Executable Enterprise Modeling with UML

Presented to:
OMG Workshop on UML in the Enterprise
December 3-6, 2001
By:
Michael Latta
Yngvar D. Tronstad
Overview

- Architecture
- OO Modeling
  - Structure
  - Behavior
  - State
- Push
- Execution Engine
- J2EE Apps Server
- JMS/JCA
- Lessons Learned
Problem space and Challenges

An example integration environment, using TIBCO middleware

Configured piece by piece, it requires organizations to carefully coordinate all of the implementation artifacts to build the final end-to-end solution.
UML Authoring in EAI

- UML OO modeling support
- Collaborative Modeling Environment
  - Different but related modeling domains/facets
  - Consistency checking
  - Multi-user support
  - Persistence
  - Versioning and Configuration Management
- Model based design with validation
- Direct deployment and execution of the models
Why Executable Models?

- **The Model and the Implementation will not drift apart**
  - Automate the generation of the execution system from the model itself
  - Allows verification prior to deployment
- **Removes Ambiguity**
  - Forces semantics to be defined
  - Makes behavior more visible than when in code
- **Provides medium for Communication**
  - Business Owners and Designers can more easily relate
  - Highlights general process flow over details
What is an Executable Model?

- A model complete enough to be executed
- Can be augmented by custom code, but only at the leaf level of the behavior tree
- Defines structure of business objects and their behavior, “smart objects” that can be distributed
- Embraces interface to external systems
- Scales to support the demands of an enterprise class customer
How to provide an Executable Model

- Start with UML
- Add missing semantics
- Clarify variations
- Generate database from structure model
- Execute behavior in process server
Challenge in Executable Models: Add Missing Semantics

- Define concrete semantics between state machine and their context
  - How are state machines started?
  - Can an instance have multiple active state machines?
- How is an incoming message converted to an event to be delivered to a state machine
- How is a generated event converted to an outgoing message
- Ambiguity in handling SynchState
Challenge in Executable Models: Clarify Variations

- Resolving transition selection
- Processing transitions in series or concurrently
- Transaction boundaries around state machine semantics
- Mapping of UML structure models to RDBMS
- Selection of action representation (1.x or Action Semantics) and the concrete syntax for actions
- Selection of primitive actions, functions, data types
- Implementation of custom code actions
Executable Model Framework

JMS MSG → J2EE Application Server → JMS MSG

Authoring
- PEAR
  - bus objs

Business Objects
- Message Definitions
- Event Definitions
- Transformations
- Behaviors

OHANA

Process Manager Execution Engine

- EE
- MG
- Action Semantics

- POAL
  - Meta-schema API

- PEAR

- RDB
  - Bus Obj Schema
  - PM Base Schema

EE
MG
Action Semantics
POAL
PEAR
RDB
Bus Obj Schema
PM Base Schema
Class Diagram

Customer
- customerID
- firstName
- lastName
- address
- city
- state
- zip

1 orderOwner

Order
- orderID
- quantity

0..* customerOrder

Association

UML Notation for Classes
UML Notation for Behavior

Object

Composite State
Initial State
Transition
Simple State
Transition
Final State
Transaction Model

Start Message

Start Event

Initial State

Simple State

Final State

Transaction 1

Entry Action

Do Action

Exit Action

Transaction 2

Entry Action

Do Action

Exit Action

Transaction 3

Entry Action

Do Action

Exit Action

Transaction 4

Transaction 5

Entry Action

Do Action

Exit Action
Execution Engine Processing

1. Start Message
4. Input Message

2a. DoAction Simple State
2b. DoAction Composite State
5. Do Action Final State
Message Driven Beans Support Transaction Model
Message Traffic Example

Web Front End

Process Manager
Execution Engine

Billing System

Provisioning System

Business Object

Enterprise    Customer

Create Customer

Create Customer Billing

Create Billing Account

Billing Account Created

Place Order

Provision Order

Order Provisioned

Bill For Service

Account Billed
Enterprise Objects

Enterprise
- name : String [rw]

Customer
- billingAccountNumber : String [rw]
- city : String [rw]
- firstName : String [rw]
- lastName : String [rw]
- state : String [rw]
- streetAddress : String [rw]
- telephoneNumber : String [rw]
- zip : String [rw]

Order
- orderID : String [rw]
- serviceLevelAgreement : String [rw]

DSLOrder
- connectionSpeed : String [rw]

Enterprise Object Behavior

Create Customer

1. StartEvent CreateCustomer (from web)

2. Send Message CreateCustomerBilling (to Customer)
Customer Object Behavior

Customer

Create Billing

Receive StartEvent CreateCustomerBilling
(from Enterprise)

4. Receive Message BillingAccountCreated
(from Billing Stub)

BillingAccountCreatedEvent

3. Send Message BillingAccountCreation
(to Billing Stub)
Customer Object Behavior (cont.)

Customer

Place Order

5. Receive StartEvent
   PlaceCustomerOrder
   (from Web)

7. Receive Message
   OrderProvisioned
   (from Provision Stub)

9. Receive Message
   OrderBillingCompleted
   (to Billing Stub)

CreateOrder

ProvisionDSL

OrderProvisionedEvent

BillForDSL

OrderBillingCompletedEvent

6. Send Message
   ProvisionOrder
   (to Provision Stub)

8. Send Message
   BillForOrder
   (to Billing Stub)
J2EE Deployment Architecture

**J2EE Server**
- EJB Container
- MDB Instance

**JMS Middleware**
- Topic
- Msg

**Web App**
- APACHE

**Prov**
- Bill
Summary

- **Process Manager**
  - UML Based OO Authoring and CM
  - Execution of Smart Distributed Objects

- **J2EE Application Server**
  - Scalability
  - Transaction control

- **JMS/JCA**
  - Distribution, Delivery, Status
Demonstration is available

Show Me!