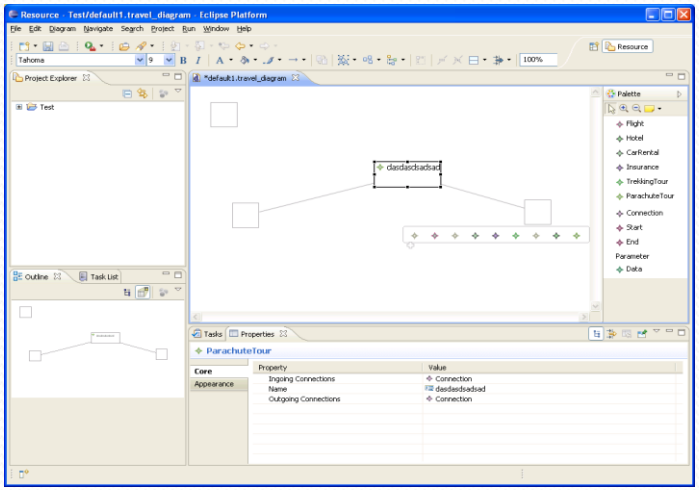
Graphical Concrete Syntax

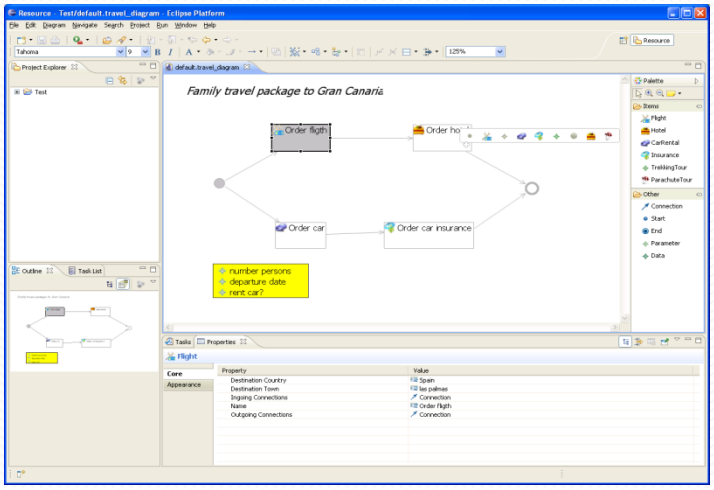
Using the Graphical Modeling Framework (GMF)

Part II

From last lecture

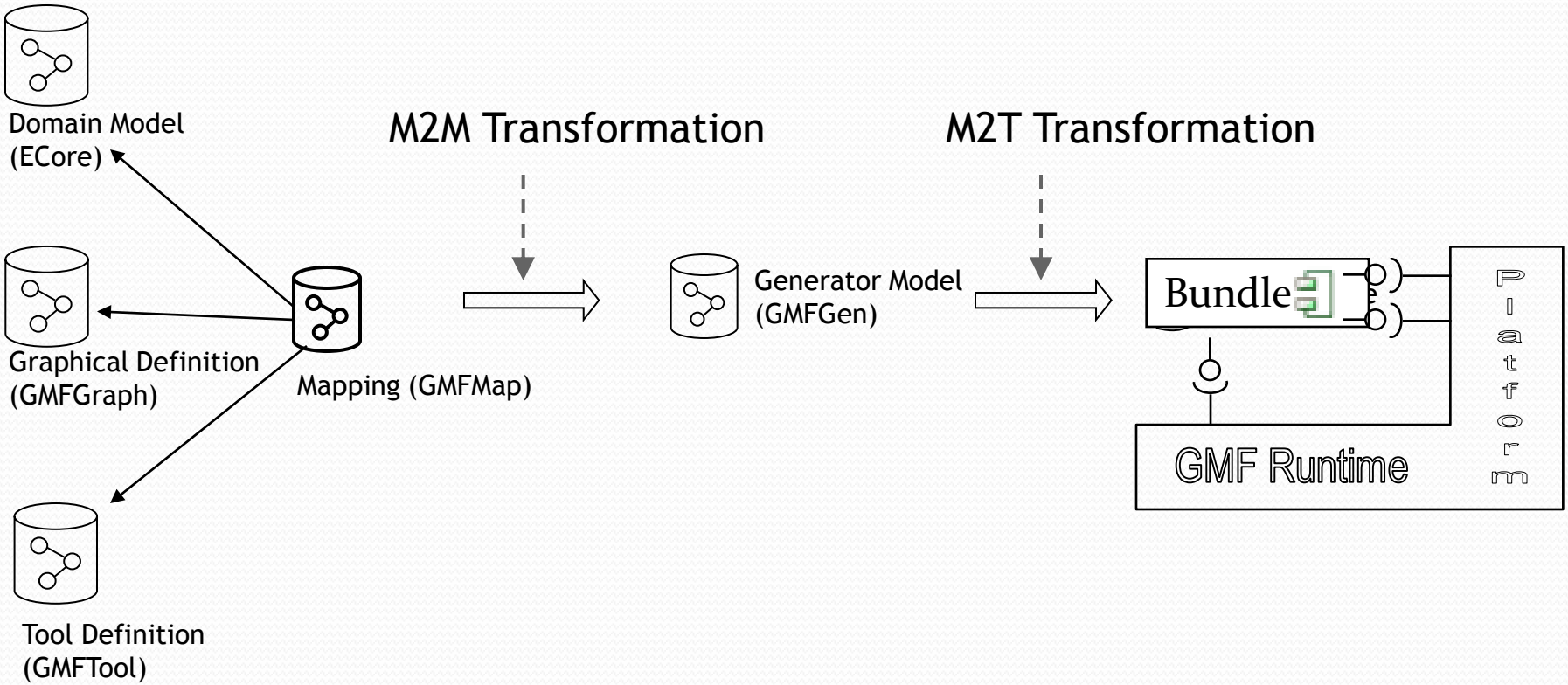
- We used GMF Tooling to create Noware travel editor
- Generate GMF models trough GMF wizards





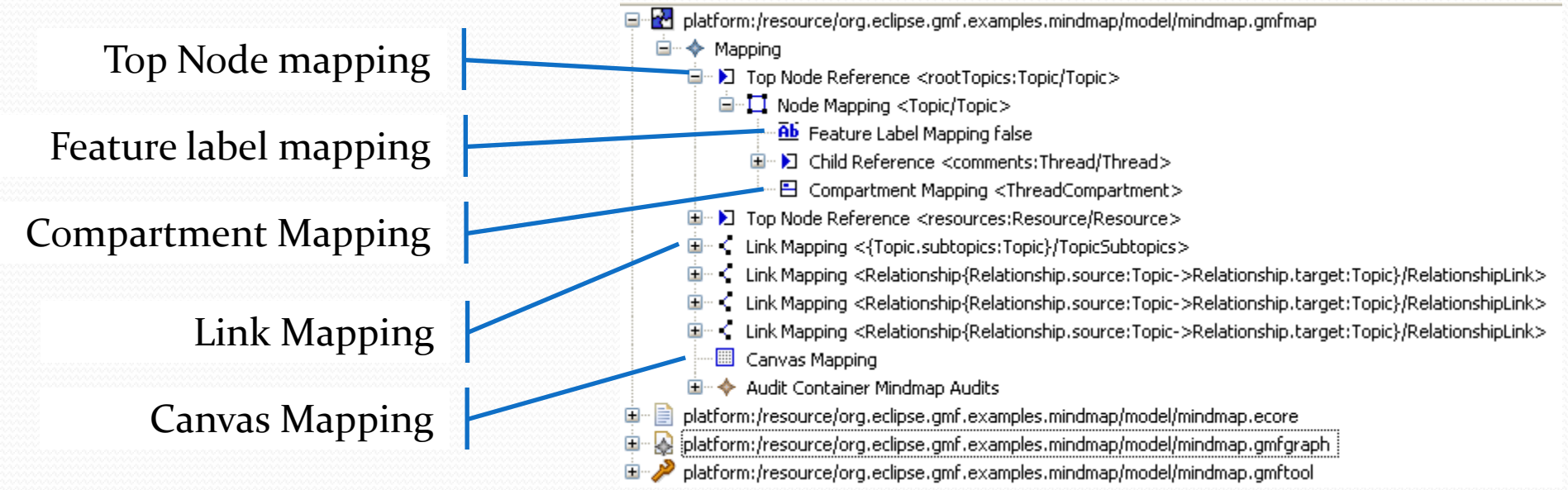
- Customized graphical syntax and palette
- Added compartment to a node

From last lecture



Mapping Model

- The heart of GMF models. Binds domain elements with palette and graphical representation.



- Any questions to the lecture or exercises?

Today

- Can we specify validations and constraints through GMF tooling ?
- What do we generate by using GMF tooling?
- How does it work?
- How can we change or extend the editor code ?

Agenda

- GMF Tooling and OCL
  - Defining constraints
  - Defining feature initialisers
  - Defining validations
- Demo
  - Add feature initialisers, constraints and validations
- GMF Runtime Architecture
  - GEF and Draw2D
  - Notation Model
  - Extensions: Extension points, code
- Demo
  - What is generated by the GMF tooling
  - Create custom action
  - Create doubleclick action on Flight items
- Summary and exercises

Remember OCL?

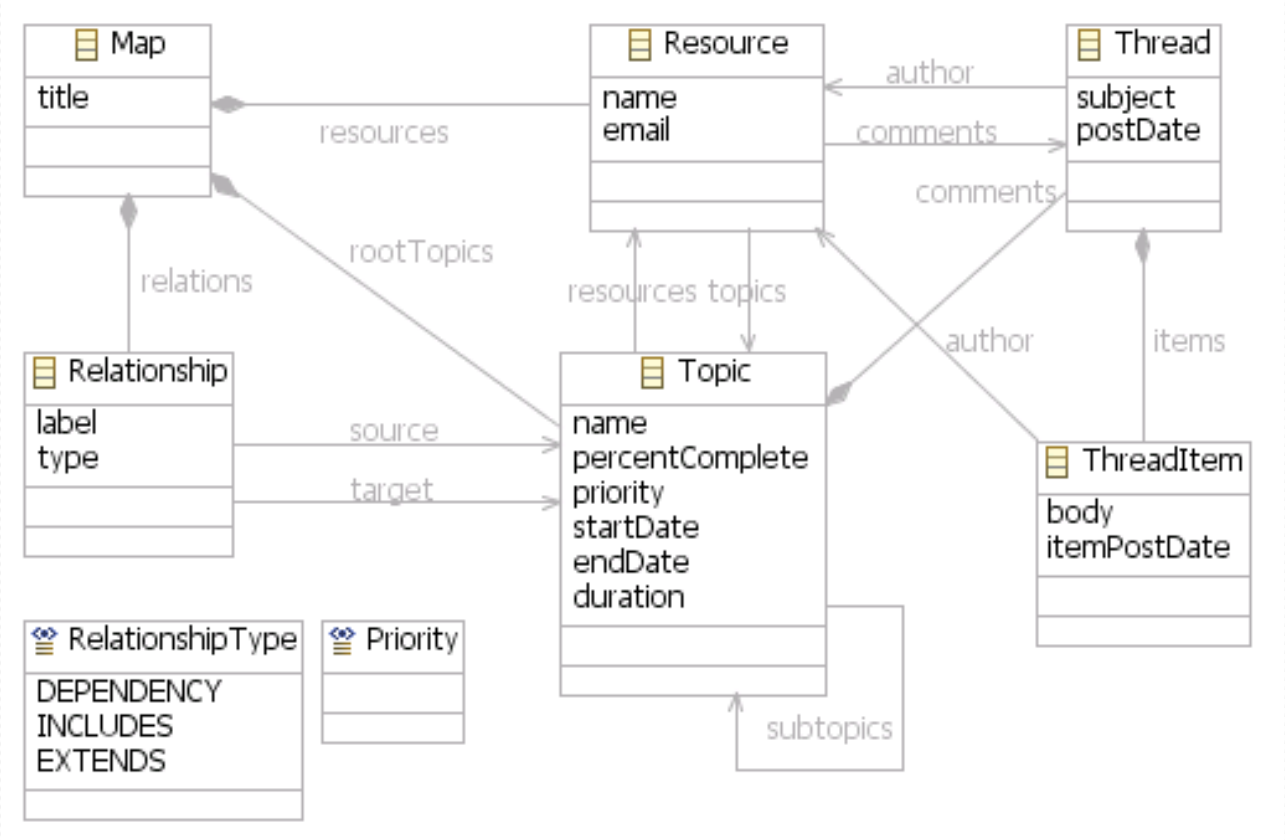
- Object Constraint language
- Language to define constraints on metamodels
- Makes us able to define static semantics that cannot be expressed in eCore
- What is a constraint in Object Constraint Language?
  - A constraint is a restriction on one or more values of (part of) a object-oriented system (metamodels in our case)

GMF Tooling and OCL

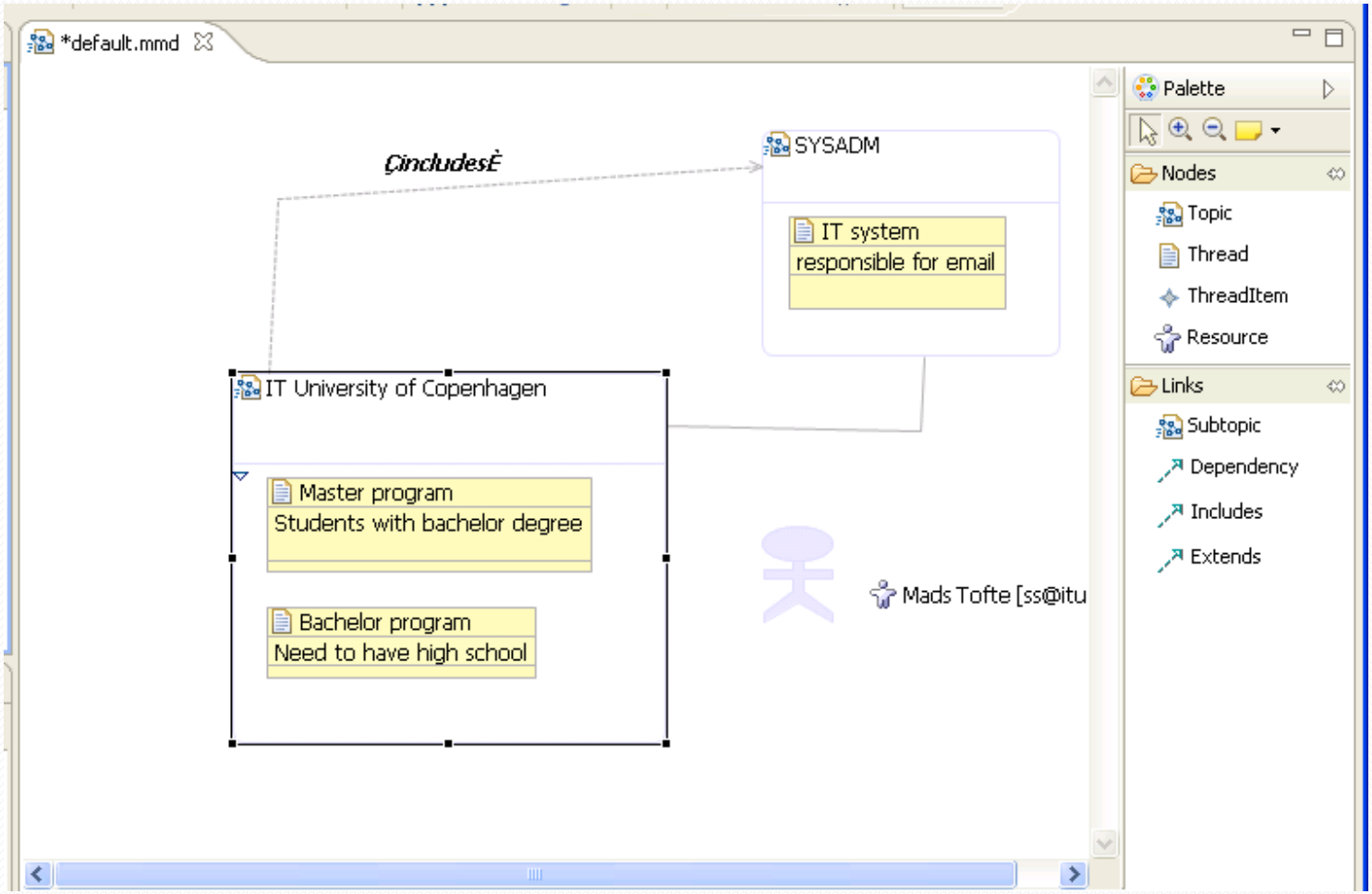
- OCL used by GMF itself
- Used to validate the GMF tooling models before generating gmfgen model and diagram code
- Used by tool smidth for defining
  - Modeling constraints
  - Initial values of created objects
  - Validations

Remember last lectures example?

- Mindmap Ecore model



Mindmap graphical editor



Constraints in GMF

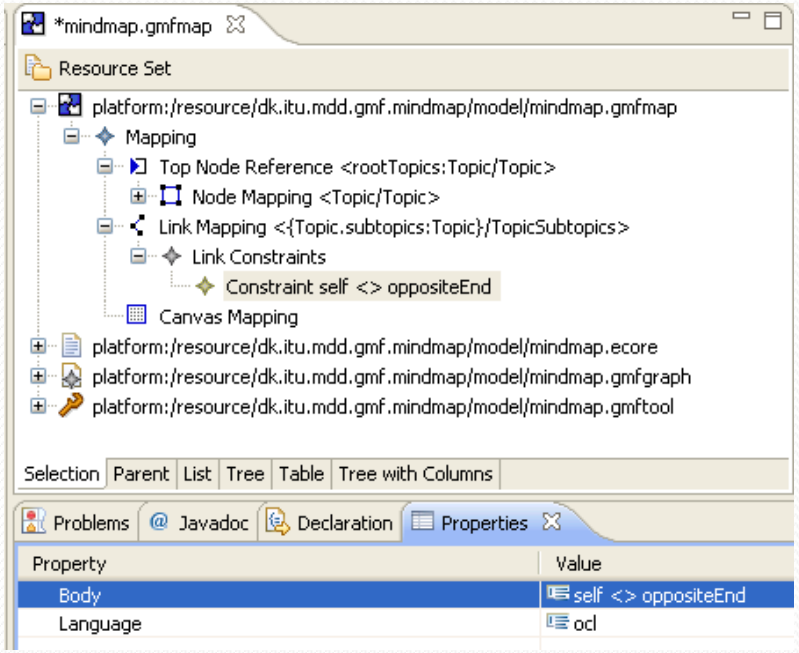
- A constraint prohibits certain modeling constructs
- Defined under a Node/Link mapping in Mapping model
- Using OCL, Java, regexp, nregexp
- The domain element is the context
- GMF tooling generator generates code that enforces the constraint

Constraints in GMF - Demo

- Mindmap example
  - A subtopic link from a Topic node must not link to itself
  - Add Link constraint
  - Add source end constraint
  - Context is the source node
  - The source node is a Topic

```
context Topic
inv: self <> oppositeEnd
```

Mapping model



'oppositeEnd' is a custom variable added to the parser environment for link constraints

Constraints in GMF - Demo

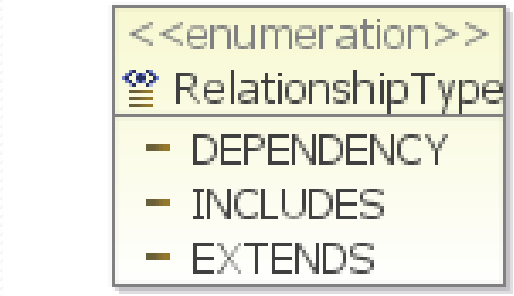
- Lets see it live!

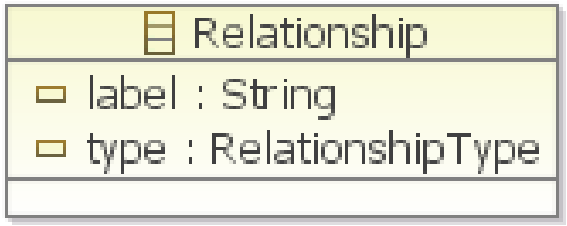
Feature Initialisers

- Specifies how to initialise a domain element and its attributes when it is added to the diagram
- Defined under Node mapping in the Mapping model
- Using OCL, Java, regexp, nregexp
- Context is the Mapping class
- Generator uses initialisers

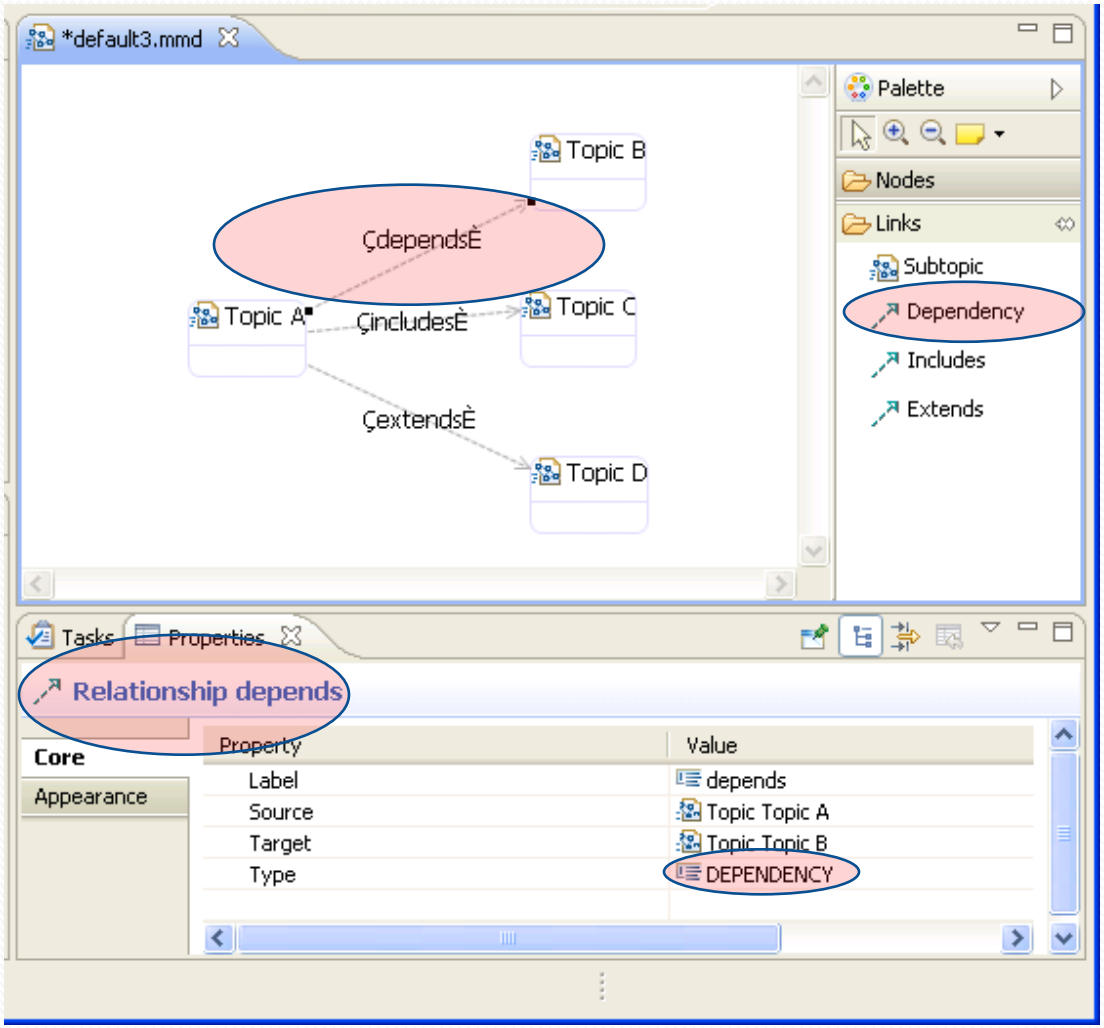
Feature Initialisers -Demo

- Mindmap example
  - Three types of relationships
  - Three tools
  - One Relationship class
  - Parameter to indicate type





Feature Initialisers -Demo



Feature Initialisers -Demo

- We need to specify the type when we are creating a Dependency relationship
- OCL assignment statement

```
context Relationship
self.type = RelationshipType::DEPENDENCY
```

- Definition in Mapping model

Feature Initialisers -Demo

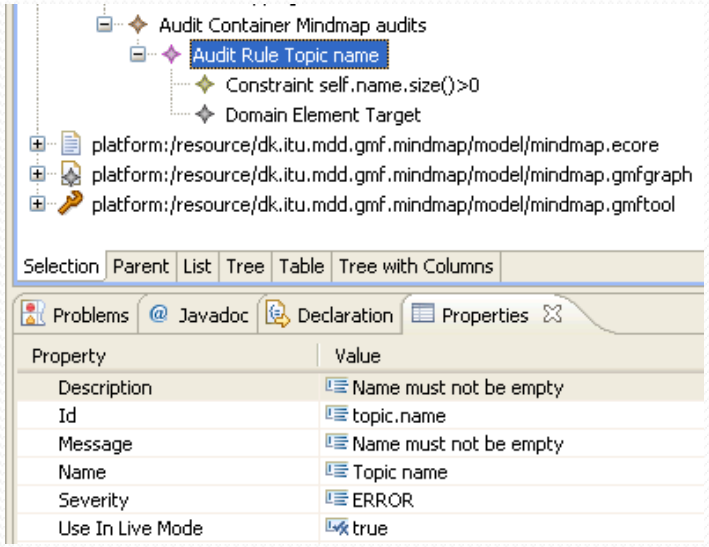
- Lets see it live!

Validations

- Validations or constraints?
- Validations defined as Audits in GMF
- Audits are defined in the mapping model
- An audit has a context and a rule
- Generators generates constraints based on the EMF Validation Framework
- Can be defined as batch or live
- Need to change .gmfgen model
  - Validation enabled=true
  - Validation decorators = true
  - Validation provider priority = Medium
  - Live validation UI feedback = true

Validations - Demo

- Mindmap demo
  - We will add two constraints (Audits)
  - A Topic must not have an empty name

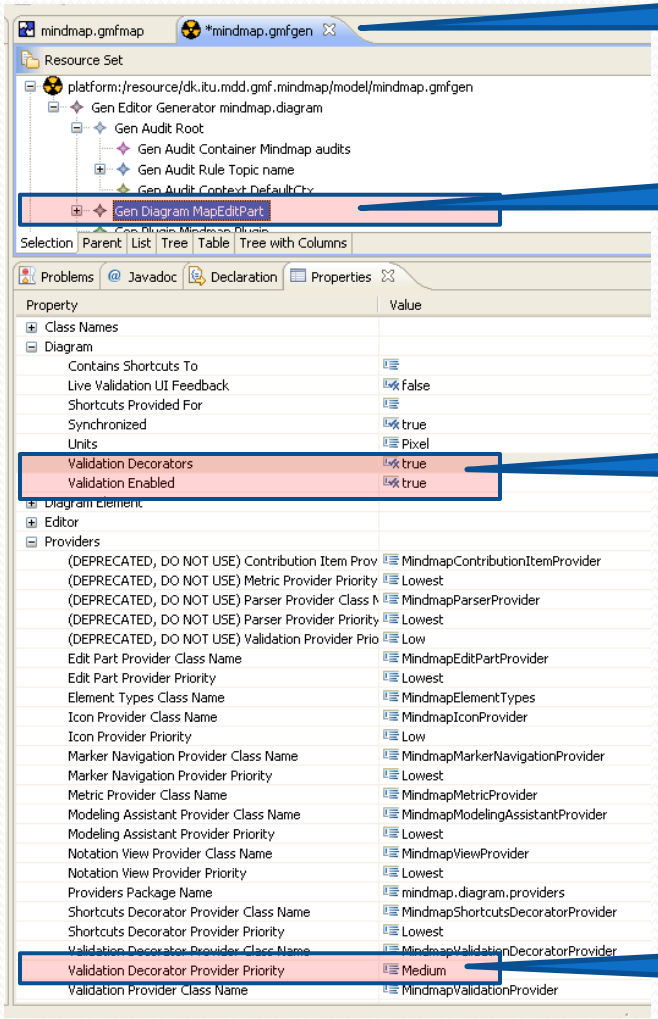



```
context Topic
inv: self.name.size()>2
```

  - Cyclic dependencies must not exist

```
context Map
inv: self.relations->forAll(r1, r2 | r1.target = r2.source and r1.type =
r2.type implies r2.target <> r1.source)
```

Validations - Demo



Generator model

Diagram map edit part

Validation decorator and Validation enabled

Validation provider priority

Validations - Demo

Generated constraint using EMF Validation Framework

```
<extension point="org.eclipse.emf.validation.constraintProviders">
   <?gmfgen generated="true"?>
         <category id="mindmap" mandatory="false" name="Mindmap audits">
      <![CDATA[Audits for mindmap]]>
   </category>
   <constraintProvider cache="true">
      <package namespaceUri="http://www.example.org/mindmap"/>
            <constraints categories="mindmap">
               <constraint id="topic.name"
         lang="OCL"
         name="Topic name"
         mode="Live"
         severity="ERROR" statusCode="200">
         <![CDATA[self.name.size()>0]]>
         <description><![CDATA[Name must not be empty]]></description>
         <message><![CDATA[Name must not be empty]]></message>
         <target class="mindmap.Topic"/>
      </constraint>
               </constraints>
   </constraintProvider>
</extension>
```

Validations - Demo

- Lets see it live!

GMF Runtime Architecture

- Build on top of - and bridges
  - Eclipse Modeling Framework (EMF)
  - Graphical Editing Framework (GEF)
- Consists of
  - Set of reusable diagramming components
  - Standard notation model for storing diagram information separate from semantic information
  - Command infrastructure that bridges EMF and GEF
  - Extensibility through 27 extension-points
  - Service provider infrastructure
- Enables open and extensible graphical editors

GMF Runtime Architecture

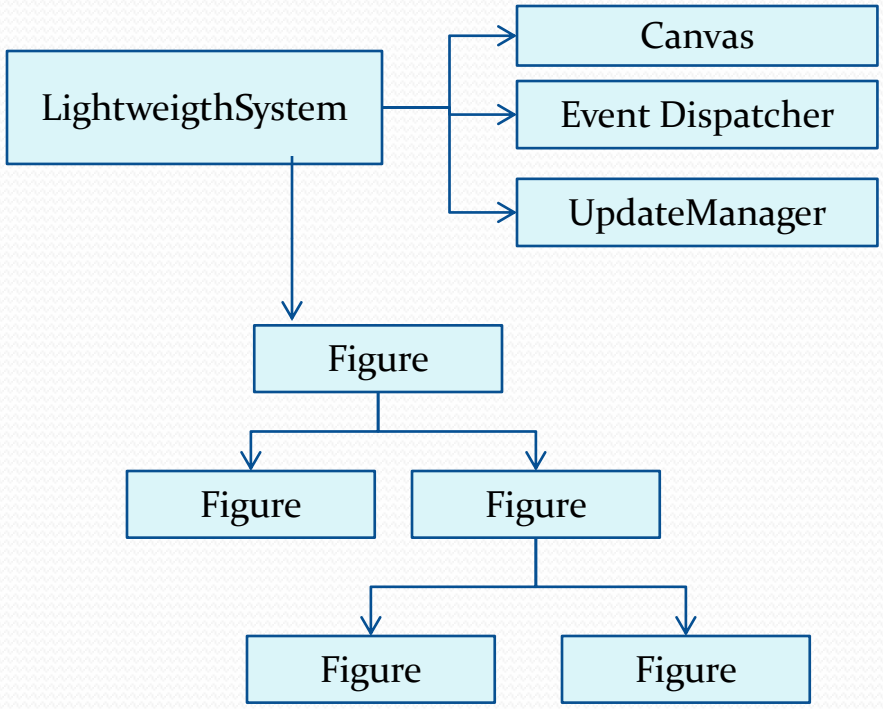
- Draw2D
- Graphical Editing Framework (GEF)
- Notational metamodel
- Key classes
- Command infrastructure
- Extensions

Draw2D

- Lightweigth toolkit of graphical components called *Figures*
- *Lightweight -> a Figure = a POJO*
- Extension of Standard Widget Toolkit (SWT)
- Provides functionality for
  - Painting, Layout, Events, Connections
- Coordinate systems

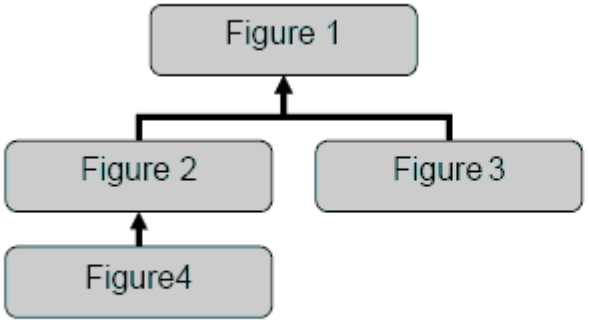
Draw2D

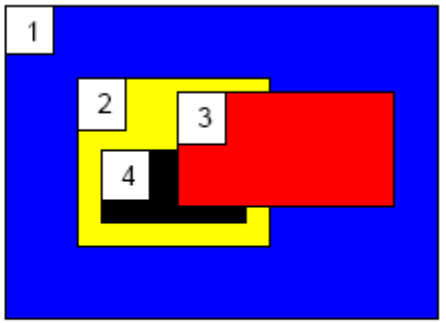
- Lightweight system
  - Associates figures with a Canvas
  - Hooks listeners for SWT events
  - Forewards SWT event to Event dispatcher
  - Forwards paint event to Update manager
- Event dispatcher
  - Translates SWT events to figure events
- Update manager
  - Coordinates painting and layout
  - Painting



Draw2D

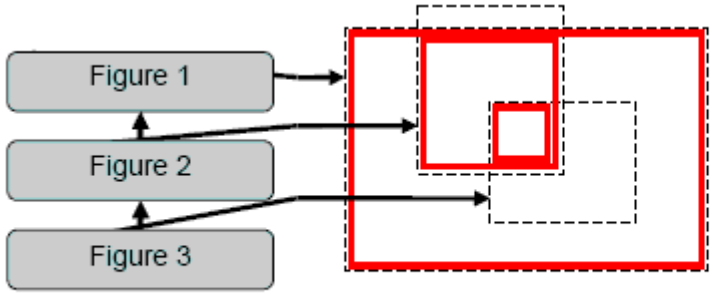
- Z-order

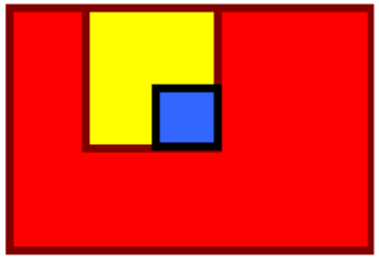




This picture shows a tree of figures and its graphical representation if each figure is painted as a full rectangle.

- Clipping



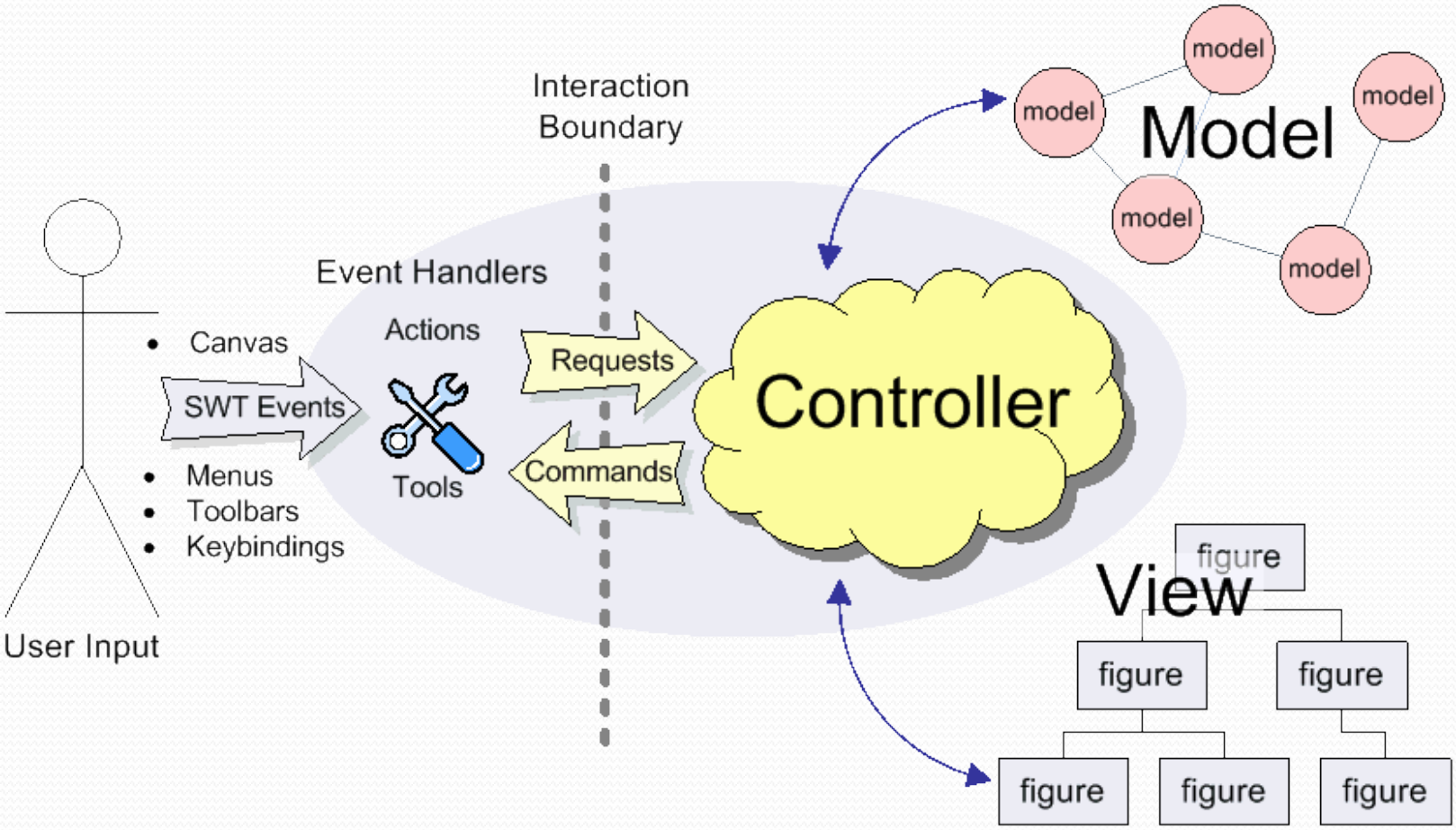


The Bounds of the figures are represented as dash lines, each figure is painted as a full rectangle with a black border, the clipping area associated with each figure is represented as a red line.

Graphical Editing Framework (GEF)

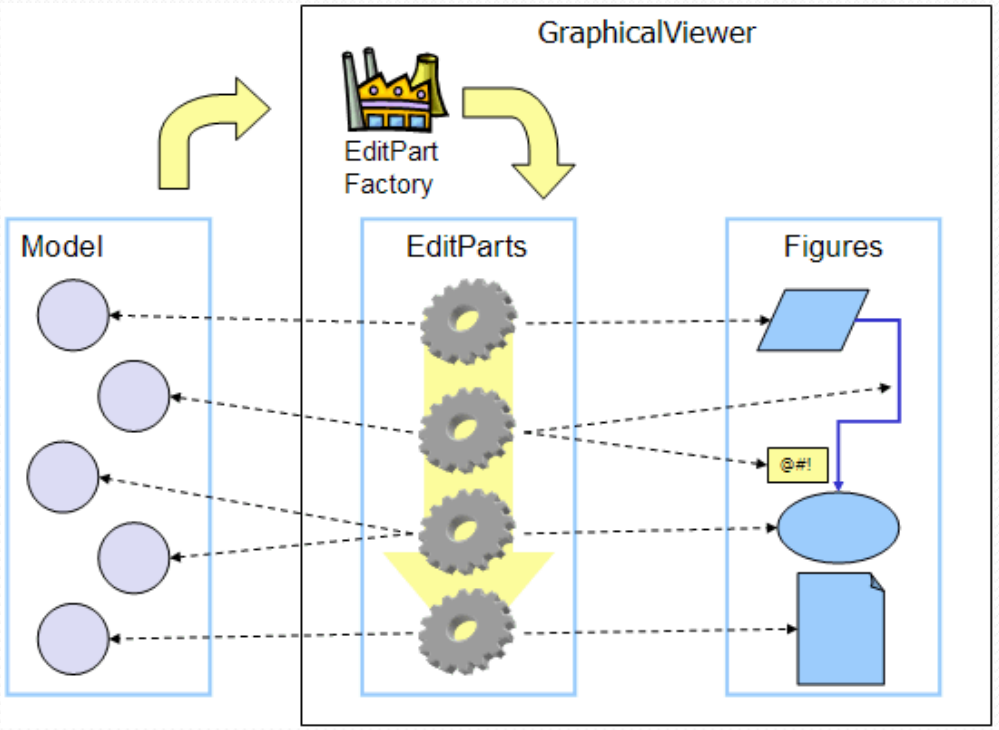
- Model-View-Controller framework for graphical editors
- Built on top of Draw2D
  - Draw2D is packaged with GEF
- Adds editing capability on top of Draw2D
- Supports interaction from mouse, keyboard
- Integration with the Eclipse platform
- Supports any kind of meta-models
- Much work to use it with an EMF model
- Much low-level editor coding must be done
- No generative tooling support

Graphical Editing Framework (GEF)



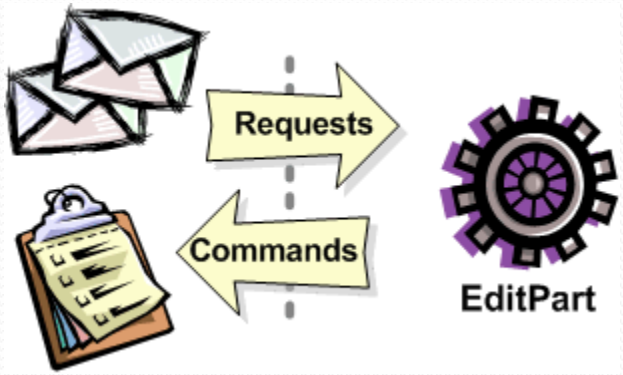
GEF Controller - EditPart

- EditParts
  - Controller for associating view and model
- Editpart Viewer
  - Graphical viewer shows the root figure on the Draw2D canvas



GEF Editing

- Editing an EditPart involves
  - Changing the underlying model
  - Showing graphical feedback during interaction
- GEF uses ***Requests*** for all interaction
  - From Palette, Actions, etc.
- Model changes are done using ***Commands***



GEF Editing

- Implementation of an EditPart

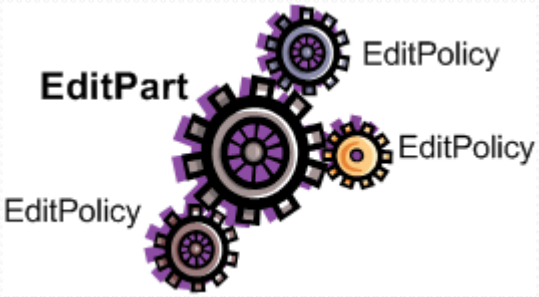
Methods on `EditPart` which take a **Request**:

1. `EditPart getTargetEditPart(Request)`
   `boolean understandsRequest(Request)`
2. `void showSourceFeedback(Request)`
   `void eraseSourceFeedback(Request)`
   `void showTargetFeedback(Request)`
   `void eraseTargetFeedback(Request)`
3. `Command getCommand(Request)`
4. `void performRequest(Request)`

1. Decide if the editpart is involved
2. Feedback during interaction
3. Command to modify the model.
4. Generic API to ”do something”

GEF Editing

- An EditPart delegates editing to ***EditPolicies***
- EditPolicies are *installed* at an EditPart in the method
  - *createEditPolicies()*
- One EditPolicy focuses on a single editing task or a group of related tasks.
- The EditPart delegates requests to its EditPolicies
- Dynamic way to implement editing functionality



GEF Example : Shape Editor

- Lets have a look at the Shape Editor code
- Available in eclipse from File->New->Examples->GEF
- ShapeEditPart
  - Install EditPolicies
  - Create Figure
- EditPolicies
- ConnectionCreateCommand

ShapeEditPart: Install EditPolicies

```
protected void createEditPolicies() {
    // allow removal of the associated model element
    installEditPolicy(EditPolicy.COMPONENT_ROLE, new ShapeComponentEditPolicy());
    // allow the creation of connections and
    // and the reconnection of connections between Shape instances
    installEditPolicy(EditPolicy.GRAPHICAL_NODE_ROLE, new GraphicalNodeEditPolicy() {
        /* (non-Javadoc)
         * @see org.eclipse.gef.editpolicies.GraphicalNodeEditPolicy#getConnectionComp
         */
        protected Command getConnectionCompleteCommand(CreateConnectionRequest request) {
            ConnectionCreateCommand cmd
                = (ConnectionCreateCommand) request.getStartCommand();
            cmd.setTarget((Shape) getHost().getModel());
            return cmd;
        }
        /* (non-Javadoc)
         * @see org.eclipse.gef.editpolicies.GraphicalNodeEditPolicy#getConnectionCreateCommand(org.eclipse
         */
        protected Command getConnectionCreateCommand(CreateConnectionRequest request) {
            Shape source = (Shape) getHost().getModel();
            int style = ((Integer) request.getNewObjectType()).intValue();
            ConnectionCreateCommand cmd = new ConnectionCreateCommand(source, style);
            request.setStartCommand(cmd);
            return cmd;
        }
```

Allows delete

Allows reconnection

Receives Request

Returns command

ConnectionCreateCommand

```java
/* (non-Javadoc)
 * @see org.eclipse.gef.commands.Command#execute()
 */
public void execute() {
    // create a new connection between source and target
    connection = new Connection(source, target);
    // use the supplied line style
    connection.setLineStyle(lineStyle);
}

/* (non-Javadoc)
 * @see org.eclipse.gef.commands.Command#redo()
 */
public void redo() {
    connection.reconnect();
}

/**
 * Set the target endpoint for the connection.
 * @param target that target endpoint (a non-null Shape instance)
 * @throws IllegalArgumentException if target is null
 */
public void setTarget(Shape target) {
    if (target == null) {
        throw new IllegalArgumentException();
    }
    this.target = target;
}

/* (non-Javadoc)
 * @see org.eclipse.gef.commands.Command#undo()
 */
public void undo() {
    connection.disconnect();
}
```

Command Superclass method

Undo/redo must be manually implemented

ShapeEditPart creating figure

```
/*(non-Javadoc)
 * @see org.eclipse.gef.editparts.AbstractGraphicalEditPart#createFigure()
 */
protected IFigure createFigure() {
    IFigure f = createFigureForModel();
    f.setOpaque(true); // non-transparent figure
    f.setBackgroundColor(ColorConstants.green);
    return f;
}

/**
 * Return a IFigure depending on the instance of the current model element.
 * This allows this EditPart to be used for both sublasses of Shape.
 */
private IFigure createFigureForModel() {
    if (getModel() instanceof EllipticalShape) {
        return new Ellipse();
    } else if (getModel() instanceof RectangularShape) {
        return new RectangleFigure();
    } else {
        // if Shapes gets extended the conditions above must be updated
        throw new IllegalArgumentException();
    }
}
```

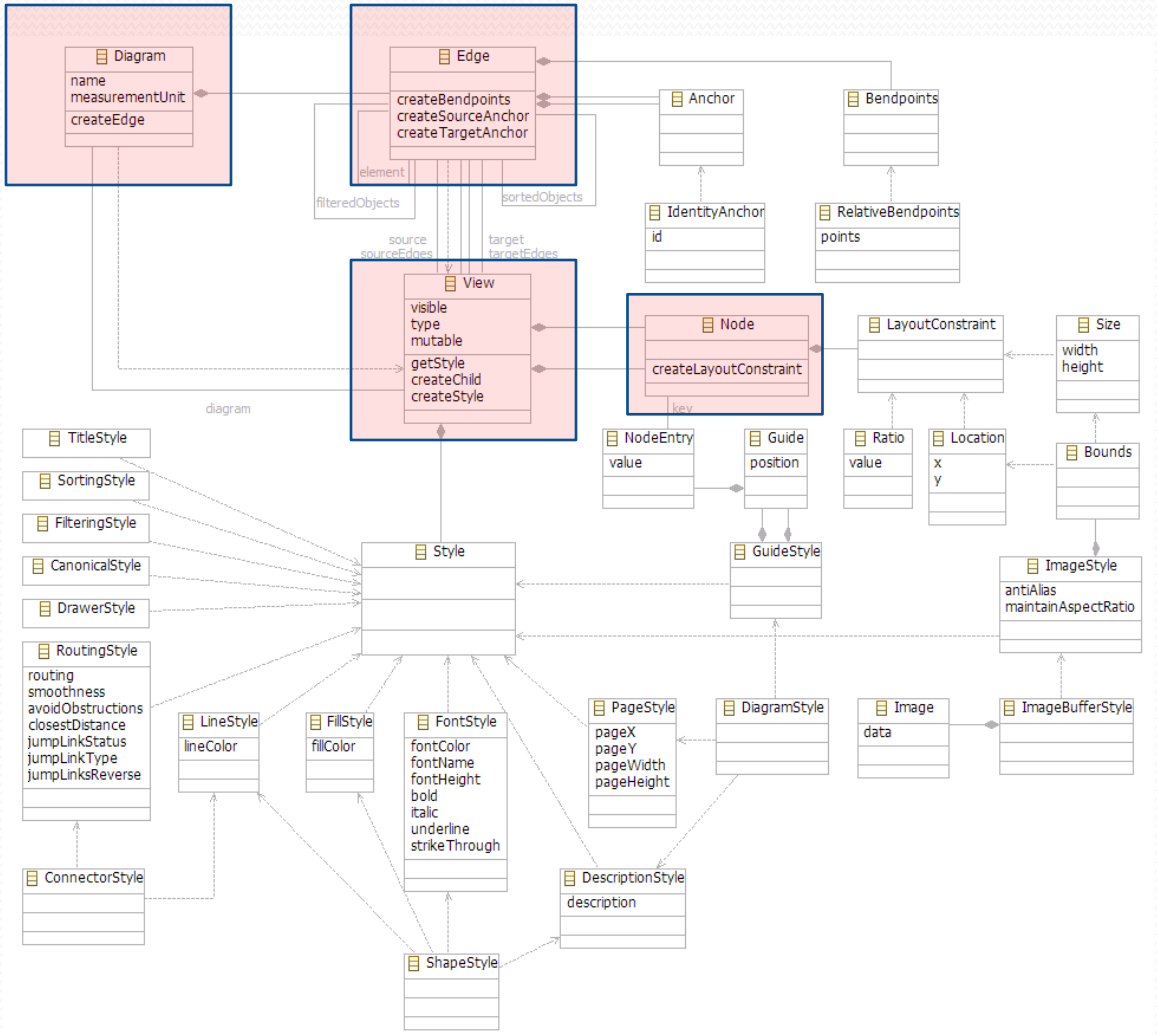
GEF

- That was all on GEF an Draw2D
- Much more could be told
- For more information see
  - Eclipse help
  - Book ”Eclipse Modeling Project” chapter 9
  - Examples in Eclipse
- Valuable to know about GEF
- Easier to use GMF!

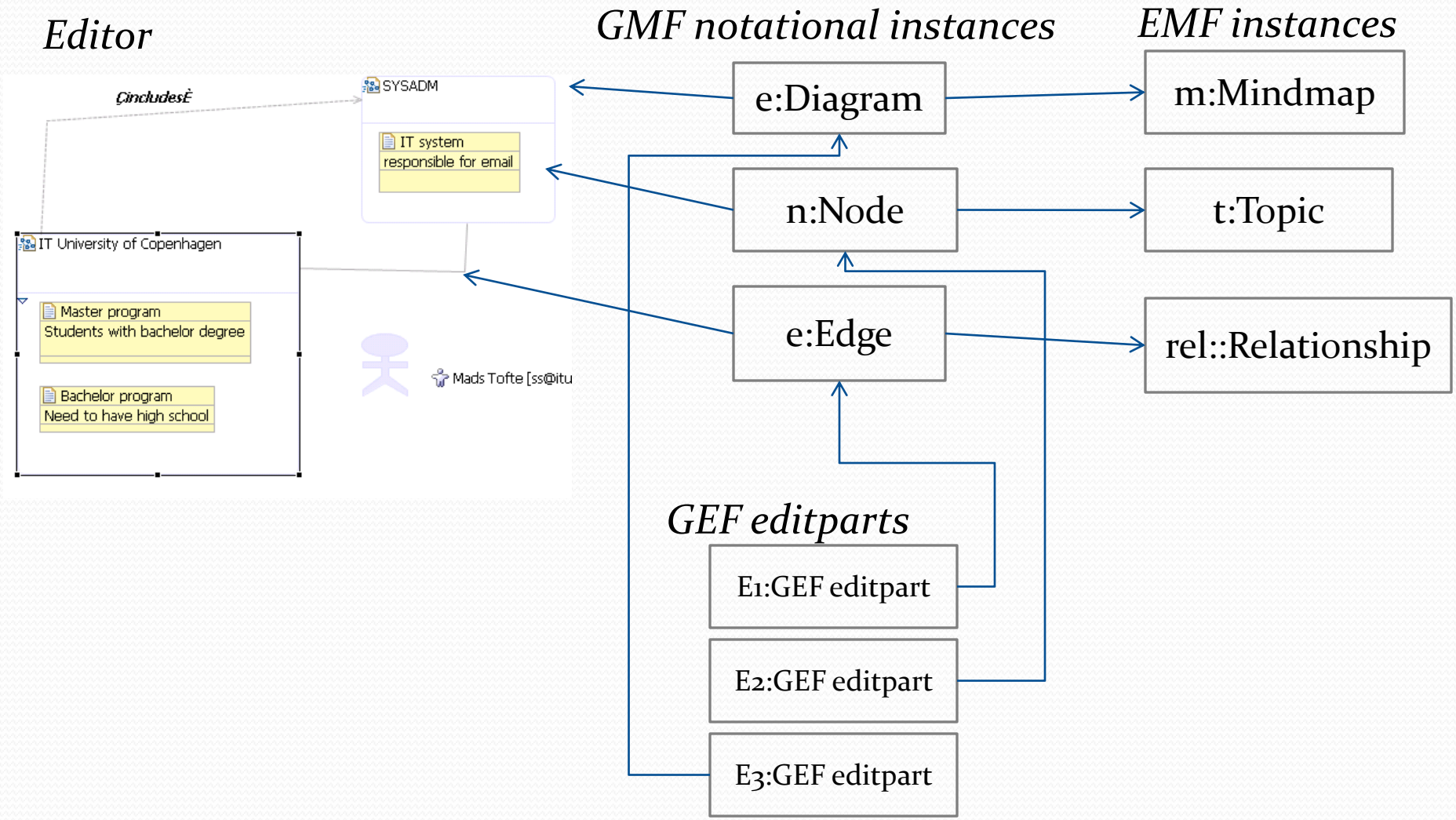
GMF Notational model

- Used to store visual information necessary for diagram drawing
  - Position and size of nodes, shapes, lines, styles, …
- Separated from domain model
  - Domain model also called semantic model
- Is the link between GEF and EMF
- API for programmically create diagrams
- Abstract View class
  - Diagram, Node, Edge sub classes

GMF Notational Ecore model



GMF Notational model



Command infrastructure

- Commands that change the semantic model
  - Built on EMF Transactional component
  - Undoable commands, support undo/redo
  - Bridges EMF commands and GEF commands
  - Key class: *AbstractTransactionalCommand*
  - Many subclasses for reuse/inspiration
    - AddCommand, CreateDiagramCommand
- Commands that do not change model
  - Open diagram, Copy image, Add node…

Extensions

- Service layer
- Service != OSGi service!!
- Collection of 27 extension points to define Service Providers
- Many of the extentions are used by the GMF tooling component
- Example of extensions
  - GlobalActionHandlerProvider
  - Palette Provider
  - EditPartProvider
  - Modeling assist provider
  - Decorator provider
  - EditPolicy provider
  - LayoutProvider
  - …

Mindmap diagram example

- Lets have a look at the generated plugin.xml
- What GMF services does it define?

Customizing generated diagram

- Changing code in the generated diagram plugin
  - Adding @generated NOT as with EMF
- Add extensions in the diagram plugin.xml
- Adding custom templates to the GMF generator
- OR
  - Create a custom plugin and add things there

Customizing Mindmap diagram

- We will see
  - How to customize the diagram in an external plugin
  - Work with the Notation API and Commands
  - Add an extension
- We will implement
  - Programmically add a Topic to the map and a Node to the diagram
  - Open popup dialog on doubleclick on a topic
    - Need to define a PolicyProvider extension
- This approach can be used to change or extend the behavior of editors where you have not access to the source code
- Eg. extend the behavior of IBM Rational modeling editors

Command Example

- We will see
  - Using a custom command
  - Using the Notational Model API
- Programmically create a Topic node
  - Use AbstractTransactionalCommand
  - Create a topic node for the model
  - Create a Node for the diagram
  - Executed from right-click on diagram
- Probably, there is a predefined command that can do the same !!! I have not tried to find it.
- Purpose of example is to illustrate use of APIs

Command Example

Context action that creates a new Topic element and adds a node to the diagram

```
public void run(IAction action) {
    if(editpart!= null){
        final View diagram = (View) editpart.getModel();
        EObject element = diagram.getElement();
        final Map mindmap = (Map) element;
        TransactionalEditingDomain domain = TransactionUtil.getEditingDomain(element);
        AbstractTransactionalCommand cmd = new AbstractTransactionalCommand(domain , "test", null) {

            @Override
            protected CommandResult doExecuteWithResult(IProgressMonitor monitor,
                    IAdaptable info) throws ExecutionException {

                // Now create the Topic
                Topic topic = MindmapFactory.eINSTANCE.createTopic();
                topic.setName("A new topic");
                mindmap.getRootTopics().add(topic);
                Node node = ViewService.createNode(diagram, mindmap, "Topic", MindmapDiagramEditorPlugin.DIAGRAM_PREFERENCES_HINT);
                if(node!= null)
                    node.setElement(topic);

                // Now we could automatically create relationships etc.
                return CommandResult.newOKCommandResult();
            }
        };

        try {
            cmd.execute(new NullProgressMonitor(), null);
            MessageDialog.openInformation(shell,"Custom","OK");
        } catch (ExecutionException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            MessageDialog.openInformation(shell,"Custom","ERROR:"+e.getMessage());
        }
    }
}

public void selectionChanged(IAction action, ISelection selection) {
    IStructuredSelection s = (IStructuredSelection)selection;
    if(!s.isEmpty() && s.getFirstElement() instanceof MapEditPart){
        editpart = (MapEditPart) s.getFirstElement();
    }
}
```

Extension example

EditPolicyProvider extension to Topic EditPart

```
<extension
      point="org.eclipse.gmf.runtime.diagram.ui.editpolicyProviders">
   <editpolicyProvider
         class="dk.itu.mdd.gmf.mindmap.diagram.custom.EditPolicyProvider">
      <Priority
            name="Medium">
      </Priority>
      <context
            editparts="mindmap.diagram.edit.parts.TopicEditPart">
      </context>
      <object
            class="mindmap.diagram.edit.parts.TopicEditPart"
            id="mindmap.diagram.edit.parts.TopicEditPart">
      </object>
   </editpolicyProvider>
</extension>
```

Extension example

EditPolicyProvider implementation

```
public class EditPolicyProvider extends AbstractProvider implements IEditPolicyProvider {

    @Override
    public void createEditPolicies(EditPart editPart) {
        editPart.installEditPolicy(EditPolicyRoles.OPEN_ROLE, new OpenEditPolicy());
    }

    @Override
    public boolean provides(IOperation operation) {
        if(operation instanceof CreateEditPoliciesOperation){
            EditPart part = ((CreateEditPoliciesOperation) operation).getEditPart();
            if(part instanceof TopicEditPart){
                return true;
            }
        }
        return false;
    }
```

Extension example

- OpenEditPolicy implementation
- Extends GMF OpenEditPolicy class
  - Receives requests of type REQ_OPEN
  - Childs specify what to do

```
public class OpenEditPolicy extends org.eclipse.gmf.runtime.diagram.ui.editpolicies.OpenEditPolicy {
    @Override
    protected Command getOpenCommand(Request request) {

        return new Command() {
            @Override
            public void execute() {
                MessageDialog.openInformation(Display.getCurrent().getActiveShell(), "Doubleclick", "It works");
            }
        };
    }
}
```

Not told about GMF

- Tooling
  - Dynamic templates
  - Extension models
  - Related diagrams
  - Phantom nodes
  - Display labels, …
- Runtime
  - Inside the Command infrastructure
  - Use of predefined commands
  - Whats going on under the hood
  - Element creation
  - GMF and GEF internal differencies
  - Details
- You can find more information in the book, in Eclipse help, and at eclipse.org under the GMF project.

Summary

- OCL in GMF tooling
  - Constraints
  - Feature initialisers
  - Validations
- GMF Runtime
  - GEF and Draw2D
  - Notational model
  - Commands
  - Extensions

Next lectures

- Four lectures by tonny
  - Exam report
  - Textural syntax
  - Textural syntax
  - Model 2 Model transformations
- Two lectures by Steen
  - Model 2 Text transformations
  - UML Profiles

Exercise

- 1: Create a constraint on diagram
  - A connection must not have a Start as target or an End as source.
- 2: Create feature initialiser
  - When creating a DataGroup, add a parameter with name =”travel date”
- 3: Create validations:
  - Items must have at least 1 ingoing and 1 outgoing connection
  - A CarRental must be followed by an Insurance of type CarInsurance
- 4: Add an EditPolicyProvider
  - It must enable the user to doubleclick on a CarRental item and get a popup dialog where the user can enter a name.
  - Create an Insurance node programmically from the EditPolicyProvider.

Evaluation next week

- Please evaluate this course!
- Its the only way to improve the course and ourselves