Modeling Principles in an EA context

Examples of patterns, frameworks and useable R-forms,
Dimensions of software complexity

Higher technical complexity
- Embedded, real-time, distributed, fault-tolerant
- Custom, unprecedented, architecture reengineering
- High performance

Lower technical complexity
- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

Higher management complexity
- Large scale
- Contractual
- Many stake holders
- “Projects”

Lower management complexity
- Small scale
- Informal
- Single stakeholder
- “Products”

An average software project:
- 5-10 people
- 10-15 month duration
- 3-5 external interfaces
- Some unknowns & risks
Key Enterprise Architecting questions

- Phase, Iterations
  - When does architecture happen?

- Process Workflows
  - Activity, steps
  - What does happen?

- Artifacts
  - models
  - reports, documents
  - What is produced?

- Worker: Architect
  - Who does it?
The domain of architecting

The “what”
- Architecture Qualities
- Architecture Representation

The “why”
- System Features
- S/W Requirements
- System Quality Attributes

The “how”
- Technology
- Process
- Organization

The “who”
- Architect
- Skills
- Stakeholders
- Organization

Wojtek Kozaczynski
“Everything should be made as simple as possible, but not simpler” – Albert Einstein
Models

- An abstract representation of reality that excludes much of the world’s infinite detail.
- The purpose of a model is to reduce the complexity of understanding or interacting with a phenomenon by eliminating the detail that does not influence its relevant behavior.
Abstraction
Modeling Point #1

- Modeling is the ‘art’ of abstraction, knowing what to include in model and what to leave out.

- A model reveals what its *creator believes is important* in understanding or predicting the phenomena modeled.
But…Africa is more than 10 times larger than Greenland!
vs. Peterson’s Projection ➔ Area Accurate
Modeling Point #2

- All models are built with a purpose, the purpose is determined by the model creator.

- Standard models have built in purposes (for example, UML activity diagrams or role activity diagrams).
Figure 1. Front view of physical object
Figure 2. Two possible top views for the same front view
Modeling Point #3

- Systems tend to be complex, our models only abstract limited parts of the entire system (called a view).

- You need multiple views to understand the entire system. We use decomposition, but instead of a hierarchy into views.

- Views must be consistent!
Enterprise Modeling

• Enterprise modeling has to fulfill several requirements to achieve efficient and effective enterprise integration:
  • provide a modeling language easily understood by non-IT professionals, but sufficient for modeling complex industrial environments.
  • provide a modeling framework which:
    • covers the life cycle of enterprise operation from requirements definition to end of life.
    • enables focus on different aspects of enterprise operation by hiding those parts of the model not relevant for the particular point of view.
    • supports re-usability of models or model parts
# Enterprise System Views

<table>
<thead>
<tr>
<th>CIMOSA</th>
<th>ARIS</th>
<th>Zachman</th>
<th>Curtis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Information Organization Resource</td>
<td>Control Data Function Organization</td>
<td>Data Process I/O</td>
<td>Function Behavior Organization or resource information</td>
</tr>
</tbody>
</table>


Frameworks

- Definition: a collection of means and procedures dedicated to a particular field of activity.
  - When used as a reference and a tool, a framework is most often presented as being complete and consistent for the field in question.

- TOGAF framework:
  - Provides a language, an approach, and a set of recommendations covering all facets of enterprise architecture, from organization and strategy, to business and technology, to planning and change management.
  - Not a universal, finished solution,
  - Instead provides a toolbox that can be used by all participants in enterprise architecture, from senior management, CIOs, and business managers to IS architects and project managers.
TOGAF
(The Open Group Architectural Framework)

- A Generic TOGAF framework
  - company adapts it to its own context, for example, by adapting the framework, identifying the specific stakeholders, and so on.
  - TOGAF allows for a phase dedicated to setting up and adapting the framework.
1. Introduction
2. ADM (Architecture Development Method)
3. ADM Guidelines
4. Architecture Content
5. Enterprise Continuum and Tools
6. Reference Models
7. Architecture Capability Framework

TOGAF Document (hundreds of artifacts)
Q: IS the sequence fixed?
Typical path of an ADM cycle

1. Scope definition, involvement of key stakeholders
2. Description of existing and target architectures
3. Gap and impact analysis
4. Architecture roadmap definition
5. Projects and schedule definition
6. Implementation governance

Preliminary phase and phase A

Phases B, C, D

Phases E, F

Phases G, H
TOGAF Repository

- Contains various elements such as models, patterns, architecture descriptions, or deliverables resulting from earlier work and also external elements from standards or other organizations.

TOGAF repository is partitioned:

1. The Metamodel – architectural elements and relationships between them
2. The "architecture landscape," which describes the existing architecture.
3. The reference library, in which templates, patterns, guides, and all elements already implemented and available for reuse are located.
4. The standard information base, containing international norms, tools, and services that must be conformed to.
5. Two parts related to:
   - Repository governance
   - Governance log and the (Figure 4.2).
Structure of the architecture repository

Landscape: ‘as-is’ architecture across enterprise

Architecture continuum:
1. Foundation architecture
2. Common systems architecture
3. Industry specific architectures
4. Organization-specific architectures
**Technical Reference Model**
- defines the components of an information system infrastructure by providing terminology, structure, and rules for interconnection between different components.

**Integrated Information Infrastructure Reference Model**
- Business applications
- Infrastructure applications,
  - such as utilities or development tools
- The application platform,
  - handles application management services, including access, deployment, or location
- Interfaces between components, with details on protocols, exchanges, and programming interfaces
- Service quality
An iteration is a sequence of activities with an established plan and evaluation criteria, resulting in an executable release.
What might some of the models (within architectures) look like?
Unified Process structure

**Process Workflows**
- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment

**Supporting Workflows**
- Configuration Mgmt
- Management
- Environment

**Phases**
- Inception
- Elaboration
- Construction
- Transition

**Iterations**
- Preliminary
- Iteration(s)
- Iter. #1
- Iter. #2
- Iter. #n
- Iter. #n+1
- Iter. #n+2
- Iter. #m
- Iter. #m+1
Architecture and Iterations

Use case Model
Design Model
Implementation Model
Deployment Model
Test Model

Content
Patterns

- A pattern is a solution to a problem in a context
- A pattern codifies specific knowledge collected from experience in a domain
- All well-structured systems are full of patterns
  - Idioms
  - Design patterns
  - Architectural patterns
Design patterns

- Creational patterns
  - Abstract factory
  - Prototype

- Structural patterns
  - Adapter
  - Bridge
  - Proxy

- Behavioral patterns
  - Chain of responsibility
  - Mediator
  - Visitor

- Mechanisms are the soul of an architecture
Modeling a design pattern
Modeling a design pattern
(cont.)
Modeling a design pattern (cont.)

```
: Client
  new
  storeCommand(c)

: Command

: Invoker
  execute()

: Receiver
  action()
```
Architectural patterns

- Distributed
- Event-driven
- Frame-based
- Batch
- Pipes and filters
- Repository-centric
- Blackboard
- Interpreter
- Rule-based

- Layered
- MVC
- IR-centric
- Subsumption
- Disposable
Logical application architecture
Physical application architecture

Client A
- Application
- Business Object Services
- Business Object Engine

Client B
- Application
- Business Object Services
- COM
- MTS
- Beans
- ETS
- Business Object Engine

Client C
- WWW Browser
- Business Object Services
- Business Object Engine

Web Server
- HTML
- CGI
- ASP
- Java
- MTS
- COM
- ETS

Business Object Server
- DCOM
- ADO/R
- CORBA
- Beans

Relational Database Server(s)

Thinner client, thicker server
Complex Internet system

The Second Wave
Paul Dreyfus, Netscape

Client

Server

Application Server

Fulfillment System
Financial System
Inventory System
RDBMS Server

Dynamic HTML, JavaScript, Java plug-ins, source code enhancements

Java, C, C++, JavaScript, CGI

Java, C, C++, JavaBeans, CORBA, DCOM

Native languages
Overview of the UML

- The UML is a language for
  - visualizing
  - specifying
  - constructing
  - documenting

the artifacts of a software-intensive system
Overview of the UML

- Modeling elements
- Relationships
- Extensibility Mechanisms
- Diagrams
Modeling Elements

- Structural elements
  - class, interface, collaboration, use case, active class, component, node

- Behavioral elements
  - interaction, state machine

- Grouping elements
  - package, subsystem

- Other elements
  - note

---

CIS 8090 Enterprise Systems Architectures  Dr. Duane Truex

42
Relationships

- Dependency
- Association
- Generalization
- Realization
Extensibility Mechanisms

- Stereotype
- Tagged value
- Constraint
A model is a complete description of a system from a particular perspective.

Models, Views, and Diagrams

Models

- Use Case Diagrams
- Class Diagrams
- Object Diagrams
- Component Diagrams
- Deployment Diagrams

- Activity Diagrams
- Sequence Diagrams
- Collaboration Diagrams
- Statechart Diagrams

CIS 8090 Enterprise Systems Architectures  Dr. Duane Truex
Diagrams

- A diagram is a view into a model
  - Presented from the aspect of a particular stakeholder
  - Provides a partial representation of the system
  - Is semantically consistent with other views

- In the UML, there are nine standard diagrams
  - Static views: use case, class, object, component, deployment
  - Dynamic views: sequence, collaboration, statechart, activity
Use Case Diagram

- Captures system functionality as seen by users

Use Case Diagram:
- Place phone call
- Place conference call
- Receive phone call
- Receive additional call
- Use scheduler

Actors:
- User
- Cellular network

Relationships:
- «extend»
- extends relationship
- use case
- system boundary
Use Case Diagram

- Captures system functionality as seen by users
- Built in early stages of development

Purpose
- Specify the context of a system
- Capture the requirements of a system
- Validate a system’s architecture
- Drive implementation and generate test cases

- Developed by analysts and domain experts
Class Diagram

- Captures the vocabulary of a system
Class Diagram

- Captures the vocabulary of a system
- Built and refined throughout development
- Purpose
  - Name and model concepts in the system
  - Specify collaborations
  - Specify logical database schemas
- Developed by analysts, designers, and implementers
Object Diagram

- Captures instances and links
Object Diagram

- Shows instances and links
- Built during analysis and design
- Purpose
  - Illustrate data/object structures
  - Specify snapshots
- Developed by analysts, designers, and implementers
Component Diagram

- Captures the physical structure of the implementation
Component Diagram

- Captures the physical structure of the implementation
- Built as part of architectural specification

Purpose
- Organize source code
- Construct an executable release
- Specify a physical database

Developed by architects and programmers
Deployment Diagram

- Captures the topology of a system’s hardware
Deployment Diagram

- Captures the topology of a system’s hardware
- Built as part of architectural specification
- Purpose
  - Specify the distribution of components
  - Identify performance bottlenecks
- Developed by architects, networking engineers, and system engineers
Sequence Diagram

- Captures dynamic behavior (time-oriented)
Sequence Diagram

- Captures dynamic behavior (time-oriented)

- Purpose
  - Model flow of control
  - Illustrate typical scenarios
Collaboration Diagram

- Captures dynamic behavior (message-oriented)
Collaboration Diagram

- Captures dynamic behavior (message-oriented)

- Purpose
  - Model flow of control
  - Illustrate coordination of object structure and control
Statechart Diagram

- Captures dynamic behavior (event-oriented)
Statechart Diagram

- Captures dynamic behavior (event-oriented)

- Purpose
  - Model object lifecycle
  - Model reactive objects (user interfaces, devices, etc.)
Activity Diagram

- Captures dynamic behavior (activity-oriented)
Activity Diagram

- Captures dynamic behavior (activity-oriented)
- Purpose
  - Model business workflows
  - Model operations
Architecture and the UML

Design View
- Classes, interfaces, collaborations
- Active classes

Implementation View
- Components
- Nodes

Process View

Deployment View

Use Case View
- Use cases

Organization
- Package, subsystem

Dynamics
- Interaction
- State machine