# Domain-specific IDEs in embedded automotive software

## EclipseCon 2005.

**Michael Rudorfer**
Paul Hoser, Tilmann Ochs, Christian Salzmann, Martin Thiede

**BMW Car IT**

**Markus Völter ***

* völter ingenieurbüro für softwaretechnologie

Burlingame,
March 2005

**Sheer**
**Driving Pleasure**

# Domain-specific IDEs in embedded automotive software.
## Overview.

Background / Challenge

BMW Car IT AUTOSAR prototype

Domain-specific IDE with Eclipse
- IDE Step 1: Projects, XML, Generator
- IDE Step 2: Model Explorers
- IDE Step 3: Graphical Editors

Graphical Editing Framework GEF
- Working with GEF
- Generating GEF editors

Summary

# Domain-specific IDEs in embedded automotive software.

## Today's Situation I.

- Embedded operating systems do not provide full hardware abstraction
- Many different hardware plattforms are used
- Limited modularity, little reuse: if hardware (processor type…) changes, the software is often re-developed from scratch
- Lifecycle gap: the lifetime of cars are usually much longer than those of ECUs
- Changing hardware requires
  - compatible re-development of software on the new hardware, or
  - available stocks of old hardware
- Unit based cost structure drives car manufacturer to use cheapest possible hardware → limited resources

# Domain-specific IDEs in embedded automotive software.

## Today‘s Situation II.

The increasing amount and complexity of software in cars requires new methods and tools

A component-based approach is aspired to master challenges in future automotive software development

# Domain-specific IDEs in embedded automotive software.

## Overview.

Background / Challenge

BMW Car IT AUTOSAR prototype

Domain-specific IDE with Eclipse
- IDE Step 1: Projects, XML, Generator
- IDE Step 2: Model Explorers
- IDE Step 3: Graphical Editors

Graphical Editing Framework GEF
- Working with GEF
- Generating GEF editors

Summary

# Domain-specific IDEs in embedded automotive software.

## Solution Approach I.

- An architecture-centric programming model will be used in the future to realize component-based development for ECUs
- The goal is a seperation of concerns:
  - Software components only deal with application logic and can be developed independently of the underlying hardware and bus systems
  - Middleware-layer handles aspects of remote communication, component lifecyle, scheduling and timing as well as OS abstraction

# Domain-specific IDEs in embedded automotive software.

## Related Technologies: What's different in AUTOSAR?

- Midlleware technologies like J2EE or Corba are not suitable for various reasons:
  - Resource limitations in the automotive domain prevent us from using heavyweight application servers or ORBs
- AUTOSAR is an upcoming automotive industry standard that implements compareable concepts using a different approach
  - As the desired system is static after system design, the middleware-layer can be tailored for the special requiremenrs of the scenario (omit features not needed)
  - A code-generator uses information from a system model to generate a highly optimized, lean middleware-layer

# Domain-specific IDEs in embedded automotive software.

## Why a prototype?

- BMW Car IT is building a proof of concept prototype of the AUTOSAR standard
- Standard based on committee discussions
- Early prototypes are vital for validating “paperware” concepts
- Provide timely feedback to the committee from an implementer’s perspective
- Give future users an early impression of the upcoming development paradigm and tooling

# Domain-specific IDEs in embedded automotive software.

## IDE Support – Why?

- Since the prototype is intended to give developers a „feeling“ of what AUTOSAR is going to be, and to convince them of the idea, developer acceptance is a major concern for the prototype.
- As a consequence, an IDE had to be developed.
- We decided to use Eclipse as a basis for the IDE.

# Domain-specific IDEs in embedded automotive software.

## Overview.

Background / Challenge

BMW Car IT AUTOSAR prototype

Domain-specific IDE with Eclipse
- IDE Step 1: Projects, XML, Generator
- IDE Step 2: Model Explorers
- IDE Step 3: Graphical Editors

Graphical Editing Framework GEF
- Working with GEF
- Generating GEF editors

Summary

# Domain-specific IDEs in embedded automotive software.
## Models.

- Models describe various aspects of the system, such as
  - Components and Interfaces
  - Component Instances and Connectors
  - Hardware characteristics, topology and networks
  - Deployment of instances on hardware topologies
  - Timing

- Format and content (metamodel and concrete syntax) is defined by the AUTOSAR committee

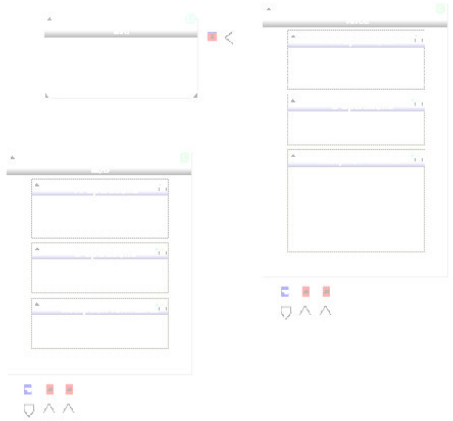
- In the prototype, a code-generator (openArchitecture Ware) verifies these models for consistency and correctness before code generation starts.

- Code generation templates define the RTE code generated for each model element

# Domain-specific IDEs in embedded automotive software.

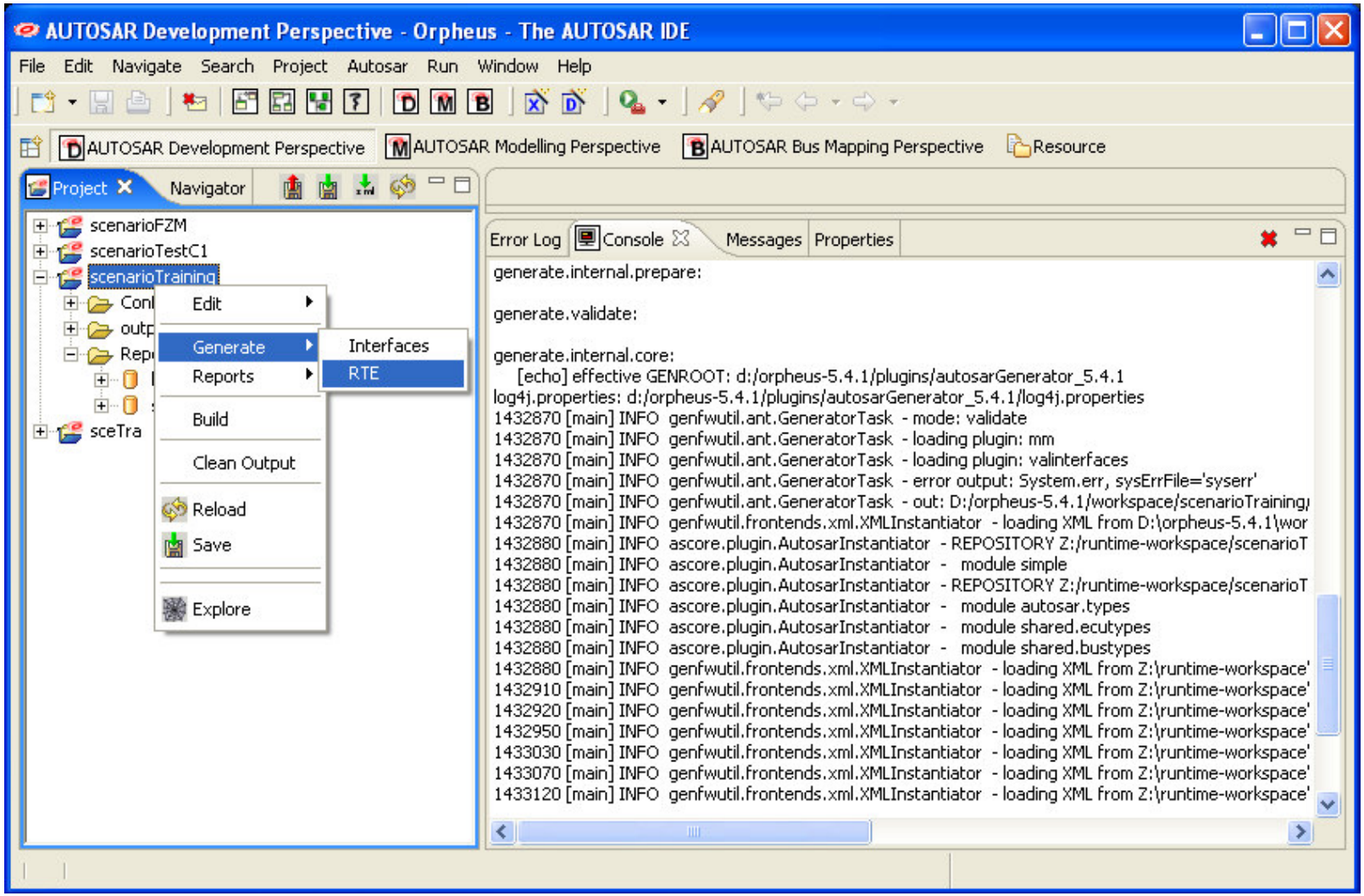
## IDE I: Projects, XML, Generator.

- The first iteration of the IDE has the following features:
  - AUTOSAR-specific projects
  - Code generator integration
  - Generator messages displayed in IDE
  - Model editing based on manual editing of XML files (models are in XML)



# Domain-specific IDEs in embedded automotive software.

## IDE I: Projects, XML, Generator.




Generator Access and Log

# Domain-specific IDEs in embedded automotive software.

## Overview.

Background / Challenge

BMW Car IT AUTOSAR prototype

Domain-specific IDE with Eclipse
- IDE Step 1: Projects, XML, Generator
- IDE Step 2: Model Explorers
- IDE Step 3: Graphical Editors

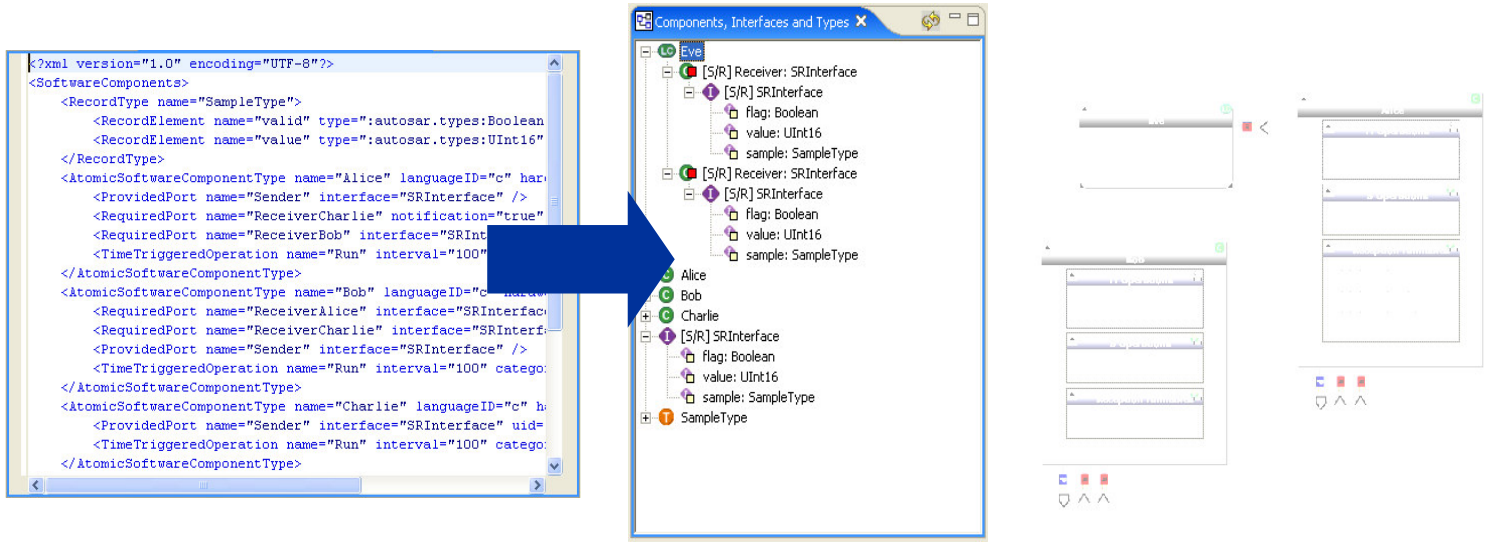
Graphical Editing Framework GEF
- Working with GEF
- Generating GEF editors

Summary

# Domain-specific IDEs in embedded automotive software.

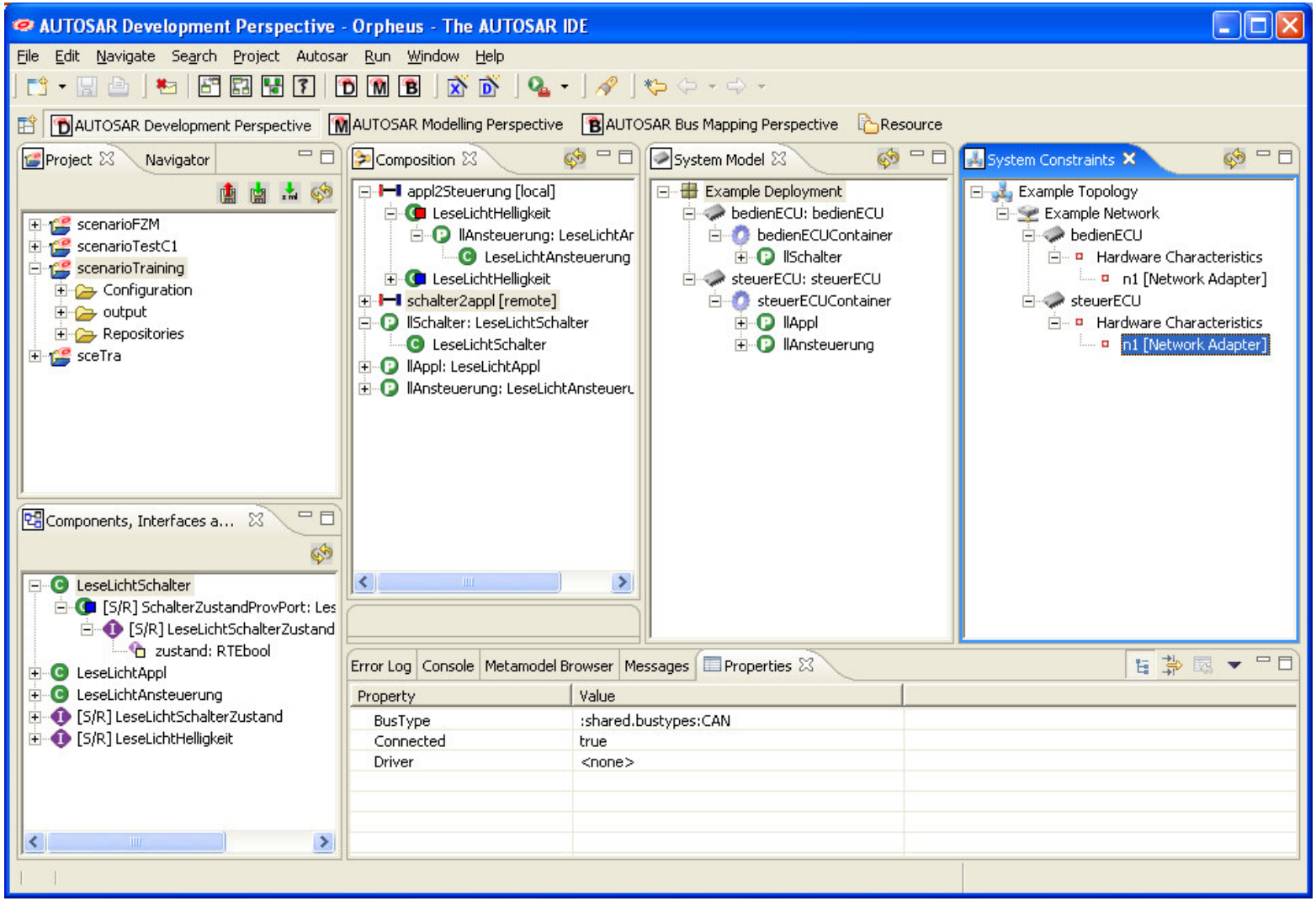
## IDE II: Model Explorers.

- Models are getting larger and more complex
- In order to provide a better picture of the XML-based models, we started developing tree-based model viewers.



# Domain-specific IDEs in embedded automotive software.

## IDE II: Model Explorers.




Various Model Browsers in the IDE

# Domain-specific IDEs in embedded automotive software.
## Overview.

Background / Challenge

BMW Car IT AUTOSAR prototype

Domain-specific IDE with Eclipse
- IDE Step 1: Projects, XML, Generator
- IDE Step 2: Model Explorers
- IDE Step 3: Graphical Editors

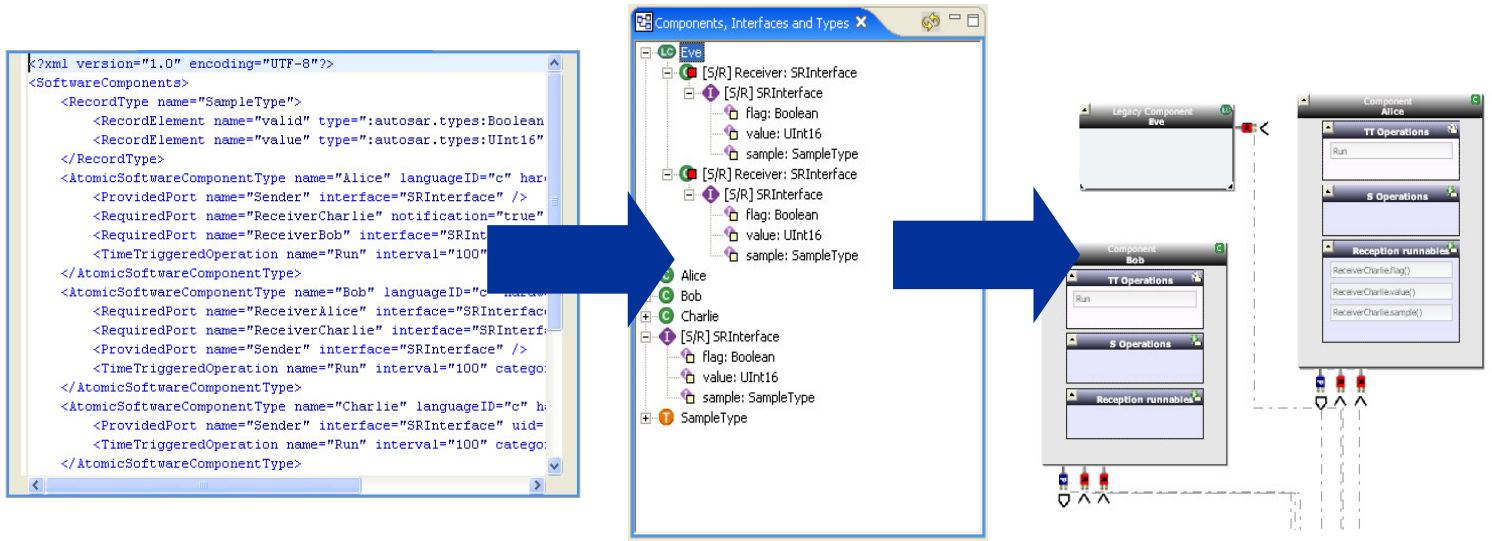
Graphical Editing Framework GEF
- Working with GEF
- Generating GEF editors

Summary

# Domain-specific IDEs in embedded automotive software.

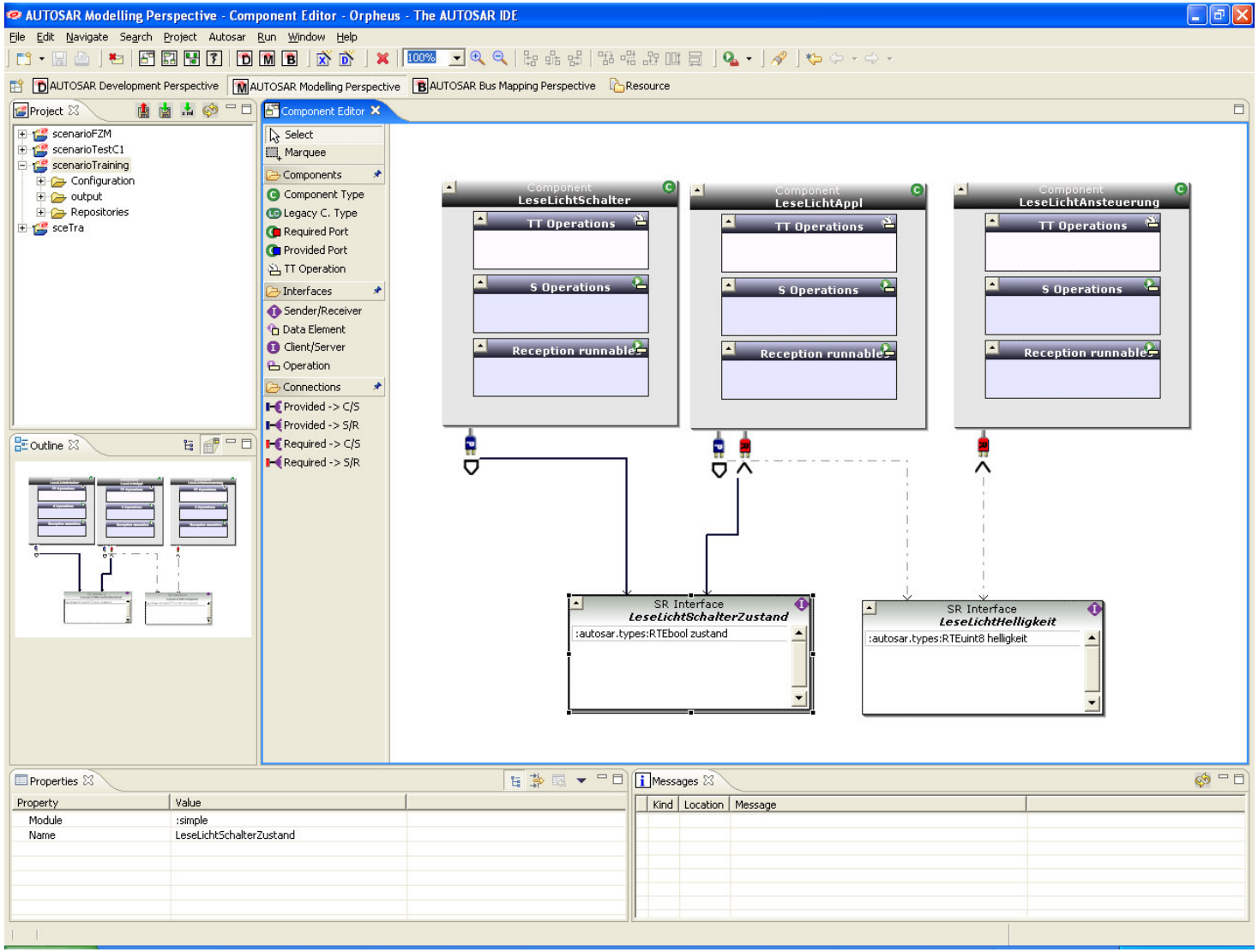
## IDE III: Graphical Editors.

- "Real" developers started using the prototype
  → Read-Only Browsers were not sufficient anymore
- AUTOSAR defines a graphical notation for the various kinds of models. This notation of AUTOSAR is not UML-based (although it does use boxes and lines).
- We use GEF to build graphical editors for the AUTOSAR specific notations.



# Domain-specific IDEs in embedded automotive software.

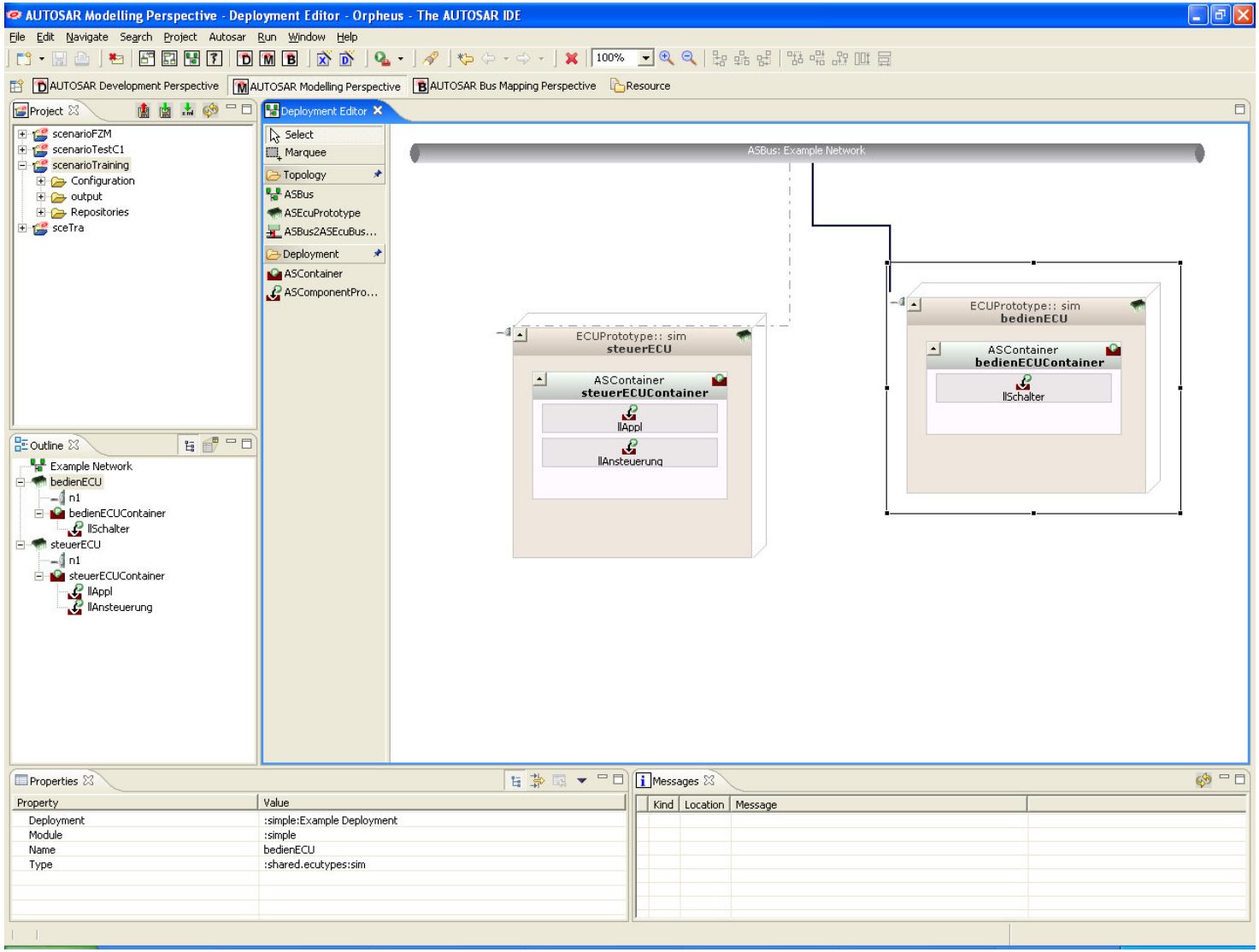
## IDE III: Graphical Editors.




Editing Components and Interfaces

# Domain-specific IDEs in embedded automotive software.

## IDE III: Graphical Editors.




Networks, ECUs and Deployments

# Domain-specific IDEs in embedded automotive software.
## Overview.

Background / Challenge

BMW Car IT AUTOSAR prototype

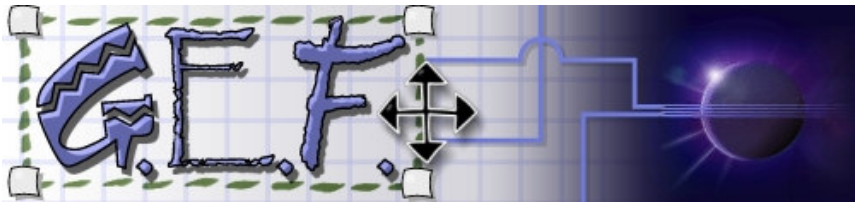
Domain-specific IDE with Eclipse
- IDE Step 1: Projects, XML, Generator
- IDE Step 2: Model Explorers
- IDE Step 3: Graphical Editors

Graphical Editing Framework GEF
- Working with GEF
- Generating GEF editors

Summary

# Domain-specific IDEs in embedded automotive software.
## Working with GEF.

- GEF is a very powerful framework for buiding graphical editors for graph-like structures on the Eclipse Platform.



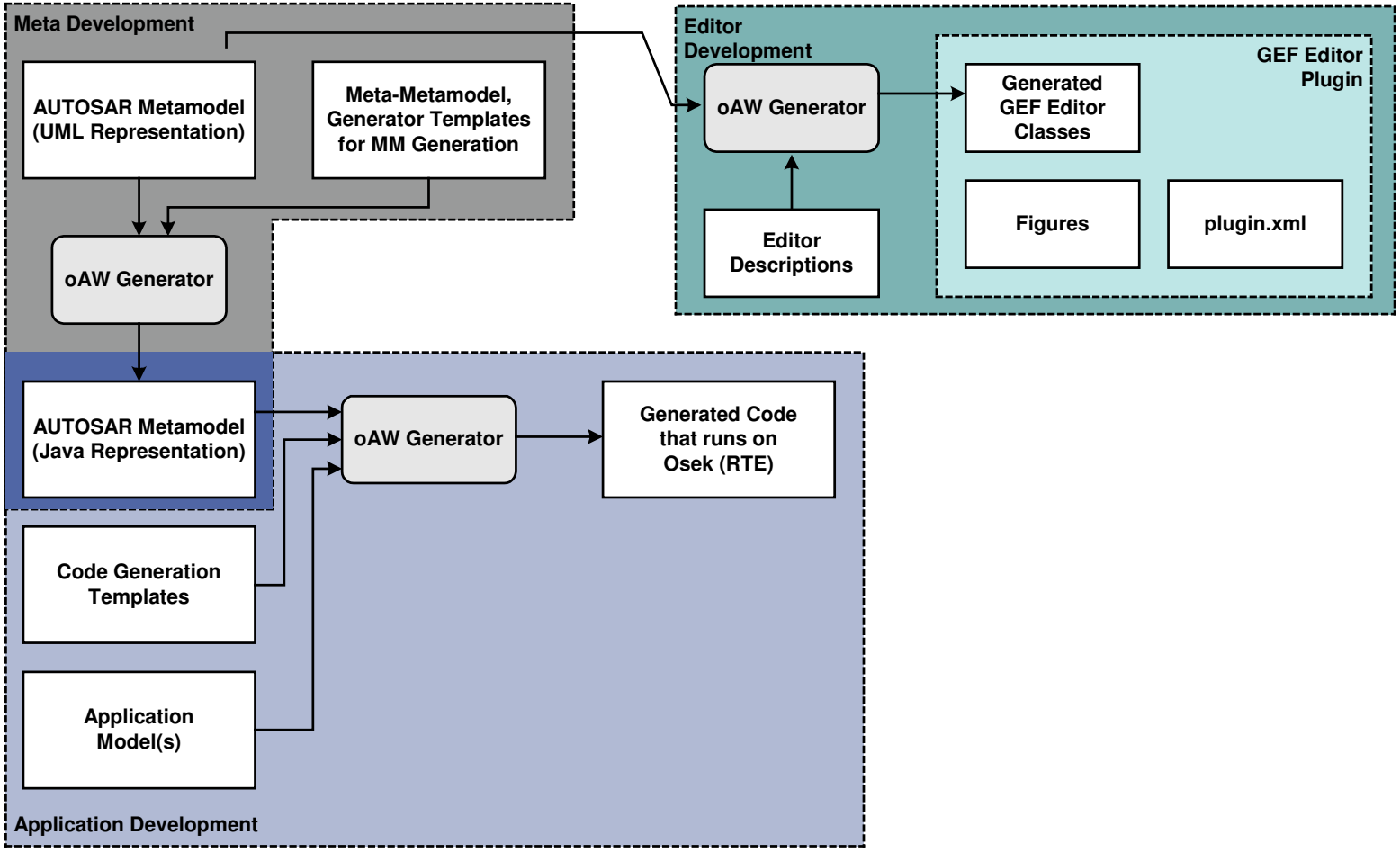

- However, it is not easy to use:
  - Documentation is improvable
  - A specific „intent“ is scattered through many classes (models, parts, policies, …); i.e. many artifacts have to be adapted to realize a new feature
  - We needed to stay in sync with the evolving metamodel we use for the AUTOSAR generator development; frequent changes!

# Domain-specific IDEs in embedded automotive software.
## Generating GEF editors.

- As a consequence, we now generate most of the GEF editors from our domain metamodel.
- Relations between meta-classes are defined in an instantiator
- Behaviour of the editors is “configured” by the instantiator
- Approach gives us serveral hooks to implement sophisticated domain model-specific behaviour
- Only figures have to be coded by hand
- → Editors are always in sync with the metamodel, changes become simpler, and GEF development in general is a pleasure now ☺

# Domain-specific IDEs in embedded automotive software.

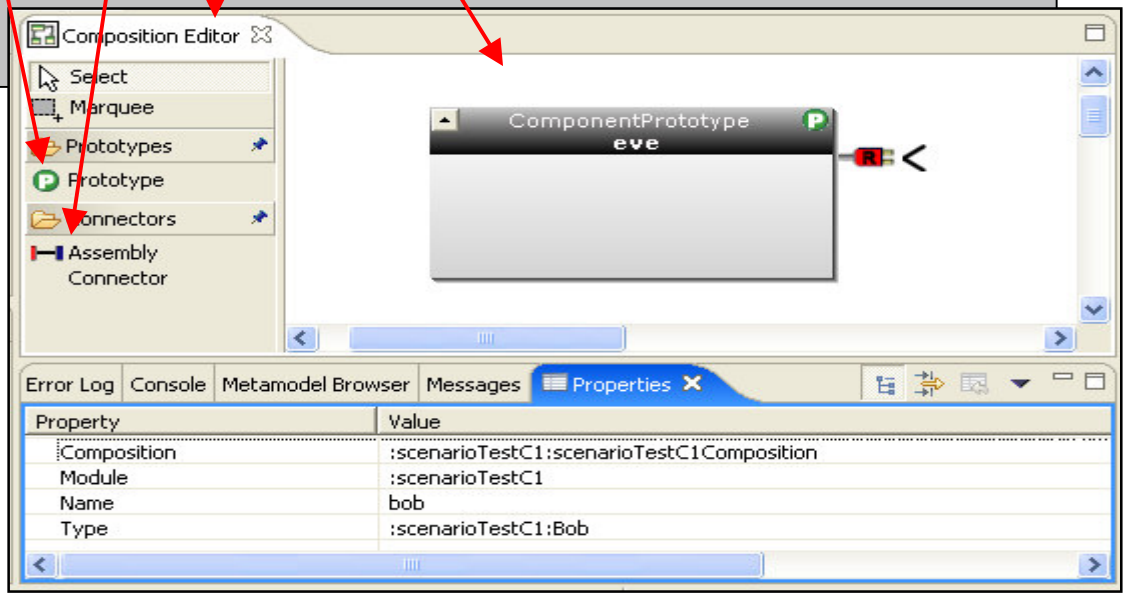
## Generator infrastructure II.



# Domain-specific IDEs in embedded automotive software.

## Example – setting up the editor.

```
EditorDesc editor = new EditorDesc();
metamodel.addEditor(ed);
editor.setName("CompositionEditor");

MetaClassDesc prototype = new MetaclassDesc( … );
editor.addItem(prototype);

OneToOneAssociationDesc assocCompositionP2P =
   new OneToOneAssociationDesc ( … );

editor.addConnectionItem(assocCompositionP2P);

OneToManyContainmentDesc editor2prototype =
    new OneToManyContainmentDesc(editor, prototype,
    "ASComponentPrototype");

editor.configure();
```



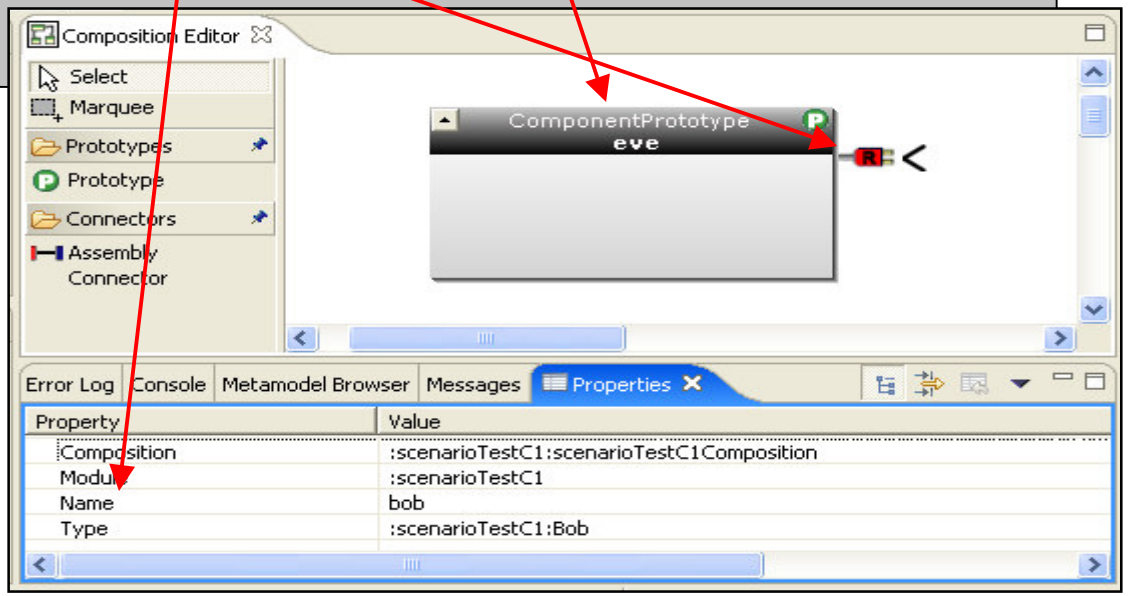
# Domain-specific IDEs in embedded automotive software.

## Example – relations between metaclasses.

```
// define class description from "ComponentPrototype"
private MetaclassDesc prototype = new MetaclassDesc(
        new RClass(metamodel, ASComponentPrototype.class),
        MetaclassDesc.NO_ASSOCIATIONTYPE, true);

prototype.setTitle("Prototype");
prototype.addStringProperty("name").setVisibleInDiagram(true);
prototype.addStringProperty("module", ASModule.class,
        "QualifiedName").setVisibleInDiagram(true);;

metamodel.addMetaclass(prototype);

MetaclassDesc portPrototype = new MetaclassDesc(… // port prototype

// define containment of port and component prototype
new OneToManyContainmentDesc(prototype, portPrototype,
"PortPrototype");
```

Generator Run

XYEditor.java
XYEditorPart.java
XYEditorModel.java
XYEditPolicy.java
...
+XYFigure.java



# Domain-specific IDEs in embedded automotive software.
## Generating GEF editors.

Some numbers:

From
- less than 100 LOC editor description
- plus a few hundred LOC for the figures

We generate
- 50+ GEF Source-Files
- containing serveral thousand LOC of GEF Code

Generating GEF editors simplyfies development drastically!

# Domain-specific IDEs in embedded automotive software.

## Overview.

Background / Challenge

BMW Car IT AUTOSAR prototype

Domain-specific IDE with Eclipse
- IDE Step 1: Projects, XML, Generator
- IDE Step 2: Model Explorers
- IDE Step 3: Graphical Editors

Graphical Editing Framework GEF
- Working with GEF
- Generating GEF editors

Summary

# Domain-specific IDEs in embedded automotive software.
## Summary.

- The increasing amount and complexity of software in cars will in future require new methods and tools for software development - Model-driven software development is a promising approach
- Early prototyping is a major contribution to a stable, reliable infrastructure
- Eclipse is a very good basis for developing custom, domain-specific IDEs
- The combination of GEF and code generation yields a very productive environment for building graphical editors.
- The approach taken here in the automotive domain could be suitable for other domains as well

# Domain-specific IDEs in embedded automotive software.
## References and Acknowledgements.

**Acknowledgements:**

Bernd Kolb
Imran Shafiq

**References:**

AUTOSAR
http://www.autosar.org

BMW Car IT
http://www.bmw-carit.de

openArchitectureWare
http://www.openarchitectureware.org