Introduction to the Eclipse Modeling Framework

Ed Merks
Macro Modeling
Model Driven Software Development

- Software is focused on manipulating data
- Data has abstract structure
  - It can be described at a high level
  - It can be represented in different ways
  - It’s always a model of something
- The description of the data is yet more data
  - It’s commonly referred to as metadata
  - Meta is a bit confusing
  - The model of a model is a model
- Whether it’s recognized or not, models drive software development
Eclipse Modeling Framework

• A simple, pragmatic, Java-based approach that provides
  – The Ecore API for describing models
  – The EObject API for manipulating instances
  – A resource framework for RESTful persistence
  – A generator framework for producing development artifacts
  – A runtime along with utilities for traversing, indexing, copy, change recording, and so on
  – Tools for working with models and their instances

• EMF was used to develop EMF
A Brief History of EMF

- Started at IBM in the late 90’s
  - It supported Object Management Group (OMG) specifications
  - It implemented Meta Object Facility (MOF)
  - It used XML Metadata Interface (XMI)
  - It’s closely related to Java Metadata Interface (JMI)

- Problems surfaced for adopters
  - The MOF model was far too complex
  - The generated code and runtime were bloated and performed poorly

- ETools Modeling Framework (EMF) was kicked off in 2000
  - Boiled MOF to its essential components resulting in Ecore
  - Revamped the runtime and tools to make them lean and mean

- Contributed to Eclipse in September 2002
  - Rebrand as the Eclipse Modeling Framework
  - Feedback to OMG resulting in Essential MOF/Complete MOF split
Ecore: The Model of Models

• A simple model for describing models
  – Classification of objects
  – Attributes of those objects
  – Relationships/associations between those objects
  – Operations on those objects
  – Simple constraints on those objects, and their attributes and relationships

• Ecore is self describing, i.e., it is its own model

• Models higher up in the meta levels tend to all look the same
  – They begin to conform to our mental model
Relationship of Ecore to Other Models
A Model is a Model is a Model

UML

XML Schema

```xml
<xs:complexType name="Node">
  <xs:sequence>
    <xs:element name="children" type="tree:Node" minOccurs="0" maxOccurs="unbounded" ecore:opposite="parent"/>
  </xs:sequence>
  <xs:attribute name="label" type="xsd:string"/>
</xs:complexType>
```

Ecore

Java

```java
public interface Node {
    String getLabel();
    void setLabel(String value);
    List<Node> getChildren();
    Node getParent();
    void setParent(Node value);
} // Node
```
Ecore Overview
Ecore Data Types
Ecore Annotations and EObject
Ecore Generics
The Tree Ecore Model

EPackage
- name: tree
- nsURI: http://www.example.org/tree
- eClassifiers: Node

EClass
- name: Node
- eStructuralFeatures: label, children, parent

EAttribute
- name: label
- eType: EString
- lowerBound: 0
- upperBound: 1

EReference
- name: children
- eType: Node
- lowerBound: 0
- upperBound: 1
- containment: true
- eOpposite: parent

EReference
- name: parent
- eType: Node
- lowerBound: 0
- upperBound: 1
- containment: false
- eOpposite: children
The Tree Ecore Model Serialized as XMI

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    name="tree"
    nsURI="http://www.example.org/tree"
    nsPrefix="tree">
<eClassifiers xsi:type="ecore:EClass" name="Node">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="children" upperBound="-1"
        eType="#/Node" containment="true" eOpposite="#/Node/parent"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="parent"
        eType="#/Node" eOpposite="#/Node/children"/>
</eClassifiers>
</ecore:EPackage>
```
<?xml version="1.0" encoding="UTF-8"?>
<emof:Package xmi:version="2.0"
   xmlns:xmi="http://www.omg.org/XMI"
   xmlns:emof="http://schema.omg.org/spec/MOF/2.0/emof.xml"
   xmi:id="tree"
   name="tree"
   uri="http://www.example.org/tree">
  <ownedType xmi:type="emof:Class" xmi:id="tree.Node" name="Node">
    <ownedAttribute xmi:id="tree.Node.label" name="label"
      isOrdered="true" lower="0">
      <type xmi:type="emof:PrimitiveType"
        href="http://schema.omg.org/spec/MOF/2.0/emof.xml#String"/>
    </ownedAttribute>
    <ownedAttribute xmi:id="tree.Node.children" name="children"
      isOrdered="true" lower="0" upper="*" type="tree.Node"
      isComposite="true" opposite="tree.Node.parent"/>
    <ownedAttribute xmi:id="tree.Node.parent" name="parent"
      isOrdered="true" lower="0" type="tree.Node"
      opposite="tree.Node.children"/>
  </ownedType>
    <nsPrefix>tree</nsPrefix>
  </xmi:Extension>
</emof:Package>
A Tree Instance Model

Node:
- label: Root
- children: A, B
- parent: 

Node:
- label: A
- children: X
- parent: Root

Node:
- label: X
- children: 
- parent: A

Node:
- label: B
- children: Y
- parent: Root

Node:
- label: Y
- children: 
- parent: B
A Tree Instance Model Serialized as XMI

<tree:Node xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:tree="http://www.example.org/tree"
    label="root">
    <children label="A">
        <children label="X"/>
    </children>
    <children label="B">
        <children label="Y"/>
    </children>
</tree:Node>
The EMF Generator Model

- The GenModel is a decorator for tailoring the generated code
EMF in Action

• Demo time!
  – Show how to create the Ecore Tree model from scratch using the Sample Ecore Editor
  – Show how to use Ecore Tools for diagrams
  – Show how to exploit dynamic models to create Tree instances
  – Demonstrate the interchangeable nature of models
    • Generate the Java realization
    • Export to XML Schema
    • Show how these round trip
    • Show how to run the example
    • Show how to run the generated editor
Summary

- EMF the defacto standard reference implementation
- EMF is a low cost modeling solution for Java
  - SD Times ranks it “top shelf” even relative to pricey commercial software
- It exploits the models already underlying the application
- It supports iterative development that facilitates both model-based changes and hand-written changes equally well
- It boosts productivity by automating routine and mundane development tasks
- It’s the foundation for data integration by providing a uniform way to access all models
Resources

• Online help
• Website
  – http://www.eclipse.org/emf
    • Downloads
    • Wiki
    • FAQ
    • Newsgroup
    • Documentation
• Books
  – Eclipse Modeling Framework
    • First Edition
      – http://safari.awprofessional.com/0131425420
    • Second Edition
      – http://my.safaribooksonline.com/9780321331885