IDE for MDA
Based on Executable Enterprise Modeling
with
UML/CWM

Presented to:
OMG’s Third Workshop on UML for
Enterprise Applications:
Model Driven Solutions for the Enterprise
October 22, 2002
By:
  Michael Latta
  Yngvar D. Tronstad
Agenda

- **Overview**
  - EAI Market Trend
  - The Challenges
  - The Next Generation EAI suites

- **Executable Model Overview**
  - ODM ➔ AIM
  - AIM (Process)
  - AIM (Process + Data)
  - AIM (Execution)

- **Summary, Future Direction, Q&A**
Cogility Product Suite

Mission Statement

To simplify the design and management of diverse artifacts of EAI by creating a single extensible tool where the artifacts can all be modeled and then generated from the model and automatically deployed into a scalable execution environment, based on OMG’s MDA concept.
Top 10 IT Spending Areas for 2002

1. Application Integration - 35%
2. Security Software - 35%
3. E-Commerce Initiatives - 32%
4. Windows 2000/XP Upgrade - Desktop - 26%
5. Web Site Enhancements - 25%
6. XML Based Applications - 23%
7. Windows 2000/XP Upgrade - Server - 23%
8. ERP Software / ERP Upgrade - 22%
9. Content management for Web Site - 21%
10. Network Equipment - 21%

Source: Morgan Stanley CIO Survey, 2002
Market Opportunity

Integration Framework

- Most integration projects are using hand-coded point-to-point approaches instead of using an integration framework
- Handcoded integration is deceptive, it seems easier and cheaper to implement then integration framework, but it ends up being far more expensive and aggravating to maintain
- Companies that use integration frameworks spend less on maintenance and more on projects that add value

Integration Framework Bottom Line
- 11% less on
  - Maintenance
  - Services
  - Headcounts

Source: AMR Research, October 2001, Survey of 686 companies over 14 key vertical markets
Problem Space and Challenges

Point-to-Point Integration

In the past, complex point-to-point integrations were used to construct an end-to-end business process.

The disadvantages:

- Interface Complexity
- Inability to quickly modify processes following initial implementation
- Expensive to maintain and extend
Large Scale Integration Problem

Required Transformations

- Point-To-Point: $n(n-1)$
- Star Schema: $2n$

Number of Transformations vs. Number of systems
Implementation Costs vs License Costs

8:1

Point to Point Integration
Problem Space and Challenges

State of the Art integration environment, using middleware solutions

Configured piece by piece, organizations must carefully coordinate all of the implementation artifacts to build the final end-to-end solution.
Typical Conceptual walls separate the Life Cycle Steps

- System Engineering
  - Ops Concept
  - System Rqmts
  - Software Reqmt

- Software & Hardware Engineering
  - Software Design
  - Code Generation
  - Compiler
Implementation Costs vs License Costs

8:1 Point to Point Integration

4:1 Message Based Middleware
Value Added by MDA Tool

- EAI complexity can be addressed by ingenious tools
  - Holistic toolset for end-to-end visibility
  - Top-down abstraction and decomposition that through successive refinements maintains the business logic as an invariant from high level design to implementation
  - Auto-deploy to multiple execution platforms
Ceira’s Two Part Architecture

Part 1: Authoring

Part 2: Execution

Third Party Messaging and Applications
Value Added by MDA

- Divorce the details of the physical implementation from the logical view of the system, while ensuring completeness
  - Future Proof: Implementation mechanism can be modified without affecting the logical model

- Build-in layers of indirection (isolation layers) to reduce the impact of changes to any one detail

- Create a mechanism for communicating the EAI strategy in the form of the EAI model that is executable, ensuring that the model and the implementation do not drift apart
Implementation Costs vs License Costs

Solution that compress Design and Implementation Timeframes

- 8:1: Point to Point Integration
- 4:1: Message Based Middleware
- 1:1: Executable Models

© 2002 Ceira Technologies Inc.™
Agenda

☐ Overview
  ☐ EAI Market Trend
  ☐ The Challenges
  ☐ The Next Generation EAI suites

☐ Executable Model Overview
  ☐ Extensible Framework
  ☐ ODM ➔ AIM
  ☐ AIM (Process)
  ☐ AIM (Process + Data)
  ☐ AIM (Execution)

☐ Summary, Future Direction, Q&A
Executable Model Overview-ODM

- Repository based multi-user model configuration management for Application Developers
- Repository based multi-user model configuration management for Application Users

ODM

ODM Data
Application Models

Application

Application Data
User Models
Extensible Tool

Executable Model Overview-Application Development

- Object Oriented UML based modeling environment
- Each application is modeled
  - The application model defines the application boundaries and artifacts
  - An application can be extended by modifying the application model and specifying the new behavior
Extensible Application Authoring Environment

Object Design Manager
This is an application that is used to create Application Models for app development

PEAR
This is used to model data structures which get pushed into a repository as UML Schema

AIM (Process)
This is the Application that allows the merging of MIM data models with behavior models (SM)

AIM (Process + Transforms)
This allows the modeling of DIMs with the MIM, with CWM based data transformations

Applications can be easily extended to tailor them to new tasks
Application Integration Modeler (AIM)

Overview

Modeling Environment

- Application Integration Model
- UML Model
- OCL Based Execution Details

Execution Environment

- J2EE Application Server
- UML Enactment
- Relational Database Persistence
The AIM tool is a multi-user, repository based, configuration managed EAI Integrated Development Environment

- Fulfills the modeling aspect of OMG’s Model Driven Architecture (MDA)
- Star-schema modeling
- OMG UML (1.4) based object modeling (classes, associations)
- Process modeling using UML State Machines
- OCL for low level logic and execution
- OMG’s Common Warehouse Metamodel (CWM 1.0 is based on MOF 1.3) based reusable transformation modeling
- Message and Event modeling
- Implementation of OMG’s Object Constraint Language for specifying low level execution details as required by the MDA.
Master Information Model
This is an abstract model of the entities that make up the enterprise

Process Model
The behavior of the entities as they participate in different business activities

Event Model
The external events that will trigger state transitions in the State Machine

Message Model
The messages that the system must send to other systems requesting actions

M2E/E2M Transforms
Transformation model that interprets messages received and events generated

Modeling Aspects
The AIM (Process + Data) application allows the user to model the business processes that an enterprise entity participates in as well as the information models of the applications to be integrated and transformations between them.

Eliminates point-to-point transformations by extending the star-schema concept to the EAI domain (the Master Information Model or MIM concept)
  - Exponential reduction in transformation artifacts through reuse
    • $2n$ vs. $n(n - 1)$

Allows for an extensible and holistic environment in which to model the entire EAI scope in a single tool.
Application Integration Modeler With Data Transformation Extensions

Modeling Aspects

- **Process Model**: This is an abstract model of the entities that make up the enterprise and their behavior (MIM).
- **Reusable Transformations**: Models transformation components that convert between information models.
- **Transformation Chains**: Models reusable transformation units that can be connected to the message triggers.
- **Distinct Information Model**: Models of the relevant schema of the remote systems that need to be integrated.
- **M2D/D2M Transforms**: Transformation models that convert messages to and from system specific definitions.
AIM (Process + Data): Star Schema

AIM uses the Star Schema pattern, extended for the EAI Domain

- Master Information Model (MIM)
- Distinct Information Model (DIM)
- Representing Enterprise Applications
Each Information Model is created using UML Modeling Techniques
The Model contains a reusable library of CWM based transformations.
The Model contains a JMS messaging layer that is active on deployment

- Reusable CWM Based Transforms
- JMS Based Messaging Layer for triggering transformations and state machines and sending notifications
- Master Information Model (MIM)
- Distinct Information Model (DIM) representing Enterprise Applications
- UML
- CWM
AIM: Transformation Chains

The Model contains Transform Chains attached to the message layer or states for sequencing transformations

- Reusable CWM Based Transforms
- JMS Based Messaging Layer for triggering transformations and state machines and sending notifications
- Transform Chains Attached to the Message Layer or states for transformation sequencing
- Master Information Model (MIM)
- Distinct Information Model (DIM) representing Enterprise Applications
Triggering Transformations: Direct Mode

- Data Object Set
Information Models may contain hierarchical class structures, associations and numerous state machines that define their object behavior.
AIM: Embedded Transformations

Messages can also trigger behavior embedded in processes to allow finer control over transformation sequencing.

- Reusable CWM Based Transforms
- JMS Based Messaging Layer for triggering transformations and state machines and sending notifications
- Transform Chains Attached to the Message Layer or states for transformation sequencing
- Master Information Model (MIM)
- Distinct Information Model (DIM) representing Enterprise Applications
AIM – Summary
- **Data Transformations based on the OMG’s Common Warehouse Metamodel (CWM) standard**
  - Even the reduced number of transformation artifacts are built using a standard that promotes reuse

- **Allows for the modeling of reusable feature-level transformations (Feature Maps)**

- **Allows for the modeling of Classifier Maps, which are transformations between DIM classes in the periphery and the MIM classes (hub), in terms of re-usuable Feature Maps**

- **Allows for the the modeling of Transformation Maps, which transform arbitrary sets of data types between a DIM and the MIM. Transformation Maps reuse Classifier maps as needed**

- **Allows for the modeling of Transformation Chains, which are comprised of re-usable Transformation Maps (or other chains) in series or parallel (Decomposable re-use)**
Transformation Chains can be triggered by messages and can also publish messages destined for remote systems

- This allows the modeling of standard system responses to multiple stimuli using reusable components for quick assembly

- The Chain itself is reusable
  - Reusable as a component in another chain
  - Directly reusable because it can respond to similar messages from different systems with the same behavior (due to the Transport Isolation Layer)

- Combines the modeling of data and behavior in the same environment with several layers of indirection providing separation of concern and ensuring that changes are localized to a limited area of influence

- Never need to build logic more than once
Application Integration Modeler: Model Execution

Product Overview

- **Transaction Integrity**: Uses J2EE semantics and transactional Persistence to maintain transactional integrity.
- **Persistent Object Access Layer**: Seamlessly performs object persistence activities (Create, Read, Update, Delete) for ERA.
- **State Machine Enactment**: Executes the UML State Machine Diagram by evaluating OCL specified state actions.
- **Data Transformations**: Executes data Transformation Units by Decomposing into constituents.
- **Message Transformation**: This component executes OCL instructions to map data between messages and DIMs.
- **OCL Execution Engine**: OCL Executor evaluates all OCL expressions that are used to specify low-level details.
- **JMS MDB Layer**: Receives messages from and sends messages to Systems that are being integrated.

**Diagram:**
- **OCL Execution**
- **State Machine Enactment**
- **Data Transformation Execution**
- **Message Transformation Execution**
- **Persistent Object Access Layer**
- **Combined Data And Message Transaction**
Model Execution is achieved by pushing the model into a scalable execution environment

- Fulfills the execution aspect of OMG’s Model Driven Architecture (MDA)
- Information Models are pushed into a relational database as database schema
- All Message Schemas are pushed into the database as execution metadata
- All CWM based transformations described using OCL are compiled and stored in the database
- All State Machines and associated actions are compiled and stored in the database
- All Message transformations are stored in the database
- Specified J2EE Application Server is configured with JMS topics and Message Driven Beans
- Web Service based Admin Console is installed on the Application Server
Agenda

- Overview
  - EAI Market Trend
  - The Challenges
  - The Next Generation EAI suites

- Executable Model Overview
  - ODM ➔ AIM
  - AIM (Process)
  - AIM (Process + Data)
  - AIM (Execution)

- Summary, Future Direction, Q&A
A MDA approach that is UML based ALL the Way from high level abstraction to low level details

- UML ➔ Business Logic (Classes, Association and StateMachine)
- CWM/UML ➔ Data modeling and transformation
- UML/OCL ➔ Authoring-time definition and Run-Time execution
- UML ➔ Run-time persistense for Classes, Associations and SM

Ensuring continuum from Analysis to Implementation
Future Direction

Overview

- Introduce aspects of UML 2.0 (U2P) Enhancements that support Executable Models
- Top to bottom seamless semantics – no gaps as before
- Introducing views that have meaning for execution
  - Sequence Diagrams containing valid execution
  - Use Cases which have defined semantics to lower levels of decomposition
- Uniform treatment of the system as classes and behaviors
- Cleaner layering, which should make it easier to present to users and at the same time provide better visibility and support for executable models
Use Cases

- Much better defined semantics.
- Less use of English to convey semantics.
- Cleaner decomposition to other model elements.
Future Direction

Sequence Diagrams

- Defined relationship to execution semantics.
- Constrains set of valid executions.
- Can contain control structures, and other aspects that make them more than just “examples” of execution.
- Each diagram defines 3 categories of sequences: valid, invalid, and out of scope.
- This allows a set of sequence diagrams to cover the space without conflict
Future Direction

Richer Execution Semantics

• Activities add new execution semantics.
• Action Semantics integrated into common behavior framework.
• State Machines still available with some enhancements.
• Components are now explicitly modeled, which should enhance reuse
Future Direction

Areas for Improvement

- Need the ability to model transaction boundaries and behavior.
  - A central part of most computing that is still missing.
- Need formalisms for failure handling, error recovery, and rework.
- Need to deal with versioning of system models and the relationship to executing systems.
  - Incremental deployment
  - Migration of active executions
  - Concurrent execution of different versions