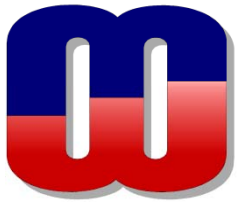
Domain-Specific Modeling Languages

12 April 2010

Juha-Pekka Tolvanen, Ph.D.
MetaCase

Company



MetaCase

- Leading provider of Domain-Specific Modeling environments
  - MetaEdit+® tool
  - supporting services
- Founded 1991
- Ownership private
- Offices in Dallas, Texas and Jyväskylä, Finland
- Several thousand licenses to 30+ countries

MetaEdit+

- Automotive
- Avionics
- Consumer electronics
- Factory automation
- Home automation
- Medical devices
- Mobile phones
- Railway systems
- Telecom systems

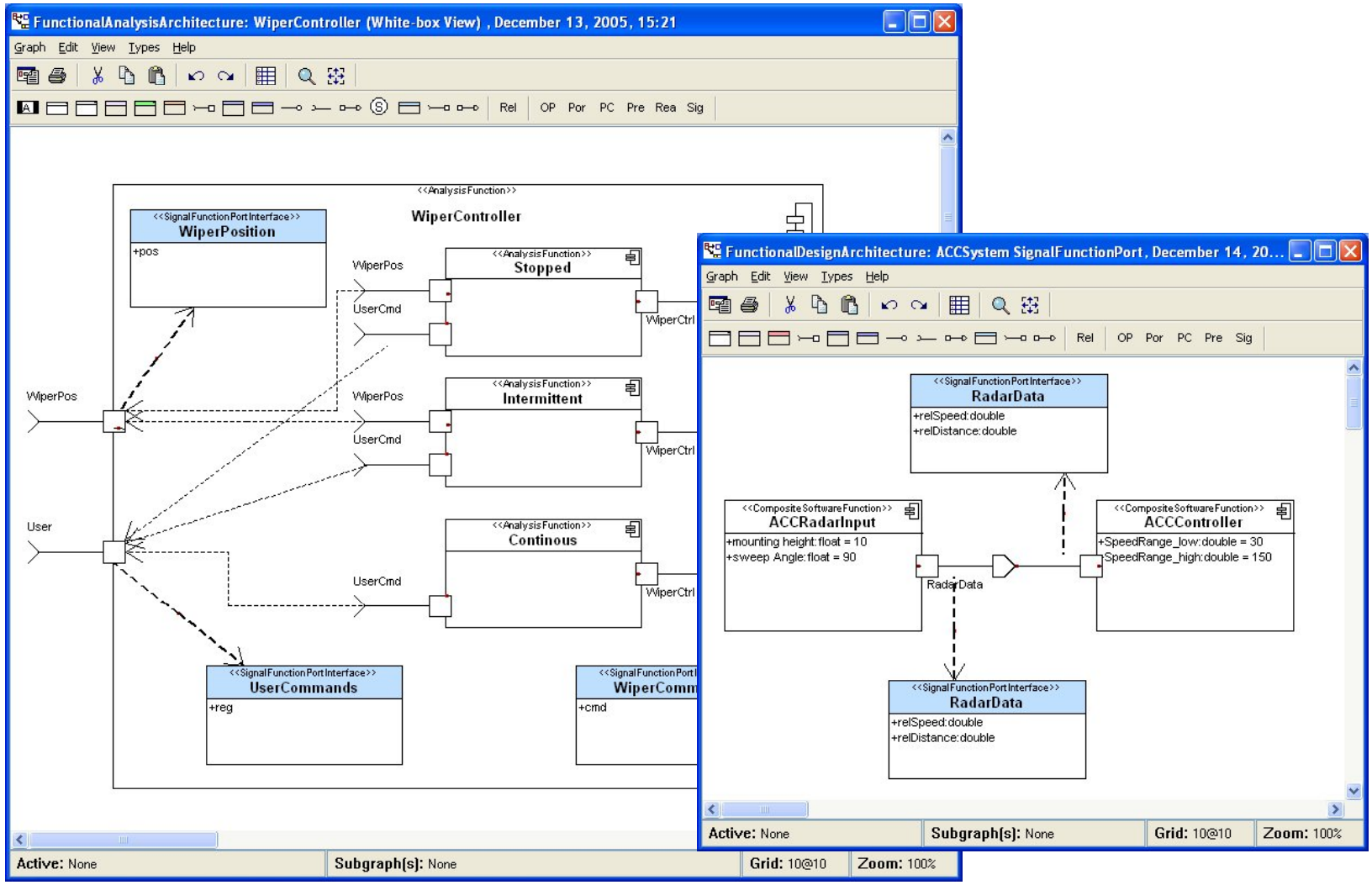
What you want to do with ADL?

- Architectures are not general purpose but designed for solving specific problems
- What you want to specify with an (AD)Langauge
  - Component structure
  - Interfaces
  - Communication protocols
  - Mapping between software and hardware architectures
  - Configuration
  - Architecture analysis
  - Application construction guidelines
  - ....
  - or just document it

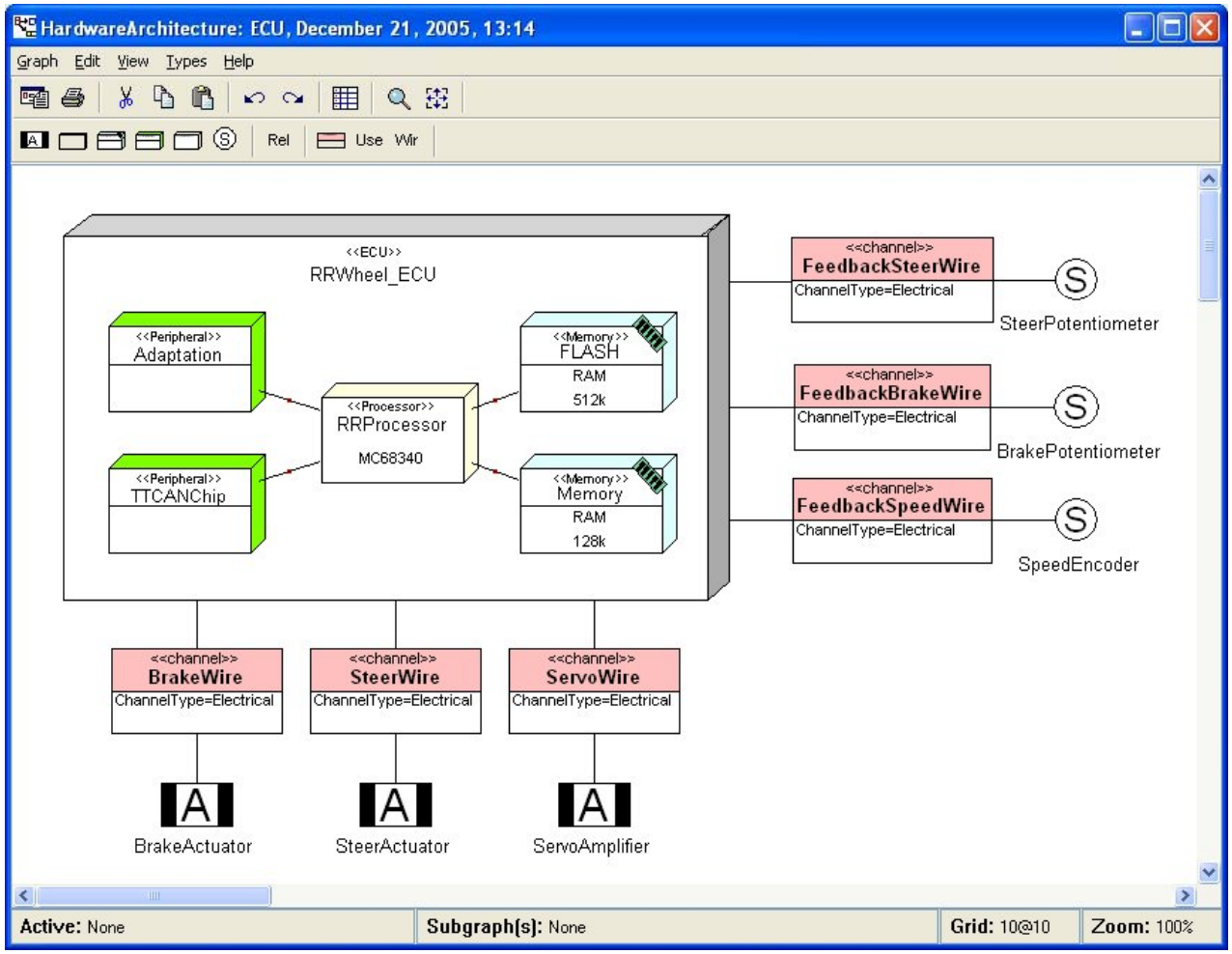
➔ Different languages support different things!

- ADLs have usually a particular focus, as in automotive related ADLs... (see examples coming)

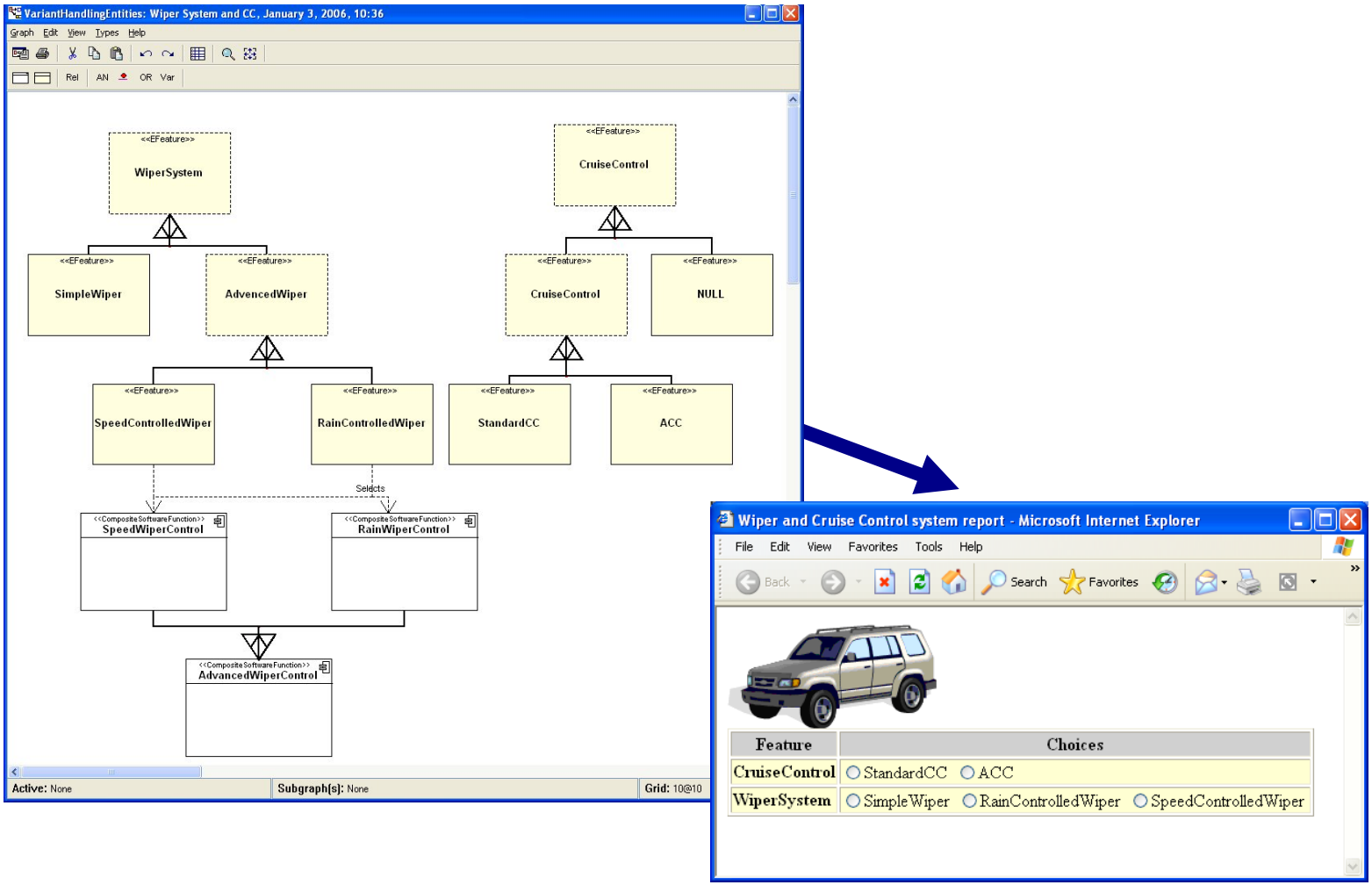
EAST-ADL: Functional Architecture



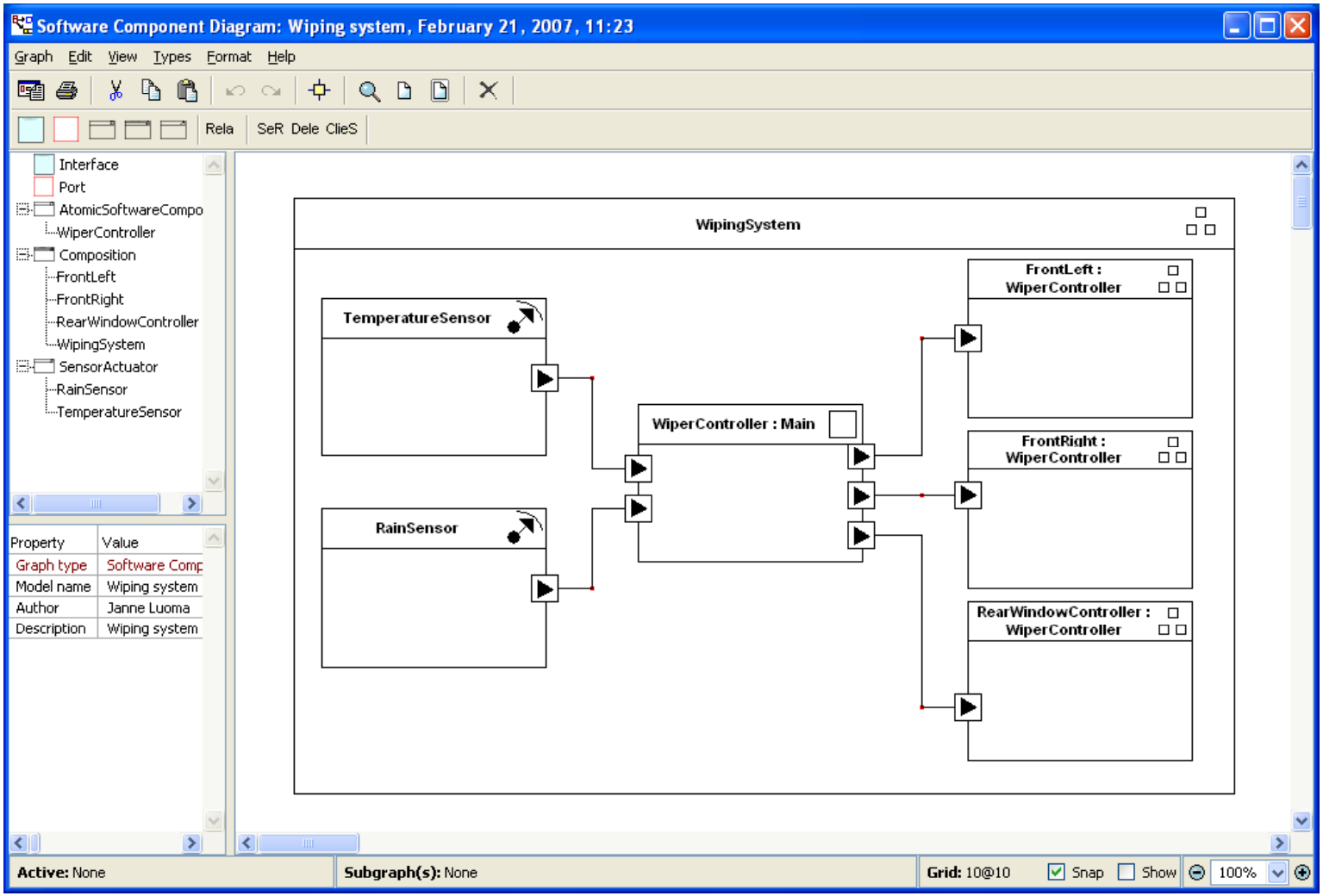
EAST-ADL: Hardware Architecture



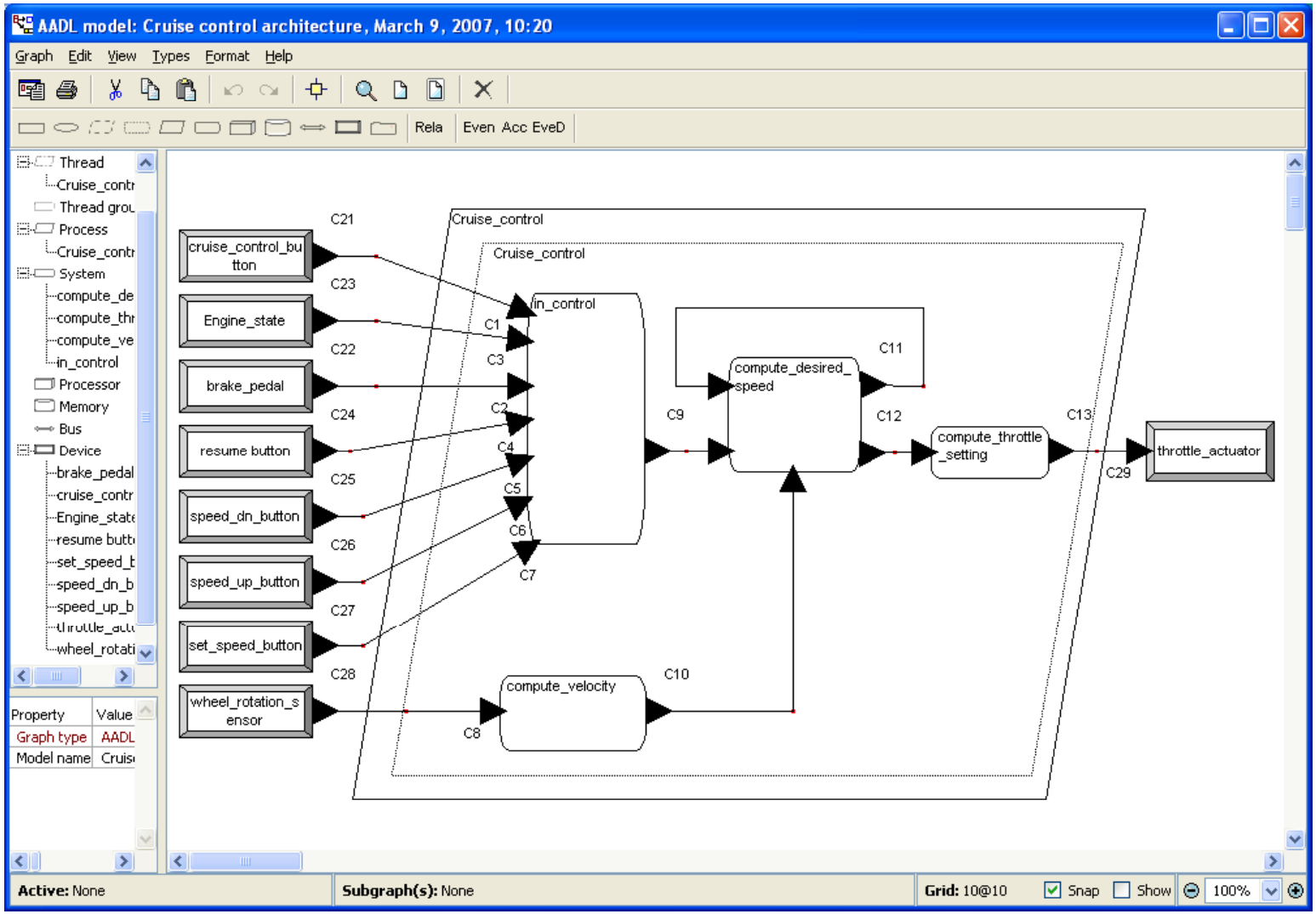
EAST-ADL: Variant Configuration



AUTOSAR: SW architecture

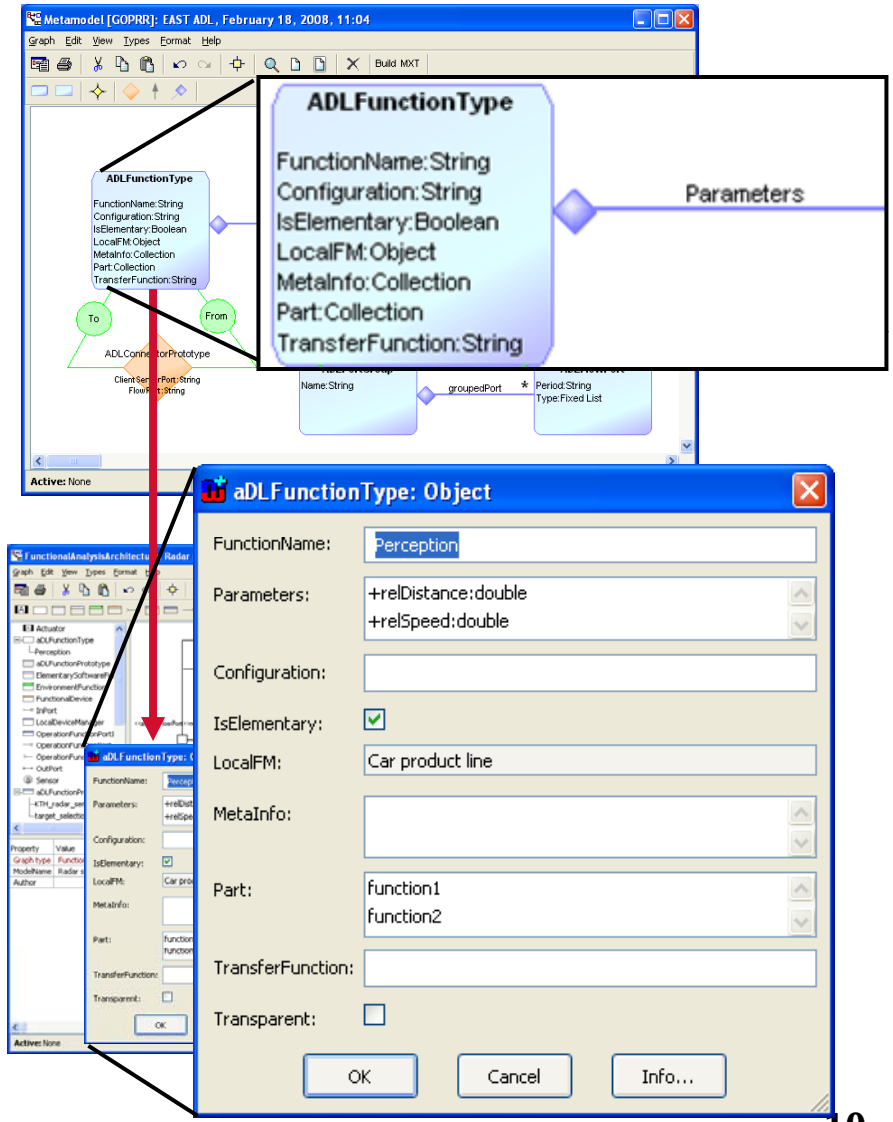


AADL: scheduling and flow control

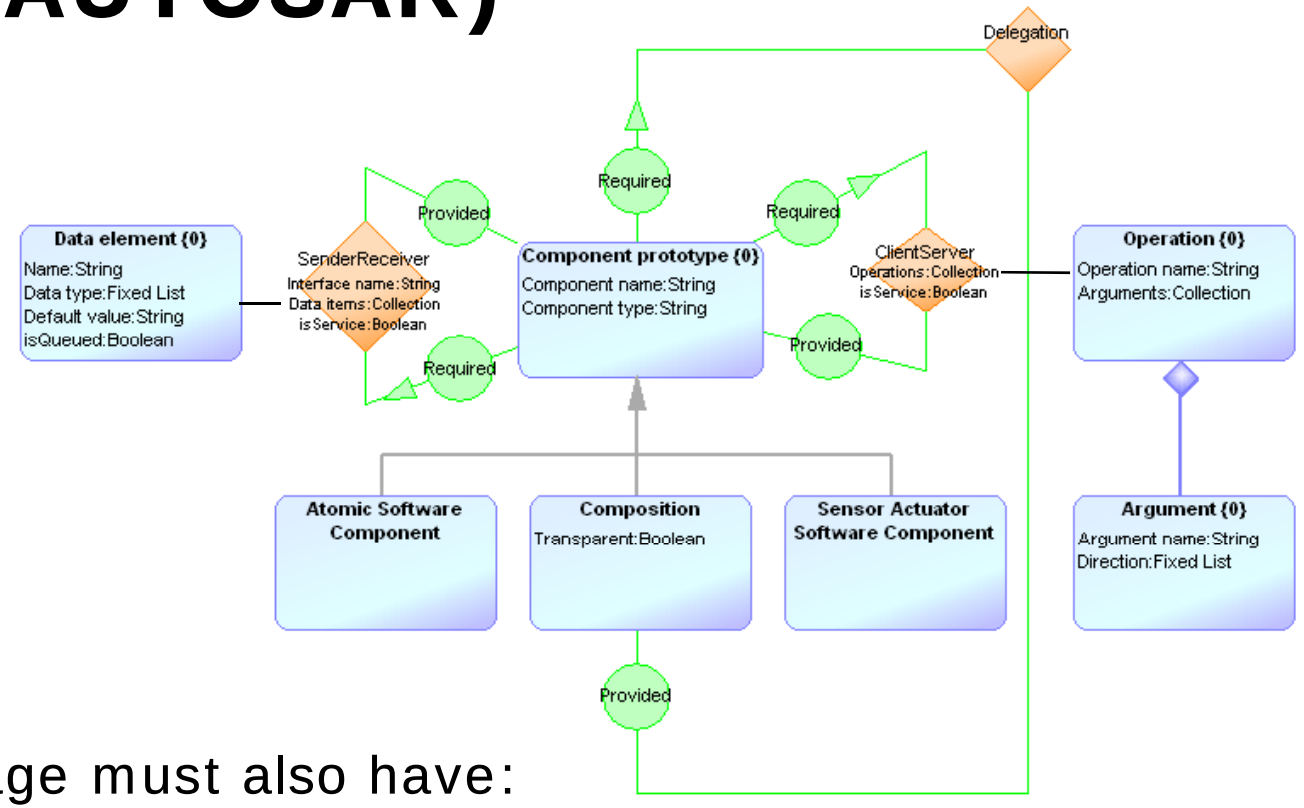


Metamodels define (AD)Languages

- Metamodeling (M2)
  - Specify domain concepts
  - Result: Metamodel
- Modeling (M1)
  - Instantiates the metamodel to specify systems and software
  - Result: Models (+ generated code etc)



Metamodel of SW component (from AUTOSAR)



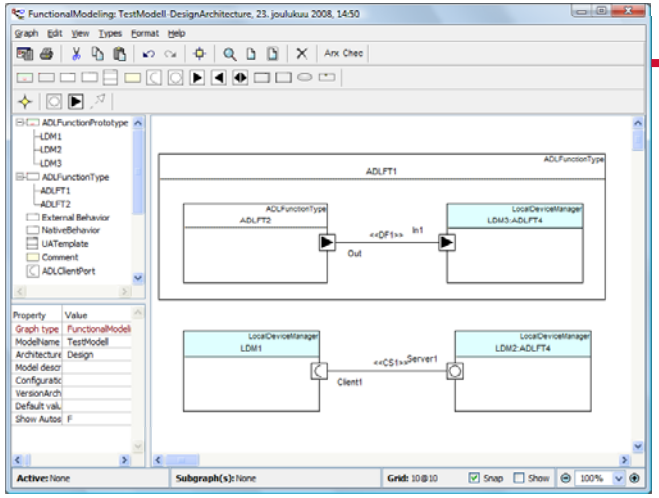
- A language must also have:
  - Notation, concrete syntax
  - Views (separation of concerns)
  - Rules and constraints (consistency, completeness, naming)
  - Semantics (operational semantics via generator)

Integrating models/views

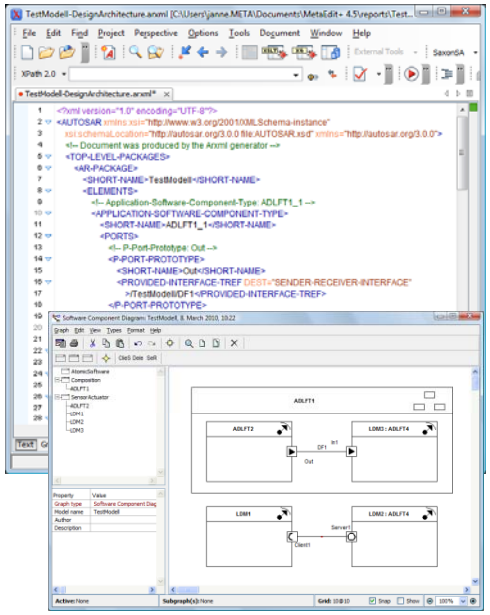
- Single language is usually not enough
  - Development processes must be integrated
    - Traceability and impact analysis
    - Generating code, metrics, test cases, documentation, etc.
- 2 main integration approaches:
  1. Transformation (separate metamodels)
     - Example: EAST-ADL2 models and AUTOSAR models
     - Challenges in integrating changes made in different models
     - Allows sharing models for different organizations/teams/ suppliers (information hiding)
  2. Common metamodel (e.g. EAST-ADL and AUTOSAR)
     - Example: integrating functional & network architecture
     - Supports collaboration, concurrent engineering
     - Partial models can be still extracted to different teams

1. Transformation example: EAST-ADL2 to AUTOSAR

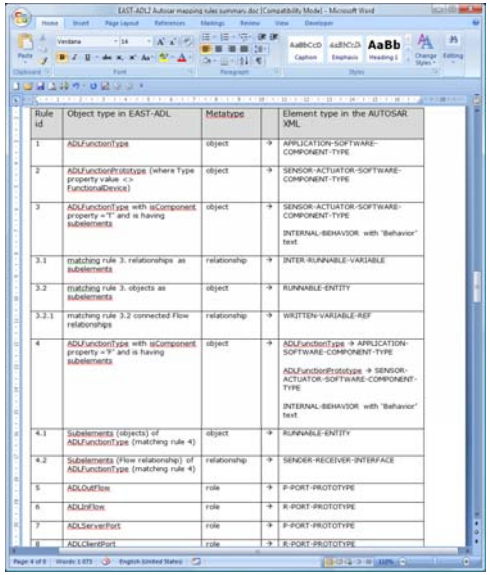
EAST-ADL2 model



Autosar



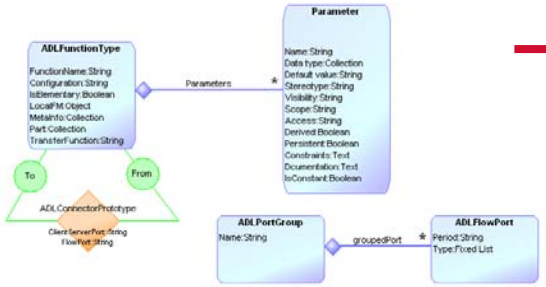
Mapping rules



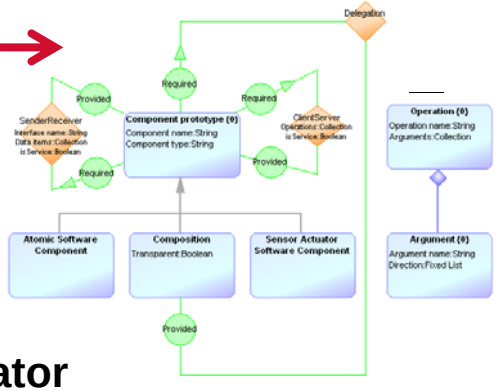
Rule id	Object type in EAST-ADL	Metatype		Element type in the AUTOSAR XML
1	ADLFunctionType	object	→	APPLICATION-SOFTWARE-COMPONENT-TYPE
2	ADLFunctionPrototype (where Type property value <> FunctionalDevice)	object	→	SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE
3	ADLFunctionType with isComponent property = "f" and is having subelements	object	→	SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE INTERNAL-BEHAVIOR with "Behavior" text
3.1	matching rule 3: relationships as subelements	relationship	→	INTER-RUNNABLE-VARIABLE
3.2	matching rule 3: objects as subelements	object	→	RUNNABLE-ENTITY
3.2.1	matching rule 3.2 connected Flow relationships	relationship	→	WRITTEN-VARIABLE-REF
4	ADLFunctionType with isComponent property = "f" and is having subelements	object	→	ADLFunctionType → APPLICATION-SOFTWARE-COMPONENT-TYPE ADLFunctionPrototype → SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE INTERNAL-BEHAVIOR with "Behavior" text
4.1	Subelements (objects) of ADLFunctionType (matching rule 4)	object	→	RUNNABLE-ENTITY
4.2	Subelements (Flow relationship) of ADLFunctionType (matching rule 4)	relationship	→	SENDER-RECEIVER-INTERFACE
5	ADLOutFlow	role	→	P-PORT-PROTOTYPE
6	ADLInFlow	role	→	R-PORT-PROTOTYPE
7	ADLServerPort	role	→	P-PORT-PROTOTYPE
8	ADLClientPort	role	→	R-PORT-PROTOTYPE

Transformations are defined based on the metamodels behind the languages

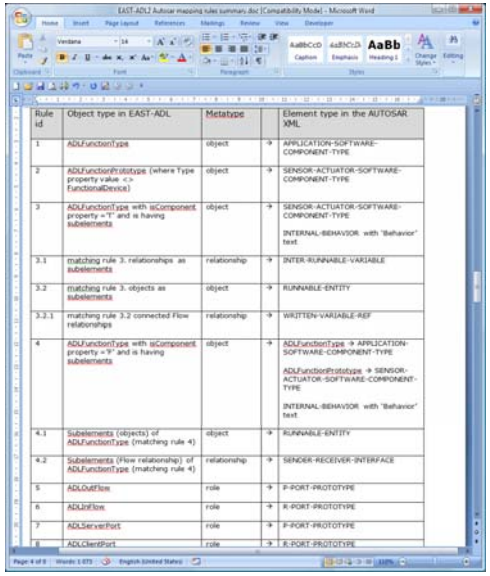
EAST-ADL2 metamodel



Autosar metamodel

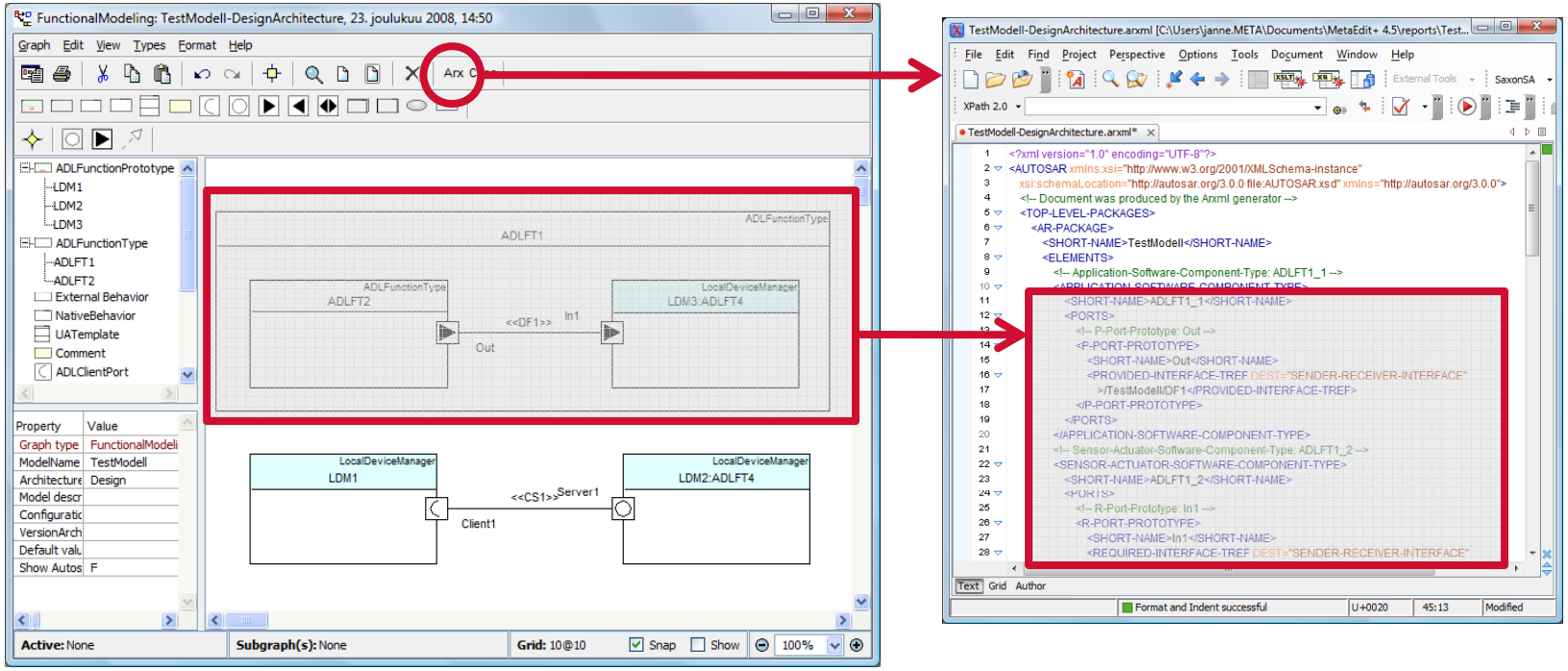


Mapping rules in generator



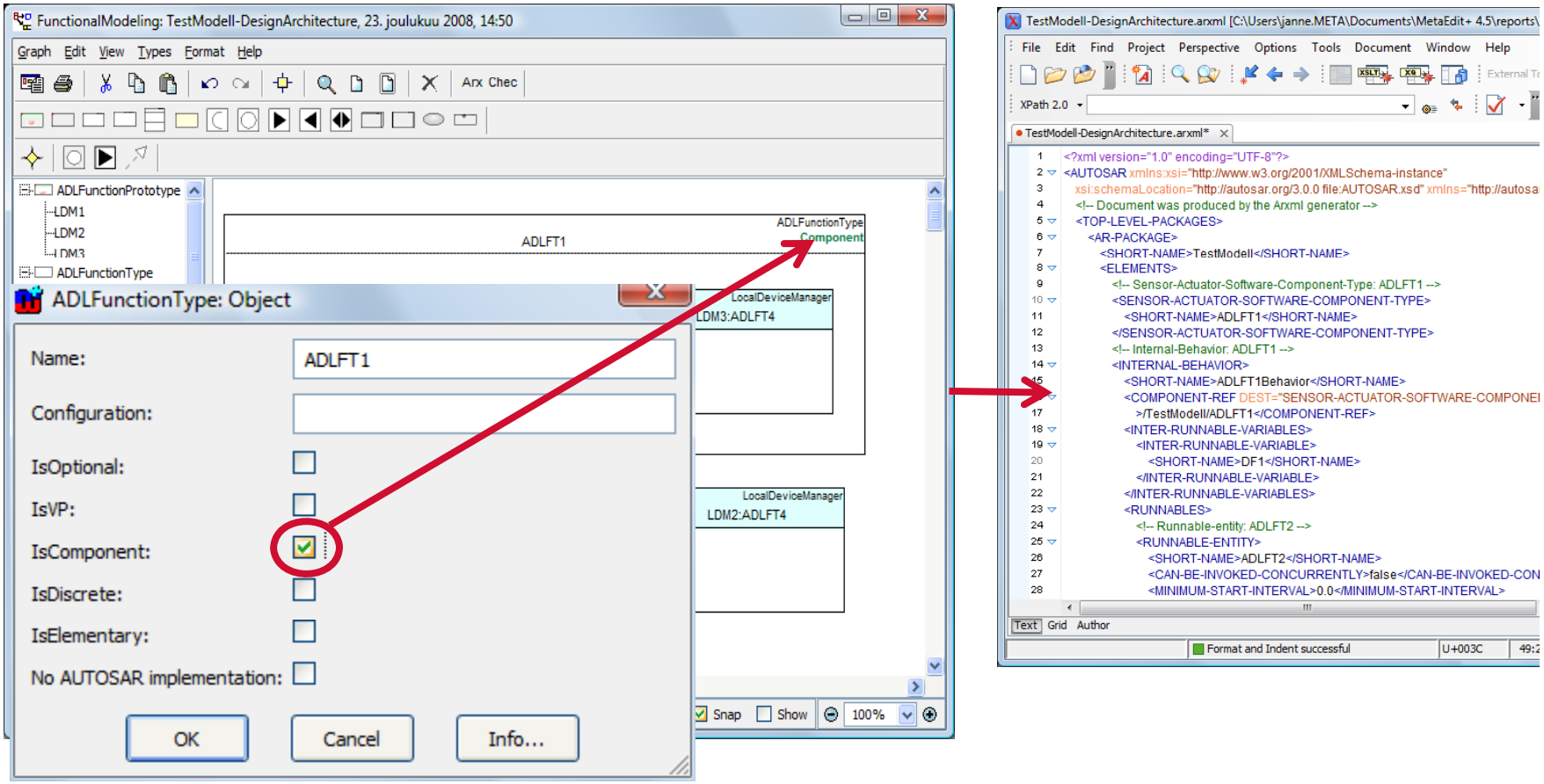
Rule id	Object type in EAST-ADL	Metatype		Element type in the AUTOSAR XML
1	ADLFunctionType	object	→	APPLICATION-SOFTWARE-COMPONENT-TYPE
2	ADLFunctionPrototype (where Type property value <> FunctionalDevice)	object	→	SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE
3	ADLFunctionType with isComponent property = "1" and is having subelements	object	→	SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE INTERNAL-BEHAVIOR with "Behavior" text
3.1	matching rule 3: relationships as subelements	relationship	→	INTER-RUNNABLE-VARIABLE
3.2	matching rule 3: objects as subelements	object	→	RUNNABLE-ENTITY
3.2.1	matching rule 3.2 connected Flow relationships	relationship	→	WRITTEN-VARIABLE-REF
4	ADLFunctionType with isComponent property = "0" and is having subelements	object	→	ADLFunctionType → APPLICATION-SOFTWARE-COMPONENT-TYPE ADLFunctionPrototype → SENSOR-ACTUATOR-SOFTWARE-COMPONENT-TYPE INTERNAL-BEHAVIOR with "Behavior" text
4.1	Subelements (objects) of ADLFunctionType (matching rule 4)	object	→	RUNNABLE-ENTITY
4.2	Subelements (Flow relationship) of ADLFunctionType (matching rule 4)	relationship	→	SENDER-RECEIVER-INTERFACE
5	ADLOutFlow	role	→	P-PORT-PROTOTYPE
6	ADLInFlow	role	→	R-PORT-PROTOTYPE
7	ADLServerPort	role	→	P-PORT-PROTOTYPE
8	ADLClientPort	role	→	R-PORT-PROTOTYPE

Mappings operate at metamodels



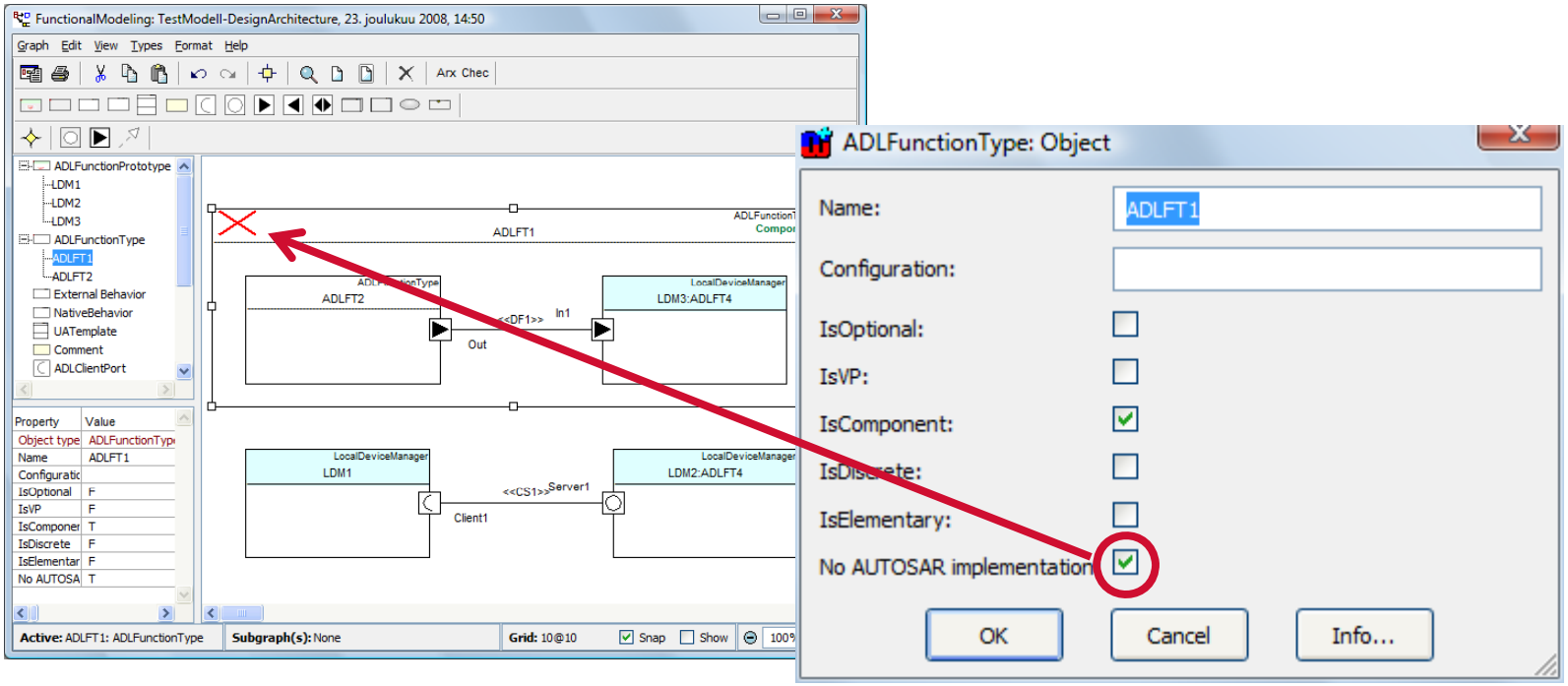
- Functional design architecture made using EAST-ADL2 is transformed into AUTOSAR
- Transformation is done by running the generator

Developer can influence how the mappings rules are applied



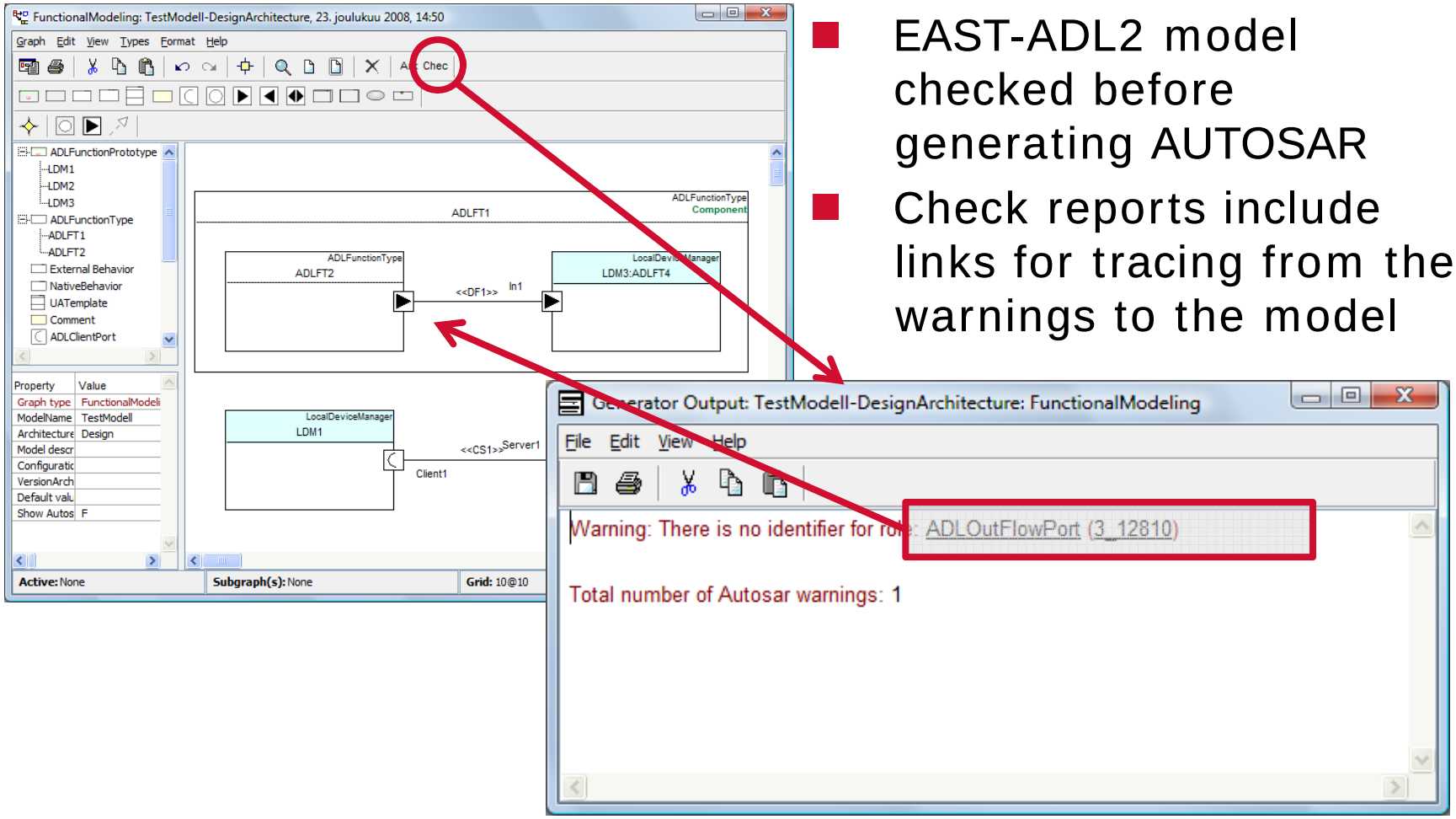
- Changes in model influence to generated mapping
  - Function is defined to be a component

AUTOSAR generation options



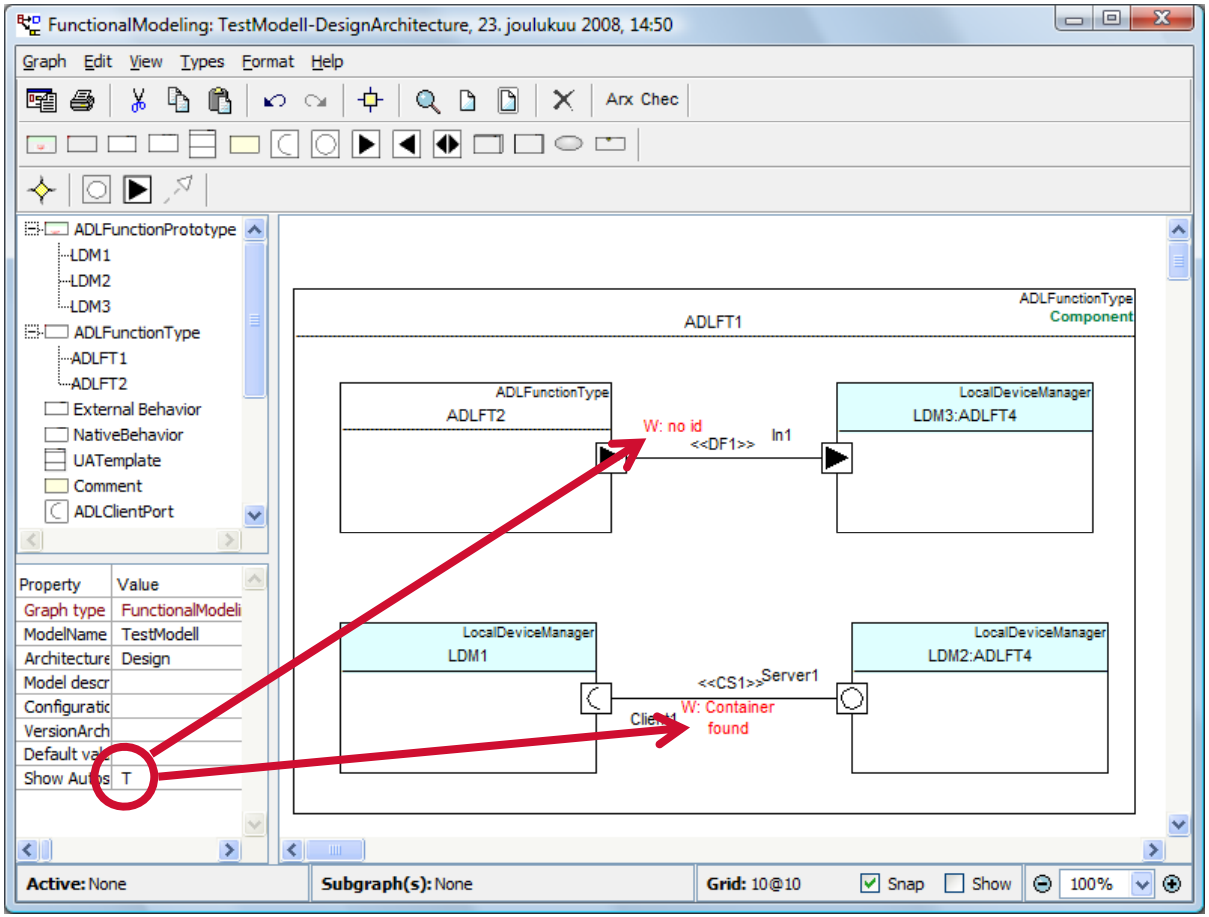
- When element is not selected for AUTOSAR implementation, a red X is shown in a symbol
  - ⇒ Element and its subelements are not transformed into AUTOSAR

EAST-ADL2 model checks



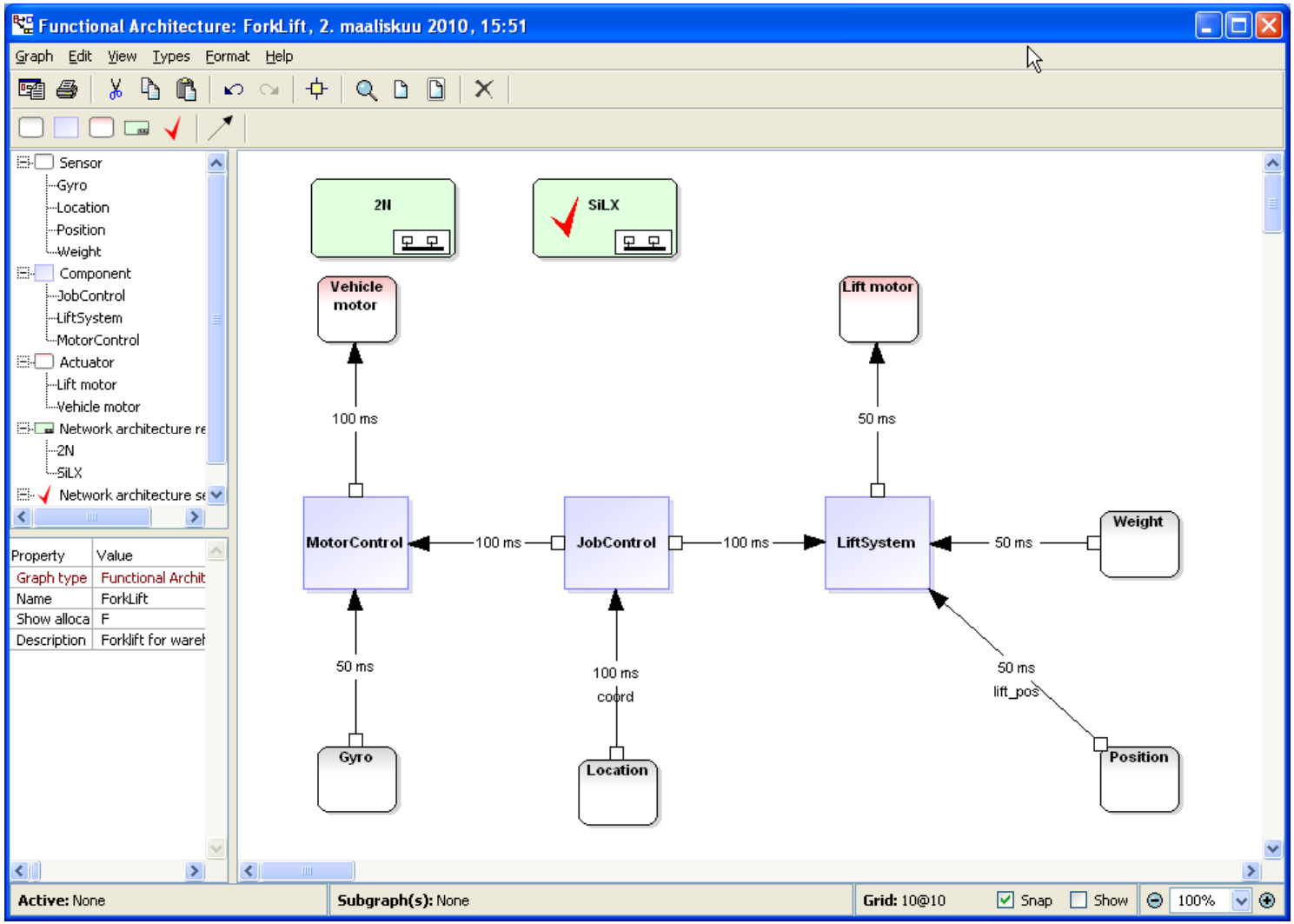
- EAST-ADL2 model checked before generating AUTOSAR
- Check reports include links for tracing from the warnings to the model

Real time model annotation



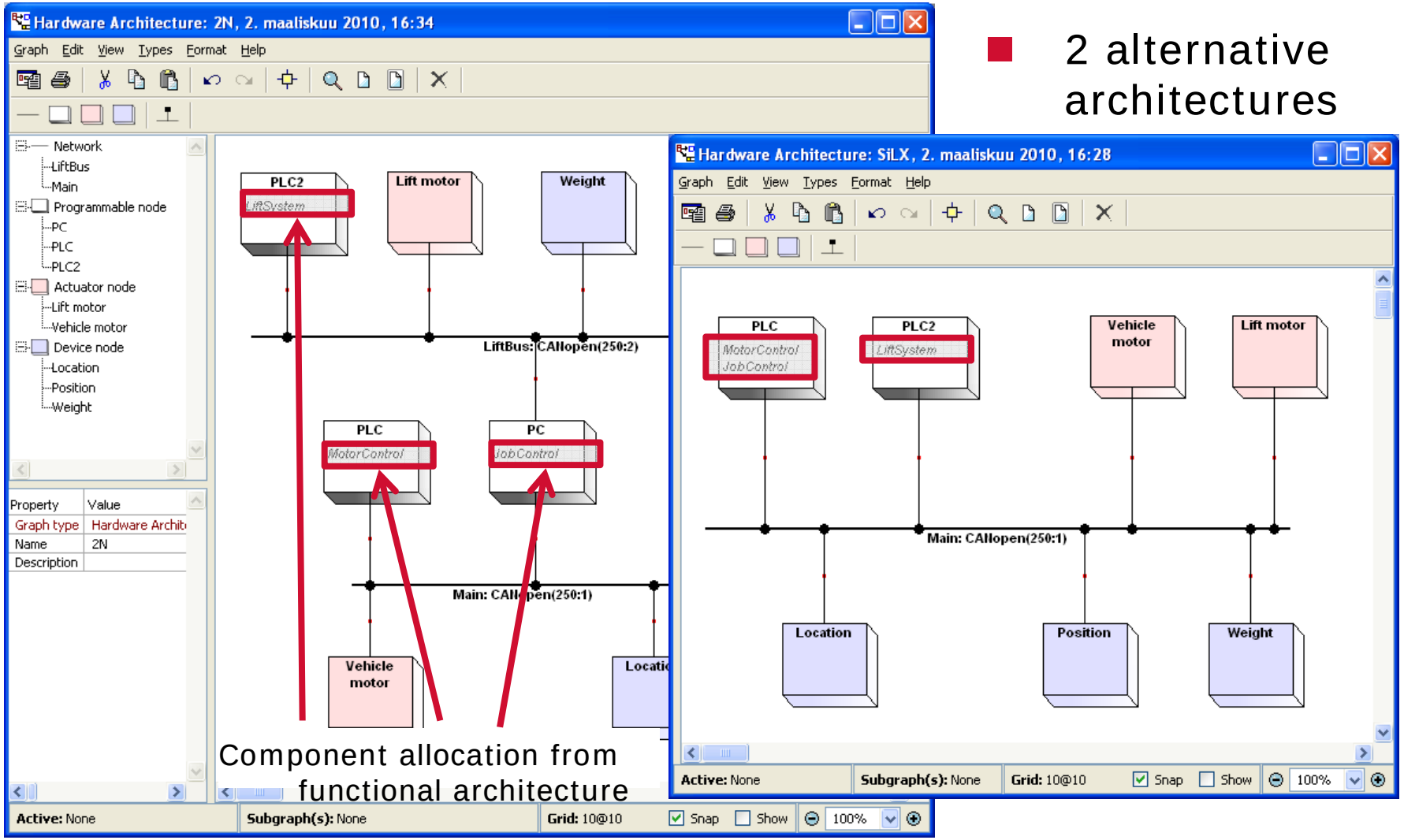
- Models can be annotated with error information
  - Graph property: Show Autosar warnings in EAST-ADL2

2. Integrated metamodel example: ADL for Functional Architecture...



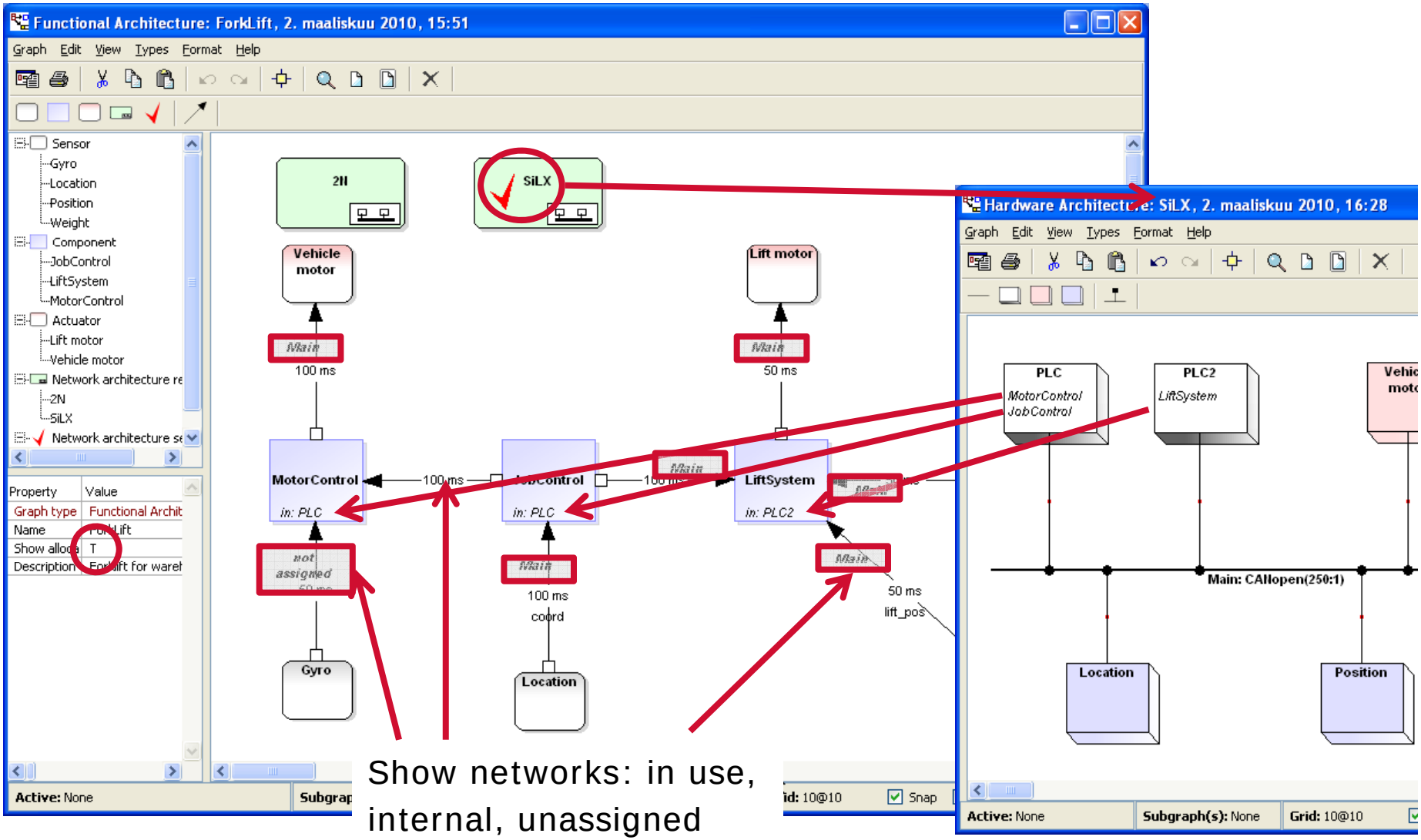
2. Integrated metamodel example: ... for HW / Network Architecture

- 2 alternative architectures



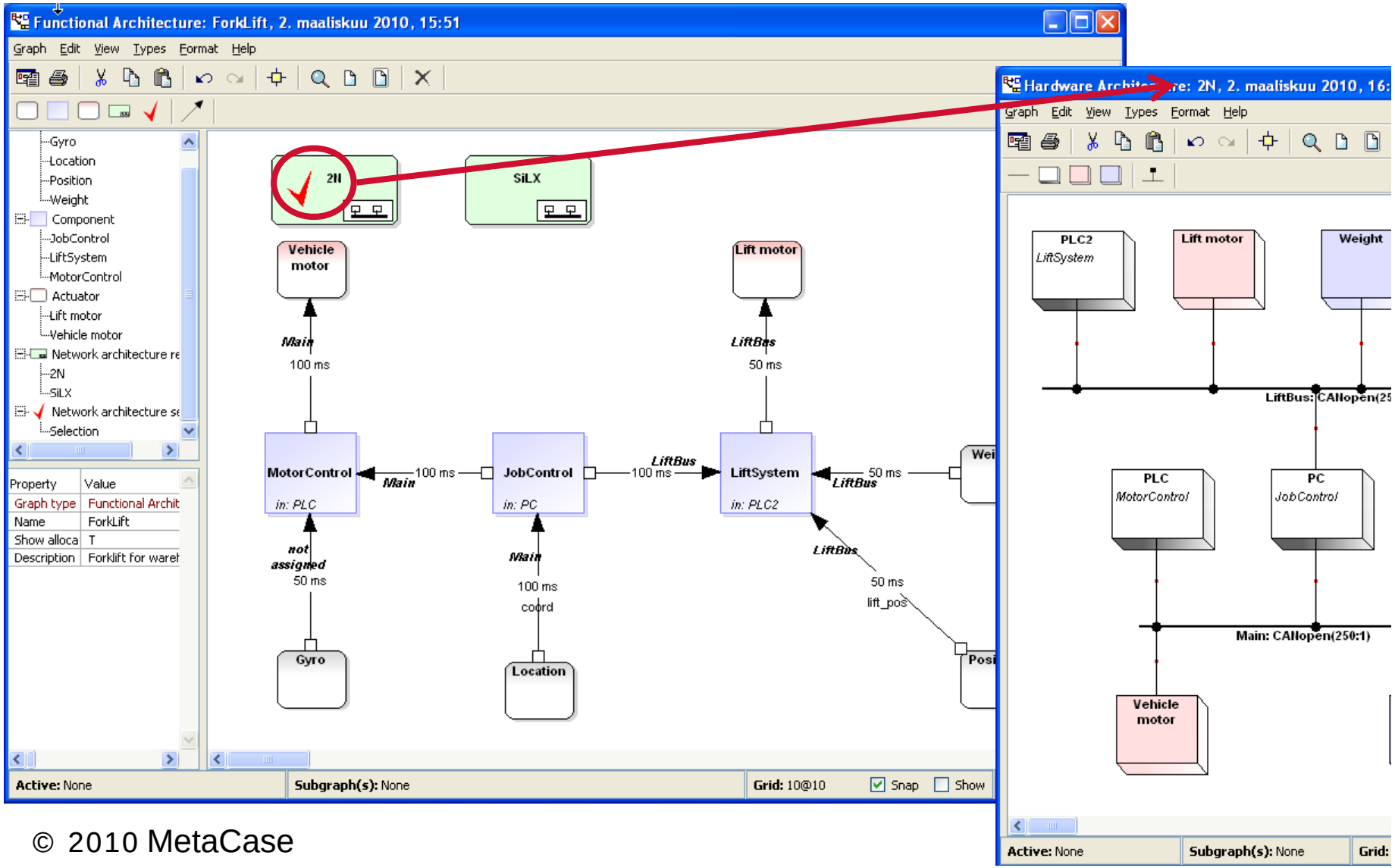
Component allocation from functional architecture

Back to functional architecture!



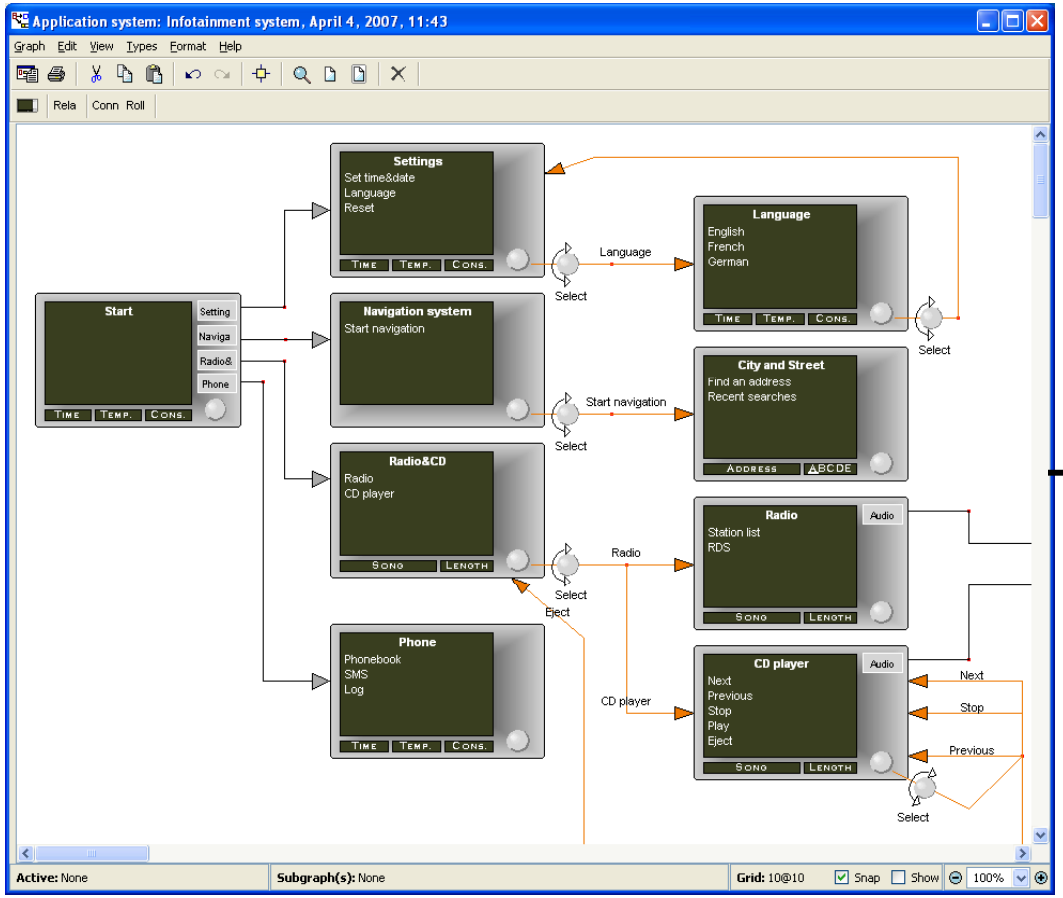
Show networks: in use, internal, unassigned

The other HW architecture...



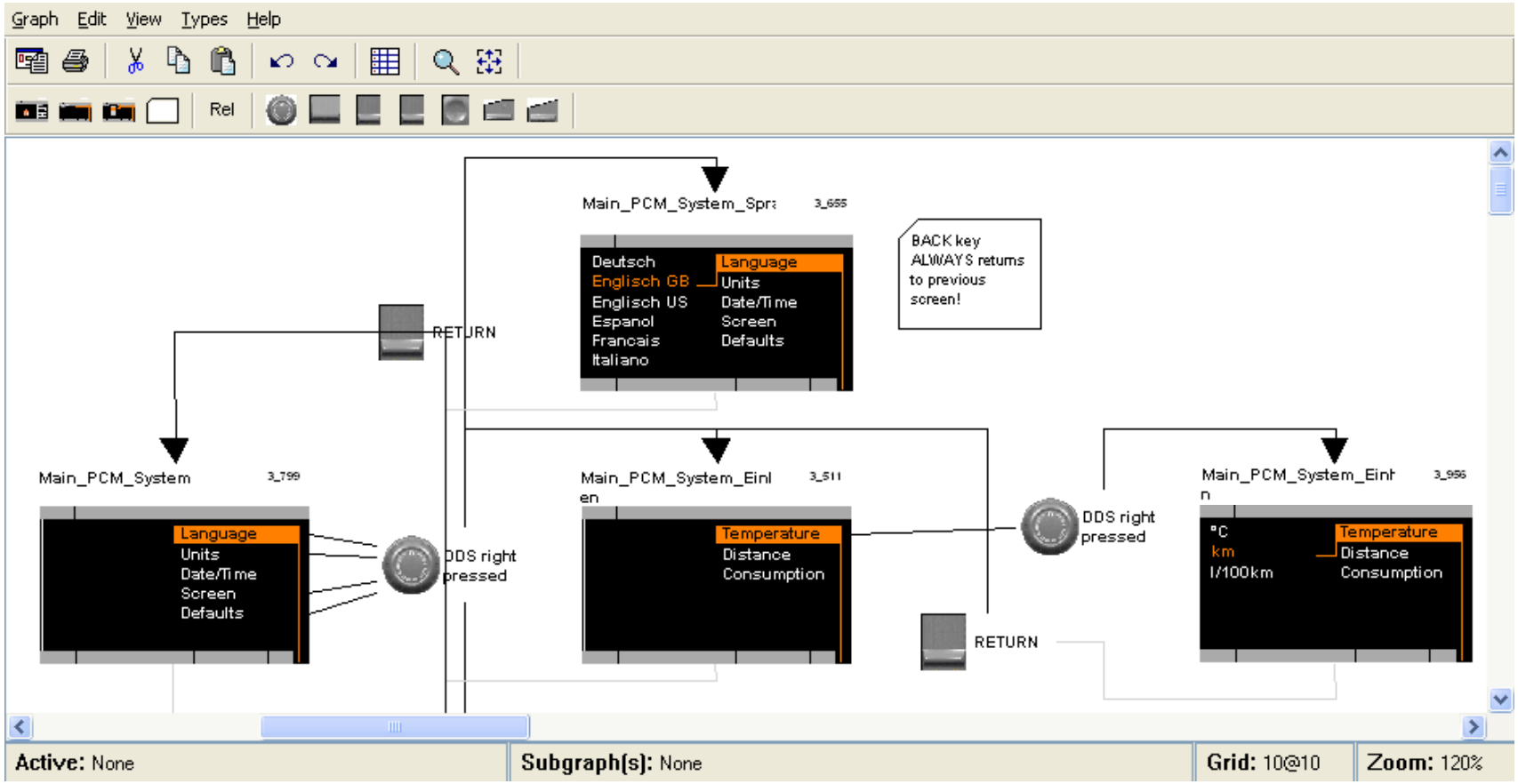
DSLs in other domains...

- Infotainment systems (1)



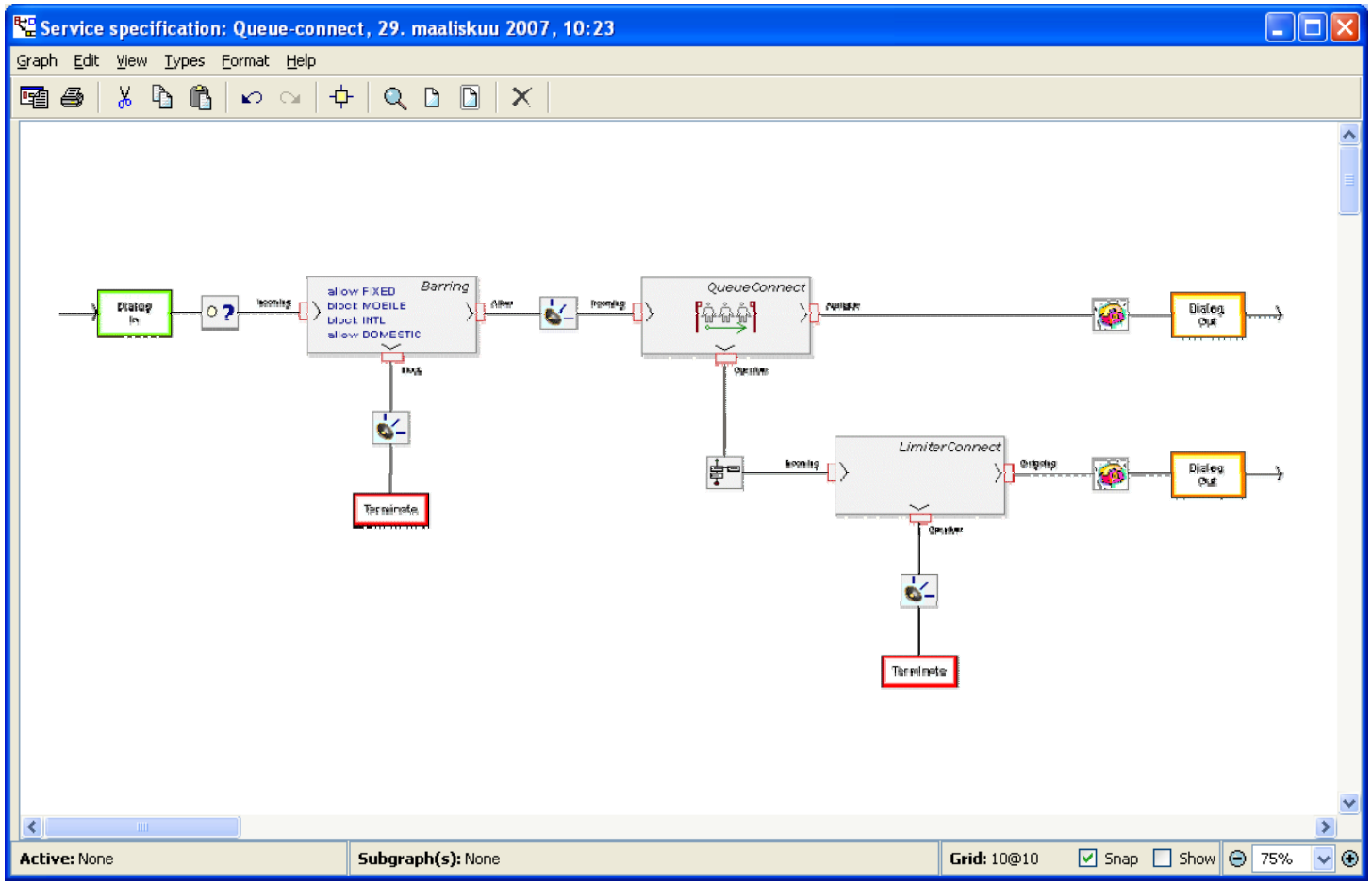
DSLs in other domains...

- Infotainment system (2)



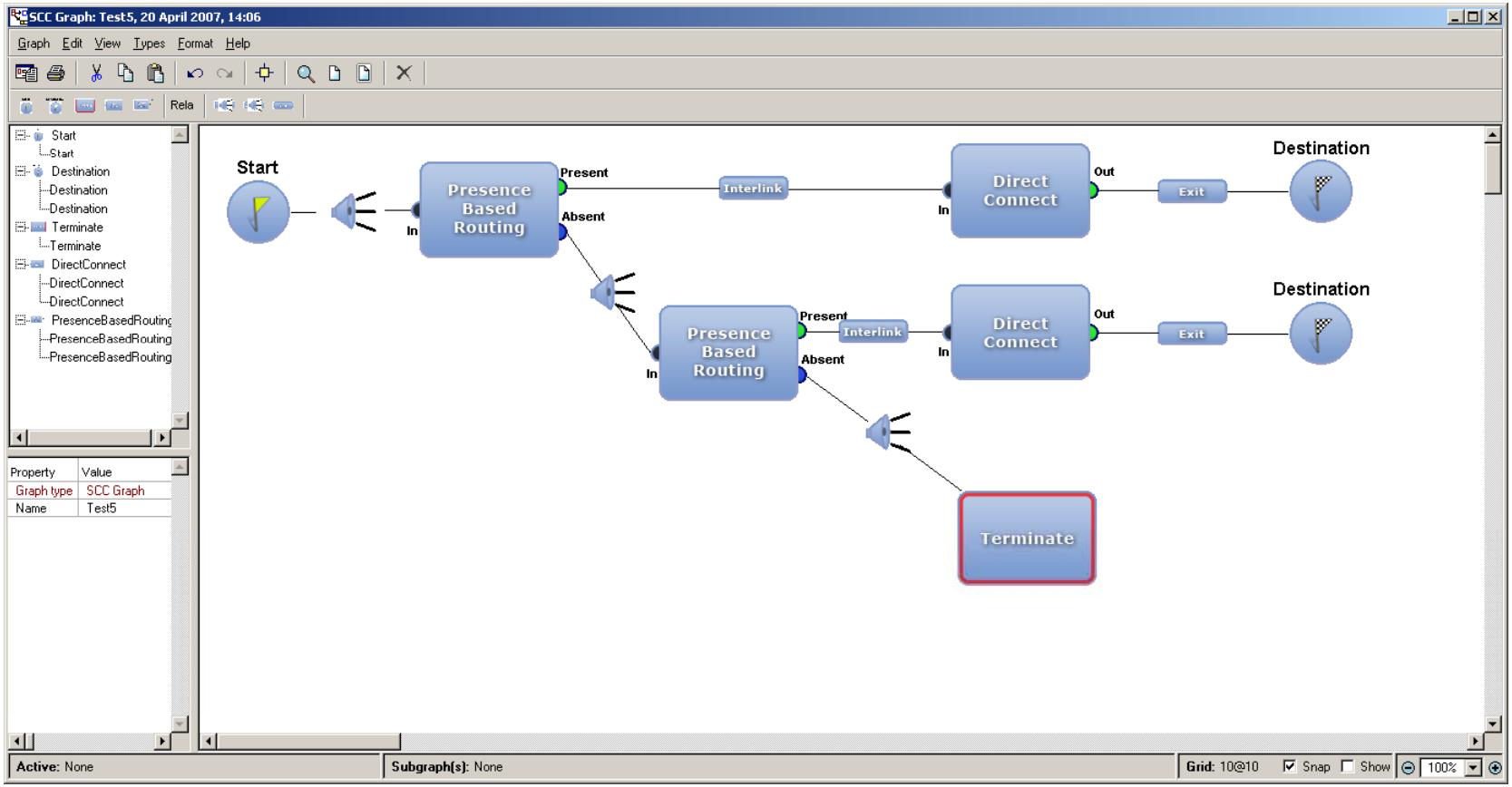
DSLs in other domains...

- Telecom services (1)



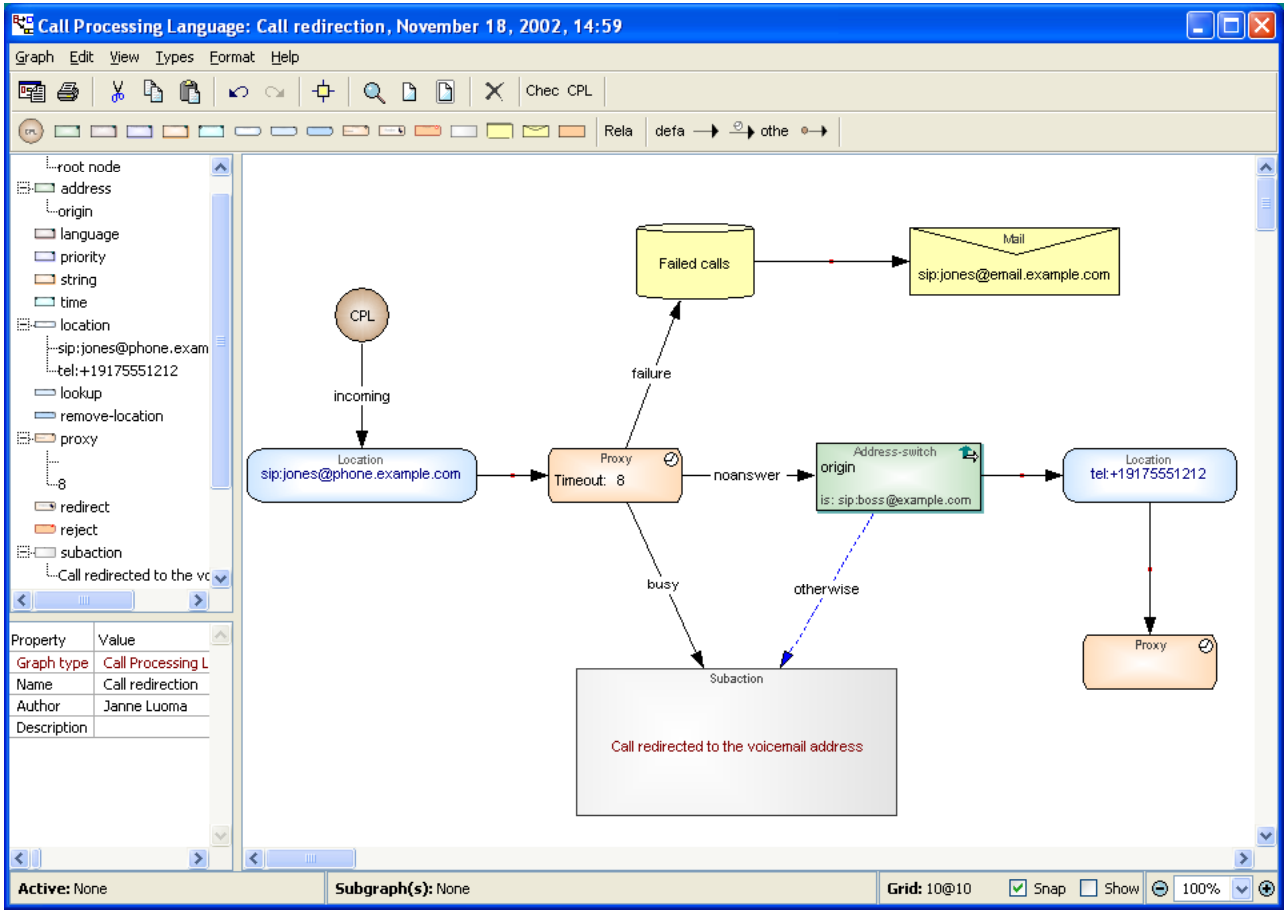
DSLs in other domains...

- Telecom services (2)



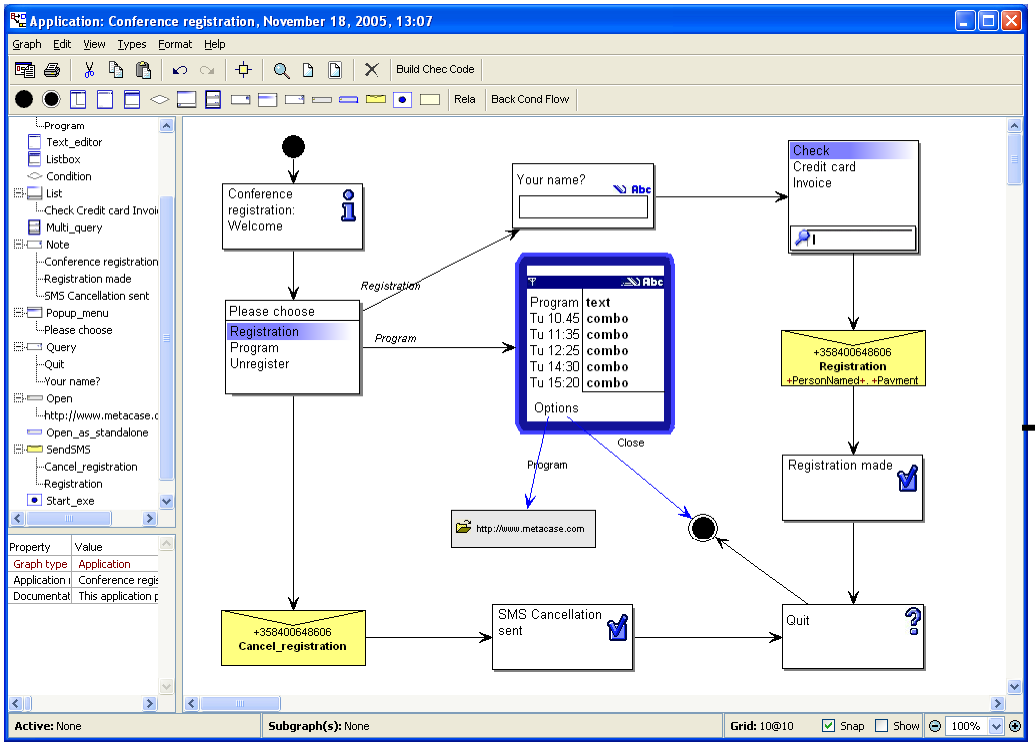
DSLs in other domains...

- Telecom services (3)



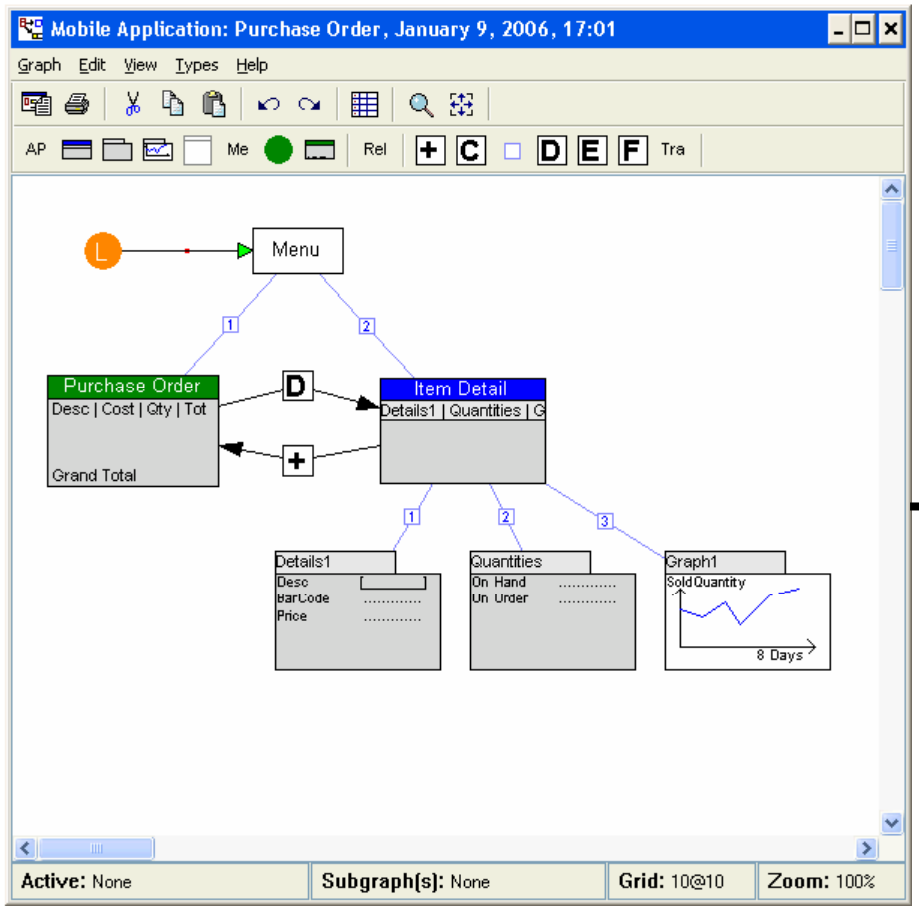
DSLs in other domains...

- Mobile phone apps (1)



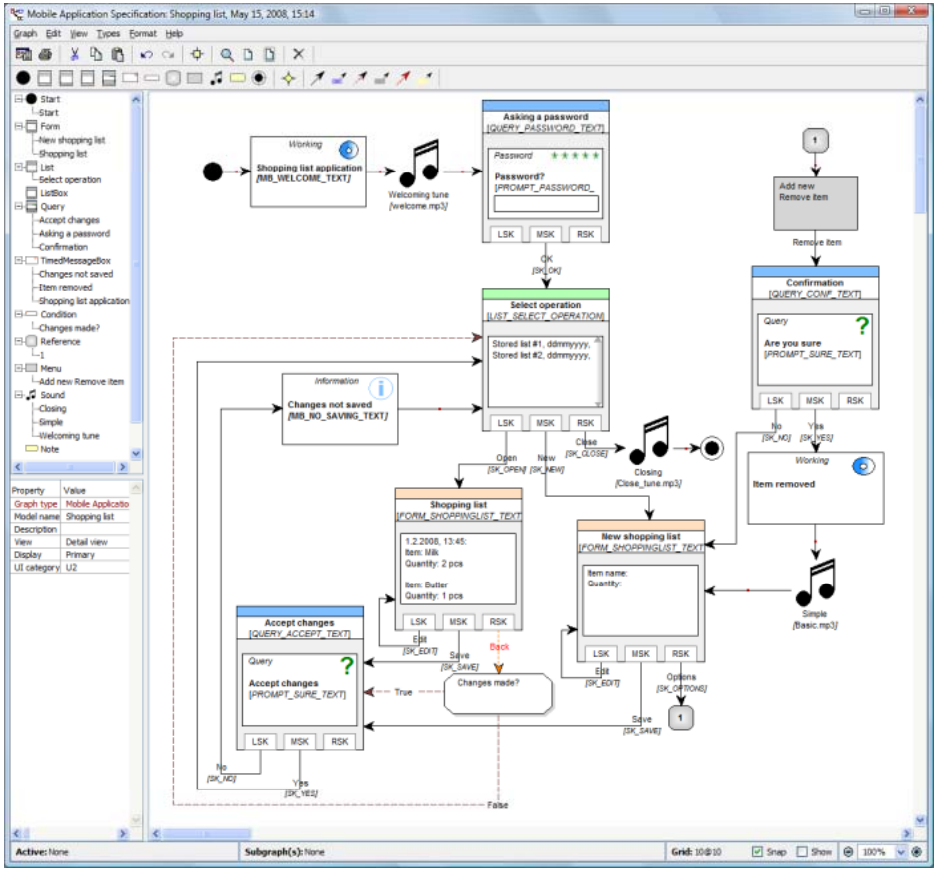
DSLs in other domains...

- Mobile phone apps (2)



DSLs in other domains...

- Mobile phone apps (3)



What industry says

Panasonic

"**5-fold** productivity increase when compared to standard development methods"

NOKIA
Connecting People

"A module that was expected to take 2 weeks now **took 1 day** from the start of the design to the finished product"

Nokia Siemens Networks

"Having the architectural rules embedded in the modeling language and code generators better **guarantees** that the rules are followed in practice **improving the quality** of the applications"

EADS

"The quality of the generated code is clearly better, simply because the modeling language **rules out errors**, eliminating them already in the design stage"

DENSO

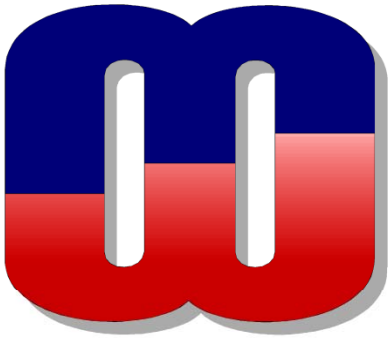
"It's now **easy to make modifications** when the AUTOSAR version is changed —implementation and testing requirements are both reduced"

Summary

- Different ADLs are needed for different purposes
- Tools for architecture description should be flexible
- Metamodel-based tools makes language definition cost effective
- Languages (and models) can be integrated
  - Via transformation
  - Via integrated metamodel
- MetaEdit+ is tried and proven technology
  - Supports different ADLs
  - Applied in different industries

Thank you!

Questions, please



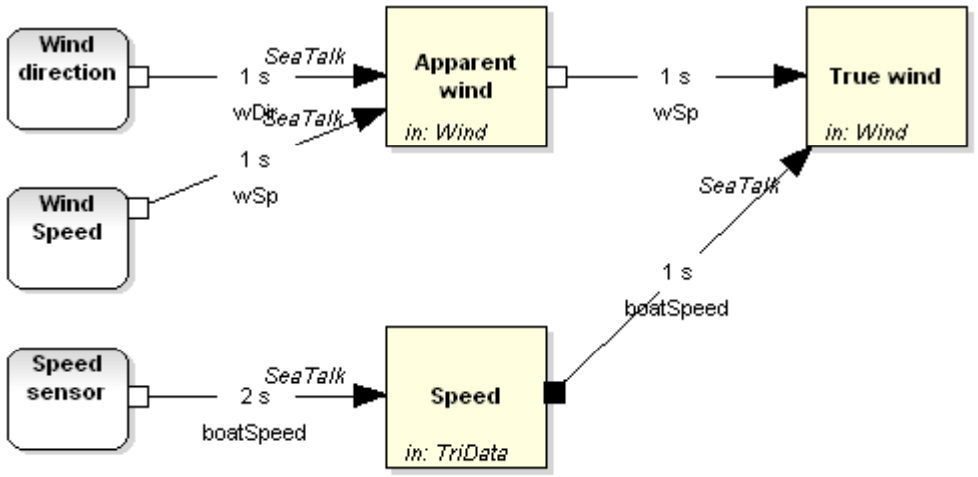
info@metacase.com
www.metacase.com

USA:
MetaCase
5605 North MacArthur Blvd.
11th Floor, Irving, Texas 75038
Phone (972) 819-2039
Fax (480) 247-5501

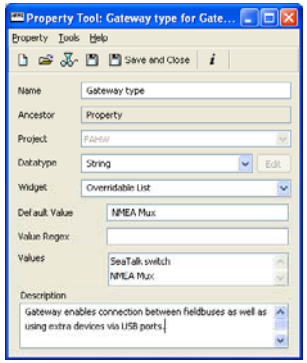
Europe:
MetaCase
Ylistönmäentie 31
FI-40500 Jyväskylä, Finland
Phone +358 14 641 000
Fax +358 420 648 606

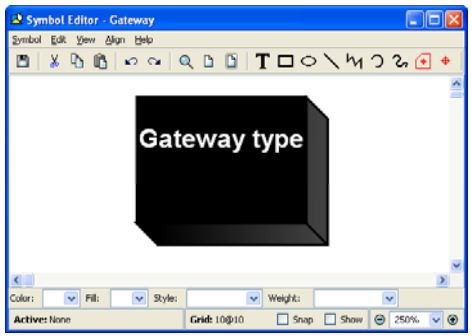
Plan for afternoon

1. Create architecture descriptions (modeling)



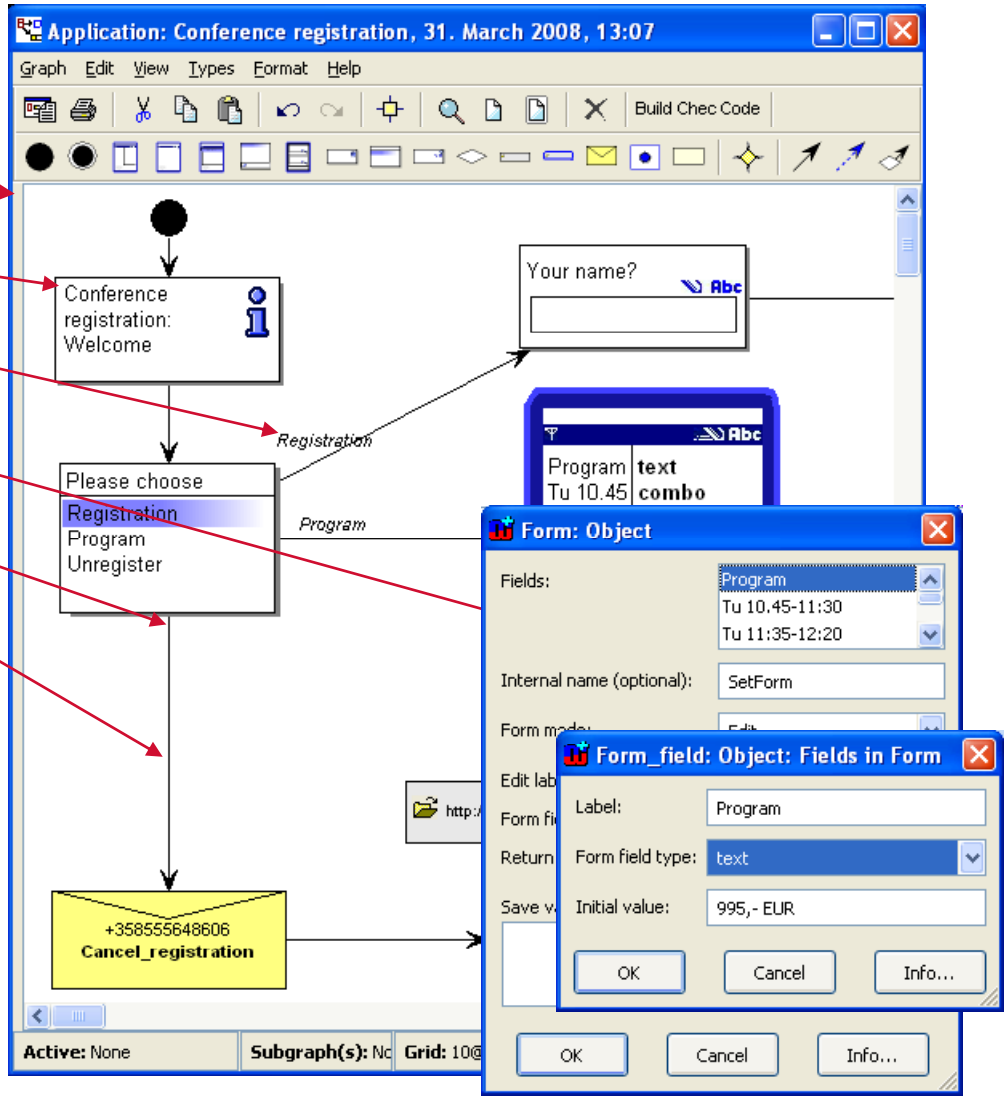
2. Create *your* ADL (metamodeling)





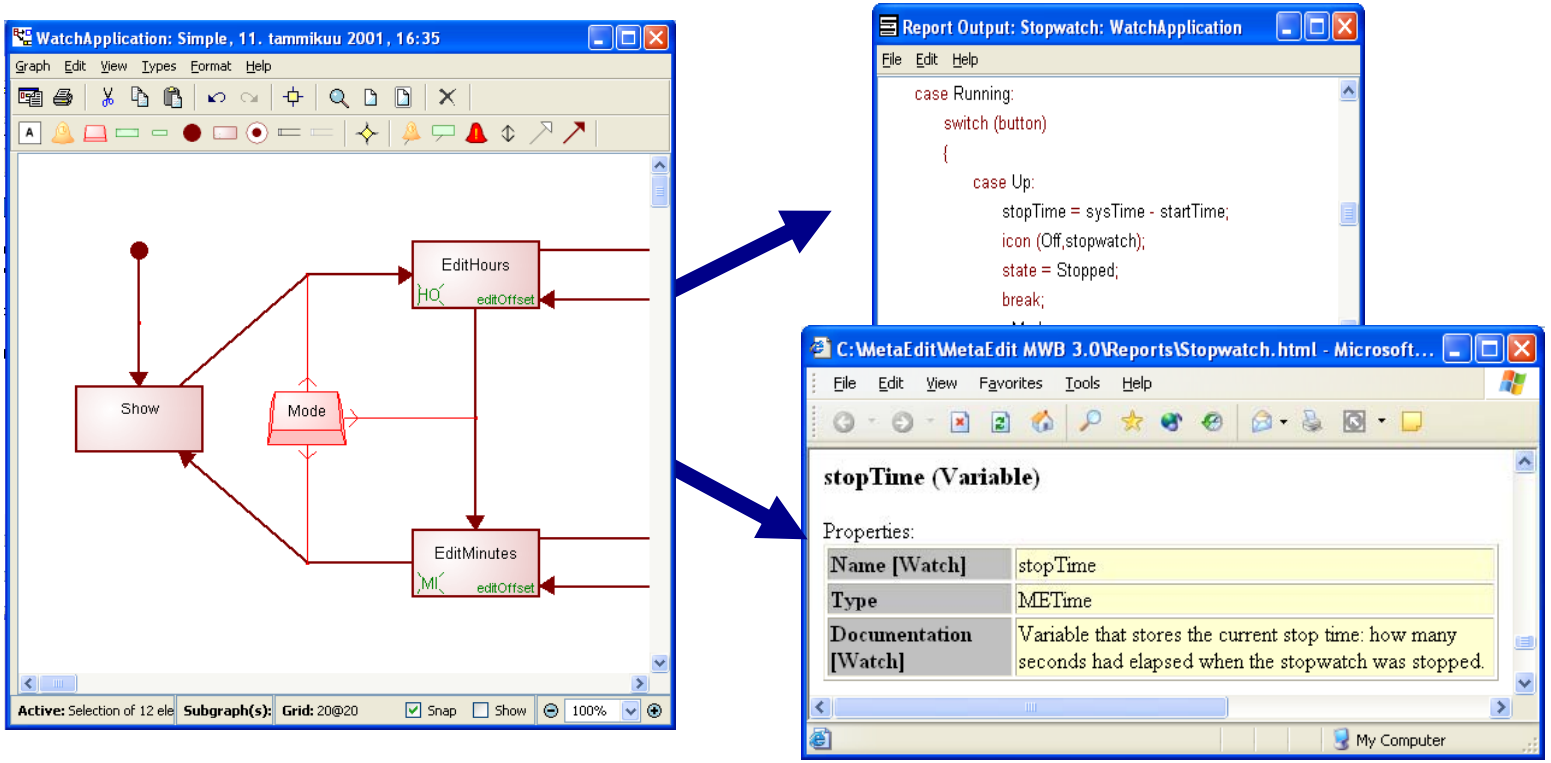
Metamodeling concepts

- Concepts
  - **G**raph
  - **O**bject
  - **P**roperty
  - **P**ort
  - **R**ole
  - **R**elationship
- Rules, e.g.
  - Max connections
  - Occurrence
  - Unique, etc
- Links between models
- Reuse rules

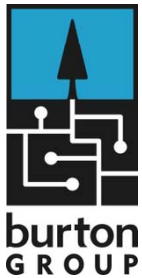


MetaEdit+ delivers the full toolset

- Editors (diagram, matrix, table), browsers, generators, multi-user, multi-project, multi-platform environment, …
- Modeling language maintenance and distribution safe and easy
  - New versions update existing models automatically, non-destructively



What analysts say on Domain-Specific Modeling languages



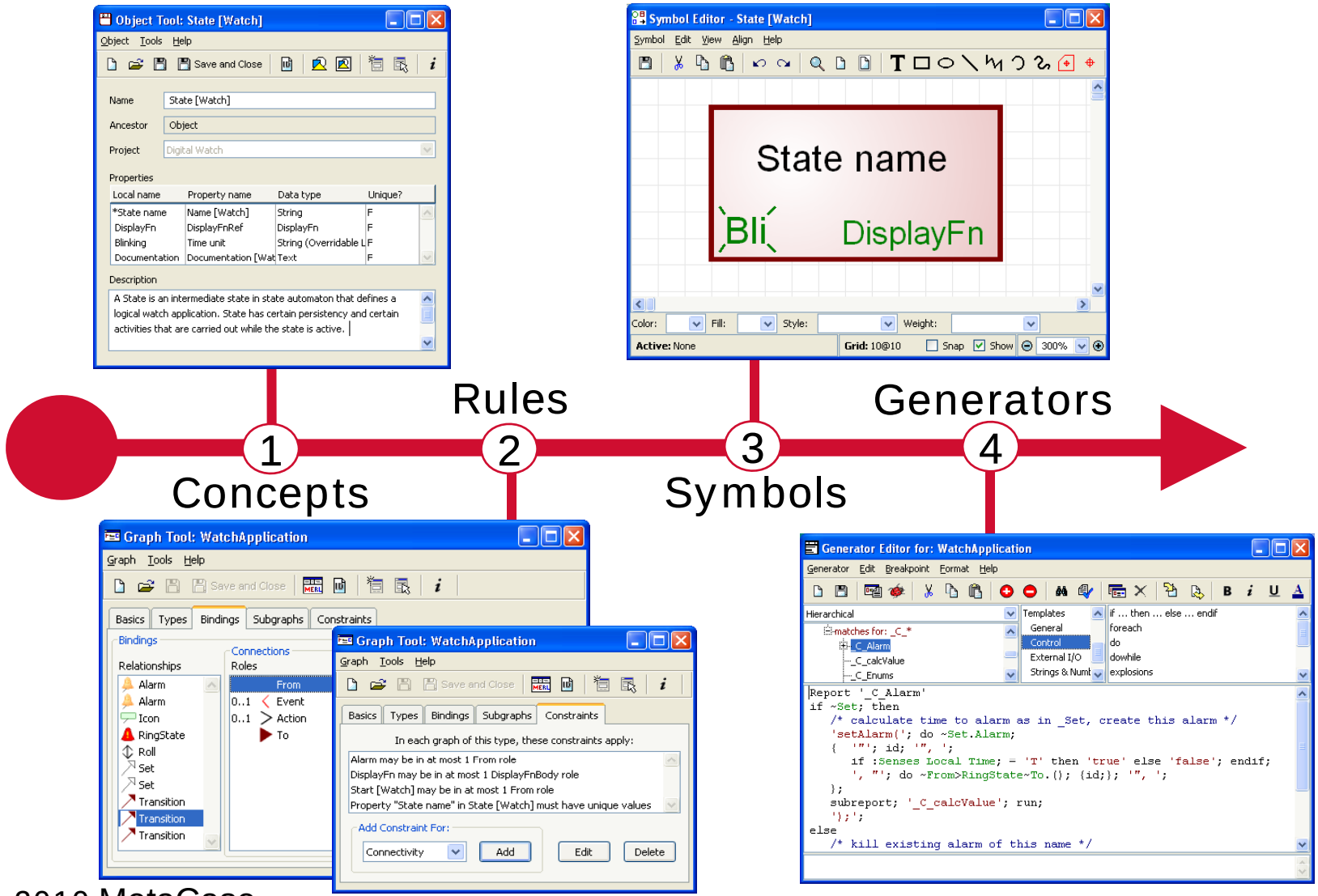
"The use of domain-specific languages and custom meta models is the **greatest aid to productivity** and making model-driven development a viable practice"

"MetaCase's approach **makes language and generator building easy**"

"Butler is **impressed with the ability of MetaEdit+** to create new languages, and test them immediately during development"

"Designers focusing on **higher-level** abstractions that are specific to the domain are **more productive** than with a general-purpose modeling language"

MetaEdit+ for language definition



Agenda

- Architectures are always domain-specific
  - No single language is enough!
- Metamodels define every (AD)Language
- Tools must to be flexible supporting your needs, e.g.
  - EAST-ADL
  - AUTOSAR
  - AADL
  - + modified, mixed and own
- Architecture models can be integrated
  - Transformation
  - Integrated metamodels
- Summary