank.util stubs from com.sun.ebank.ejb
Web browser	—
WebUI	web com.sun.ebank.web com.sun.ebank.util stubs from com.sun.eban.ejb
AccountControllerEJB	com.sun.ebank.ejb com.sun.ebank.util
AccountEJB	com.sun.ebank.ejb com.sun.ebank.util
...	...

Example from Duke's Bank

Software Architecture Document (5)

6. Architecture Analysis and Rationale

- Rationale for cross-view design decisions (including rejected alternatives)
- Results of architecture evaluation (e.g., ATAM report)

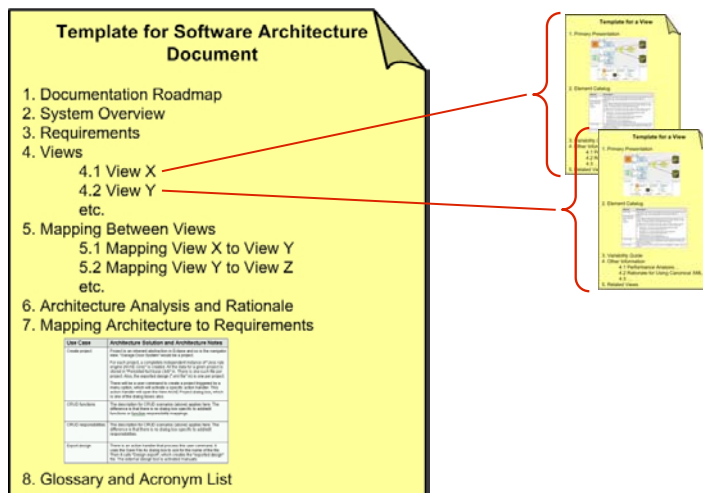
7. Mapping Requirements to Architecture

- 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





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
- “Documenting Software Architectures: Views and Beyond” by Paul Clements et al.
- UML 2.0 Superstructure Specification (www.uml.org)
- <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"?

2006 JavaOneSM Conference | Session 4619 | 57

java.sun.com/javaone/sf



the
POWER
of
JAVA™



Carnegie Mellon University
Software Engineering Institute

JavaOne

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 4619

2006 JavaOneSM Conference | Session TS-4619 |

java.sun.com/javaone/sf