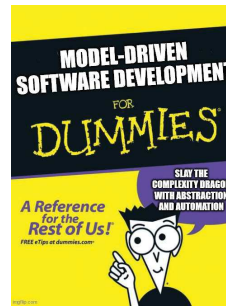


CISC 844: Beyond code: An Introduction to Model-Driven Software Development



Topic: EMF

- Meta modeling
 - Languages for meta models: Ecore
 - Using EMF and Ecore to define a data model
 - Using EMF to generate code for data model
- ⇒ Prep for Xtext

Juergen Dingel
March 2026

CISC844, Winter 2026

EMF

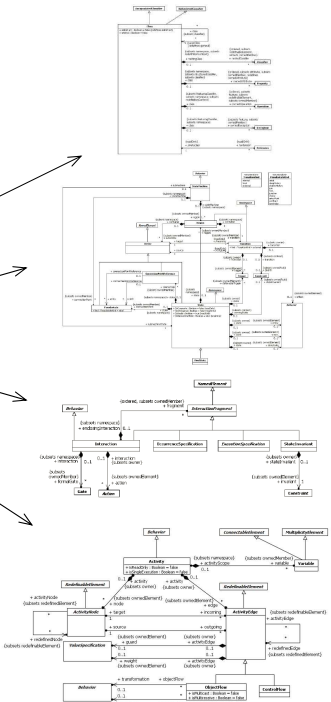
1

Metamodeling

Can use concepts from Class models/diagrams to describe structure (“abstract syntax”) of modeling concepts in UML:

- **Classes**: Section 11.4.2 in [UML2.5]
- **Statemachines**: Section 14.2.2
- **Interactions**: Section 17.2.2
- **Activities**: Section 15.2.2
- **Packages**: Section 12.2.2
- **Behaviour**
- **Classification**: Classifiers, Features, Properties, Operations

These class models are **metamodels**, i.e., models describing models

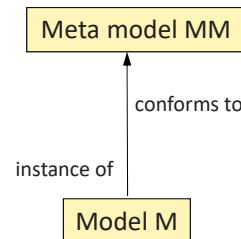


CISC844, Winter 2026

EMF

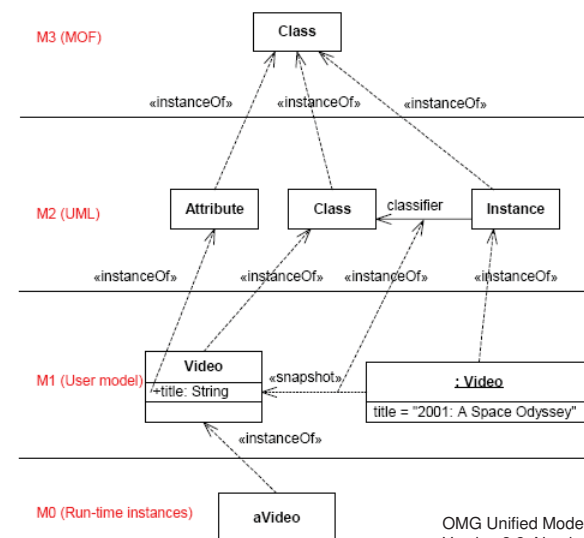
Metamodeling (Cont'd)

- **Meta model MM**: model (a specification) of a set of models (i.e., a modeling language L(MM))
- **Instance M of meta model MM**: well-formed model in modeling language L (i.e., $M \in L(MM)$)
- **Languages for expressing meta models**



- **Meta Object Facility (MOF)**:
 - OMG standardized language for defining modeling languages
 - **subset of UML class diagrams**: types (classes, primitive, enumeration), generalization, attributes, associations, operations
- **ECore**:
 - Eclipse version of MOF; used by Xtext
- **Object Constraint Language (OCL)**:
 - declarative language to express well-formedness rules (e.g., “the inheritance hierarchy is acyclic”)

Metamodeling (Cont'd)



OMG Unified Modeling Language, Infrastructure, Version 2.2. Number: formal/2009-02-04

Figure 7.8 - An example of the four-layer metamodel hierarchy

CISC844, Winter 2026

EMF

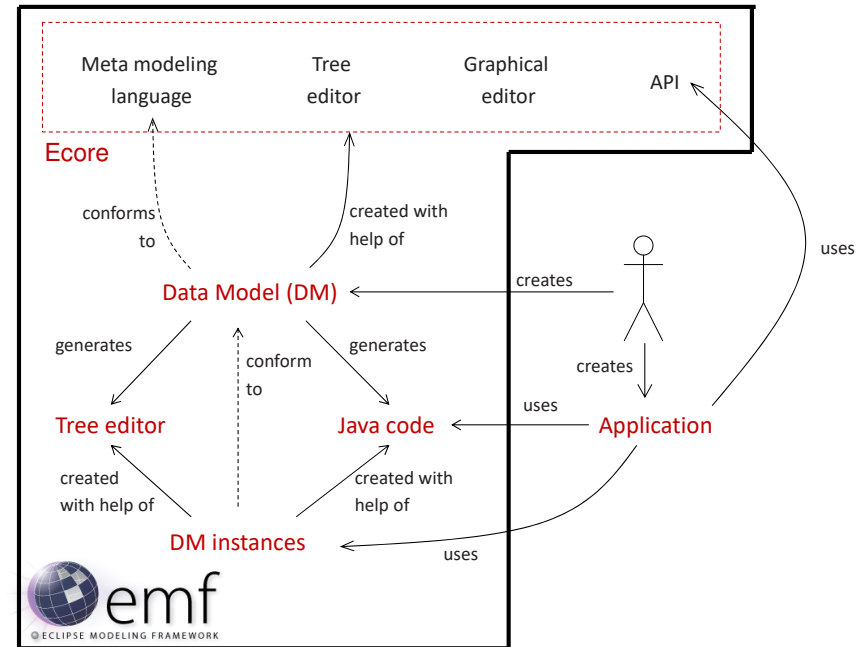
3

4

Eclipse Modeling Framework

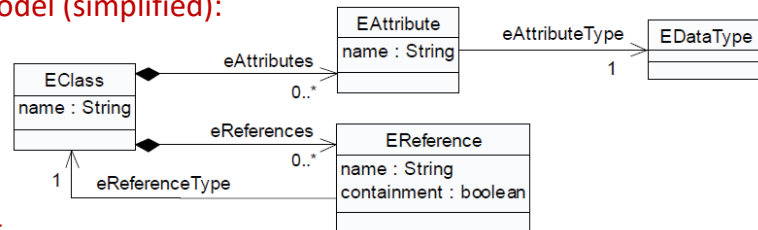


- Eclipse-based open-source framework supporting the development of applications requiring a structured **data model (DM)**
- Consists of
 - **Ecore**
 - **Meta modelling language** for the description of DMs with support for
 - change notification (via the Observer design pattern),
 - persistence (via XML serialization), and
 - generic object manipulation (via reflective API)
 - **EMF.Codegen**
 - Generates code implementing the DM and supporting the development of tools creating and manipulating data model instances:
 - **Model**: Java interfaces and implementation of DM (using Factory design pattern)
 - **Editor**: plugin for tree-based editor of DM instances
 - **Edit**: implementation classes adapting DM classes for editing and display (via Adapter Factory design pattern)
 - **Large user community**: <http://www.eclipse.org/emf>



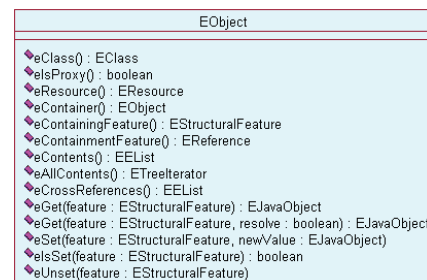
Ecore: Metamodeling Language

- Metamodel (simplified):



- EObject

- Base of every Ecore class and every generated class
- Provides support for **notification** and **persistence**

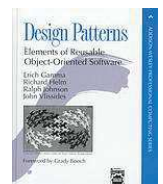


[Summary of package org.eclipse.emf.ecore]

[Steinberg, Budinsky, Paternostro, Merks. EMF: Eclipse Modeling Framework (2nd Ed.). 2008]

EMF: Supporting Technology

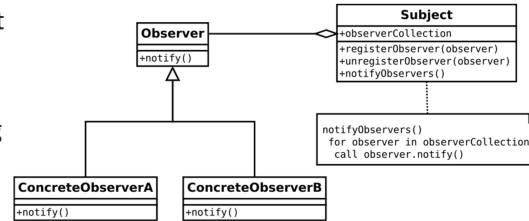
- **Eclipse**
 - Editors are Eclipse plugins
- **Sirius**
 - Framework for automatic generation
- **JUnit**
 - Supported by generated test code
- **Design patterns**
 - Observer
 - Factory
 - Adapter Factory



Observer Design Pattern

Object, called **subject**, maintains list of its dependents, called **observers**, and notifies them automatically of **any state changes**, usually by calling one of their methods.

Used for MVC pattern.



```

import java.util.Observable;
import static java.lang.System.out;

class MyApp {
    public static void main(String[] args) {
        out.println("Enter Text >");
        EventSource eventSource = new EventSource();

        eventSource.addObserver((Observable obj, Object arg) -> {
            out.println("\nReceived response: " + arg);
        });

        new Thread(eventSource).start();
    }
}

import java.util.Observable;
import java.util.Scanner;

class EventSource extends Observable implements Runnable {
    public void run() {
        while (true) {
            String response = new Scanner(System.in).next();
            setChanged();
            notifyObservers(response);
        }
    }
}

```

[Wikipedia. Observer pattern. https://en.wikipedia.org/wiki/Observer_pattern. 2016]

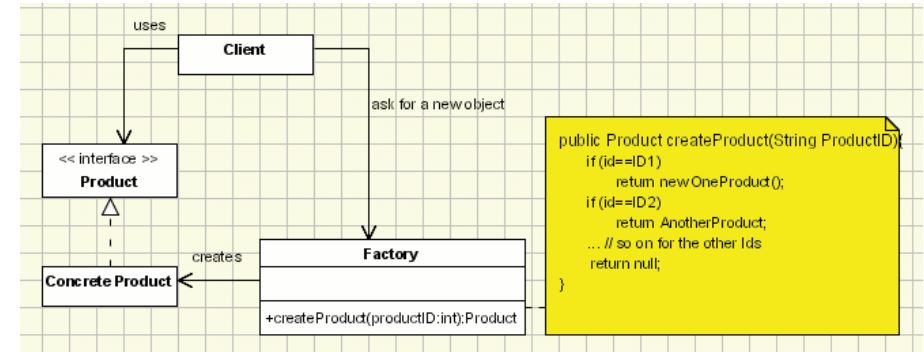
CISC844, Winter 2026

EMF

9

Factory Design Pattern

Object, called the **Factory**, encapsulates details of creation of different **Products**



[OODesign.com. Factory pattern. <http://www.oodeign.com/factory-pattern.html>. 2016]

CISC844, Winter 2026

EMF

10

Using EMF: Getting Started

- Download and installation
 - <http://www.eclipse.org/modeling/emf>
- Documentation
 - <http://www.eclipse.org/modeling/emf/docs/>
- Tutorials
 - <http://www.vogella.com/tutorials/EclipseEMF/article.html>
 - <http://eclipsesource.com/blogs/tutorials/emf-tutorial/>
 - <http://www.philmann-dark.de/EMFDocs/tutorial.html>

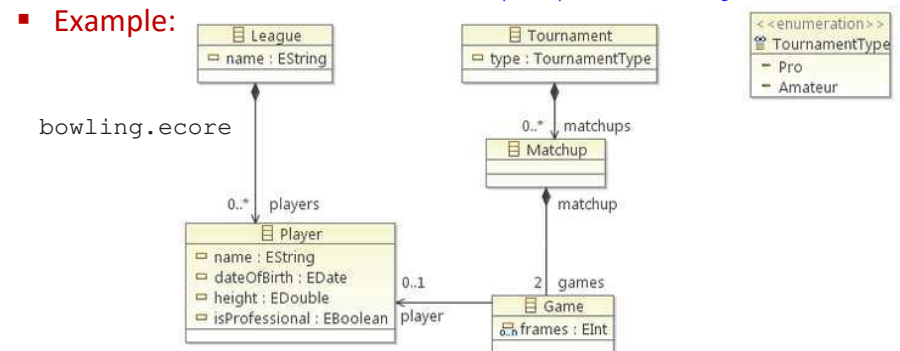
CISC844, Winter 2026

EMF

11

Using EMF: Create Domain Model (DM)

<http://eclipsesource.com/blogs/tutorials/emf-tutorial/>



- Root classes: League, Tournament
- Containment relationship b/w Matchup and Game is bi-directional
 - "Containment = true" for reference games
 - matchup is opposite of games:
 - for all m:Matchup, if g:Game in m.games, then g.matchup == m

CISC844, Winter 2026

EMF

12

EMF Sample Models



Beyond Code: An Introduction to Model-Driven Software Development (CISC 836, Winter 2021)

Sample Eclipse Modeling Framework (EMF) Projects

The sample projects below assume the Eclipse IDE for Java and DSL Developers (version 2020-12 R, it will contain EMF version 2.24). See the beginning of [Assignment 4](#) for information on how to obtain and install it. The origin of these projects is as follows:

- Extended library, Java: Part of the EMF distribution
- Bowling: Running example of the EclipseSource EMF tutorial [here](#)
- Linked lists: Own

All artifacts are made available under the terms of the [Eclipse Public Licence v1.0](#).

1. Extended library:
 - class (data) model for a library:
 - illustrates metamodeling with ECore (concepts: classes (concrete, abstract, subclasses), associations (multiplicities, inverses), attributes (types, defaults, multiplicities, uniqueness, unchangeable), enumerations), editor plugin generation
 - ECore metamodel:



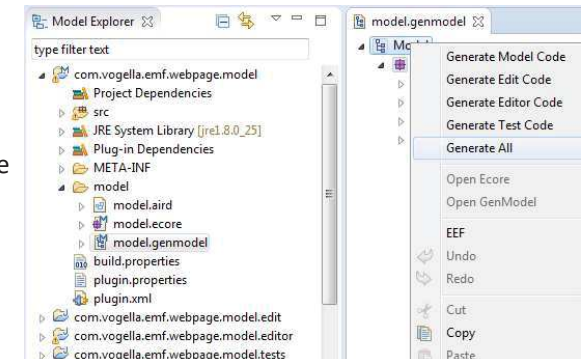
Using EMF: Generate Code from DM

Genmodel

- Generated from .ecore model
- Contains additional info necessary for code generation (package names, source paths, project names, generator settings)
- Automatically synchronized with .ecore model when .ecore model is saved

To generate:

- Open .genmodel
- right-click root node
- select plugin to generate



EMF Sample Models

CISC844, Winter 2026

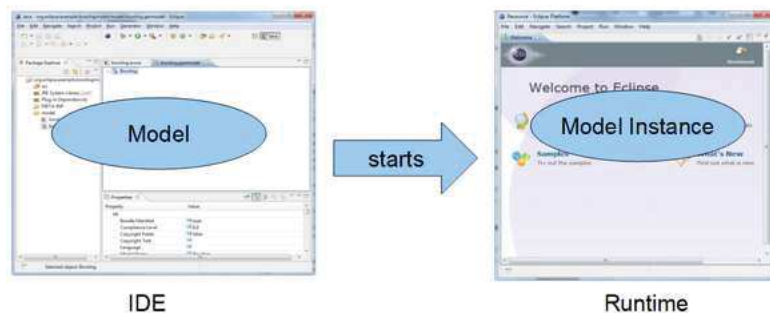
EMF

13

CISC844, Winter 2026

Using EMF: Use Generated DM Editor

- Code for DM editor generated as Eclipse plugin
- Invocation of the editor will create another instance of Eclipse in which the editor will execute

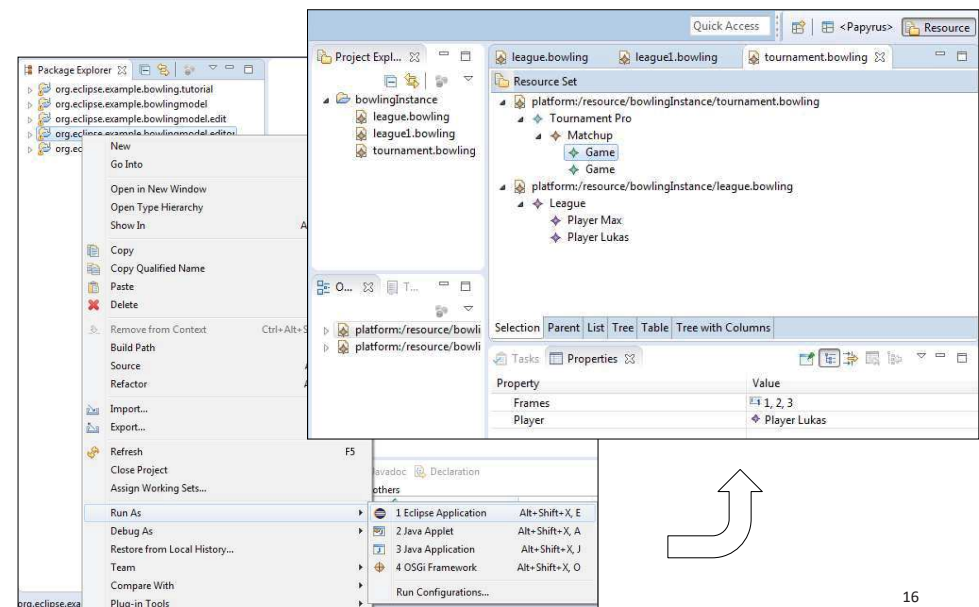


CISC844, Winter 2026

EMF

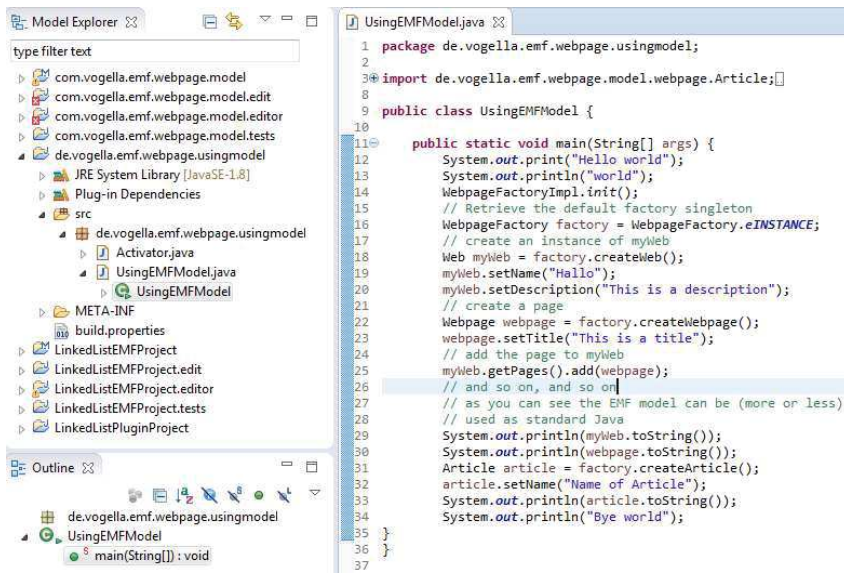
15

Using EMF: Use Generated DM Editor (Cont'd)



16

Using EMF: Use Generated DM Java Code



CISC844, Winter 2026

EMF

17

Using EMF: Demo

- Using generated test code
- Generating JavaDoc
- Generating method bodies



CISC844, Winter 2026

EMF

18

EMF: Pros and Cons

- Pros
 - Quite powerful, stable, adequate documentation
 - Integral part of Eclipse Modeling ecosystem:
 - EMF Forms: for generation of form-based UI
 - Xtext: for DSL implementation
 - ATL: for model-to-model transformation
- Cons
 - Inherit Eclipse "baggage"

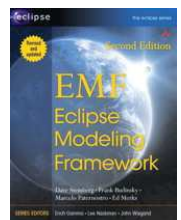
CISC844, Winter 2026

EMF

19

EMF: More Info

- Installation
 - <http://www.eclipse.org/modeling/emf>
- Documentation
 - <http://www.eclipse.org/modeling/emf/docs/>
- Tutorials
 - <http://eclipsesource.com/blogs/tutorials/emf-tutorial/>
 - <http://www.vogella.com/tutorials/EclipseEMF/article.html>
 - <http://www.philmann-dark.de/EMFDocs/tutorial.html>
- Book
 - Steinberg, Budinsky, Paternostro, Merks. EMF: Eclipse Modeling Framework (2nd Ed.). Addison-Wesley Professional. 2008.



CISC844, Winter 2026

EMF