



# Languages for modeling embedded system architectures

アーキテクチャ記述言語に活用される MetaEdit+  
EAST-ADL、AUTOSAR、SysML など

*September 2010*

Juha-Pekka Tolvanen, Ph.D.



MetaCase



# ADL に活用される MetaEdit+ の優位性

- あらゆるモデリング言語(ADL)を作ることが出来る
  - EAST-ADL、AUTOSAR、SysML など
- マルチ言語対応(ADLモデル間の連携)
  - 単一メタモデル上に複数のADLを統合
- ジェネレータ(ADLモデル間の連携)
  - モデル変換による統合
- ADL、ADL間で厳密なルール、コンストレインツによる形式化
  - モデル、モデル間の整合性チェック
- モデリング言語、モデルへの変更・修正を自動更新できる
  - 既存モデルが破壊されない、変更のインパクトを可視化

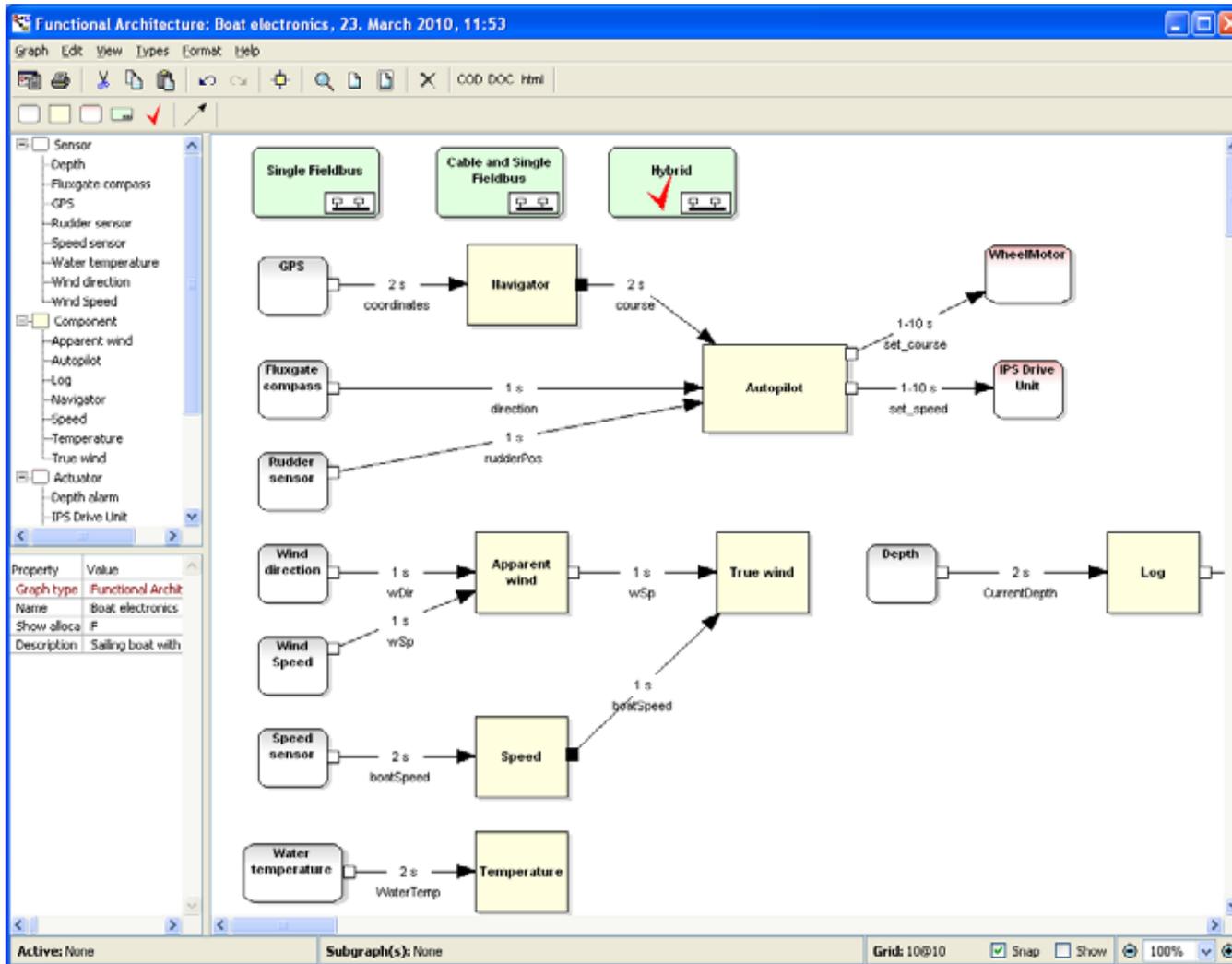


# Contents

- 個々のアーキテクチャは、それぞれドメインスペシフィック
  - 単一言語では間に合わない!
- ツールは柔軟にあらゆるドメイン、需要を満たすこと
  - SysML
  - EAST-ADL
  - AUTOSAR
  - AADL
  - + 修正・追加、統合・連携、独自拡張
- メタモデルで全てのADLを定義
- アーキテクチャモデルは統合・連携できる
  - 変換
  - メタモデルレベルの統合
- まとめ



# Sample ADL: a demo



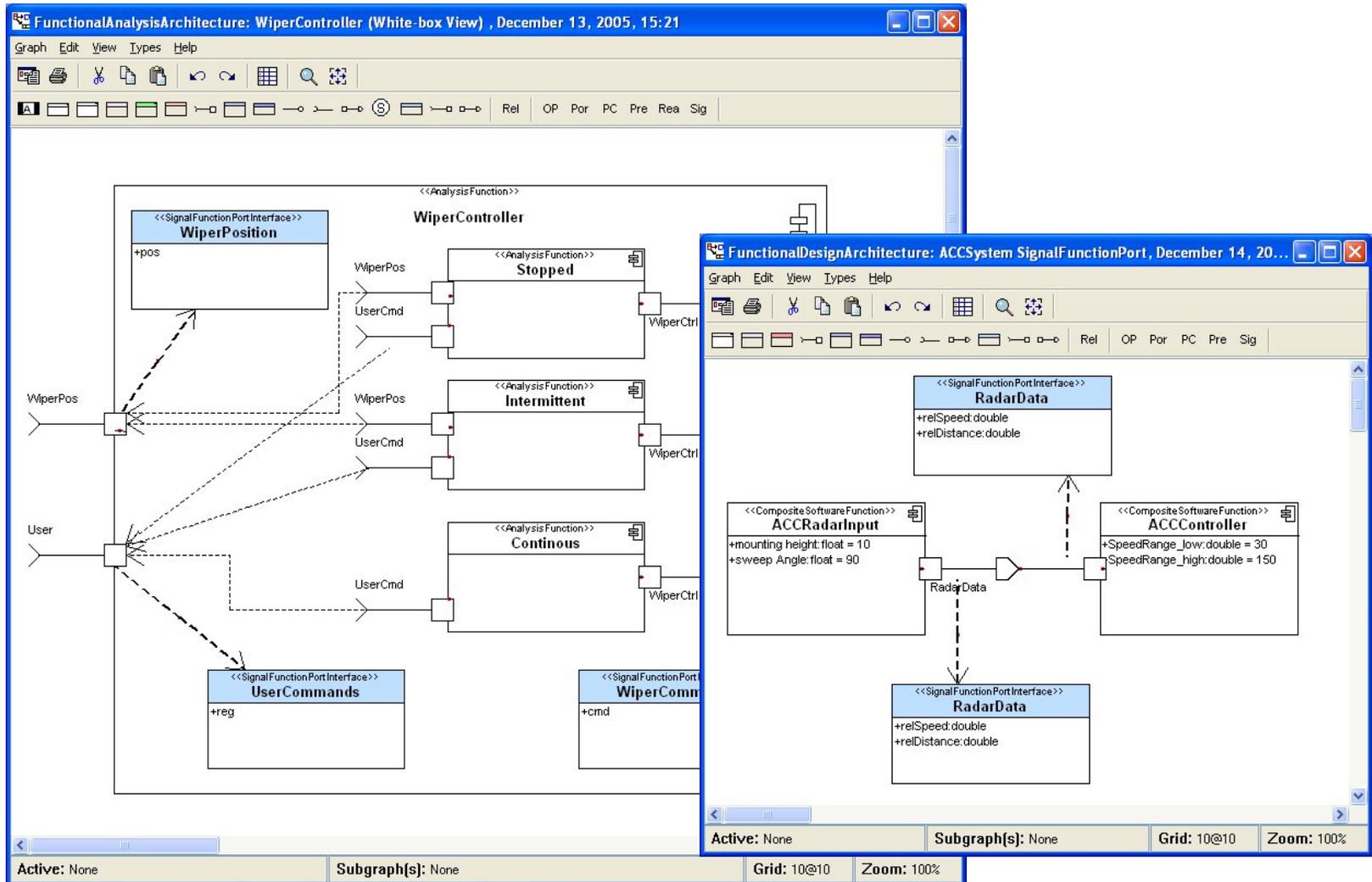


# What you want to do with ADL?

- アーキテクチャは汎用ではなく、特定の問題解決のためにデザインされる
- ADLで仕様化したいこと
  - コンポーネントの構造
  - インターフェイス
  - 通信プロトコル
  - SWとHWアーキテクチャのマッピング
  - コンフィグレーション
  - アーキテクチャの分析
  - アプリケーション構成のためのガイドライン
- ➔ 異なるADL言語ごとにそれぞれの支援対象がある！
- 一般にADLは特定用途のもの 自動車のADLなど

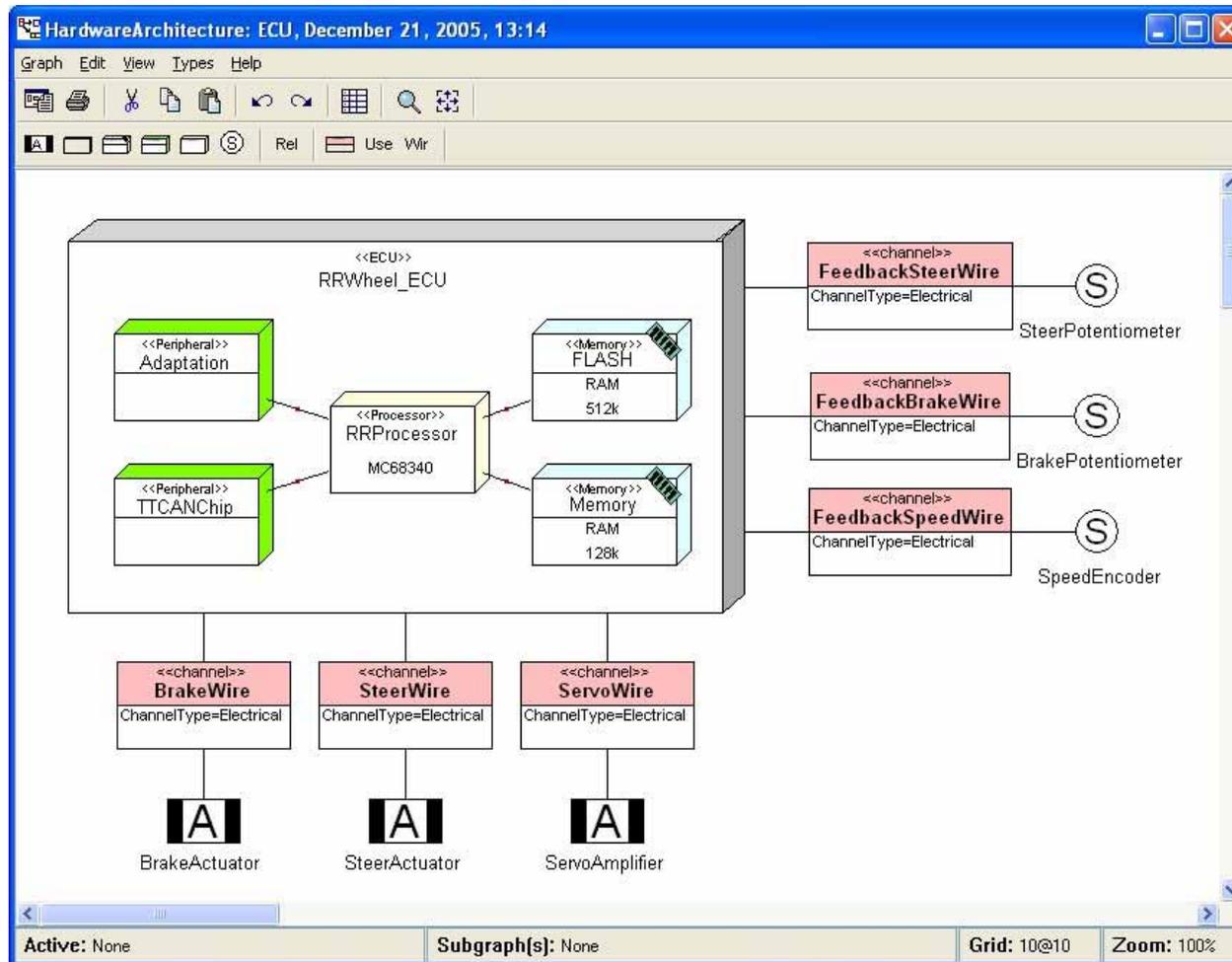


# EAST-ADL: Functional Architecture



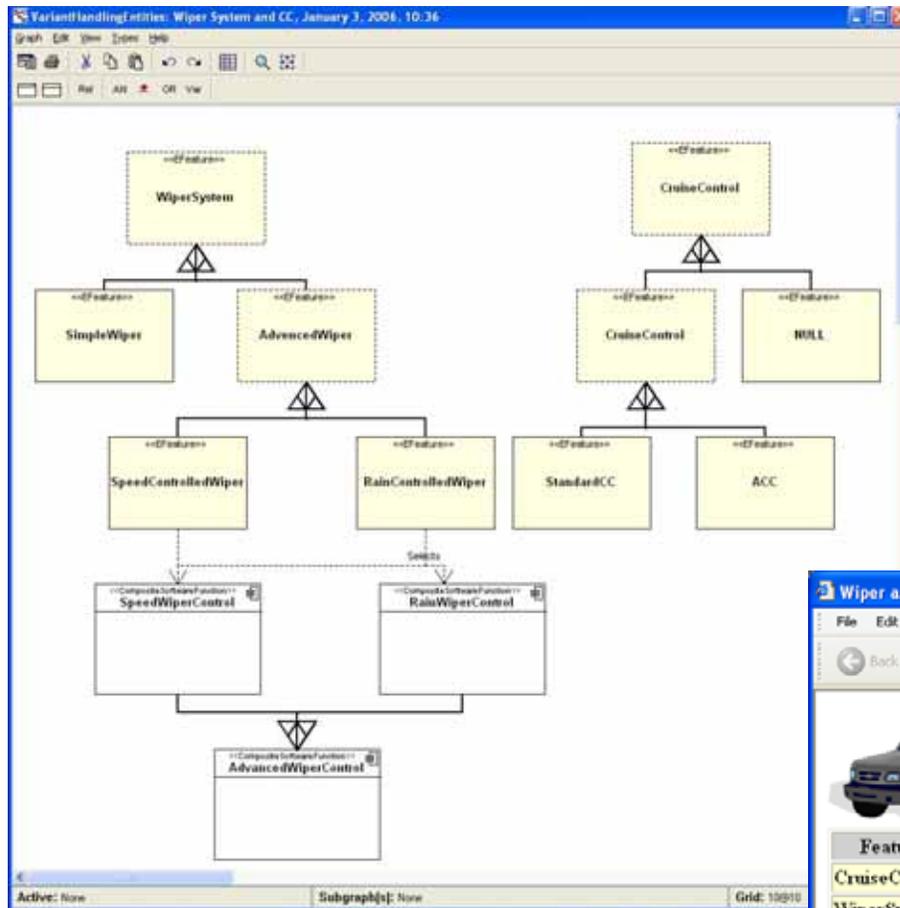


# EAST-ADL: Hardware Architecture





# EAST-ADL: Variant Configuration

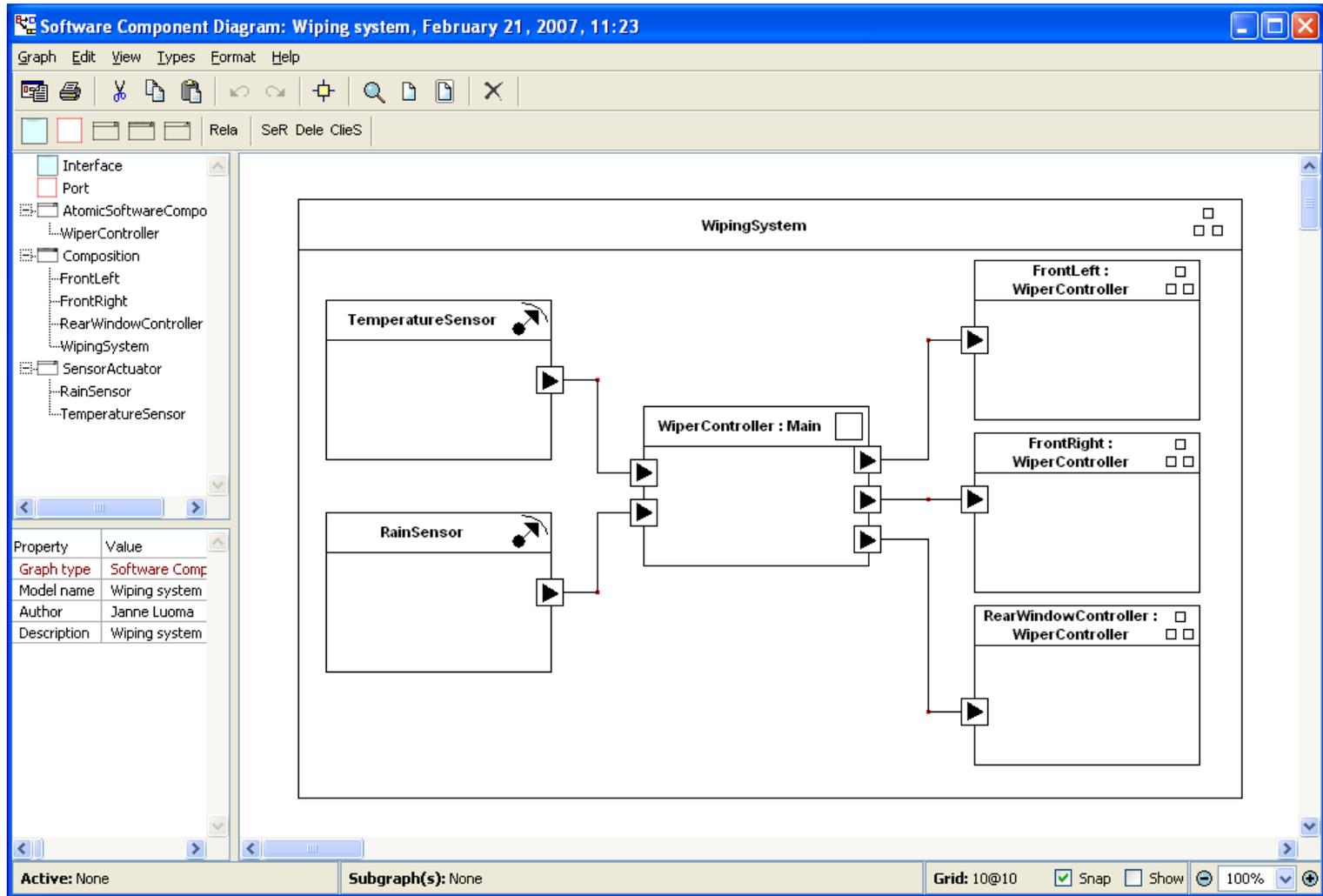


The screenshot shows a web browser window titled 'Wiper and Cruise Control system report - Microsoft Internet Explorer'. The browser displays a report for a vehicle, represented by an image of a silver SUV. Below the image is a table with two columns: 'Feature' and 'Choices'. The table lists the following features and their choices:

Feature	Choices
CruiseControl	<input type="radio"/> StandardCC <input type="radio"/> ACC
WiperSystem	<input type="radio"/> SimpleWiper <input type="radio"/> RainControlledWiper <input type="radio"/> SpeedControlledWiper

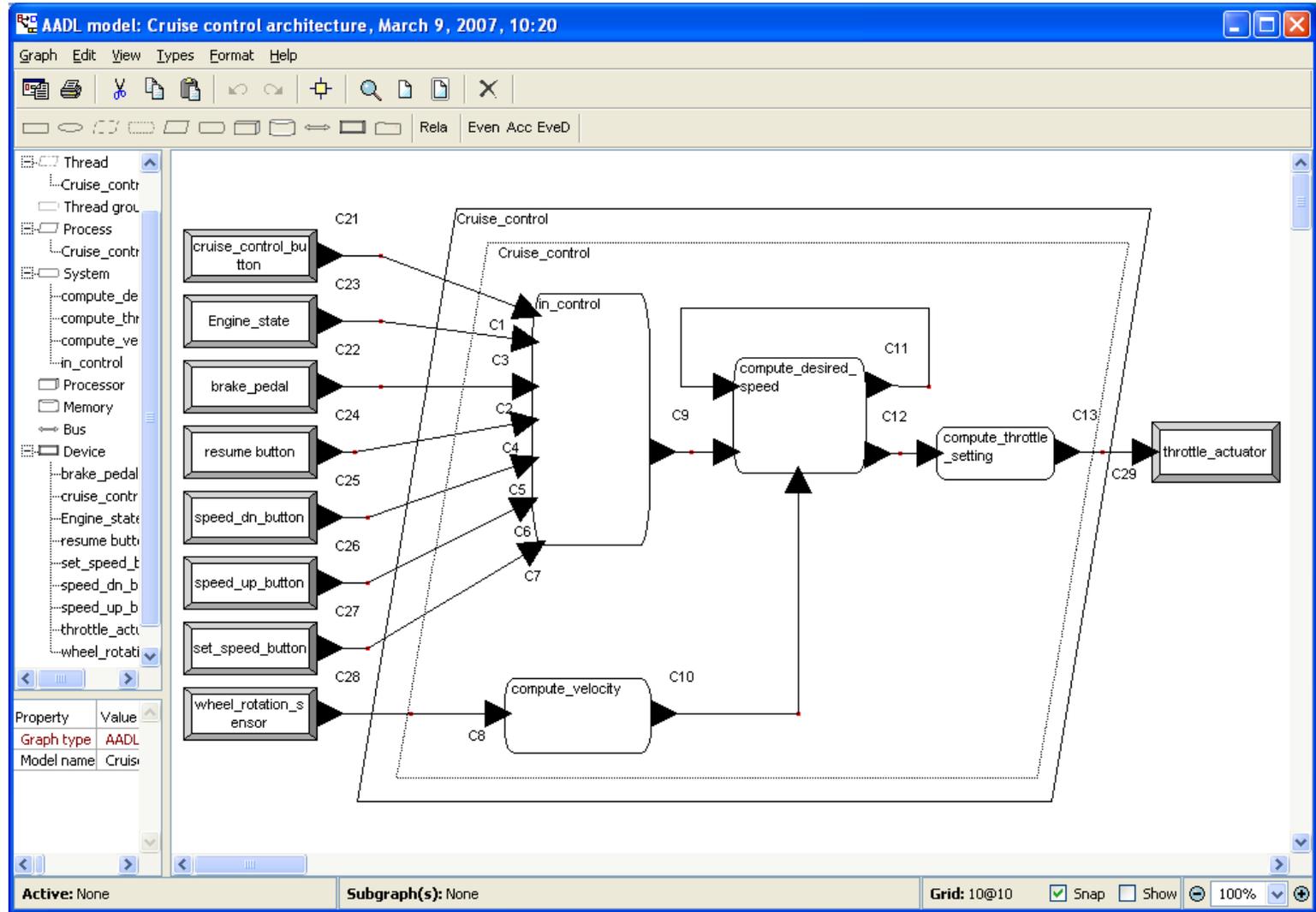


# AUTOSAR: SW architecture



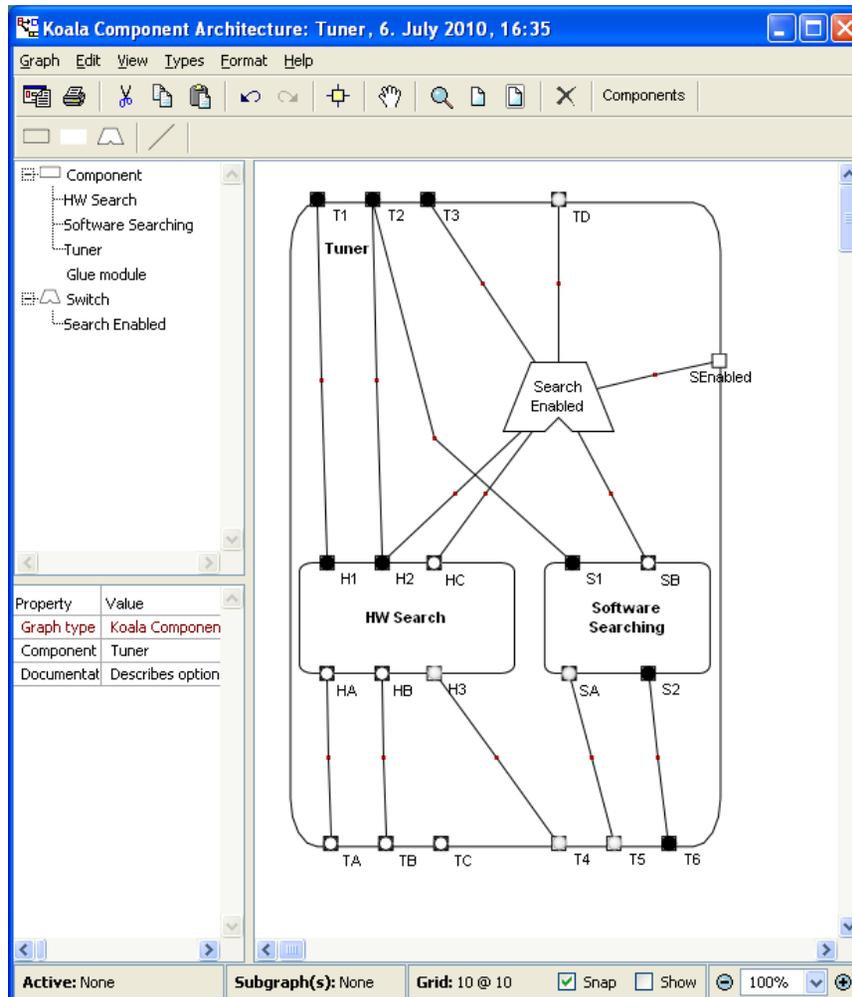


# AADL: scheduling and flow control





# Koala (ADL from Philips)



```
Component Tuner {  
  Provided interfaces:  
  - T1  
  - T2  
  - T3  
  - T4 optional  
  - T5 optional  
  - T6  
  Required interfaces:  
  - TA  
  - TB  
  - TC  
  - TD optional  
  Diversity interfaces:  
  - SEnabled  
}  
  
Component HW Search {  
  Provided interfaces:  
  - H1  
  - H2
```



# メタモデルでADLを定義

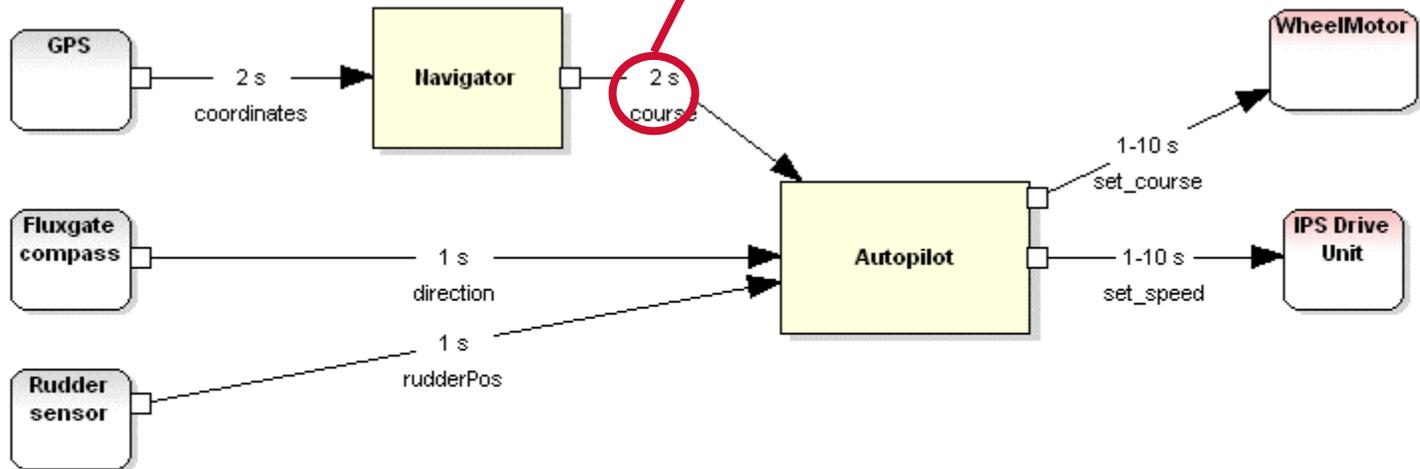
- Metamodeling (M2)
  - ドメインのコンセプトを特定・仕様化
  - 結果:メタモデル
- Modeling (M1)
  - メタモデルでシステムやソフトウェアを定義
  - 結果:モデル(+生成されるコードなど)

The image shows two screenshots from the Metamodel [GPSR] application. The top screenshot displays a class diagram for 'ADLFunctionType' with attributes: FunctionName: String, Configuration: String, IsElementary: Boolean, LocalFM: Object, MetaInfo: Collection, Part: Collection, and TransferFunction: String. It also shows a 'Parameters' association. The bottom screenshot is a configuration dialog for 'ADLFunctionType: Object' with the following fields: FunctionName (Perception), Parameters (+relDistance:double, +relSpeed:double), Configuration, IsElementary (checked), LocalFM (Car product line), MetaInfo, Part (function1, function2), TransferFunction, and Transparent (unchecked). Buttons for OK, Cancel, and Info... are at the bottom.



# ADLを進化させる

- プッシュモード、プルモードの違いを定義したい
- 信号に新しく 追加のプロパティが必要







# Integrating models/views

## ■ 単一言語では間に合わない

### – 開発工程上で連携されるべき

- トレーサビリティとインパクトの解析
- コード、メトリクス、テストケース、ドキュメントなどの生成

## ■ 主に2つの統合・連携アプローチ

### 1. 変換(別のメタモデル間)

- 例: EAST-ADL2 モデルと AUTOSAR モデル
- 課題は片側モデルへの変更の反映
- 異なる組織・チーム・提供者ごとにモデルを共有できる(情報の隠蔽ができる)

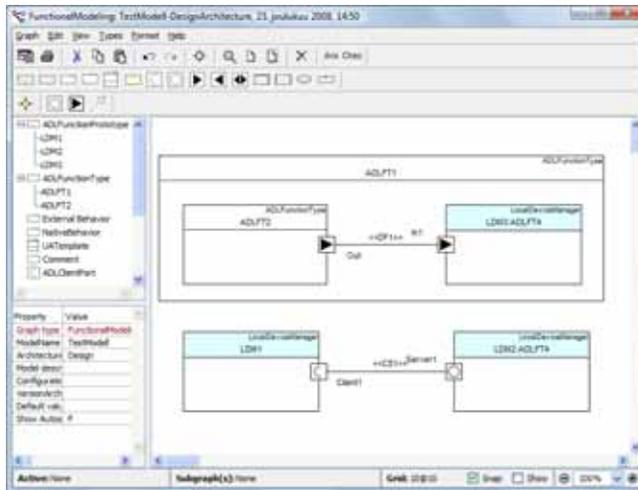
### 2. 共通のメタモデル(e.g. EAST-ADL and AUTOSAR)

- 例: ファンクションとネットワークのアーキテクチャー間の統合
- 協調作業、共同作業を支援
- 異なるチームごとに部分モデルを抽出・提供できる

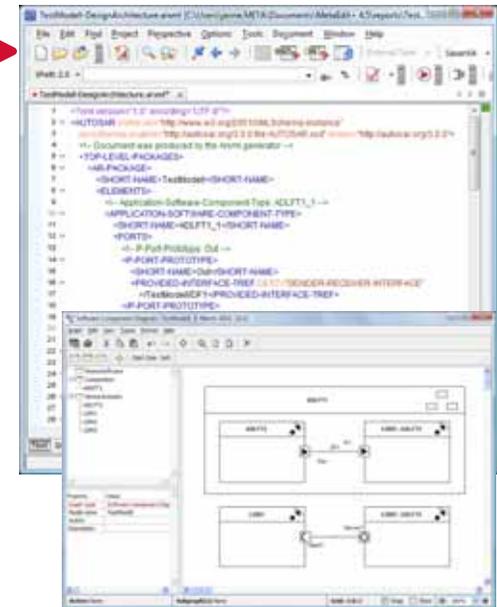


# 1. モデル変換の例: EAST-ADL2 to AUTOSAR

EAST-ADL2 model



Autosar



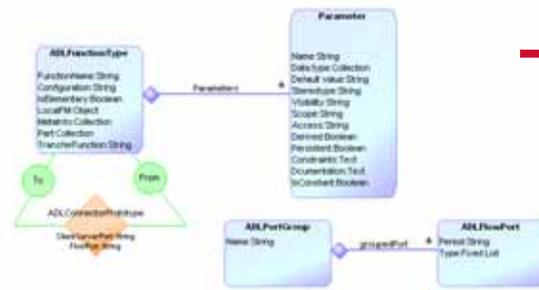
Mapping rules

Rule ID	Element type in EAST-ADL	Notation	Element type in the AUTOSAR SWS
1	ADLFunction	class	APPLICATION SOFTWARE COMPONENT TYPE
2	ADLFunctionType (class type property = 1 and a having subelements)	class	SEMI-STRUCTURED SOFTWARE COMPONENT TYPE
3	ADLFunctionType with allocated property = 1 and a having subelements	class	SEMI-STRUCTURED SOFTWARE COMPONENT TYPE
3.1	including rule 3.1.1 included in subelements	relationship	REFERRABLE OBJECT
3.2	including rule 3.1.2 included in subelements	class	REFERENCE
3.3	including rule 3.2 connected flow relationship	relationship	CONNECTED PART
4	ADLFunctionType with allocated property = 1 and a having subelements	class	APPLICATION SOFTWARE COMPONENT TYPE
4.1	Subelements (except of ADLFunctionType) (including rule 4)	class	REFERENCE
4.2	Subelements (except of ADLFunctionType) (including rule 4)	relationship	PROVIDED-INTERFACE
5	ADLPort	class	PORT PROTOTYPE
6	ADLPort	class	PORT PROTOTYPE
7	ADLPort	class	PORT PROTOTYPE
8	ADLPort	class	PORT PROTOTYPE

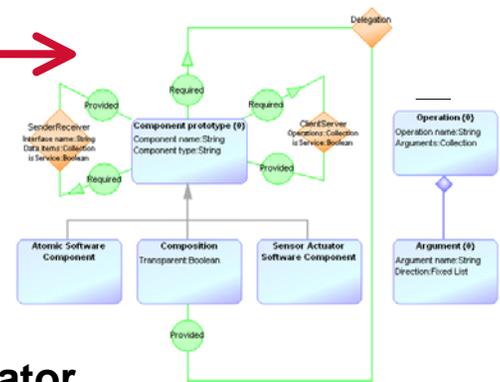


# 各ADL言語のメタモデル同士で変換を定義

## EAST-ADL2 metamodel



## Autosar metamodel

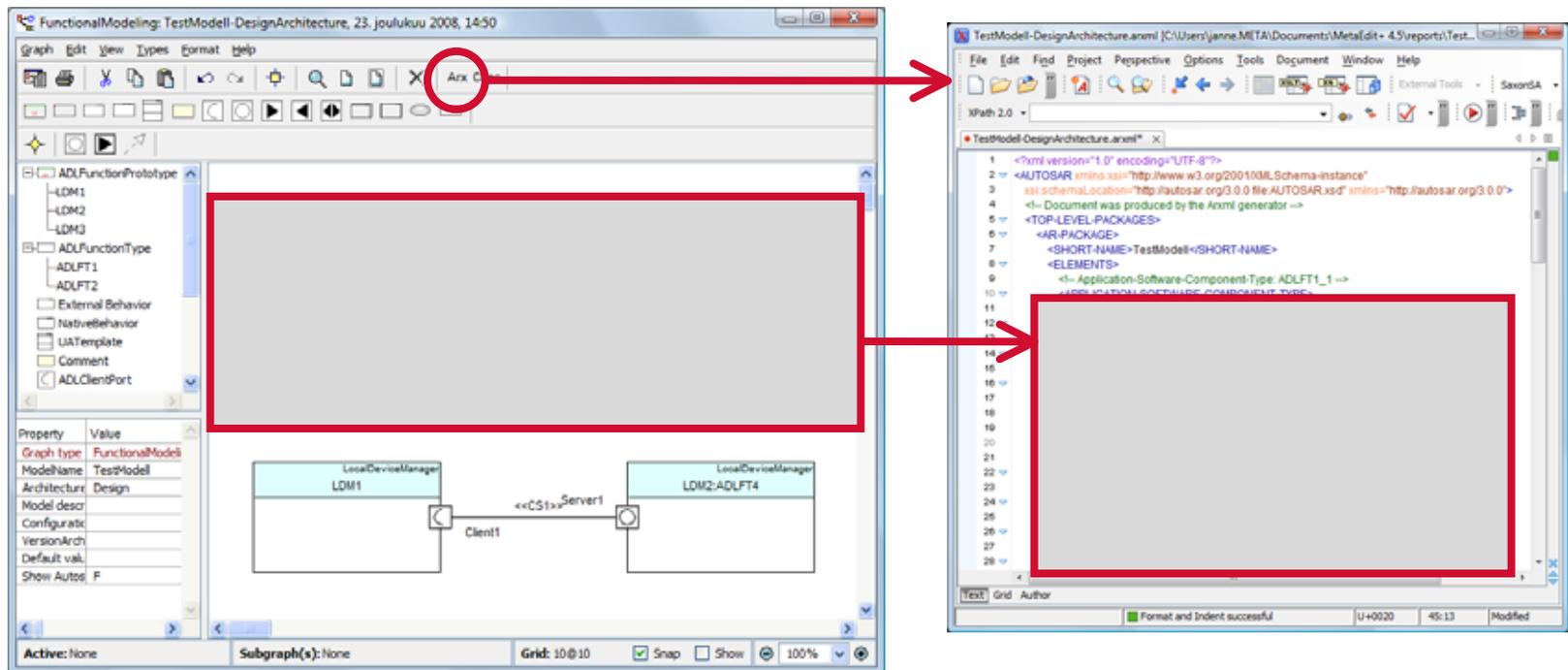


## Mapping rules in generator

Rule	Element type in EAST-ADL	Metaclass	Element type in the AUTOSAR metamodel
1	ABLFunctionType	class	APPLICATION SOFTWARE COMPONENT TYPE
2	ABLFunctionType (when type property value is "S")	class	SENSOR ACTUATOR SOFTWARE COMPONENT TYPE
3	ABLFunctionType (when type property is "A" and is having subelements)	class	SENSOR ACTUATOR SOFTWARE COMPONENT TYPE
3.1	including rule 3's subelements in subelements	relationship	REFERRABLE INCLUDE
3.2	including rule 3's include in subelements	class	REFERENCE
3.3	including rule 3.2's connected flow relationship	relationship	SWTCHEN UNPLUGGED REF
4	ABLFunctionType (when type property is "A" and is having subelements)	class	APPLICATION SOFTWARE COMPONENT TYPE
4.1	Subelements (except of ABLFunctionType (including rule 4))	class	REFERENCE
4.2	Subelements (not including of ABLFunctionType (including rule 4))	relationship	SENDER RECEIVER INTERFACE
5	ABLFlow	class	PORT MESSAGE
6	ABLFlow	class	PORT MESSAGE
7	ABLFlow	class	PORT MESSAGE
8	ABLFlow	class	PORT MESSAGE



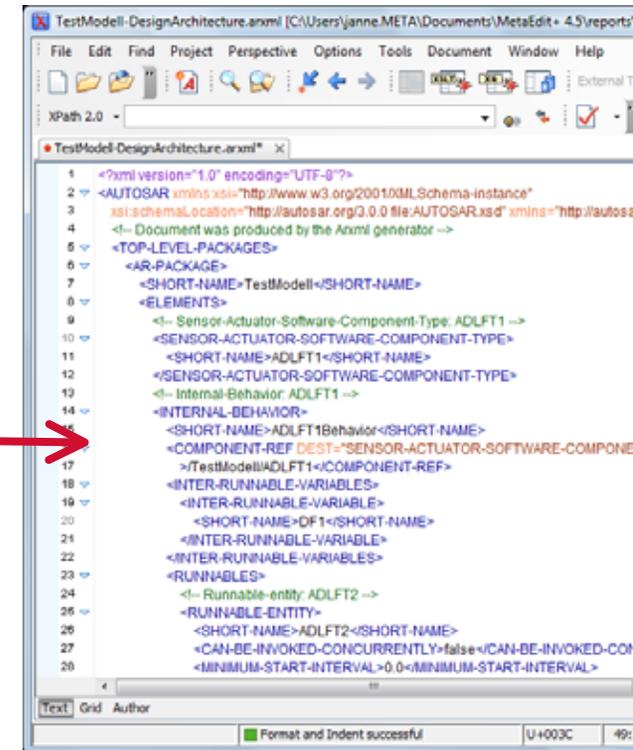
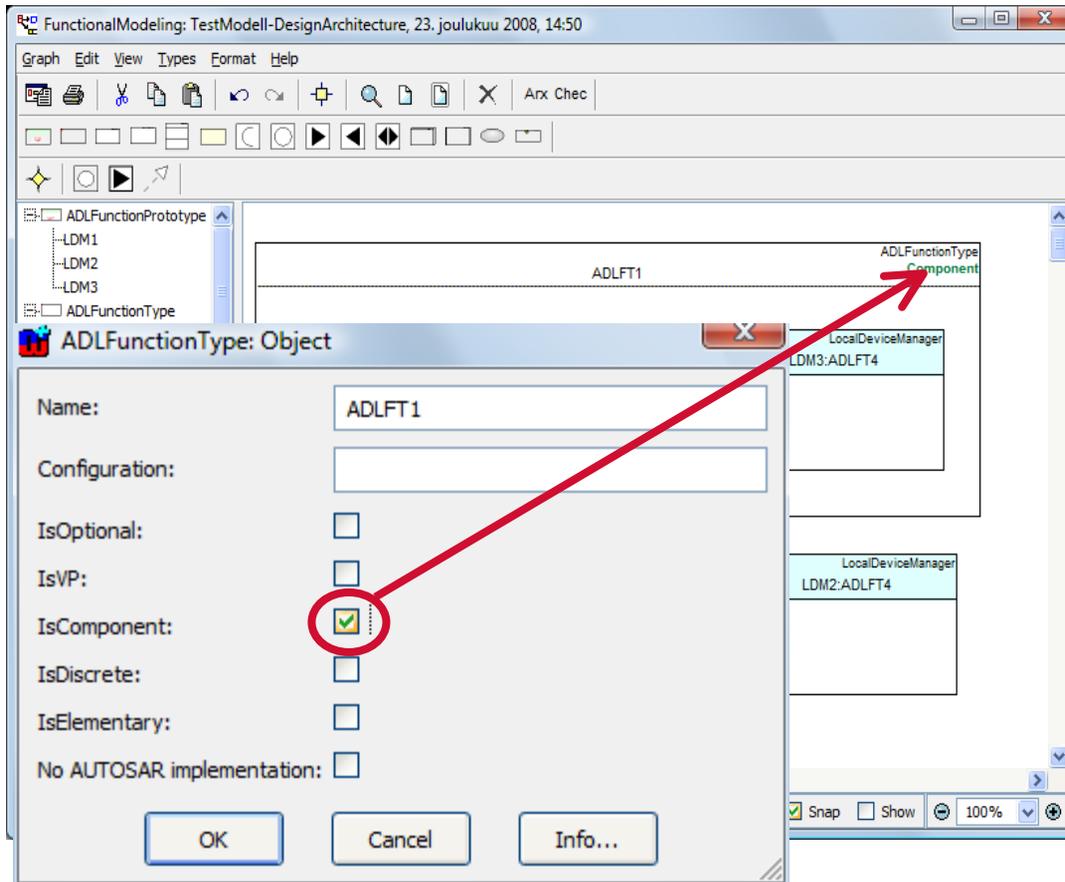
# メタモデル同士でマッピング



- EAST-ADL2のファンクショナルデザインアーキテクチャがAUTOSARへ変換
- ジェネレータを実行することで変換を実施



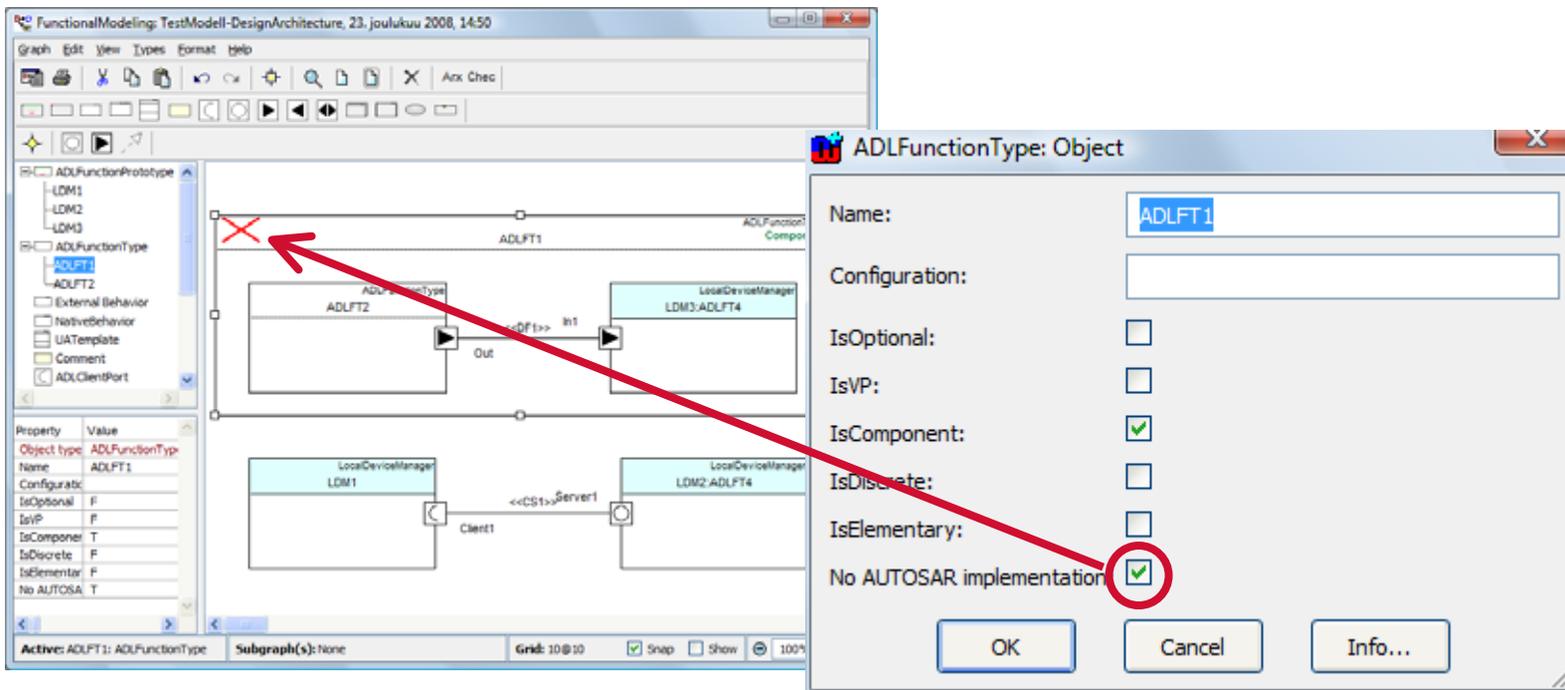
# 開発者はマッピングルールを選択できる



- モデルへの設定次第で生成されるマッピング情報が変わる
  - ファンクションをコンポーネントとして定義



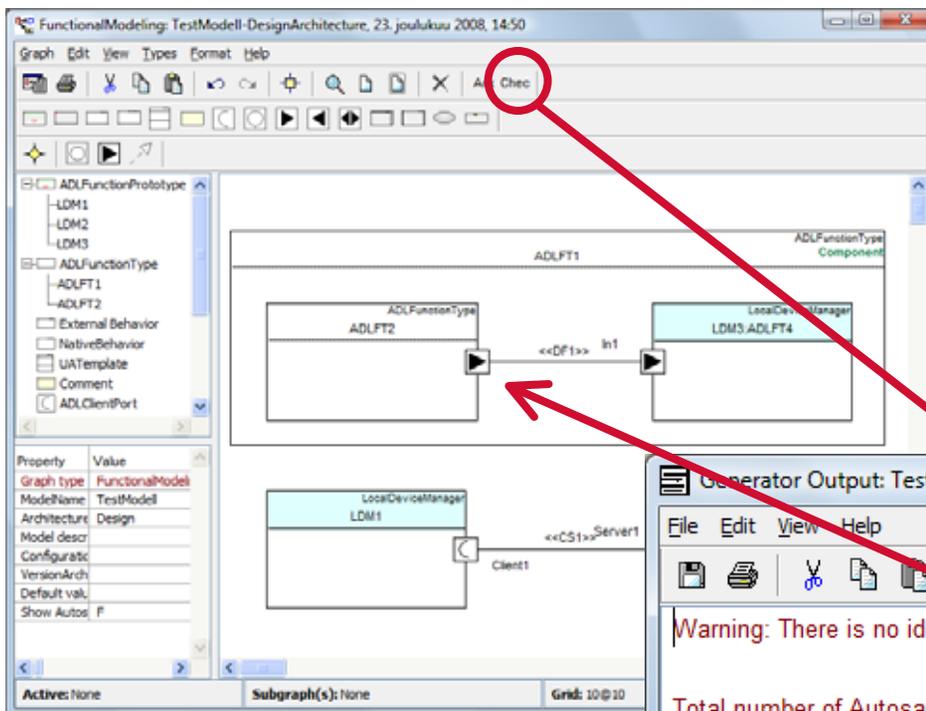
# AUTOSAR 生成オプション



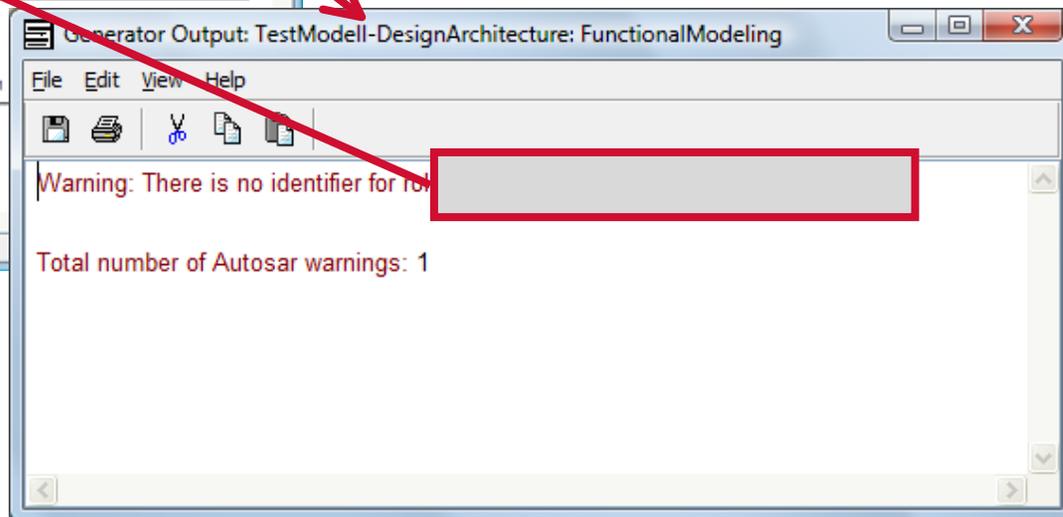
- AUTOSAR 変換に対してエレメントが選択されなければ赤Xがシンボルに表示  
⇒ エレメントとサブエレメントがAUTOSARに変換されない



# EAST-ADL2 モデルチェック機能

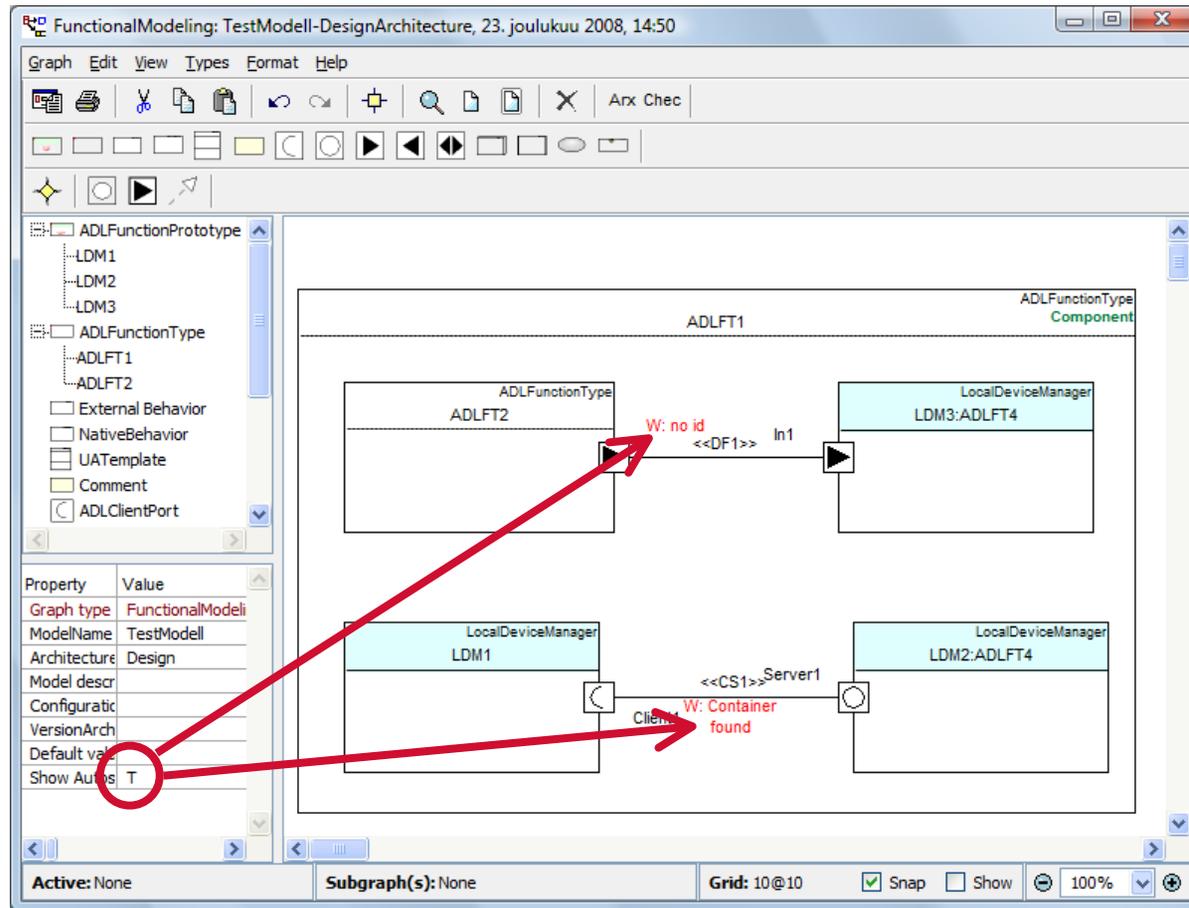


- 生成前にEAST-ADL2 モデルをチェック
- ウォーニングからモデルにナビゲートできる





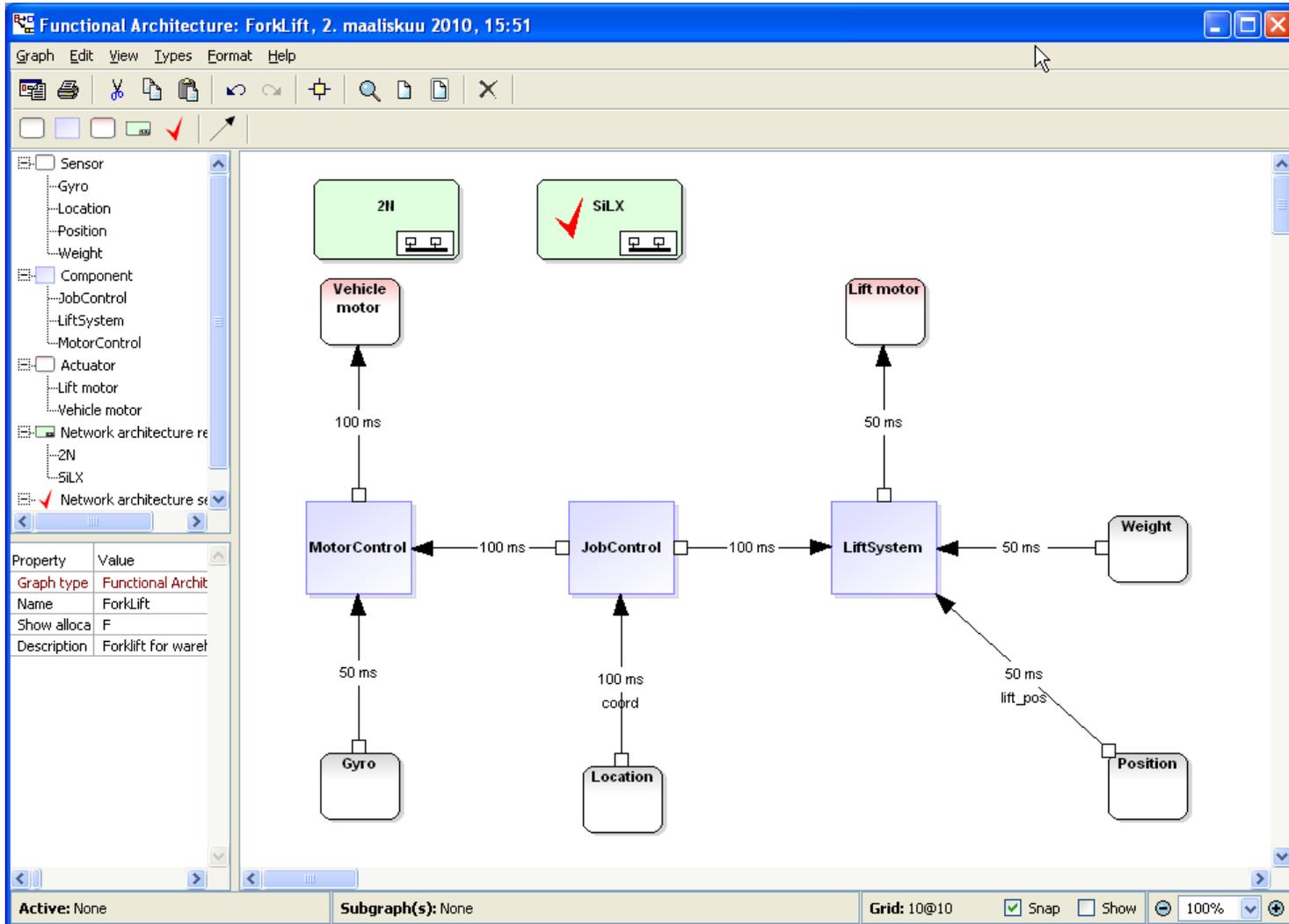
# リアルタイムにモデルへエラーを注釈



- エラー情報をモデル上に注釈
  - グラフプロパティ: AUTOSARのウォーニングをEAST-ADL2内に

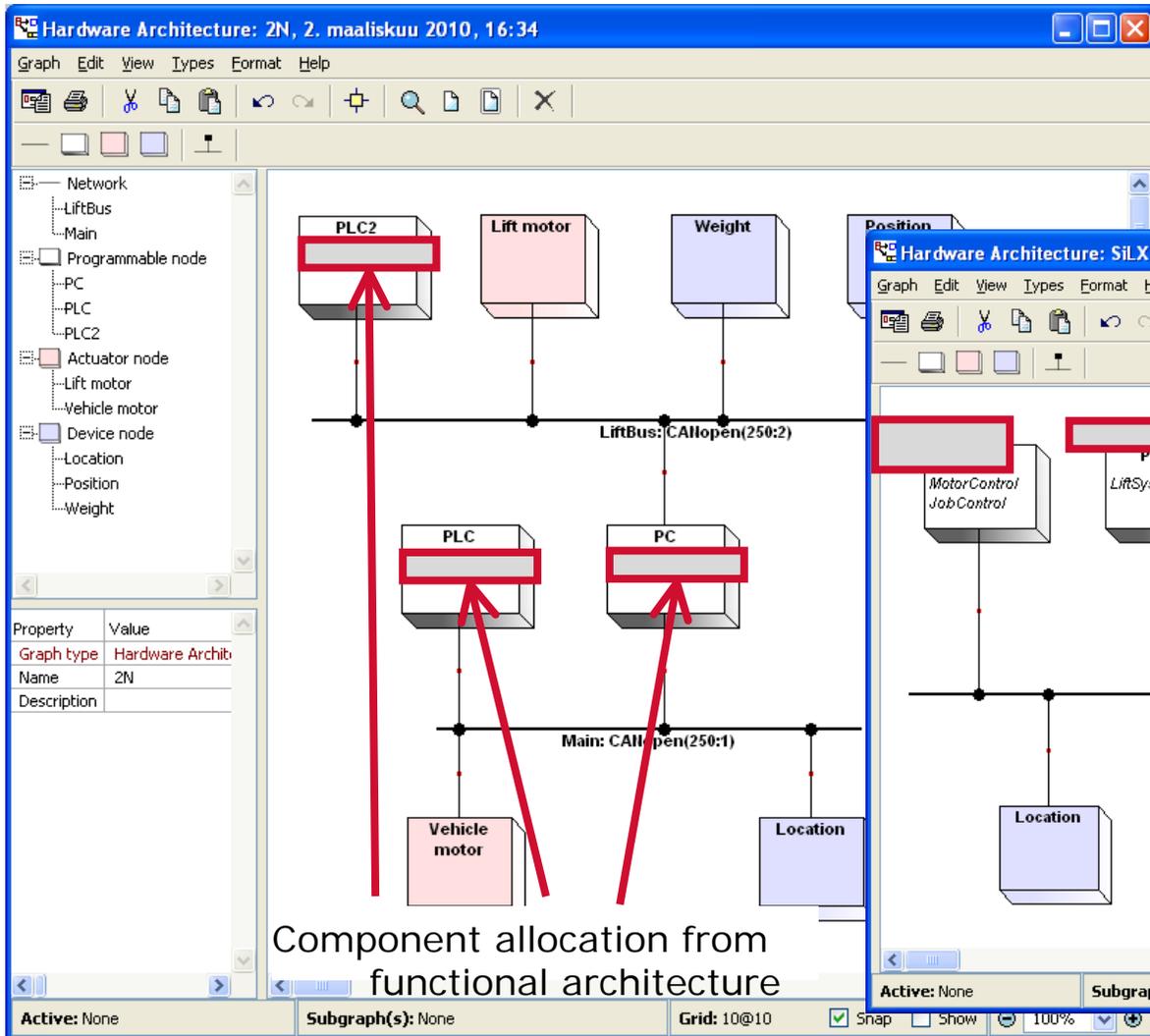


## 2. メタモデル上で統合例：ファンクショナルアーキテクチャモデル図

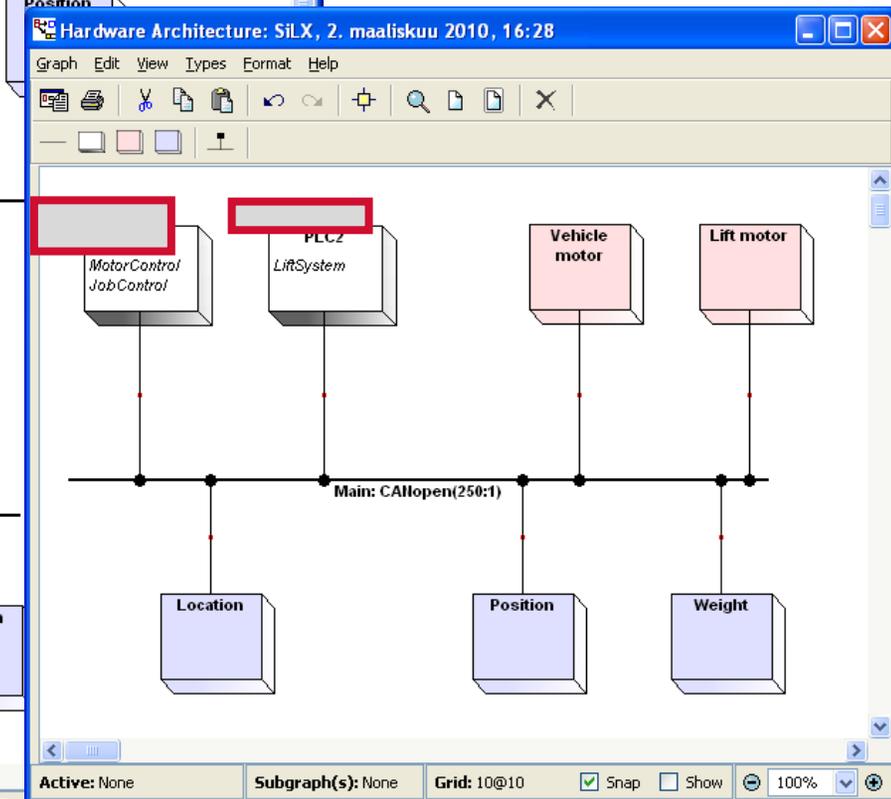




## 2. メタモデル上で統合例：HWネットワークアーキテクチャモデル図

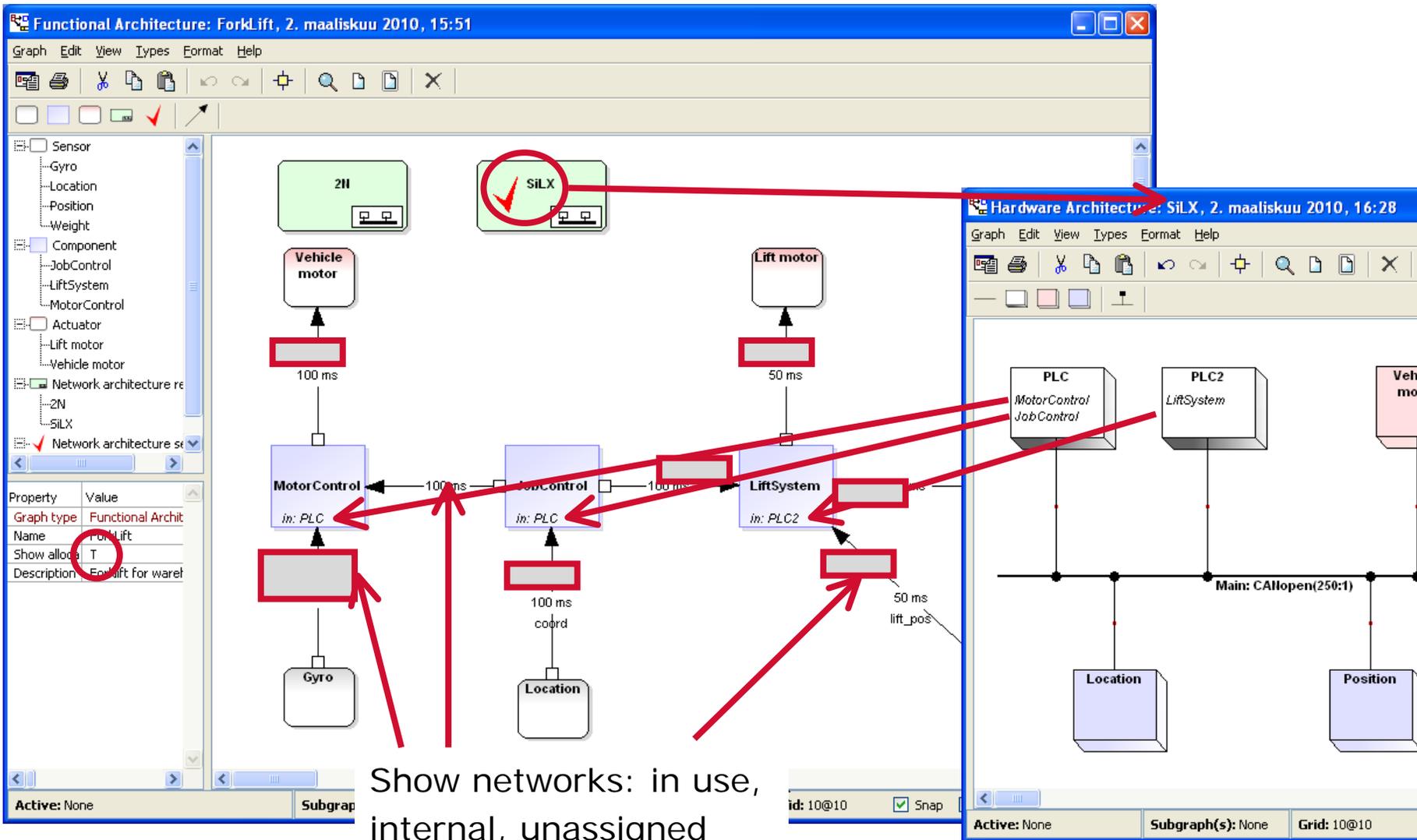


- 2種のHWアーキテクチャモデル フィールドバスのみ、フィールドバスとケーブル接続の両方



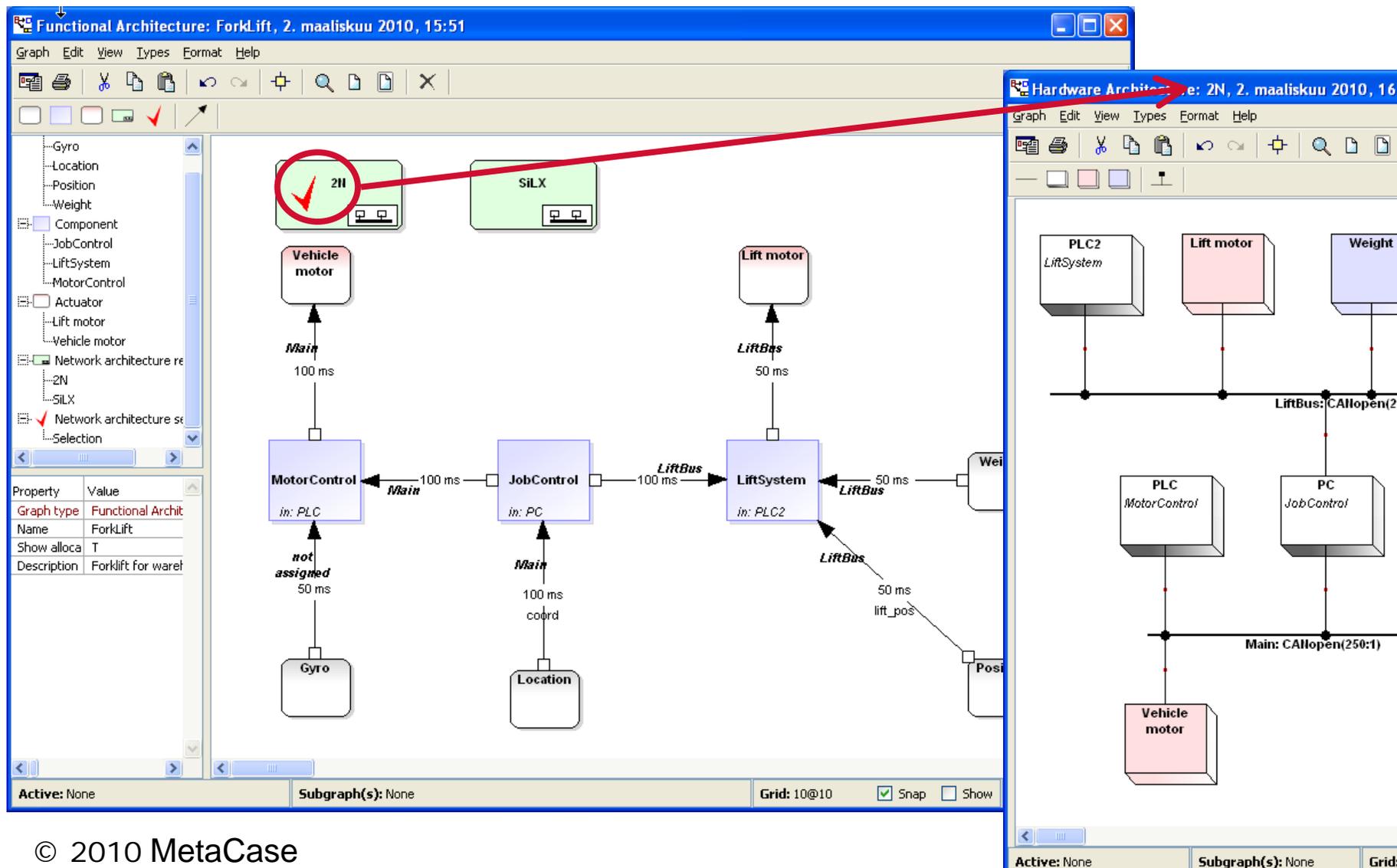


# ファンクショナルアーキテクチャモデル図と HWネットワークアーキテクチャモデル図の連携





# 別のHWネットワークアーキテクチャモデル図





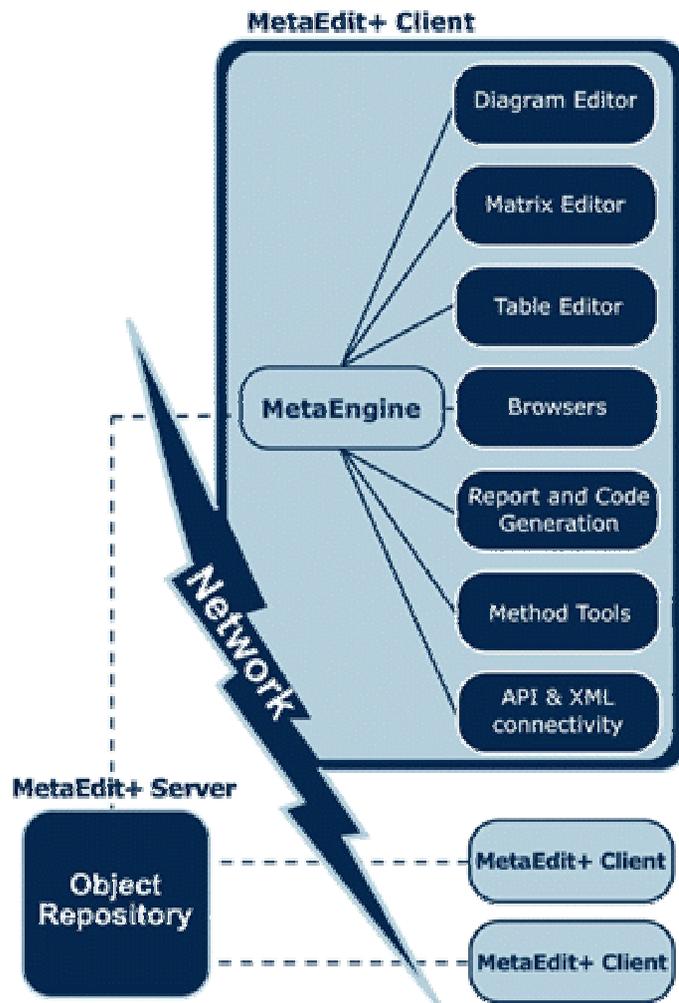
# ADLをサポートするツールに必要な機能

- 様々なモデリングをサポート
  - 様々なエディタ(ダイアグラム、マトリックス、表)
  - モデルブラウザー
  - モデルの自動バリデーション
  - コード、ドキュメント、テストなどあらゆる成果物の自動生成
- 拡張性
  - マルチユーザサポート
  - モデル、モデルエレメントの共有、再利用の支援
- オープン性
  - あらゆるツールとの連携
- ADL言語、ジェネレータが柔軟に変更できること



# マルチユーザーとマルチプラットフォーム

- Windows
- Linux
- Solaris
- HP-UX
- Mac OS X
  
- ツールの統合
  1. SOAP / Webサービス / .NETを使ったAPI
  2. XMLファイル
  3. システムコール / コマンドラインでの利用





# Case: Nokia Siemens Networks



- 通信システムプラットフォーム向けのモデリング言語を構築
  - アーキテクチャ上のルール、コンストレインツをサポートした
  - コーディングルールに準拠できるジェネレータ
- MetaEdit+ で モデリング言語、ジェネレータの実装に要した期間は1週間
- 効果:
  - アプリケーション開発者がアーキテクチャの詳細をマスターしなくても済んだ
  - アーキテクチャ上の変更が容易に展開できる
  - より良く アーキテクチャのルール、コーディングルールが守られる
  - コードジェネレータにより実装工程が自動化され工数削減
  - ドキュメントやメトリクスも自動生成
  - モデルの下、それら成果物の一貫性が保たれる



## まとめ

- 目的に応じて複数のADLが必要
- アーキテクチャ記述のためのツールは柔軟でなければならない
- メタモデルをベースにしたツールならADL言語の定義は低コストで実現できる
- ADL言語、モデル間の統合
  - 変換
  - 単一メタモデル内統合
- MetaEdit+ には多くの実績がある
  - 様々なADLをサポート
  - 様々な業界で活用されている