Model-Driven Software Engineering

MDSE with the Eclipse Modeling Framework I

Dr. Jochen Küster  (jku@zurich.ibm.com)
Contents

- Brief Introduction to Eclipse
- Introduction to the Eclipse Modeling Framework (EMF)
- EMF Modeling
- Summary and Literature
Eclipse

- Eclipse is a software development environment comprising and integrated development environment (IDE)
- Eclipse is an open source community whose projects are focused on an open development platform
- Eclipse is extensible by a plug-in architecture
- The Eclipse IDE is available for Java, C, C++, Ada and many more
- Many commercial products rely on Eclipse or use Eclipse as a basis
The Eclipse Integrated Development Environment

```java
package books.impl;

import books.Book;

/**
 * An implementation of the model object '<code>Book</code>'.
 */

public class BookImpl extends EObjectImpl implements Book {
  /**
   * The cached value of the '{@link #getAuthor() &lt;em&gt;get&lt;/em&gt;} author' feature.
   */
  protected Writer author = null;

  public Writer getAuthor() {
    return author;
  }

  public void setAuthor(Writer newAuthor) {
    if (newAuthor != author) {
      author = newAuthor;
      notifyChanged();
    }
  }

  @Generated
  public boolean eIsSet(int featureID) {
    switch (featureID)
    {
      case BooksPackage.LIBRARY__AUTHOR:
        return author != null;
      default:
        return super.eIsSet(featureID);
    }
  }

  @Override
  public int eGet(int featureID, boolean resolve, boolean coreType)
```
Brief Overview of History of Eclipse

- Eclipse originated as a replacement of IBM VisualAge and was originally developed by IBM Canada and Object Technology International (OTI)
- In 2001 a consortium was formed to continue development as open source, source code of Eclipse was released to open source by IBM
- In 2004 the Eclipse Foundation was formed to foster further development of Eclipse
- The Eclipse Foundation is a non-profit organization
- Eclipse is available under the Eclipse Public License
- Current version is *Indigo* (Eclipse 3.7)
- A large open source community has formed itself around Eclipse
The Eclipse Universe

Indigo is here!

Explore the Eclipse universe...

Announcements

New Java 7 Summit To Be Held at EclipseCon Europe
The Eclipse Foundation, in collaboration with Oracle, is pleased to announce the Java 7 Summit, to be held at the upcoming EclipseCon Europe in Luxembourg, Germany on November 2-4, 2011.

New Open Source Initiative for Automotive Software Development Tools
The Eclipse Foundation is pleased to announce the creation of a new open source initiative to define and implement a standard platform for the software development tools used in the automotive industry. [German]

New Location Announced for EclipseCon 2012
We are pleased to announce that EclipseCon 2012 is moving to a new location just outside Washington, D.C. The annual Eclipse North America conference will be March 26-29 at the Hyatt Regency in Reston, Virginia.

Reminder: EclipseCon Europe Call For Papers
Reminder to send in your CCF submissions! The deadline is the week away on August 17. If you submit soon, you could be an easy-listed winner. The program committee will choose the talks for early acceptance from those submitted by August 2.

Eclipse Indigo Release Train Is Now Available
The Eclipse Foundation is pleased to announce the availability of Indigo, the 2011 annual release train. Indigo is available for immediate download. [German] [French]

Community News

Jaspersoft Open Source Reporting Now Available for Eclipse Developers
Jaspersoft, one of the world’s most widely used business intelligence (BI) software, today released Jaspersoft Studio, the most complete open source BI design environment for Eclipse. Bringing new BI capabilities to Eclipse Java developers, Jaspersoft Studio provides a complete open source platform...

Seven Java projects that changed the world
A Large Number of Open Source Projects and Technology
Eclipse Platform Overview and Benefits

- Eclipse platform is a foundation for building arbitrary tools and applications in areas such as banking, automotive, medical etc.
- Eclipse offers tool builders mechanisms to build tools more systematically and more easily
- Eclipse supports development of graphical user interfaces and runs on a wide range of operating systems
- Eclipse facilitates integration of tools and reuse via a specific platform architecture using the plug-in concept
## Popular Eclipse Plug-in Examples

<table>
<thead>
<tr>
<th>Solution</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclipse</td>
<td>5125</td>
</tr>
<tr>
<td>Subversive – SVN Team Provider</td>
<td>2991</td>
</tr>
<tr>
<td>Spring IDE</td>
<td>2524</td>
</tr>
<tr>
<td>JBoss Tools</td>
<td>1977</td>
</tr>
<tr>
<td>Pydev – Python IDE for Eclipse</td>
<td>1924</td>
</tr>
<tr>
<td>EGit – Git Team Provider</td>
<td>1852</td>
</tr>
<tr>
<td>Google Plugin for Eclipse</td>
<td>1812</td>
</tr>
<tr>
<td>FindBugs Eclipse Plugin</td>
<td>1691</td>
</tr>
<tr>
<td>GlassFish Java EE Application Server Plugin for Eclipse</td>
<td>1256</td>
</tr>
<tr>
<td>EclEmma Java Code Coverage</td>
<td>1187</td>
</tr>
</tbody>
</table>
In Eclipse the basic component is called a **plug-in**

The platform runtime engine is responsible for managing plug-ins during runtime
Important Components of Eclipse

- **Plug-in:**
  - pluggable component that contributes functionality
  - can depend on other plug-ins
  - can extend other plug-ins by adding additional functionality

- **SWT: Standard Widget Toolkit (SWT)**
  - provides a common OS-independent API for widgets and graphics
  - tight integration with the underlying native window system.
  - Eclipse Platform UI, and the tools that plug in to it, use SWT for presenting information to the user.

- **JFace:**
  - a UI toolkit with classes for handling many common UI programming tasks
  - window-system-independent in both its API and implementation
  - is designed to work with SWT without hiding it

- **Workbench:**
  - desktop development environment
Key Sources of Information

Eclipse Tutorials and Help for

- Workbench User Guide
- Java Development User Guide
- Platform Plugin Developer Guide
- Plugin Development Environment Guide

And many more…
How Popular is Eclipse?

Top Countries by Eclipse Usage (April 08 - March 10)

<table>
<thead>
<tr>
<th>Country</th>
<th>Max Users</th>
<th>Min Users</th>
<th>Median Users</th>
<th>Average Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>41,521</td>
<td>262</td>
<td>27,803.5</td>
<td>24,481.8</td>
</tr>
<tr>
<td>China</td>
<td>25,437</td>
<td>83</td>
<td>16,176.5</td>
<td>13,633.7</td>
</tr>
<tr>
<td>Korea, Republic of</td>
<td>15,768</td>
<td>52</td>
<td>9,490.0</td>
<td>8,495.9</td>
</tr>
<tr>
<td>Germany</td>
<td>14,800</td>
<td>128</td>
<td>9,619.0</td>
<td>8,563.5</td>
</tr>
<tr>
<td>Japan</td>
<td>11,920</td>
<td>51</td>
<td>6,873.0</td>
<td>6,773.1</td>
</tr>
<tr>
<td>France</td>
<td>11,651</td>
<td>85</td>
<td>7,008.0</td>
<td>6,556.9</td>
</tr>
<tr>
<td>Brazil</td>
<td>10,888</td>
<td>22</td>
<td>7,357.0</td>
<td>6,265.2</td>
</tr>
<tr>
<td>Spain</td>
<td>8,907</td>
<td>15</td>
<td>4,082.5</td>
<td>3,846.0</td>
</tr>
<tr>
<td>Italy</td>
<td>5,943</td>
<td>33</td>
<td>3,651.0</td>
<td>3,477.2</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>5,802</td>
<td>57</td>
<td>3,944.5</td>
<td>3,641.2</td>
</tr>
<tr>
<td>Canada</td>
<td>5,700</td>
<td>53</td>
<td>3,640.5</td>
<td>3,336.4</td>
</tr>
<tr>
<td>India</td>
<td>5,293</td>
<td>12</td>
<td>2,651.5</td>
<td>2,583.3</td>
</tr>
<tr>
<td>Taiwan, Province of China</td>
<td>4,242</td>
<td>12</td>
<td>2,678.0</td>
<td>2,327.4</td>
</tr>
<tr>
<td>Poland</td>
<td>4,099</td>
<td>24</td>
<td>2,674.5</td>
<td>2,344.8</td>
</tr>
<tr>
<td>Australia</td>
<td>3,676</td>
<td>16</td>
<td>1,520.0</td>
<td>1,628.1</td>
</tr>
<tr>
<td>Russian Federation</td>
<td>2,807</td>
<td>27</td>
<td>1,365.5</td>
<td>1,506.1</td>
</tr>
</tbody>
</table>

Introduction to the Eclipse Modeling Framework (EMF)
EMF Overview

- Eclipse Modeling Framework (EMF)
- EMF unifies XML, UML and Java
- EMF can be seen as an instance of MDA
- EMF follows the model-driven software engineering vision
- EMF increases productivity
- EMF is open-source
EMF Concept Overview

- Domain model describes the concepts of the domain
- Can be considered as the data of the application
  - objects and their relationships
- Realizes an implementation of the domain model
- Contains persistence capabilities
- Contains editing capabilities
EMF Example – Demo
Benefits of EMF

- Domain model is made explicit and can directly be used for implementation
- Domain model is used to generate Java functionality for representing the model in Java
  - No code needs to be written, speedup of development
- Additional functionality for persisting the domain model is provided
  - No serialization/deserialization code needs to be written
- EMF also provides a validation framework for the domain model
  - To support evaluation of constraints
- EMF provides further functionality for generating other components for the domain model (e.g. an editor)
- Changes of the domain model can easily be implemented by regenerating Java from the domain model
EMF Architecture Overview

Adapted and extended from: Effective Use of the Eclipse Modeling Framework, EclipseCon 2007
Description of Main Components

- Model
  - Expressed in Ecore

- EMF Codegen
  - Code generator for core and edit based components
  - Generator templates

- EMF Core
  - Persistence and serialization
  - Model change notification
  - Model validation

- EMF Edit
  - Used to build editors and viewers for the model
  - Predefined runtime libraries
EMF as Architecture-Centric MDSD

Application

Domain Model

Infrastructure code

Business Logic Code
manually written

Generative Architecture

Ecore

expressed in

Generator Templates

Infrastructure Components

EMF Generator
EMF Modeling
EMF Road Map

Adapted and extended from: Effective Use of the Eclipse Modeling Framework, EclipseCon 2007
EMF Technology Overview

- Domain model can be considered as the metamodel of the data
- Imports from UML, Java, Annotated code
- Domain model is expressed in Ecore modeling language

- EMF.Model
- EMF.Edit and EMF.Editor
Domain Model Example
Creating Domain Models using the Ecore Diagram Editor
Typical steps when using EMF

- Metamodel (domain model) can be defined using Ecore Diagram editor, using UML editor, or using annotated Java
- .genmodel needs to be generated
- Code generation includes model, edit, and editor code
- Customization of code with additional functionality
Domain model is expressed by instantiating the Ecore metamodel
Ecore implements Essential MOF (Meta Object Facility)
Core Elements of the Ecore Metamodel

- **EClass** models classes
  - attribute “name” refers to the name of the class
  - can have attributes
  - can have references to other elements
  - can have super types for supporting inheritance

- **EAttribute** models attributes
  - type refers to the type of the attribute

- **EDataType** is used to represent simple types
  - EString for modeling strings

- **EReference** is used for modeling one end of an associations between classes
  - references are identified by name and have a type
  - type must be of type EClass

- **EStructuralFeature** for modeling features such as attributes and references of a class
Elements of the Ecore Metamodel - Example

**EClass**

**EReference**

**EAttribute**
EClass and EReference

- **EClass** is used to model a class in the domain model
  - Name is the name of the class

- **EReference** is used to model a reference in the domain model
  - Name is the name of the reference
  - Lower and Upper Bound model the cardinalities
Pairs of references realize a bidirectional association

- Relieves the developer from error-prone manual implementation
  - Dangling references
- Ecore models can contain packages for structuring
  - EPackage
**EPackage Properties**

- **EPackage attributes:**
  - name: name of the package
  - nsURI: URI for identification
  - nsPrefix: prefix of the namespace

- **EPackage references:**
  - eSuperPackage: super package
  - eSubPackage: sub package

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>schoollibrary</td>
</tr>
<tr>
<td>Ns Prefix</td>
<td>schoollibrary</td>
</tr>
<tr>
<td>Ns URI</td>
<td><a href="http://schoollibrary">http://schoollibrary</a></td>
</tr>
</tbody>
</table>
Packages and referenced Elements

- Elements can be referenced from other packages
Datatypes: EDataType

- Represent single piece of simple data
- EDataType models single Java type (primitive, class, interface, array)

Guideline: If values cannot be represented as strings, do not use EDataType but EClass

EDataTypes are created implicitly for any Java type used by a structural feature or operation
Annotations: EAnnotation

- Annotations for attaching additional information to an object in an Ecore model
- All classes can be annotated by zero or more EAnnotations

Properties of EAnnotation:
- source: defines URI for type of the annotation
- details: map of key value pairs
- contents: a list of EObjects
- references: a list of EObject referenced
Generation of .genmodel

- After the metamodel has been defined, it is transformed into a .genmodel
Summary and Literature
Summary of Lecture and References

- EMF is one example of architecture centric MDSD
- EMF is widely used today in the Java world
- EMF modeling provides a means of expressing a domain model
- EMF is a basis for a rich platform of diverse technology

References:

- Eclipse EMF tutorials online.