



Eclipse Modelling Framework

*S. de Lorenzo, E. Ebensperger,
P. Neulinger, M. Schrempf
and S. Vikoler*

Table of Contents

- Introduction
 - EMF, UML and Composite Pattern explanations
- Framework
 - Generell and detailed description
- EMF Demonstration and GUI Code Presentation
- Example of Possibilities and GUI Demonstration

Assignment

- Create a common framework for the simulation of almost everything, e.g.: computer parts
 - This parts have a:
 - Name
 - Properties
 - Ports
 - Other parts as children
- We used the composite pattern to realize the framework.

Assignment

- Components and compositions of components, called composites, should be able to send data via connected I/O and hybrid ports to each other
- A GUI controls the framework and sets its properties
- We created a modified Composite-Pattern-UML-Diagram with the EMF to accomplish the task

Hypothesis

- We are able to simulate a computer with different composites, which contain components
- These composites are connected via ports to each other
- A composite and a component can have n ports

Hypothesis

- Ports can only have one other connected port
- The three types of ports must be considered
 - Input
 - Output
 - Hybrid
- Data is stored in the input and hybrid ports

Materials

- Eclipse Modeling Framework
 - Books in Works Cited
 - Tutorials
 - Examples
- Unified Modeling Language
 - Software Engineering Lecture
 - Book in Works Cited
- Composite Pattern
 - Books in Works Cited

EMF – What is that?

- Eclipse Modeling Framework (EMF) is an Eclipse-based modeling framework and code generation facility
- You can build tools and other applications based on a structured data model
- Models can be specified using annotated Java, UML, XML documents, or modeling tools, then imported into EMF
- We used a UML-Diagram to draw the appendences from the interfaces and classes we created

EMF – Start problems

- Compliance level automatically turned down
 - Non generic Elists
- Hand written code implementations of the body of a method
- "Delete" deletes references from the diagram, not from the model
- No java framework imports

EMF – Start problems

- Only byte arrays
- Only primitive data types for global variables
- Useless error messages
- Interfaces cannot inherit from interfaces
 - Leads to useless code

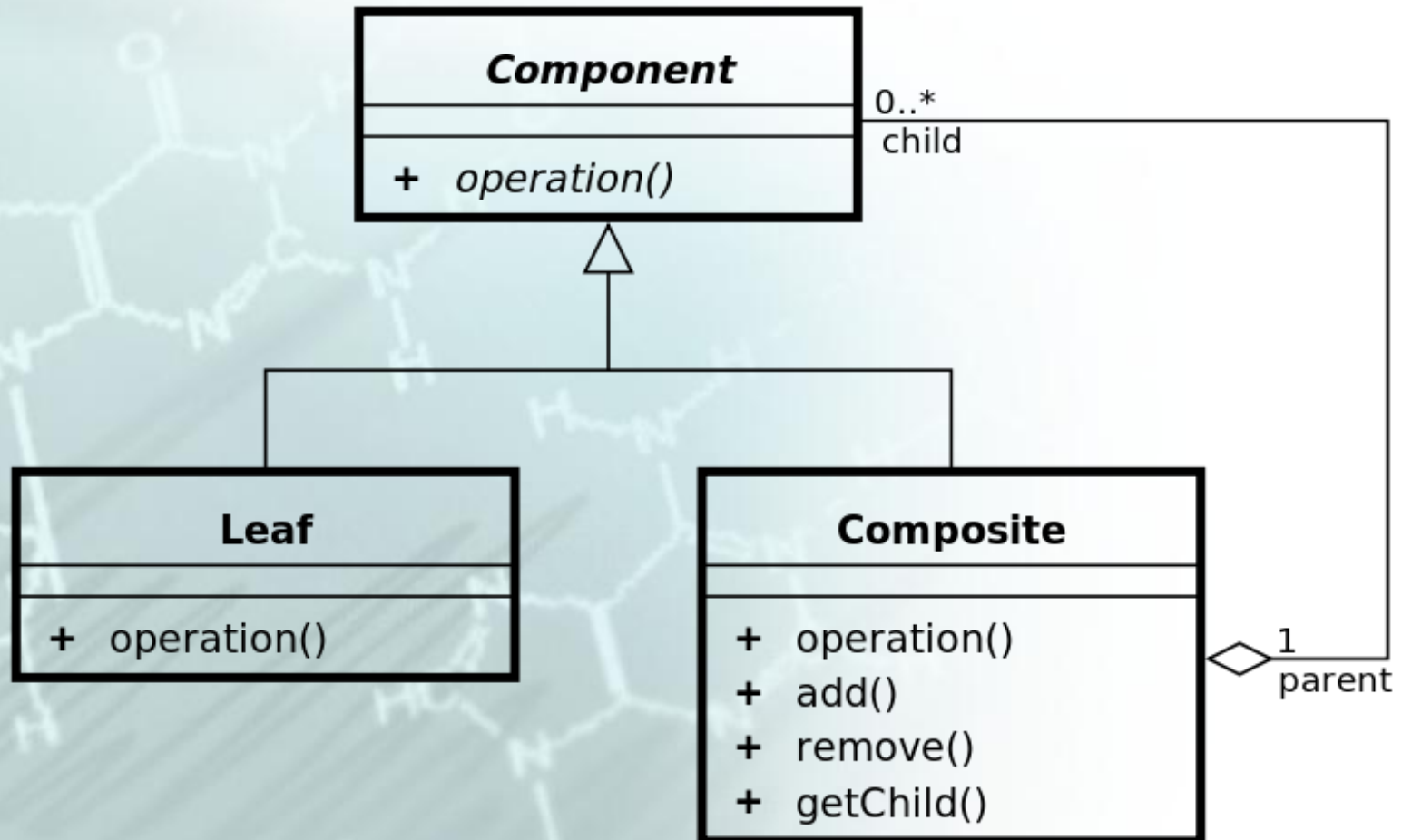
EMF – Advantages

- Easy:
 - UML diagrams
 - Installation
 - Code generation
 - Simple handling
 - Documentation
 - Editing
- Insert own code in methods
- You actually see the data structure

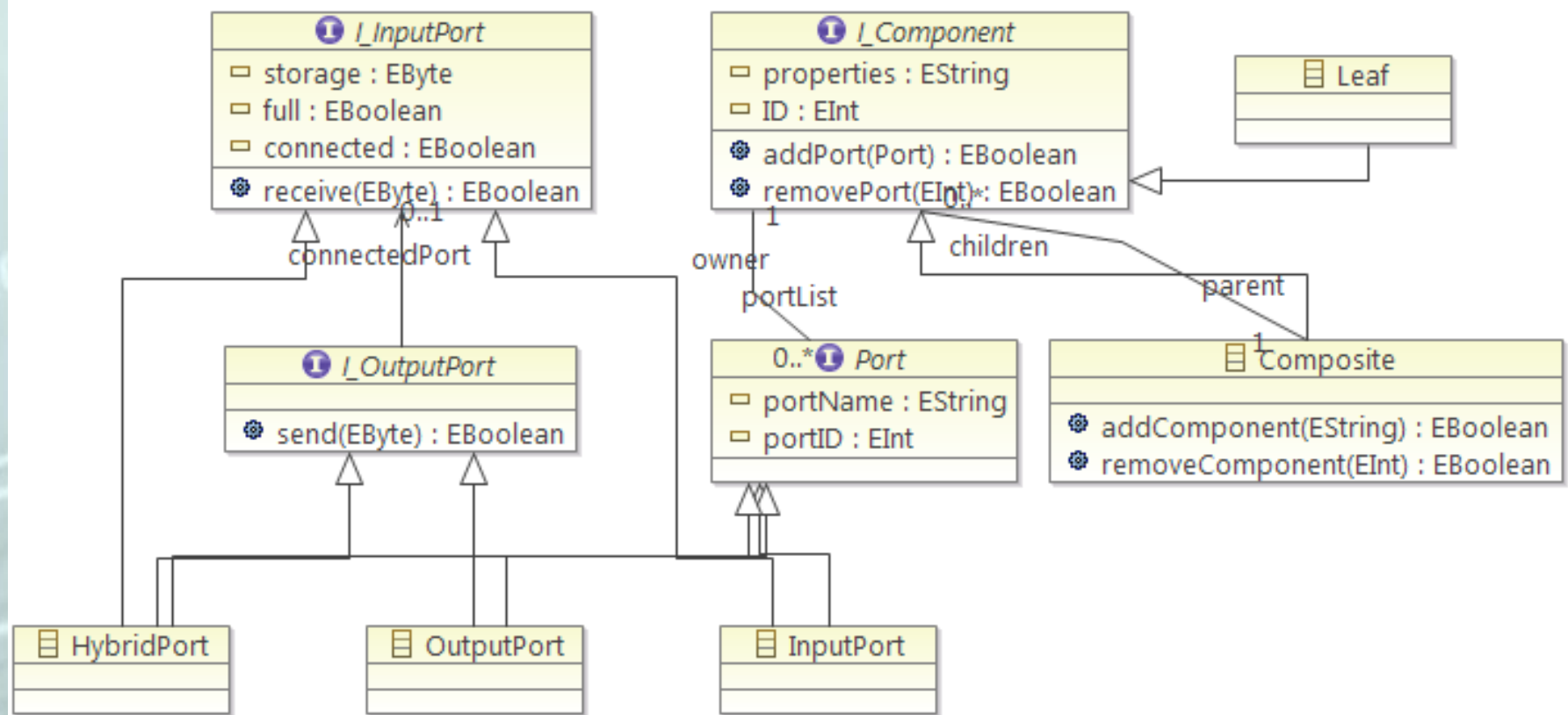
Unified Modeling Language

- Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering
- The UML is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development
- Standardized visual data structure representation

Composite Pattern



The Framework



I_Component

 I_Component

□ properties : EString

□ ID : EInt

⚙ addPort(Port) : EBoolean

⚙ removePort(EInt): EBoolean

- Properties, specified by the user
- ID, Integer identification number of the component
- addPort, Adds a Input-, Output- or Hybrid Port to the Component
- removePort, Removes a Port from the component, specified by its ID

I_Component

I_Component

□ properties : EString

□ ID : EInt

⚙ addPort(Port) : EBoolean

⚙ removePort(EInt)*: EBoolean

- The child contains the reference of its parent object. The object itself is called parent
- EList<Port> portList 0:N, contains all ports of a composite

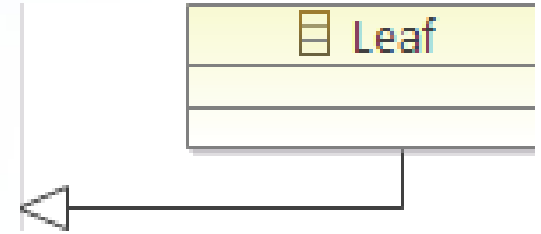
Composite

Composite

⚙️ addComponent(EString) : EBoolean
⚙️ removeComponent(EInt) : EBoolean

- EList<Component> children 0:N, contains all children from this composite
- AddComponent, adds a component as a new child in the EList
- RemoveComponent, deletes a component, specified by an ID, from the EList
- Inherits from I_Component

Leaf



- Is a atomic component without children
- Always the last element in a composite-component structure
- Has no own attributes and methods
- Is needed to know where an end is
- Inherits from I_Component

I_Port


0..*  Port

▣ portName : EString

▣ portID : EInt

- The general interface for all ports
- portName, Name of the Port
- portID, Integer identification number of the Port
- All ports inherit from this interface
- In EMF an interface cannot inherit from another interface!

I_InputPort

 I_InputPort

 storage : EByte

 full : EBoolean

 connected : EBoolean

 receive(EByte) : EBoolean

- Interface for all ports which can receive data:
 - Input
 - Hybrid
- storage, To stores the received data
- full, This is a flag its true if the input port is full, false if not
- connected, is also a flag which declares if a port is already connected or not
- Receive, Receives data and stores it in the storage and considers the full flag


I_OutputPort

- Interface for all ports which can send data
 - Output
 - Hybrid
- Send, Uses an inputPort class object and calls its receive method

OutputPort

- Sends data to an input port
- Checks if the connected port is full, or not connected
- Returns false if anything goes wrong
- `connectedPort 0:1`, is the object which is connected

InputPort

 InputPort

- Gets data from an output or hybrid port and stores it
- For safety, it returns also false if the storage is already full or anything got wrong

HybridPort

- Only port who inherits from I_InputPort and I_OutputPort
- Can perform all actions of an input and output port
- connectedPort 0:1, is the object which is connected

The background of the slide features faint, light blue chemical structures. These include a complex polycyclic aromatic hydrocarbon (PAH) on the left, a nucleotide-like structure in the center, and a purine-like base on the right. The structures are rendered in a low-contrast, sketchy style.

EMF

Demonstration

GUI

- Graphical user interface is a type of user interface that allows users to interact with electronic devices with images rather than text commands.
- Programmed with Java Swing
- Controls the components, you can do everything with the GUI:
 - Add/Delete Components
 - Add/Delete Ports
 - Add/Delete Connections
 - Send Data
 - Get an basic overview of the already existing data structure
 - Navigate through it

Changes

- Added package: gui
 - Main
 - Controller
 - MainViewPanel
- Added in package sp.impl, the class MainModel
 - API for GUI
 - Controls the other models
 - Model code is untouched

The background of the slide features faint, overlapping chemical structures. On the left, there is a complex molecule with a fused ring system, including a pyrimidine-like ring and a carbonyl group. Below it, a polymer chain is visible, consisting of repeating units with a central carbon atom bonded to two other groups. To the right, there is a smaller, more complex molecule with multiple rings and functional groups, including what appears to be a carboxylic acid or ester group. The overall background is a light blue gradient with these chemical structures in a slightly darker blue.

GUI

Demonstration

An Example

- So after all, we know how our framework works.
- To show, that it is a useful tool, we implemented an example to demonstrate how it works.
- Therefore we decided to implement a production flow of the game, ANNO 2070, to demonstrate how much different working stations are required for a balanced production.

An Example



An Example



- First we need sand and copper to construct electrical components.
- From corn and alga we create biopolymer.
- After that, we combine these two parts to construct little robots, which will help our habitants in all-day-situations.

Works Cited

- Eclipse Modeling Framework, Second Edition
 - Dave Steinberg
 - Frank Budinsky
 - Marcelo Paternostro
 - Ed Merks
- Software Engineering: Architektur-Design und Prozessorientierung
 - Wolfgang Pree
 - Gustav Pomberger
- Wikipedia



Eclipse Modeling Framework

Thank's for your attention.
