Driving MDA with UML:
Principles and Practices

Junichi Suzuki, Ph.D.
jxs@computer.org
http://www.jks.la/jxs/
School of Information and Computer Science
University of California, Irvine

Who am I?

• Research fellow, UC Irvine (2000–)
  – biologically-inspired software designs for scalable and adaptable distributed computing
• Ph.D. from Keio U (2001)
• ex– Technical director, Object Management Group Japan
• ex.ex– Technical director, Soken Planning Co., Ltd.
Where is UC Irvine?

• UCI (U of California, Irvine)
  – One of eight UC system universities
• Irvine
  – in between LA and San Diego
  – reported by FBI, as the safest city in the US
  – 1 hour to LA downtown
  – 10 minutes to Newport Beach
  – 20 minutes to Huntington Beach
  – 20 minutes to Anaheim Disneyland
  – 5 hours to Las Vegas

Overview

• MDA (Model Driven Architecture)
  – Model transformation and integration
    • Patterns and technologies for model transformations
• MDA Practices
  – Standardization effort based on MDA principles
    • OMG Super Distributed Objects specification
  – MDA practice for ubiquitous computing
    • Bio-Networking Architecture
Traditional Modeling and Development

Domain analysts, Modelers, Designers, Developers

Traditional modeling/dev tools

Platform/technology expertise

Applications

MDA–based Modeling and Development

Platform experts

Platform expertise

MDA tools

Application developers

Technology (logic impl) expertise

Applications

Domain experts

Domain expertise
Goals in MDA

- Model continuation
  - Maximizing model continuation during software development process.
- Separation of concerns
  - Maximizing separation of concerns

Benefits from MDA

- Reduced software development cost
- Reduced software development time
- Rapid and smooth integration of legacy and emerging technologies
Model Transformation and Integration

- Model transformation
  - Domain specialization
  - Platform specialization
- Model integration
  - Model weaving

![Diagram of Model Transformation Process]

- **MDA tools**
  - Platform experts
  - Application developers
  - Domain experts
  - Technology (logic impl) expertise

**PIM**
- MOF
- XMI

**PSM**
- Model maintenance and exchange
- Model transformations
- Generates/derives

**Applications**
- Source code
- Configuration files
- Application
Model Transformation

Platform Independent Model (PIM)

- Modeled with
  - UML
  - ADL/ASL
  - Conceptual drawings
- may incorporate several software patterns
  - Architectural, analysis and design patterns
Model Transformation

- 2 dimensions of model transformation
  - Domain specialization
  - Platform specialization
- Several forms of model transformation
  - Manual transformation
  - Automatic transformation
Technologies for Model Transformations

• UML profiles
  – for EJB
  – for CORBA
  – for Realtime scheduling

• Action semantics
  – allows modelers to embed actions (behaviors) into model elements.

UML Profiles

• A UML profile
  – provides a means to specialize UML models to a specific domain or implementation technology.
  – is defined with the UML extension mechanism
    • i.e. stereotypes, tag definition/tagged values, and constraints
  – may extend the UML standard meta model.
    • Virtual meta model
## UML Profile for EJB

- [http://jcp.org/jsr/detail/26.jsp](http://jcp.org/jsr/detail/26.jsp)

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaInterface</td>
<td>Java interface</td>
</tr>
<tr>
<td>EJBHome Interface</td>
<td>Home interface</td>
</tr>
<tr>
<td>EJBRemote Interface</td>
<td>Remote interface</td>
</tr>
<tr>
<td>EJBImplementation</td>
<td>Implementation class of a bean</td>
</tr>
<tr>
<td>EJBSessionBean</td>
<td>Session bean</td>
</tr>
<tr>
<td>EJBEntityBean</td>
<td>Entity bean</td>
</tr>
</tbody>
</table>

```java
import javax.ejb.*;
import java.rmi.Remote;
import java.rmi.RemoteException;

public class CustomerServerBean implements Javax.ejb.SessionBean {
    public CustomerServerBean() {
        // Constructor
    }
    // EJB operations
    public String getCustomerEntry(String name) {
        // Implementation
    }
    // Other methods...
}
```
Super Distributed Objects (SDOs)

- The goals of the OMG Super Distributed Objects (SDOs) DSIG (domain SIG) are to
  - provide a standard computing infrastructure that incorporates massive number of objects (SDOs) including hardware devices and software components
  - deploy SDOs in highly-distributed and ubiquitous environments, and
  - allow SDOs to seamlessly interwork with each other in a less centralized manner.

- SDO is...
  - a logical representation of hardware devices and software components operating on highly-distributed and ubiquitous networks.

History and status:
- The SDO RFI issued (‘00), and responses gathered (‘01)
  - from 10 organizations including UCI
- The SDO white paper published (‘01)
  - by Hitachi, GMD Fokus and UCI
- The first RFP published (Jan. 02), which
  - solicits the resource data model for SDOs, and interfaces to access and manipulate resource data model.
  - sdo/02–01–04
- The initial proposals submitted (Sept. 02)
  - by Hitachi, GMD Fokus and UCI
  - sdo/02–09–01, sdo/02–09–02
  - 28 organizations on the voting list
- The revised joint proposal was submitted in March 2003.
  - by Hitachi, GMD Fokus and UCI
  - sdo/02–01–04
- The submission was recommended for adoption.
SDO PIM and PSM Specification

- Addresses information and computational aspects for SDOs
  - Information aspect
    - Resource data model, used to define the capabilities and properties of SDOs.
  - Computational aspect
    - A set of interfaces, used to access and manipulate resource data model.
- Defines a PIM and PSM for each of the aspects.
  - UML used to define PIM.
  - CORBA IDL used to define PSM.

SDO Resource Data Model
SDO Interfaces

SDO

Interfaces:
- SDO
- Organization
- Configuration
- Monitoring
- Callback

<<Interface>>
Configuration
+ setDeviceProfile(dProfile : DeviceProfile) : void
+ addServiceProfile(sProfile : ServiceProfile) : void
+ addOrganization(organization : Organization) : void
+ removeDeviceProfile() : void
+ removeServiceProfile(serviceID : String) : void
+ removeOrganization(organization : Organization) : void
+ getConfigParameter() : ParameterList
+ getParameterValue(name : String) : any
+ setParameterValue(name : String, value : any) : void
+ getConfigurationSets() : ConfigurationSetList
+ addConfigurationSet(configurationSet : ConfigurationSet) : void
+ removeConfigurationSet(configurationSetID : String) : void
+ activateConfigurationSet(configID : String) : void

SDO/application

SDO

DeviceProfile
- deviceType : String
- manufacturer : String
- model : String
- version : String
- properties : NVList

ServiceProfile
- id : String
- InterfaceType : String
- properties : NVList
- serviceRef : SDOService

SDOEntity
- id : String
- doType : String
- members
- owner
- organizations
- organizations
+ properties : NVList

Organization
- direction : boolean

OrganizationProperty
- properties : NVList

SDO Interfaces

SDO/application

SDO

<<Interface>>
CORBA PSM

- CORBA PSM for SDO resource data model and interfaces

```plaintext
module SDOPackage {
    interface SDO;
    interface SDOService;
    interface SDOSystemElement;
    interface Configuration;
    interface Monitoring;
    interface Organization;
    interface SDO : SDOSystemElement {
        UnqiueIdentifier get_id();
        string get_SDO_type();
    }
}
```

Scope of SDO PIM/PSM
The Bio-Networking Architecture: An Example of SDO Implementations

- Computer network environment is seamlessly spanning locations engaged in human endeavor.
- Need a self-organizing network that supports
  - *scalability* in terms of # of objects and network nodes,
  - *adaptability* to changes in network conditions,
  - *availability/survivability* from massive failures and attacks,
  - *simplicity* to design and maintain.

- Our solution: *apply biological concepts and mechanisms to network application design*
  - Biological systems have overcome the above features.
    - e.g. bee colony, bird flock, fish school, etc.
- The Bio-Networking Architecture is a new framework
  - for developing large-scale, highly distributed, heterogeneous, and dynamic network applications.

Biological Concepts Applied

- Decentralized system organization
  - Biological systems
    - consist of autonomous entities (e.g. bees in a bee colony)
    - no centralized (leader) entity (e.g. a leader in a bird flock)
      - Decentralization increases scalability and survivability of biological systems.
  - The Bio-Networking Architecture
    - biological entities = cyber-entities (CEs)
      - the smallest component in an application
      - provides a functional service related to the application
      - autonomous with simple behaviors
        » replication, reproduction, migration, death, etc.
      » makes its own behavioral decision according to its own policy
    - no centralized entity among CEs
• **Emergence**
  - Biological systems
    • Useful group behavior (e.g. adaptability and survivability) emerges from autonomous local interaction of individuals with simple behaviors.
      - i.e. not by direction of a centralized (leader) entity
      - e.g. food gathering function
        » When a bee colony needs more food, a number of bees will go to the flower patches to gather nectar.
        » When food storage is near its capacity, only a few bees will leave the hive.
  - The Bio-Networking Architecture
    • CEs autonomously
      - sense local/nearby environment
        » e.g. existence of neighboring CEs, existence/movement of users, workload, availability of resources (e.g. memory space), etc.
      - invoke behaviors according to the condition in a local/nearby environment
      - interacts with each other

• **Lifecycle**
  - Biological systems
    • Each entity strives to seek and consume food for living.
    • Some entities replicate and/or reproduce children with partners.
  - The Bio-Networking Architecture
    • Each CE stores and expends *energy* for living.
      - gains energy in exchange for providing its service to other CEs
      - expends energy for performing its behaviors, utilizing resources (e.g. CPU and memory), and invoking another CE’s service.
    • Each CE replicates itself and reproduce a child with a partner.
• Evolution
  – Biological system
    • adjusts itself for environmental changes through species diversity and natural selection
  – The Bio-Networking Architecture
    • CEs evolve by
      – generating behavioral diversity among them, and
        » CEs with a variety of behavioral policies are created by human developers manually, or through mutation (during replication and reproduction) and crossover (during reproduction)
      – executing natural selection.
        » death from energy starvation
        » tendency to replicate/reproduce from energy abundance

• Social networking
  – Biological systems (social systems)
    • Any two entities can be linked in a short path through relationships among entities.
      – not through any centralized entity (e.g. directory), rather in a decentralized manner.
      – six decrees of separation
  – The Bio-Networking Architecture
    • CEs are linked with each other using relationships.
      – A relationship contains some properties about other CEs (e.g. unique ID, name, reference, service type, etc.)
    • Relationships are used for a CE to search other CEs.
      – Search queries originate from a CE, and travel from CE to CE through relationships.
    • The strength of relationship is used for prioritizing different relationships in discovery.
      – A CE may change its relationship strength based on the degree of similarity between two CEs.
      – The stronger relationship is likely to lead a query to a successful discovery result.
CE’s Structure and Behaviors

- **Attributes**
  - ID
  - Relationship list
  - Age
  - …etc.

- **Body**
  - Executable code
  - Non-executable data

- **Behaviors**
  - Energy exchange and storage
  - Communication
  - Migration
  - Replication and reproduction
  - Death
  - Relationship establishment
  - Social networking (discovery)
  - Resource sensing
  - State change

Design Strategies of the Bio-Networking Architecture

- **Separate cyber-entity (CE) and Bio-Networking Platform (bionet platform),**
  - **Cyber-entity (CE)**
    - mobile object (agent) that provides any service logic
  - **Bionet platform**
    - middleware system for deploying and executing cyber-entities

- **Model CE and bionet platform with UML**
  - Using SDO PIM

- **Implement CE and bionet platform in Java and CORBA**
  - Using SDO CORBA PSM
Architecture of the Bio-Networking Platform

A Cyber-entity (CE) is an autonomous mobile object. CEs communicate with each other using FIPA ACL.

A CE context provides references to available bionet services.

Bionet services are runtime services that CEs use frequently.

Bionet container dispatches incoming messages to target CEs.

Bionet message transport takes care of I/O, low-level messaging and concurrency.

Bionet class loader loads byte code of CEs to Java VM.

Scope of Bionet Implementation

SDO PIM ➔ SDO (CORBA) PIM

SDO PIM/PSM spec

Bionet domain model ➔ Bionet implementation
Bionet domain model

SDO PIM

Patterns

PIM

model transformations

model transformation

PSM

Model transformations

generates/derives

Source code

Configuration files

Application

SDO CORBA PSM
Java interface
CORBA IDL

UML