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

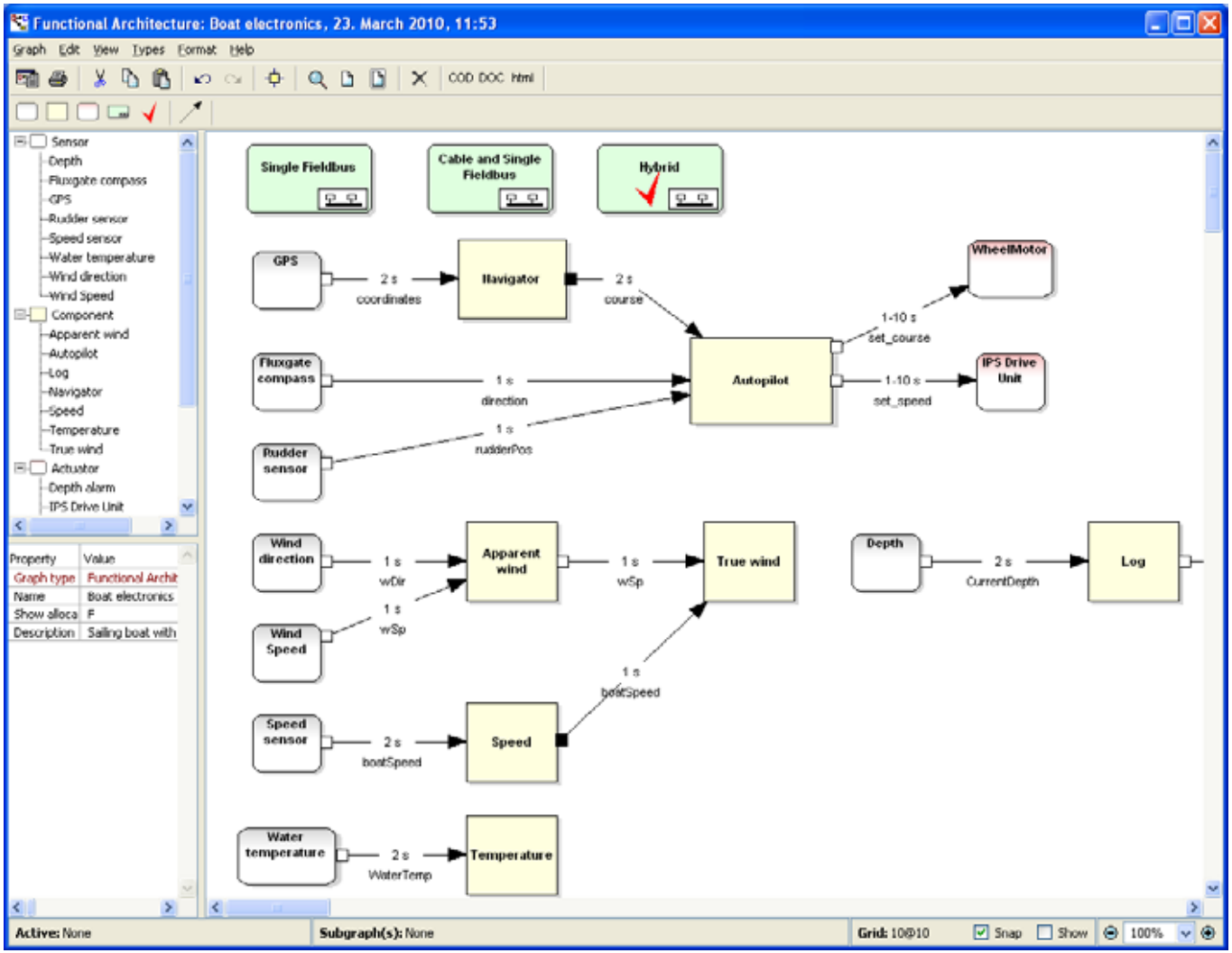
ADL に活用される MetaEdit+ の優位性

- 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



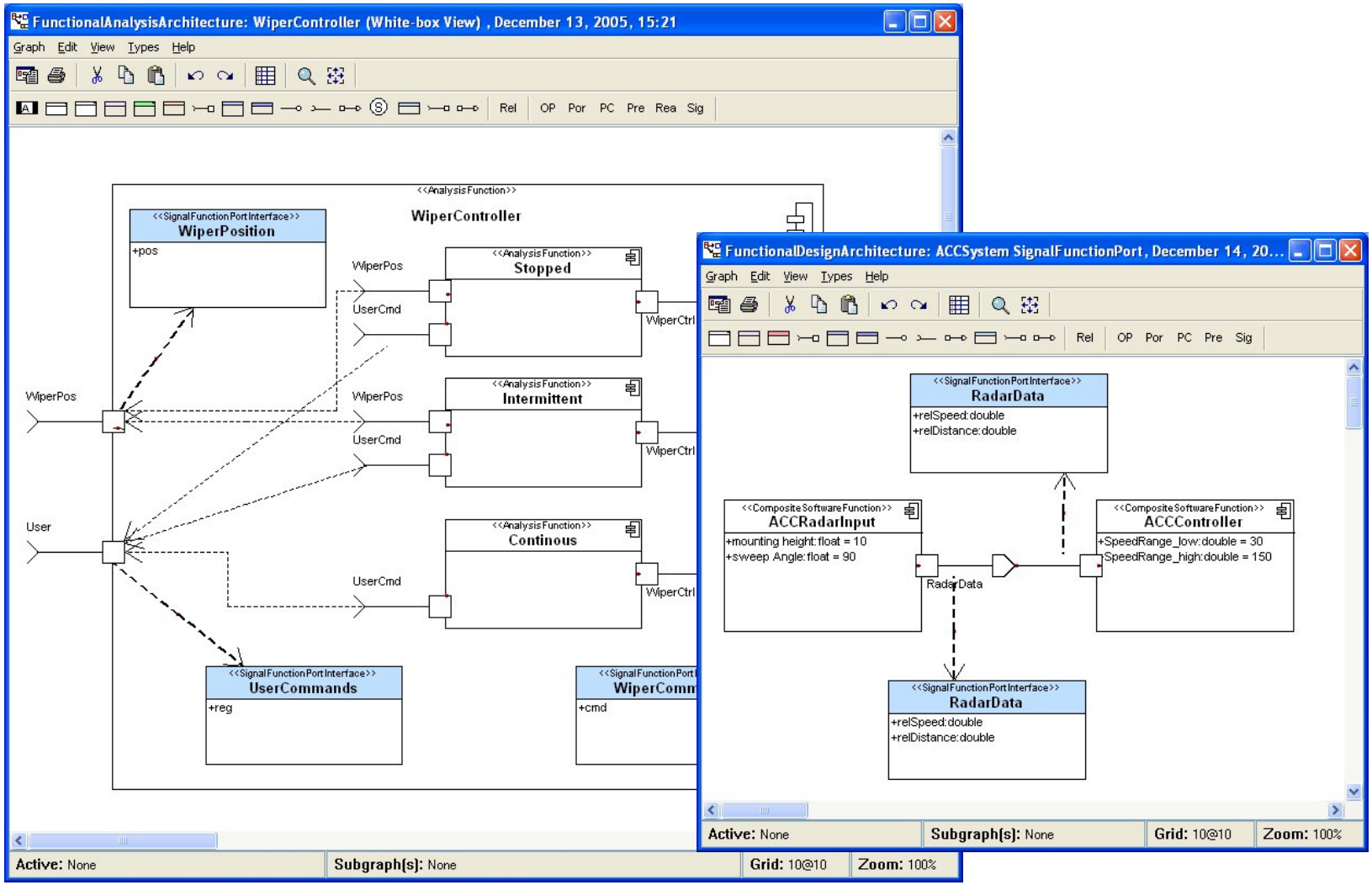
What you want to do with ADL?

- 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

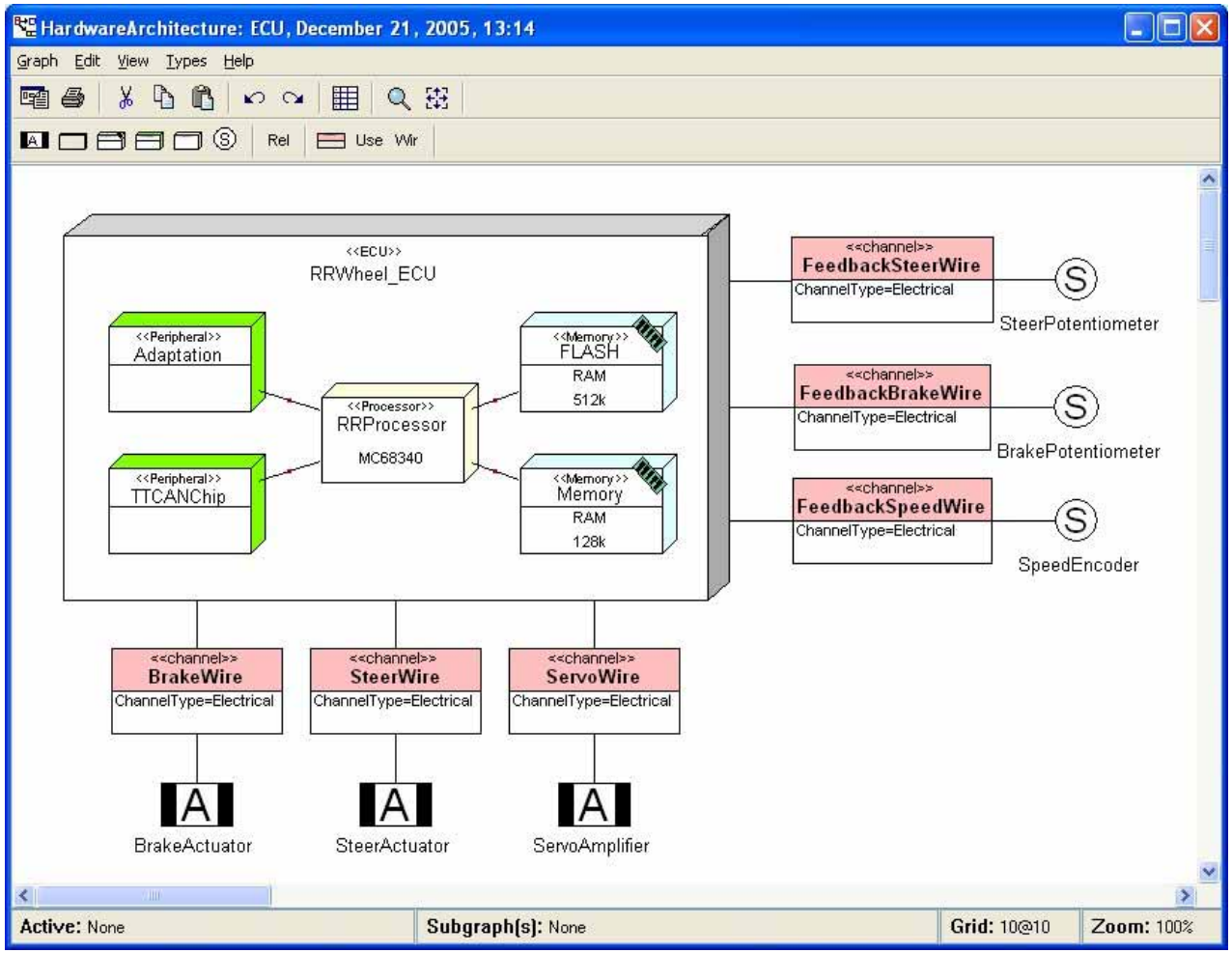
➔ 異なるADL言語ごとにそれぞれの支援対象がある！

- 一般にADLは特定用途のもの　自動車のADLなど

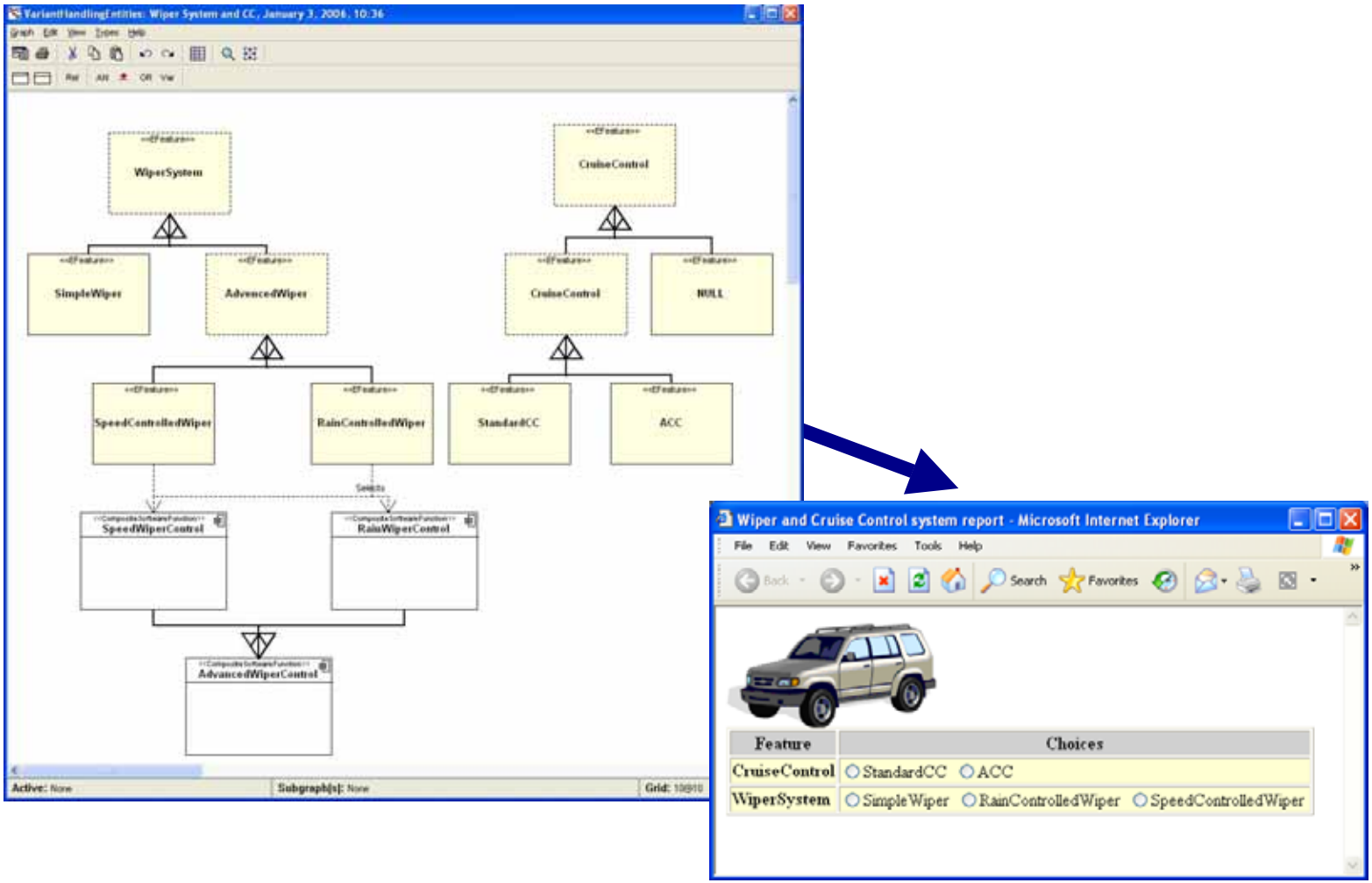
EAST-ADL: Functional Architecture



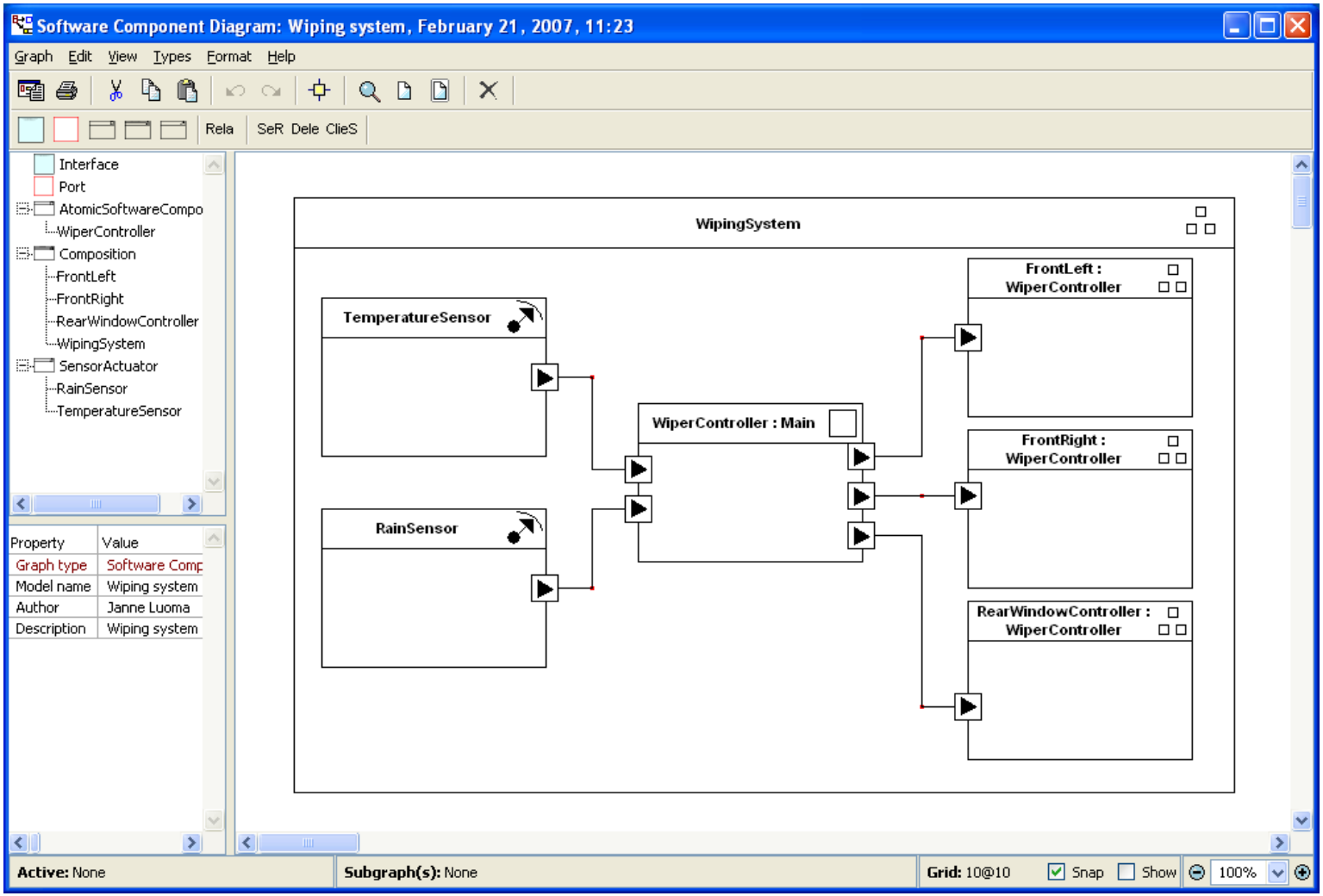
EAST-ADL: Hardware Architecture



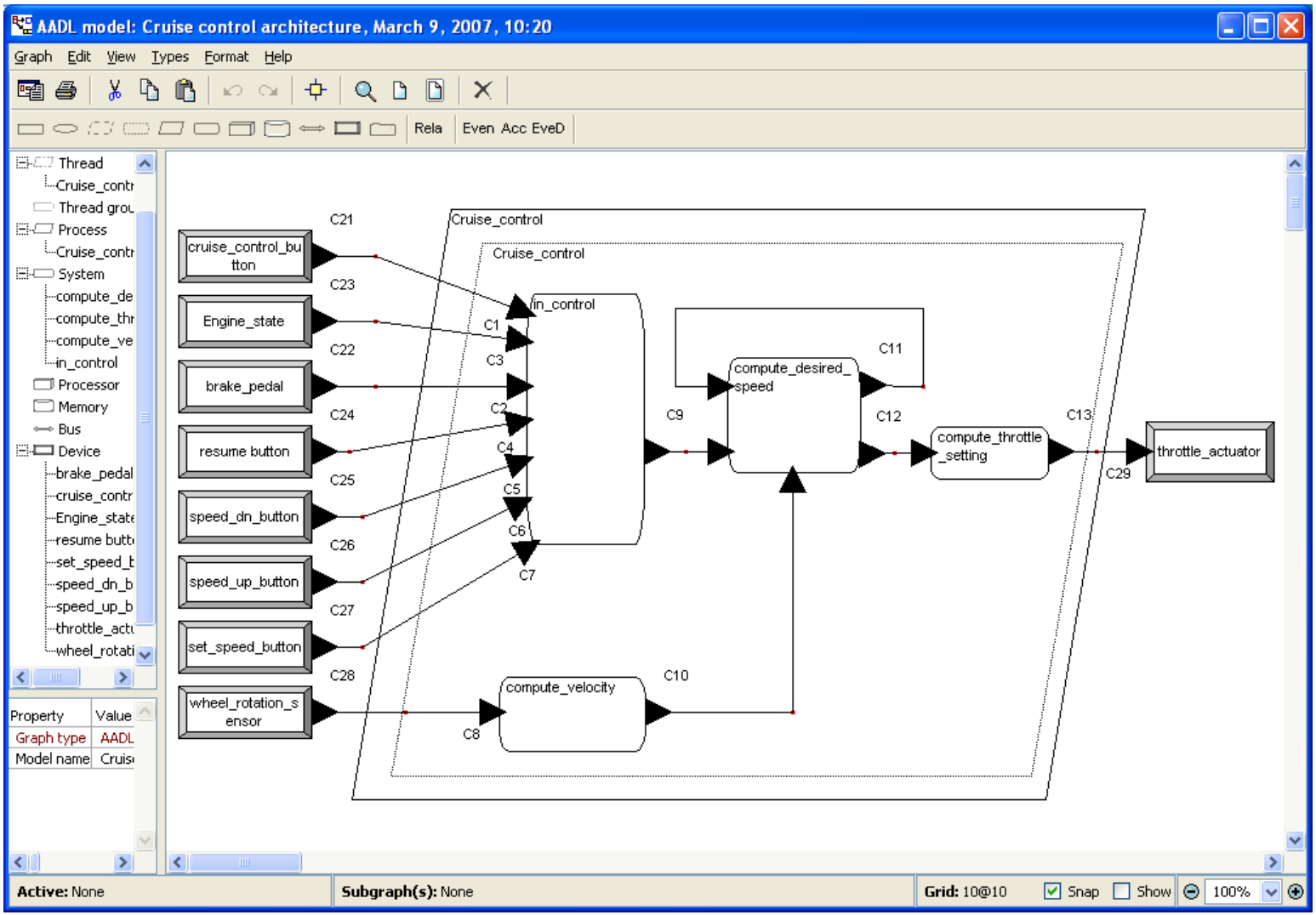
EAST-ADL: Variant Configuration



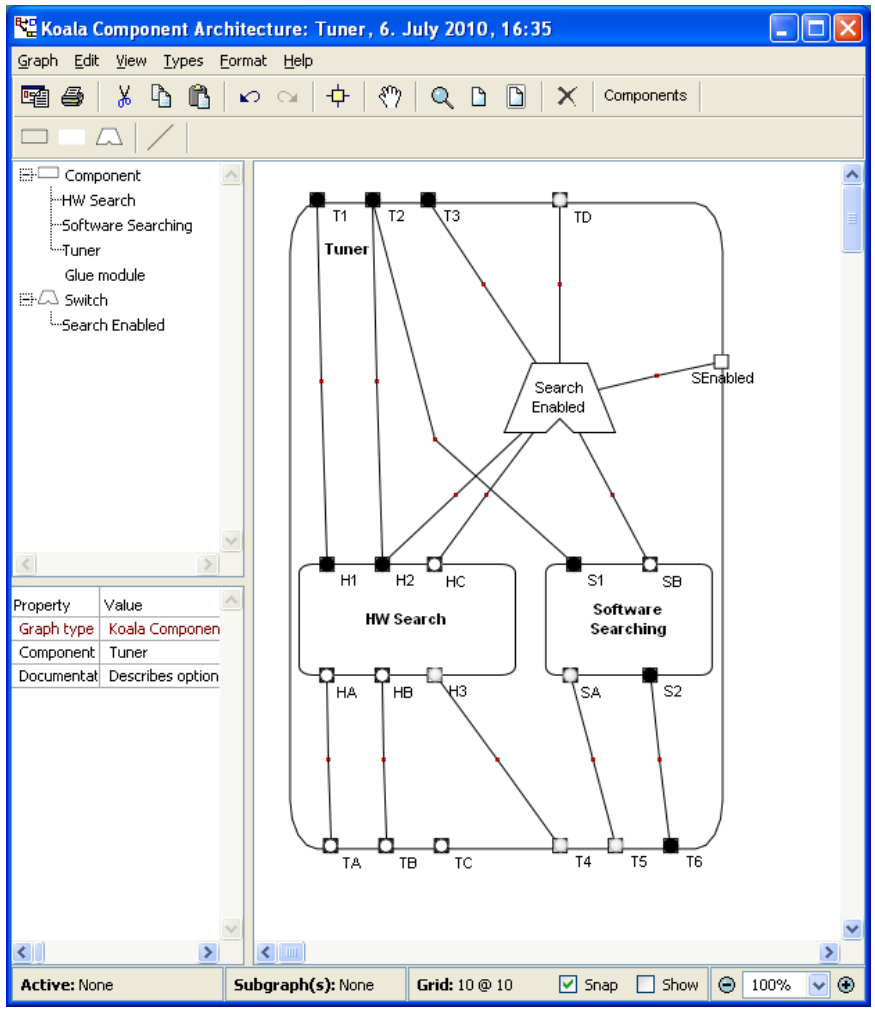
AUTOSAR: SW architecture

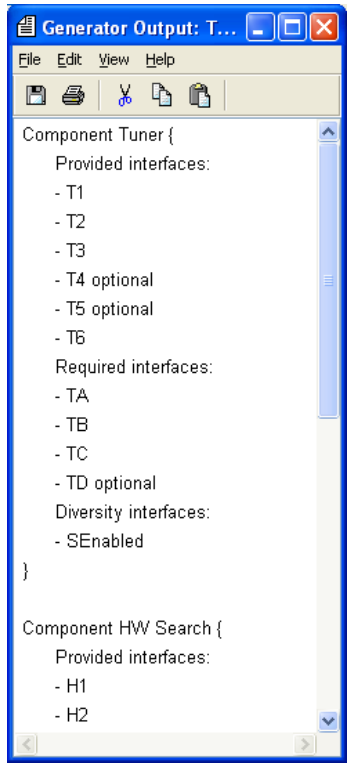


AADL: scheduling and flow control



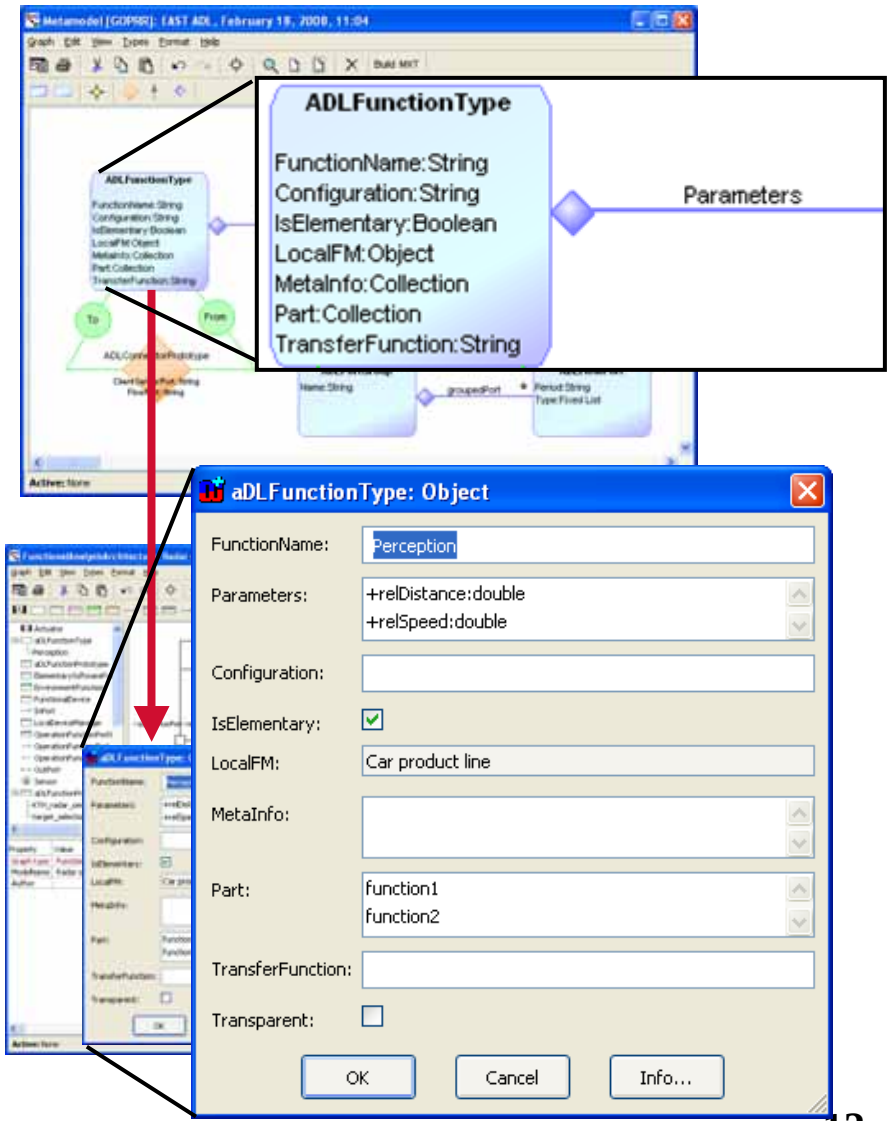
Koala (ADL from Philips)





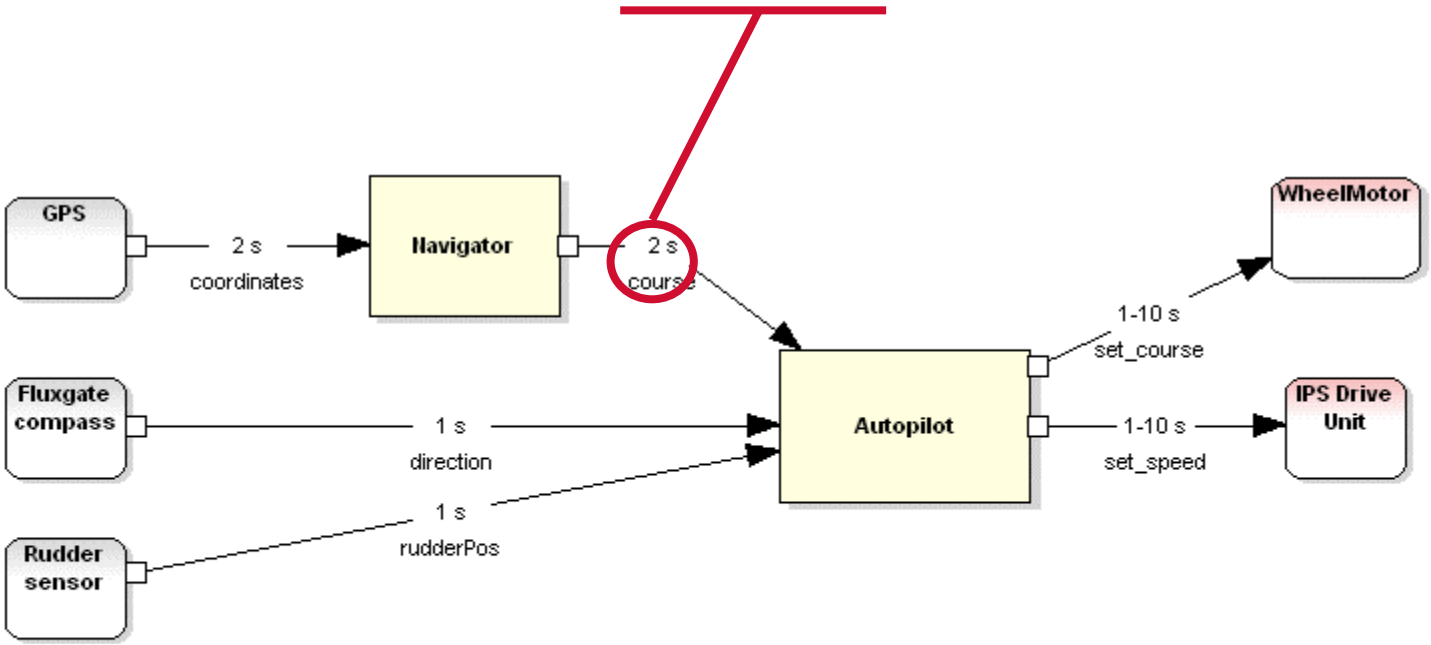
メタモデルでADLを定義

- 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エアを定義
  - 結果：モデル（＋生成されるコードなど）

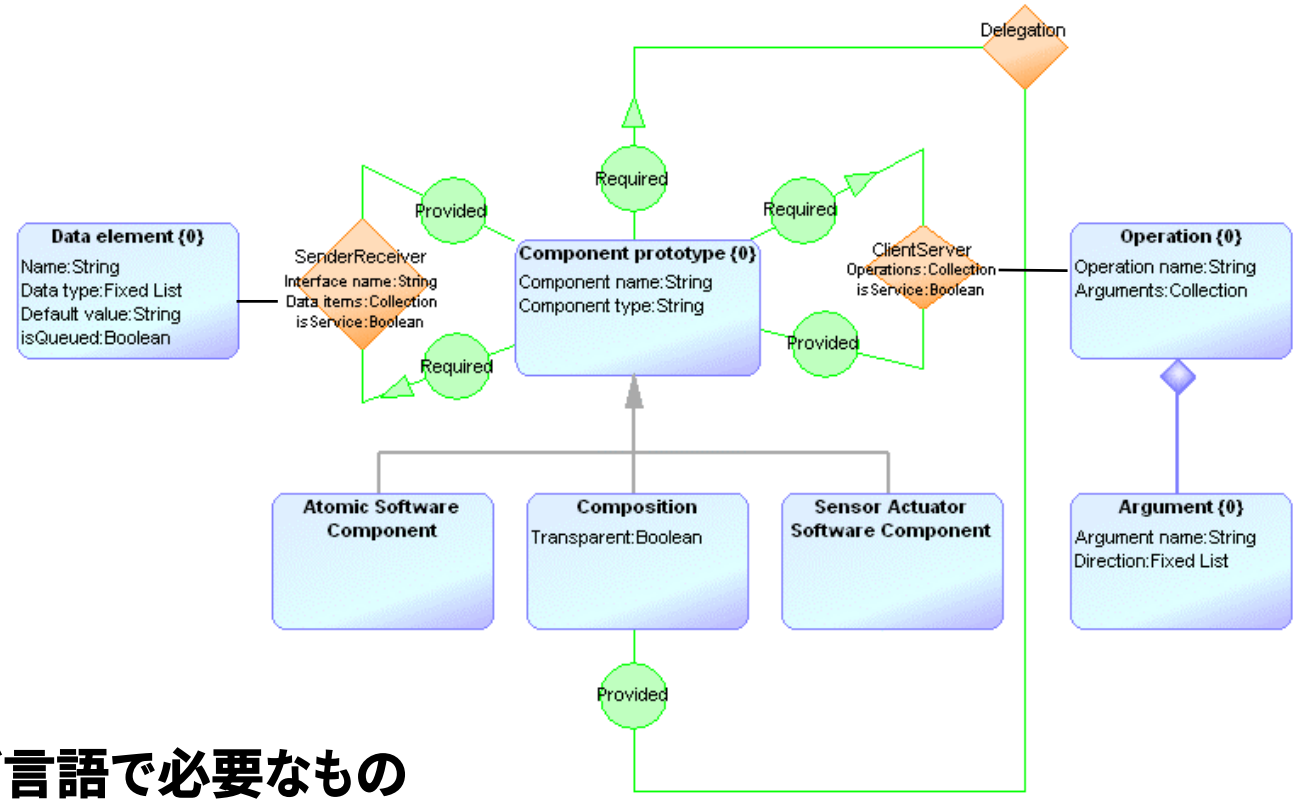


ADLを進化させる

- 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



AUTOSAR SWコンポーネントのメタモデル



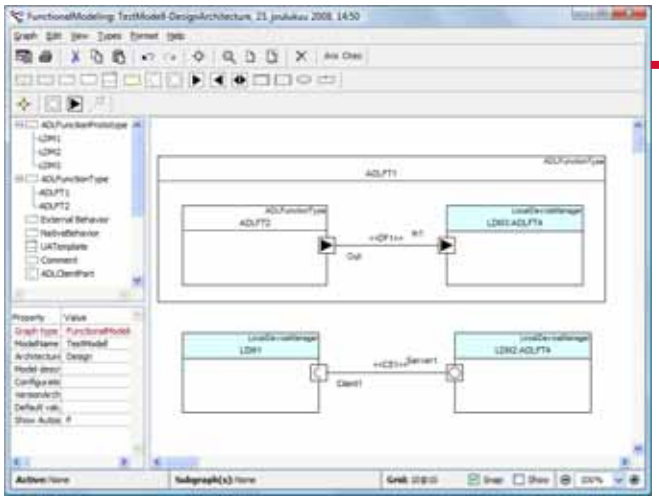
- モデリング言語で必要なもの
  - 表記、具象構文
  - 複数のビュー (関心事の分離)
  - ルール、コンストレインツ (一貫性、完全性、ネーミング)
  - セマンチックス (ジェネレータを介してのオペレーショナルセマンチックス)

Integrating models/views

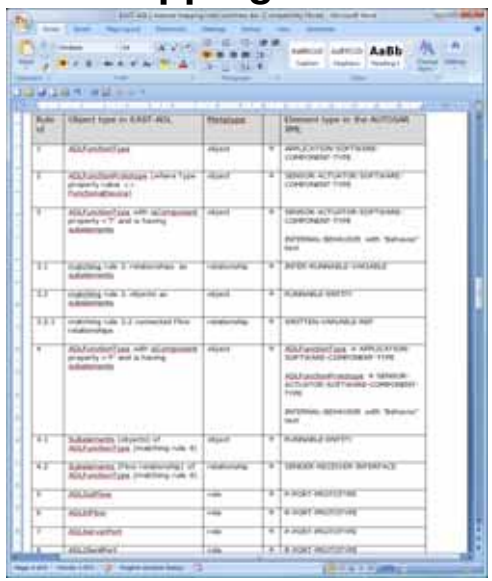
- **単一言語では間に合わない**
  - 開発工程上で連携されるべき
    - トレーサビリティとインパクトの解析
    - コード、メトリクス、テストケース、ドキュメントなどの生成
- **主に2つの統合・連携アプローチ**
  1. 変換 (別のメタモデル間)
     - 例: EAST-ADL2 モデルと AUTOSAR モデル
     - 課題は片側モデルへの変更の反映
     - 異なる組織・チーム・提供者ごとでモデルを共有できる (情報の隠蔽ができる)
  2. 共通のメタモデル(e.g. EAST-ADL and AUTOSAR)
     - 例:ファンクションと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

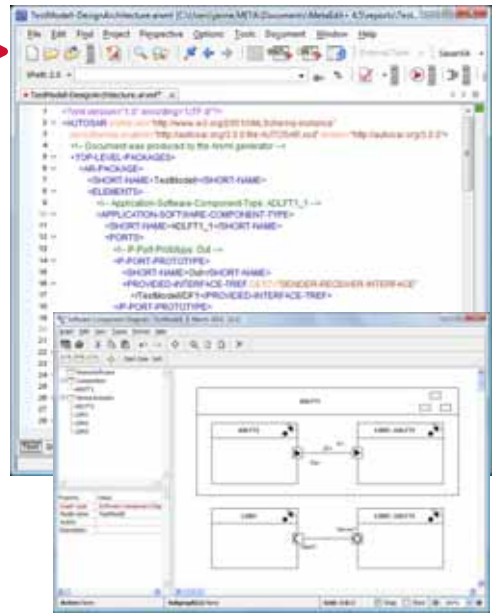
EAST-ADL2 model



Mapping rules



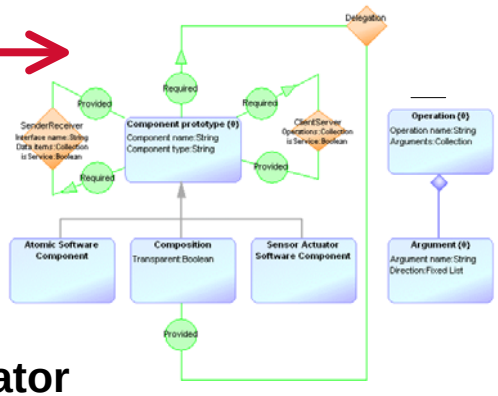
Autosar



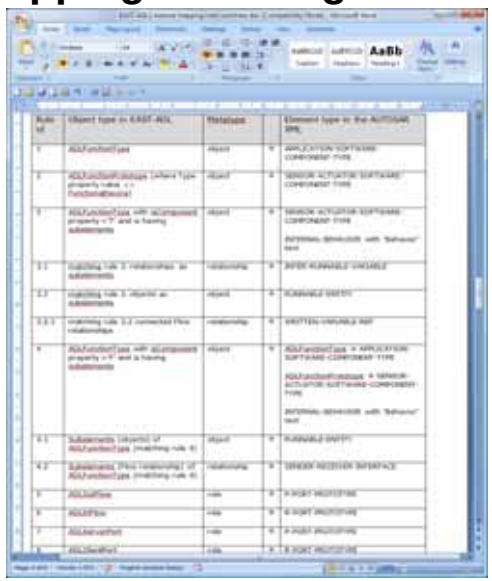
各ADL言語のメタモデル同士で変換を定義

EAST-ADL2 metamodel

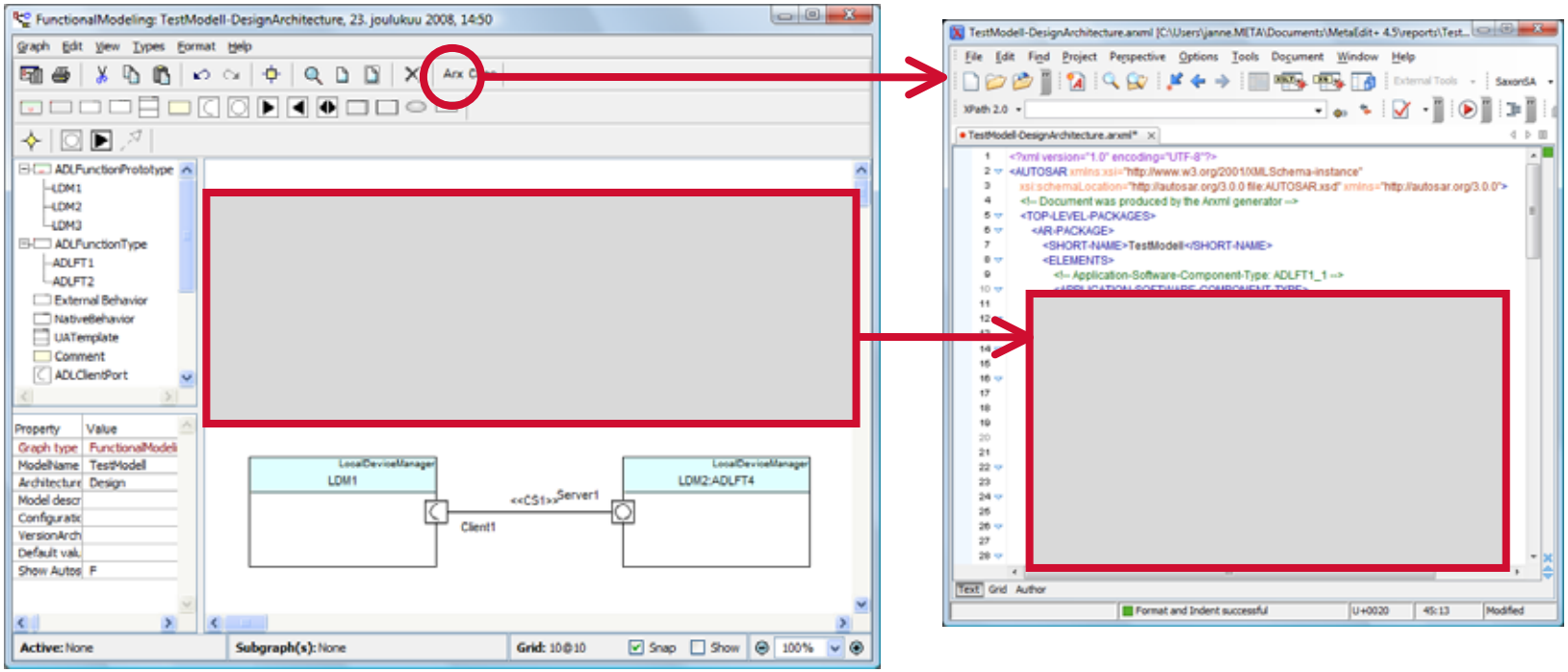
Autosar metamodel



Mapping rules in generator

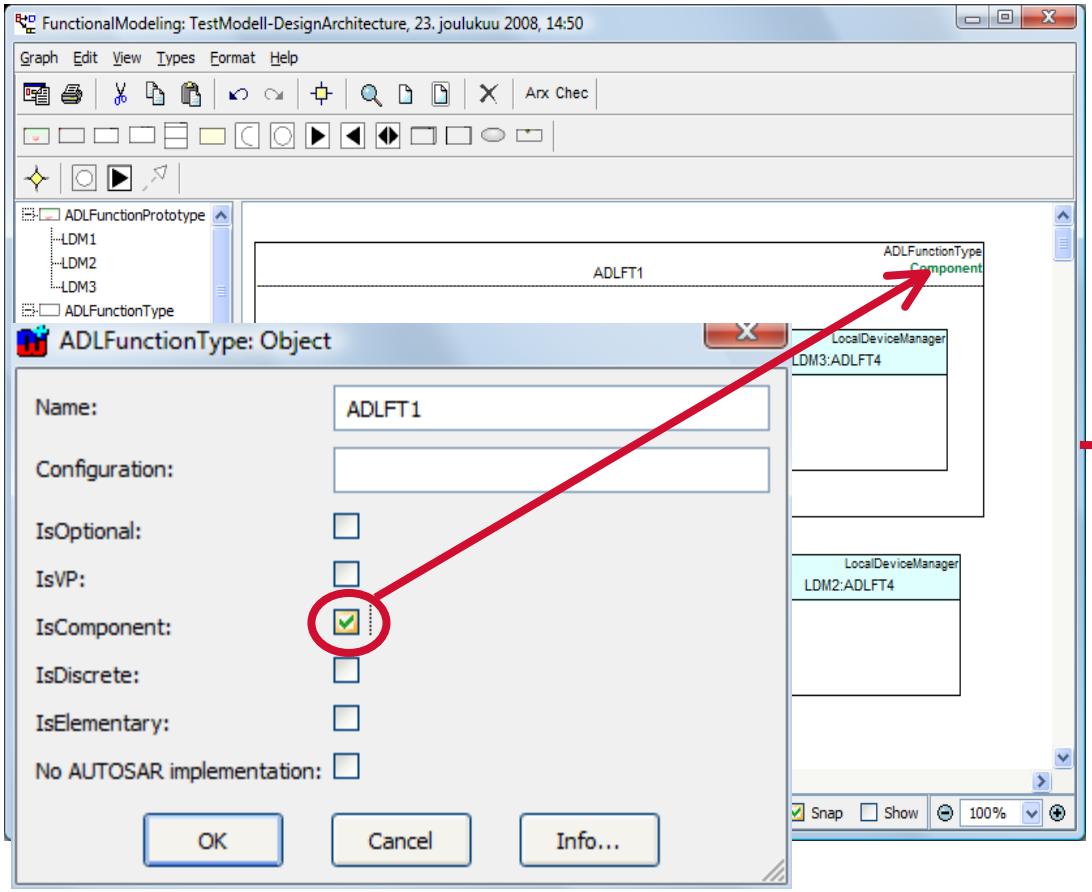


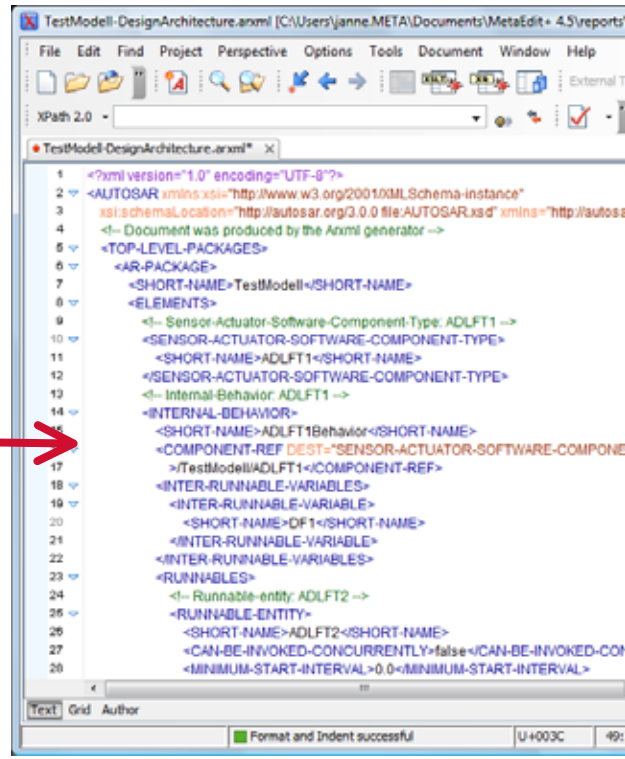
メタモデル同士でマッピング



- 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

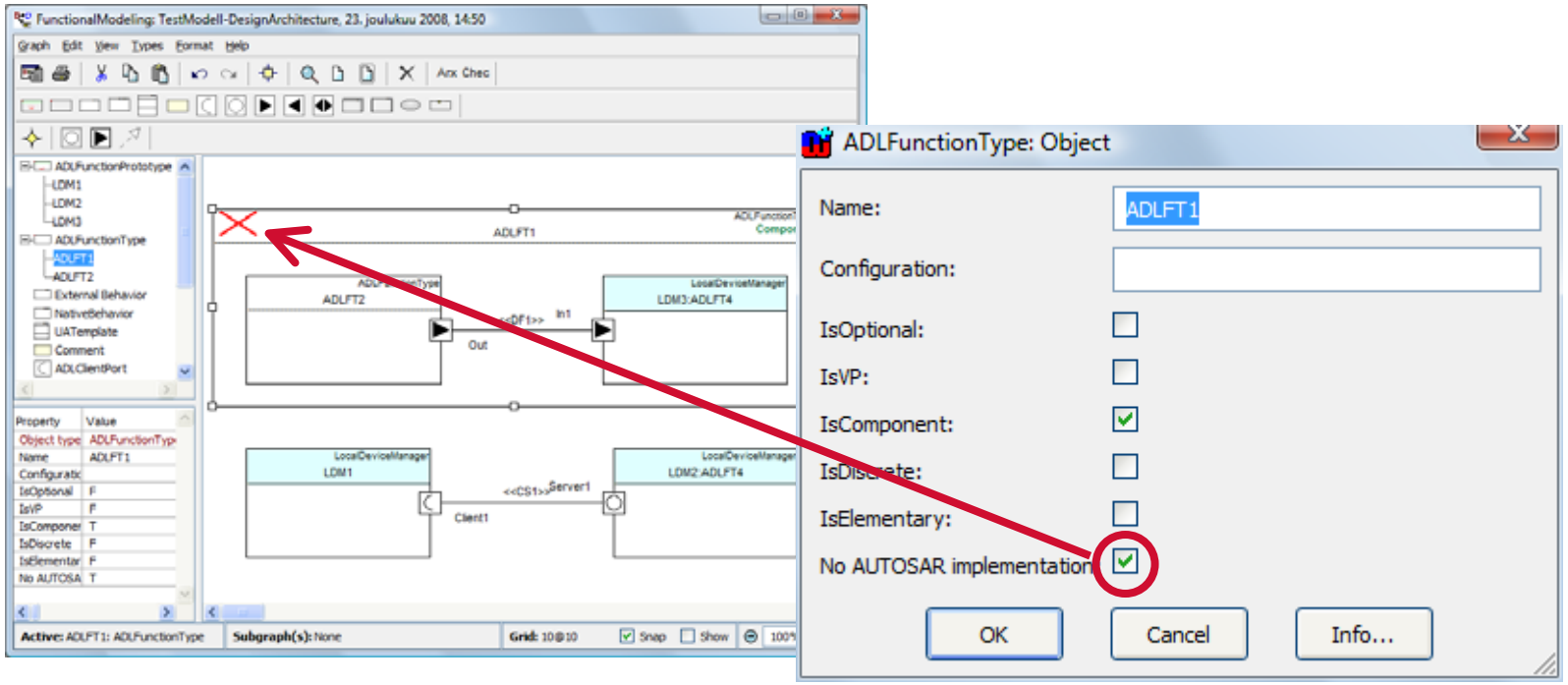
開発者はマッピングルールを選択できる





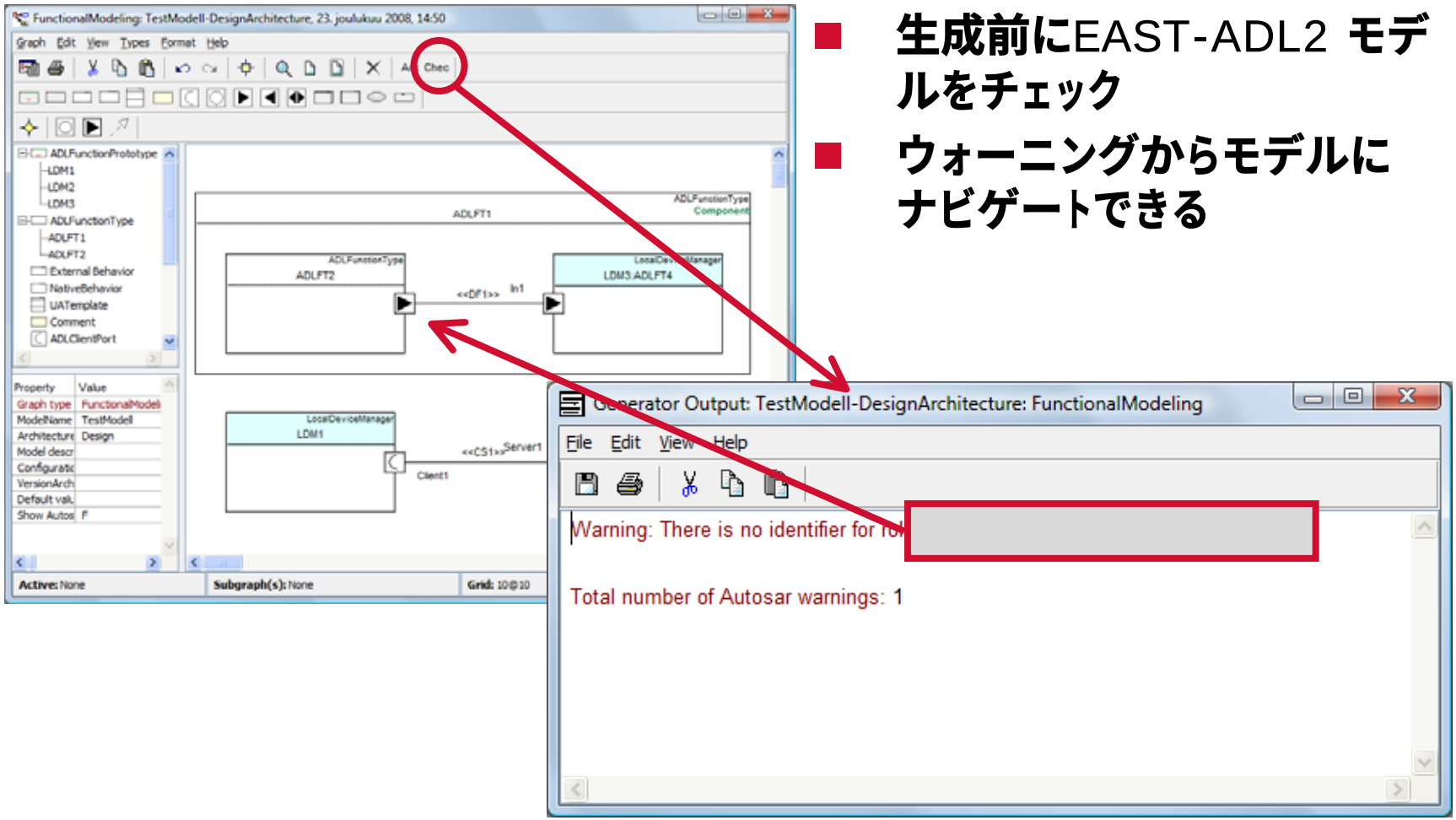
- **モデルへの設定次第で生成されるマッピング情報が変わる**
  - ファンクションをコンポーネントとして定義

AUTOSAR 生成オプション



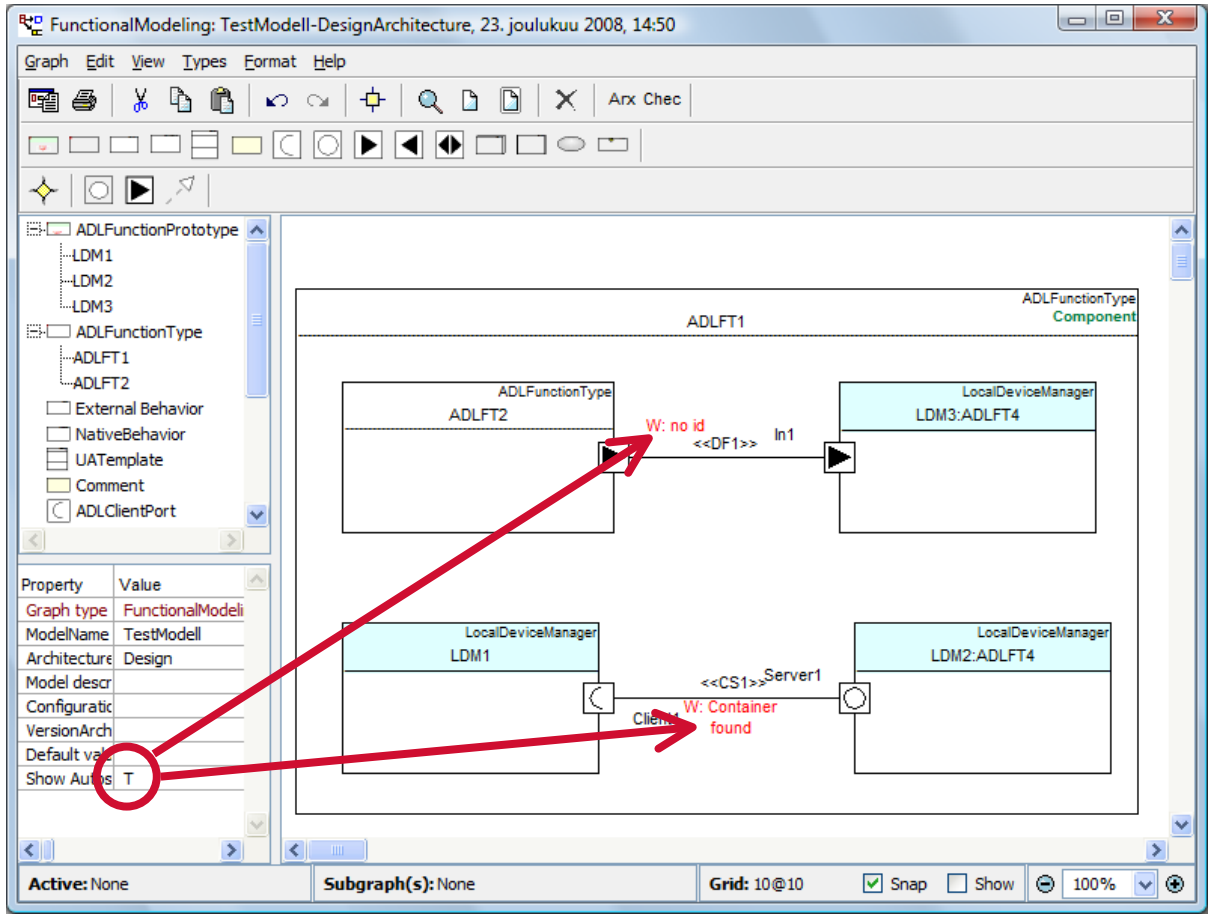
- AUTOSAR **変換に対してエレメントが選択されなければ赤Xがシンボルに表示**
  - ⇒ **エレメントとサブエレメントがAUTOSARに変換されない**

EAST-ADL2 モデルチェック機能



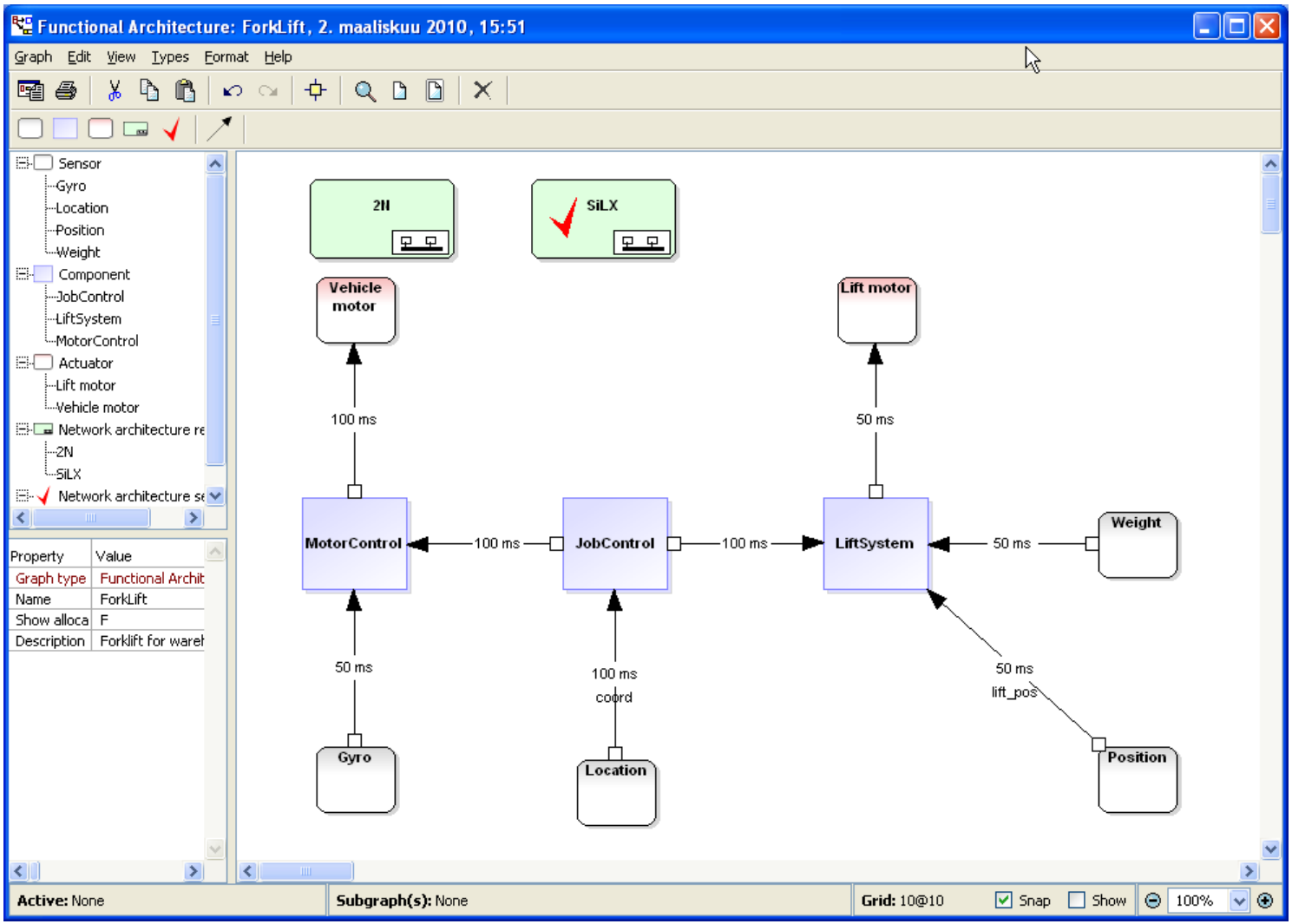
- **生成前にEAST-ADL2 モデルをチェック**
- **ウォーニングからモデルにナビゲートできる**

リアルタイムにモデルへエラーを注釈

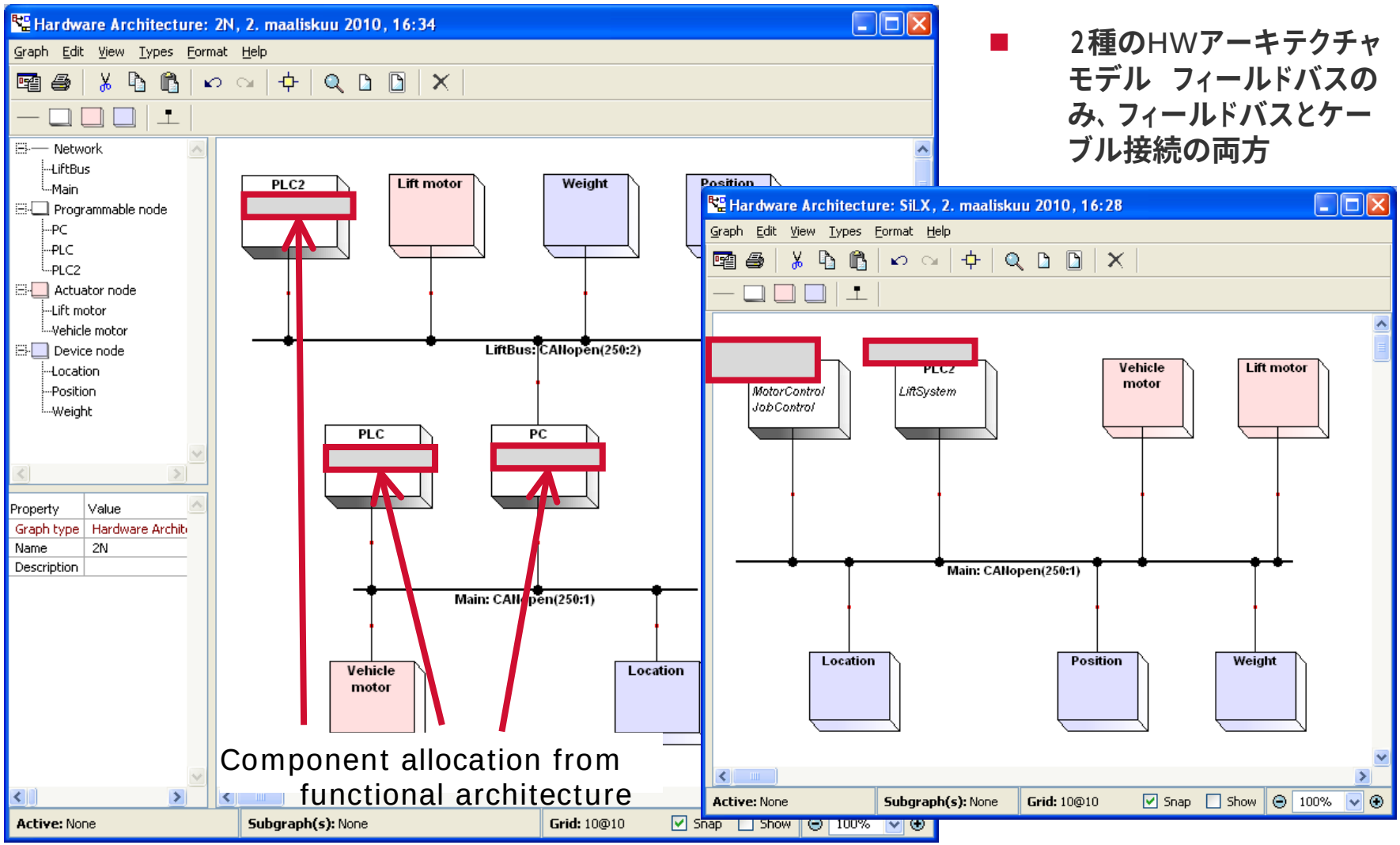


- **エラー情報をモデル上に注釈**
  - **グラフプロパティ：AUTOSARのウォーニングをEAST-ADL2内に**

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

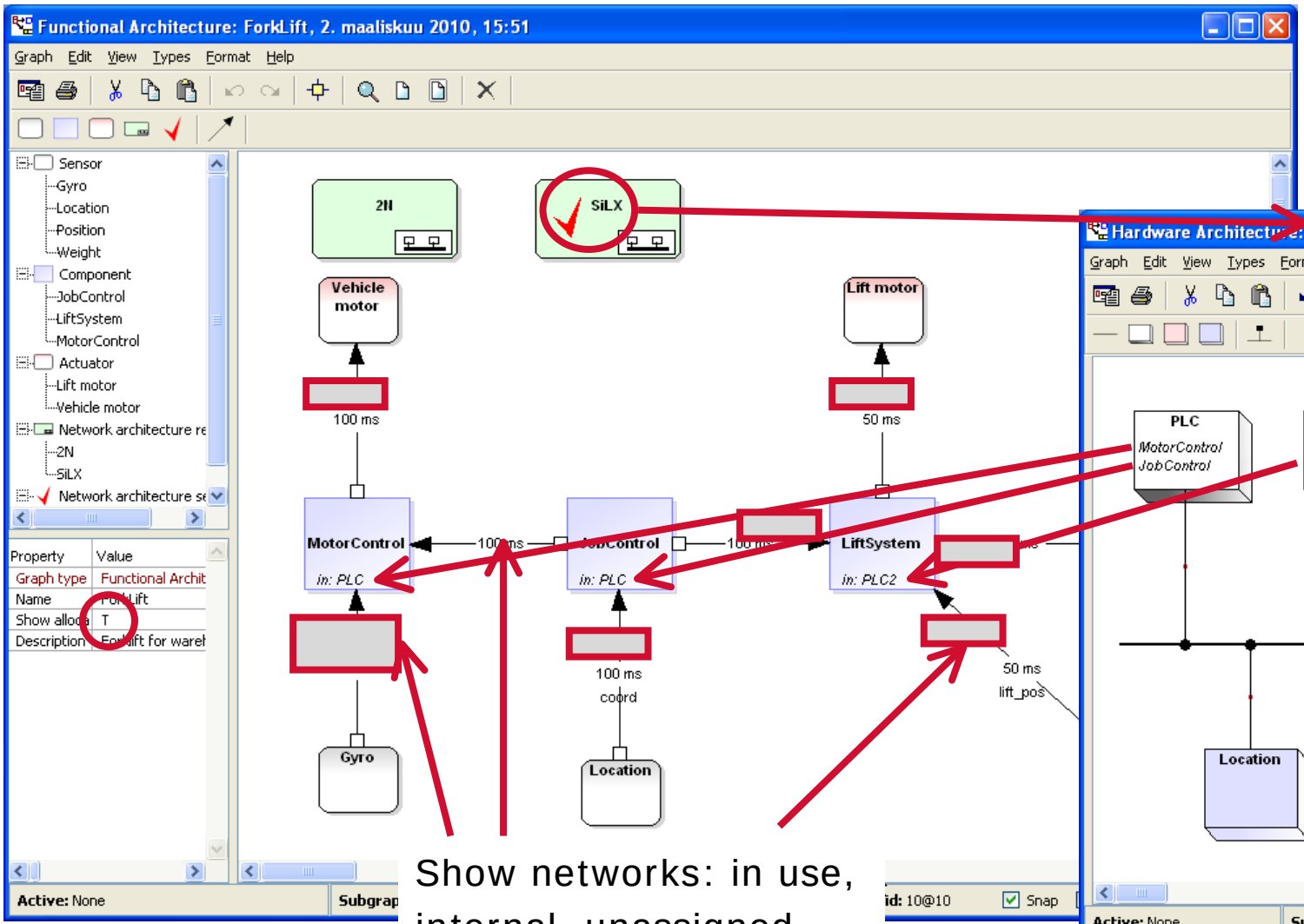


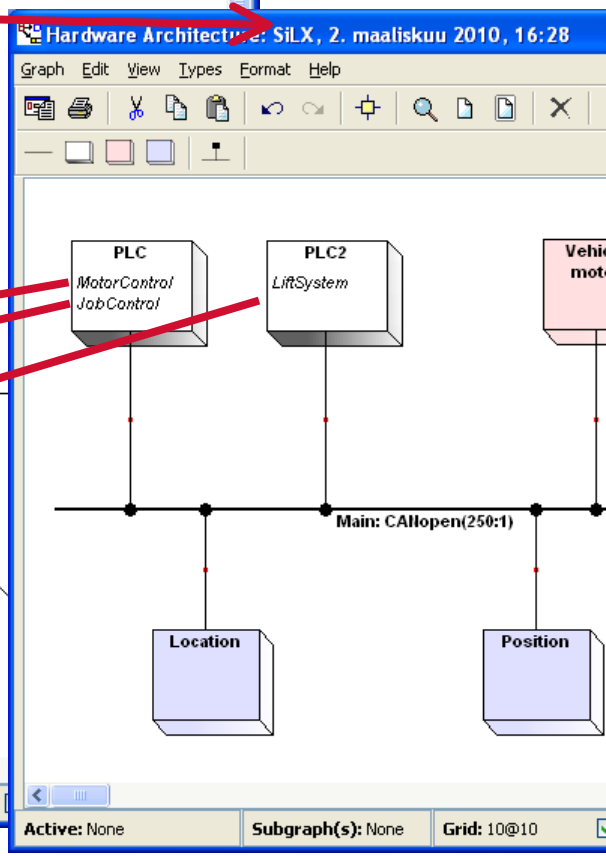
2. メタモデル上で統合例：HWネットワークアーキテクチャモデル図



- 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

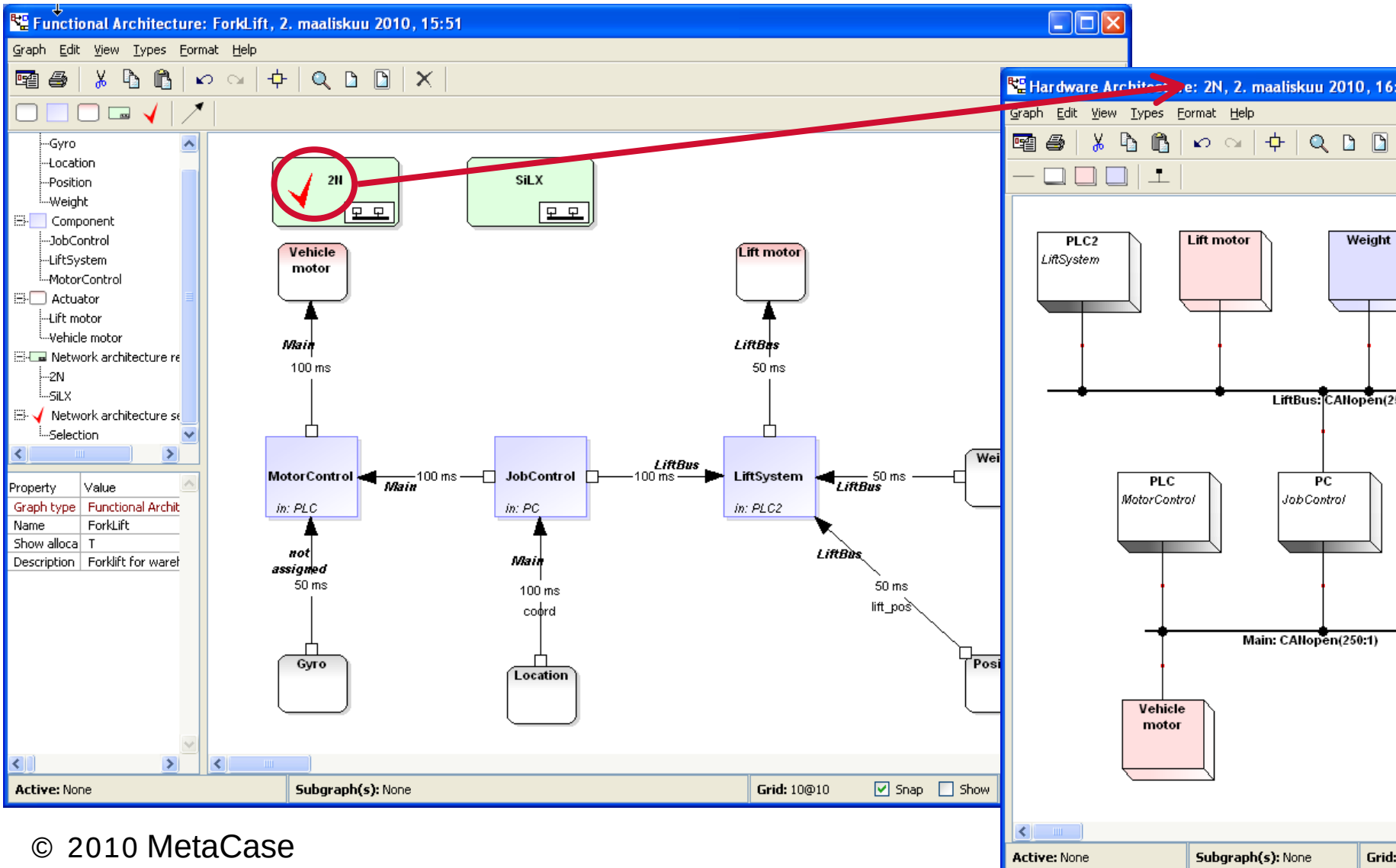
ファンクショナルアーキテクチャモデル図　と
HWネットワークアーキテクチャモデル図の連携





Show networks: in use, internal, unassigned

別のHWネットワークアーキテクチャモデル図



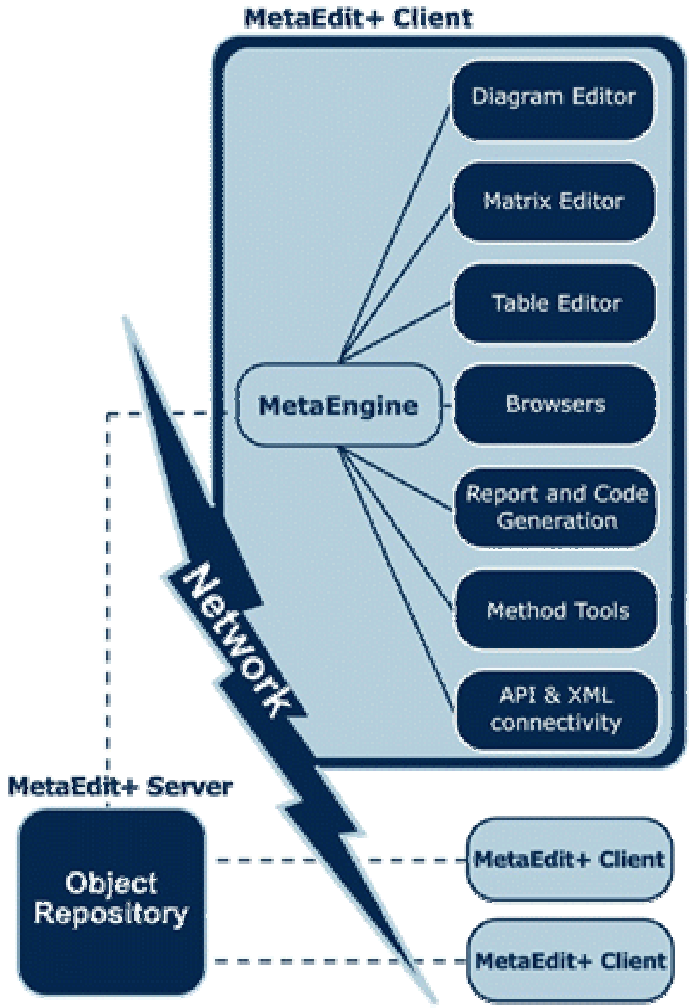
ADLをサポートするツールに必要な機能

- 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

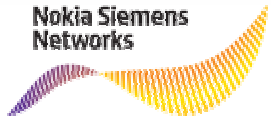
マルチユーザーとマルチプラットフォーム

- 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