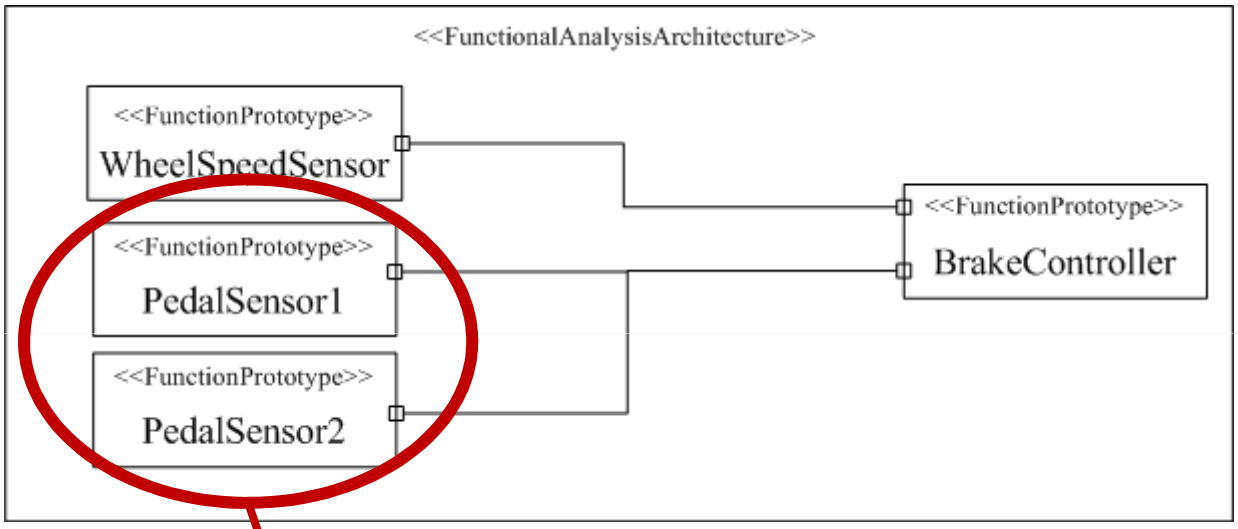
Documenting Stepwise Model Refinement using Executable Design Decisions

Matthias Biehl
Embedded Control Systems
Royal Institute of Technology
Stockholm, Sweden

- What kind of models are considered?
  - Implementation: EMF-based models, independent of metamodel
  - Case study: models of the HW and SW architecture of automotive embedded systems
- What kind of evolution challenges are addressed?
  - Documentation of design decisions for models
  - Linking documentation to the affected model elements
  - Consistency between design decision documentation and model

Example: Documentation of Design Decisions for Models

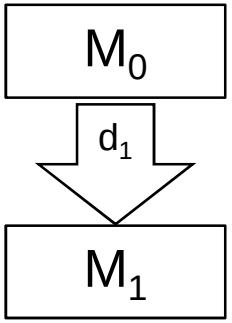


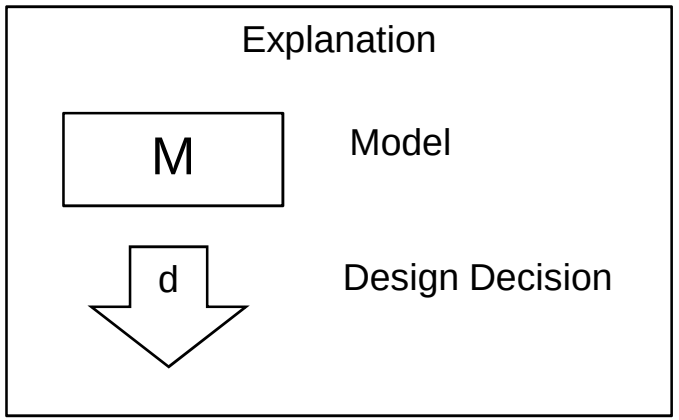
Double redundancy to improve the reliability of a safety critical component

Requirements for Documentation of Design Decisions for Models

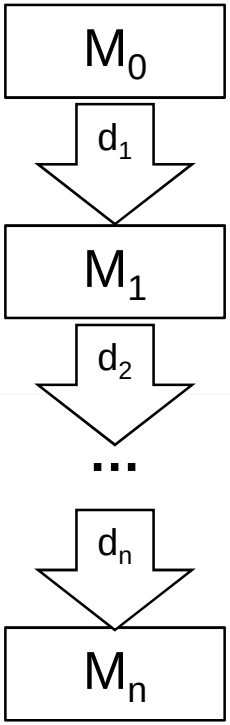
- **Explicit documentation** of the design decision, for example by a textual description.
- Applicable for an **arbitrary metamodel**: it should not be necessary to change existing metamodels to document design decisions.
- **Link** between the design decision documentation and the model: a link between the design decision documentation, the context of the design decision and the model elements affected by the design decision
- **Low capture overhead**: the effort for linking to the documentation to the model should be low.
- **Consistency** between the design decision documentation and the model: the change described by the design decision should be actually be realized in the model.

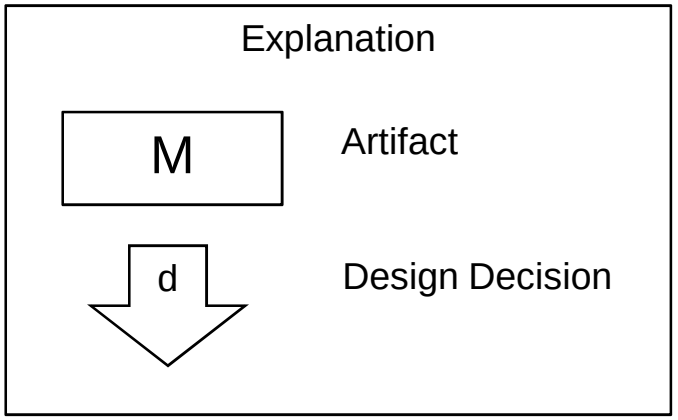
Stepwise Refinement





Lack of Design Decision Documentation

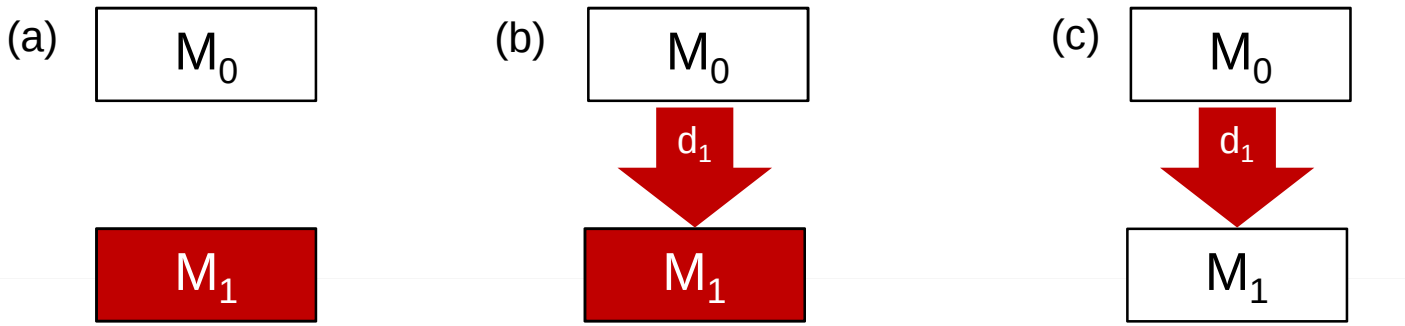




Design Decision Capture Problem

- Additional cost and effort for documenting design decisions, no immediate benefit [10]
- Documentation is not always performed in practice [18]
- Knowledge vaporization
- Model inconsistencies, erosion, aging, architectural drift
- Problems during maintenance

How to deal with the design decision capture problem?



artifact based,
e.g. SVN, CVS

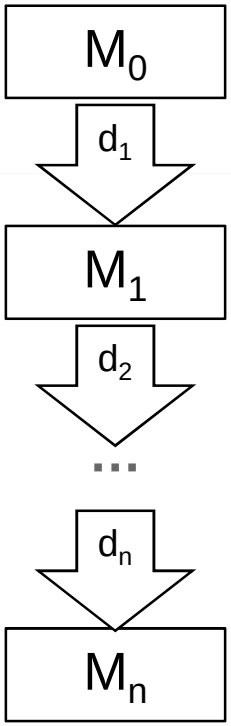
artifact based
+ design decision
documentation

executable
design decisions

Red = needs to be specified and changed manually

Formalization: Stepwise Model Refinement with Executable Design Decisions

- Design Decisions as executable specification is both
  - descriptive documentation
  - constructive tool for creation of the artifact

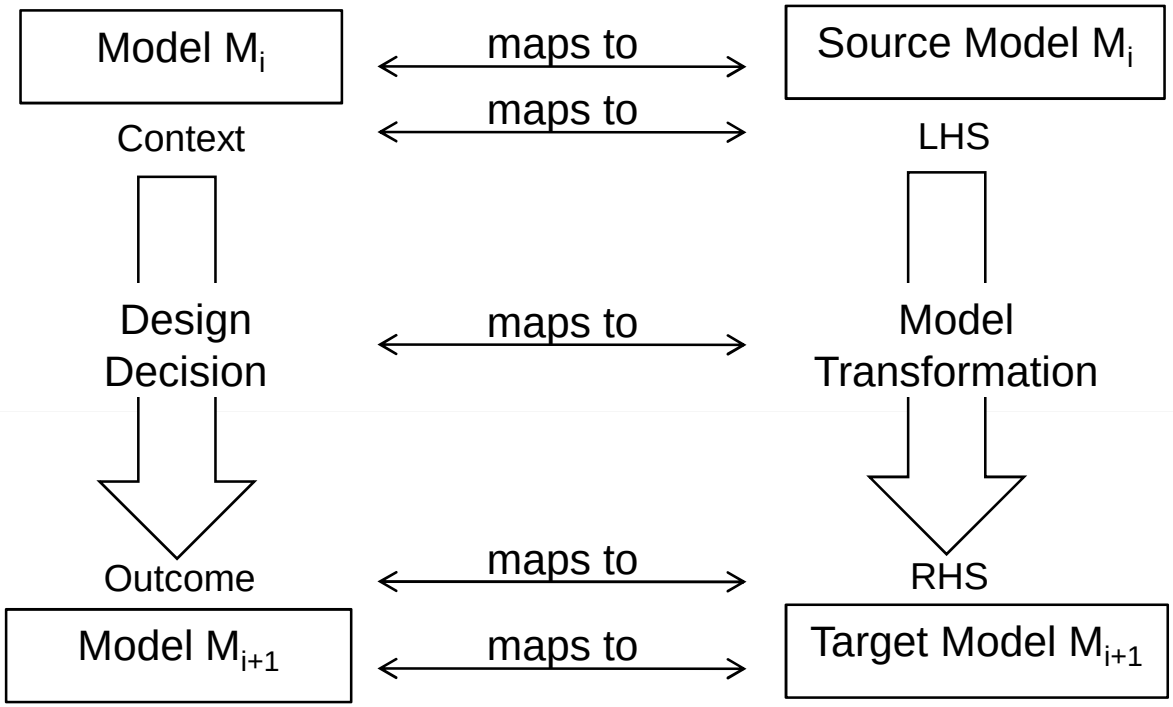


M_0 initial architecture, can be empty

$M_1 = d_1(M_0)$

$M_n = (d_n \circ d_{n-1} \circ \ldots \circ d_1)(M_0)$

Mapping between Design Decisions and Model Transformations



- Requirements for Model Transformation Language/Engine
  - **Model-to-Model**: input and output of the transformation are models
  - **Endogeneous:** metamodels of source and target are identical
  - **In-Place**: create the target model by modifying specific parts in the source model

Which part of the design decision can be described by a Model Transformation?

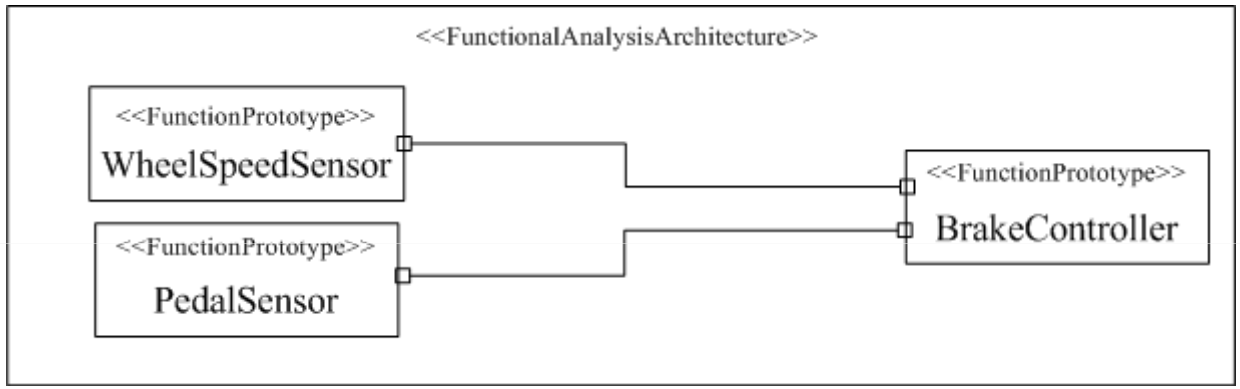
- Basis: Design Decision Ontology [9]
  - Context/Scope, Outcome
  - Author, Timestamp, Risk, Cost, Design Decision Rationale
  - Relation between design decisions
- Model Transformation
  - Context/Scope, Outcome, Trace link to the model elements
- Additional Information: Design Decision Management Container
  - Author, Timestamp, Risk, Cost, Design Decision Rationale
  - Relation between design decisions
  - Link to the Model Transformation

Capturing Design Decisions

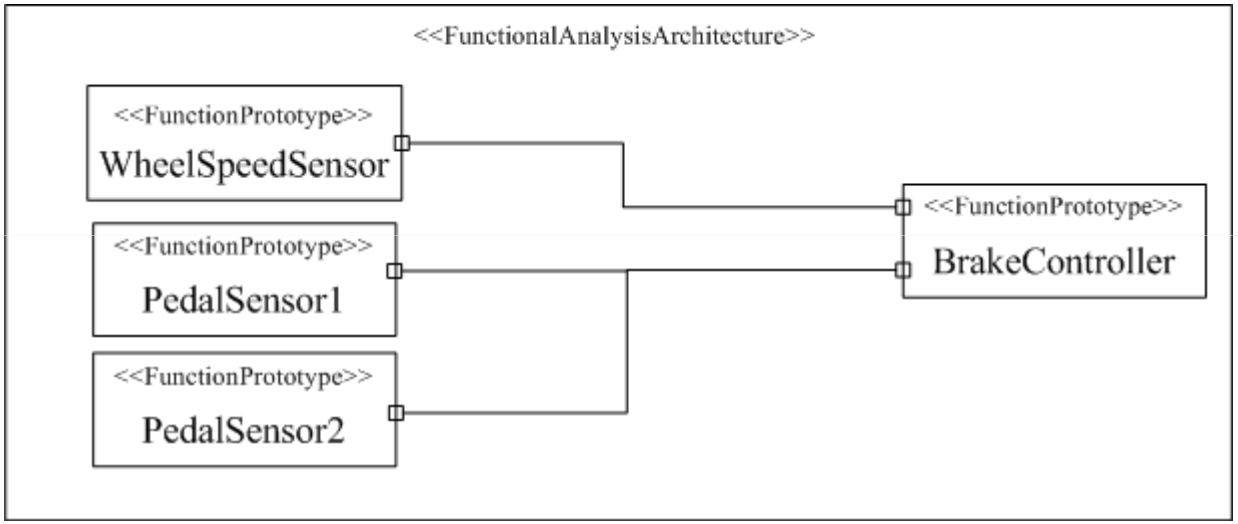
- Steps for manual capturing
  1. Performing an **update of the model**, so it reflects the new design decision
  2. Documenting **where** in the model the change applies and **what** is changed
  3. L**inking** the **documentation** to the **model elements** affected by the change
  4. Documenting the **rationale of the change**

- Executable Design Decisions use intrinsic relation of steps 1, 2 and 3
  1. Update of the model = outcome of executing the Executable Design Decision
  2. Where and What = LHS and RHS of Executable Design Decision
  3. Traces of Executable Design Decision link the design decision to the model

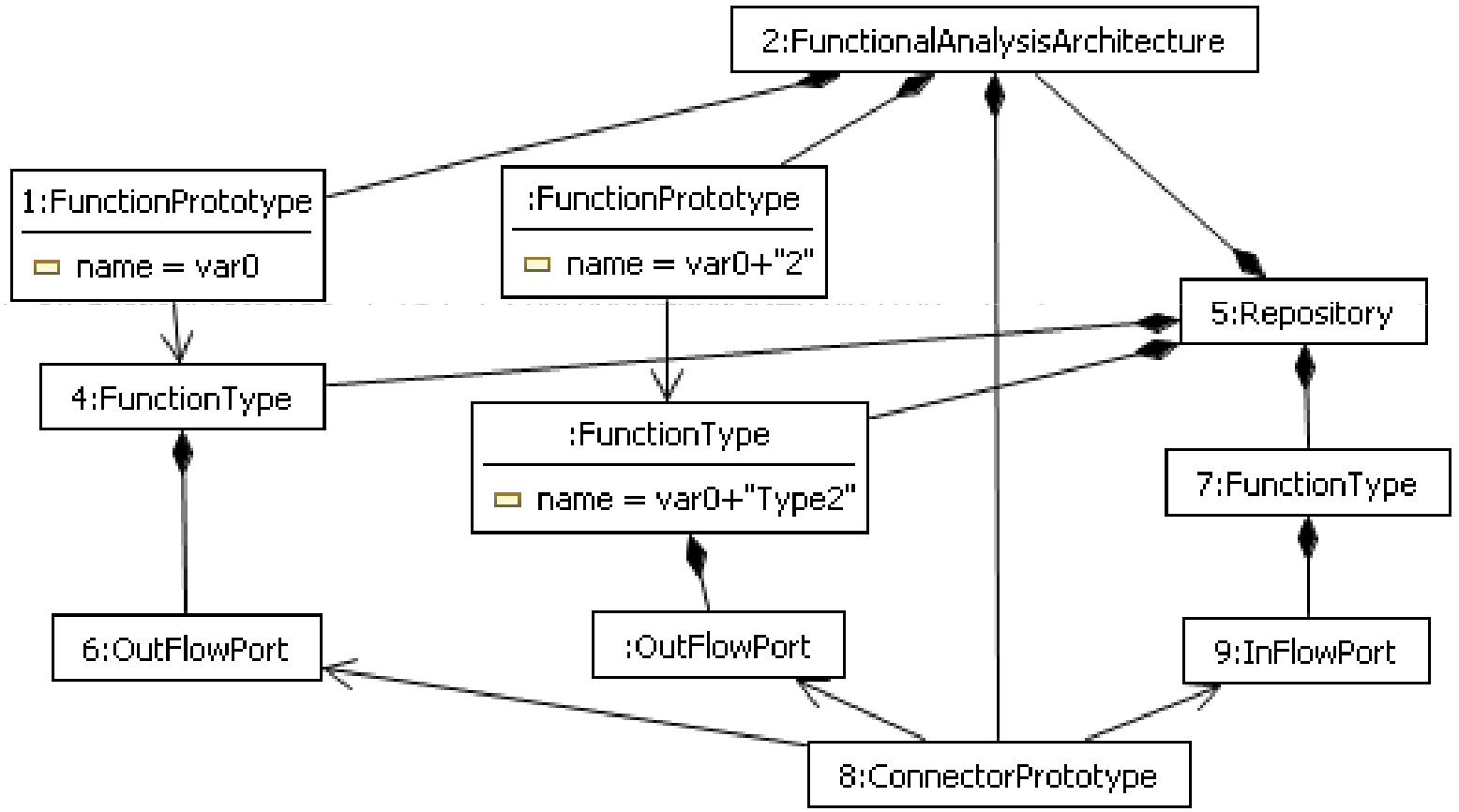
Simplified Brake-by-wire Model



Brake-by-wire Architecture with Double Redundancy



Design Decision Documentation using Triple Graph Grammars (TGG)

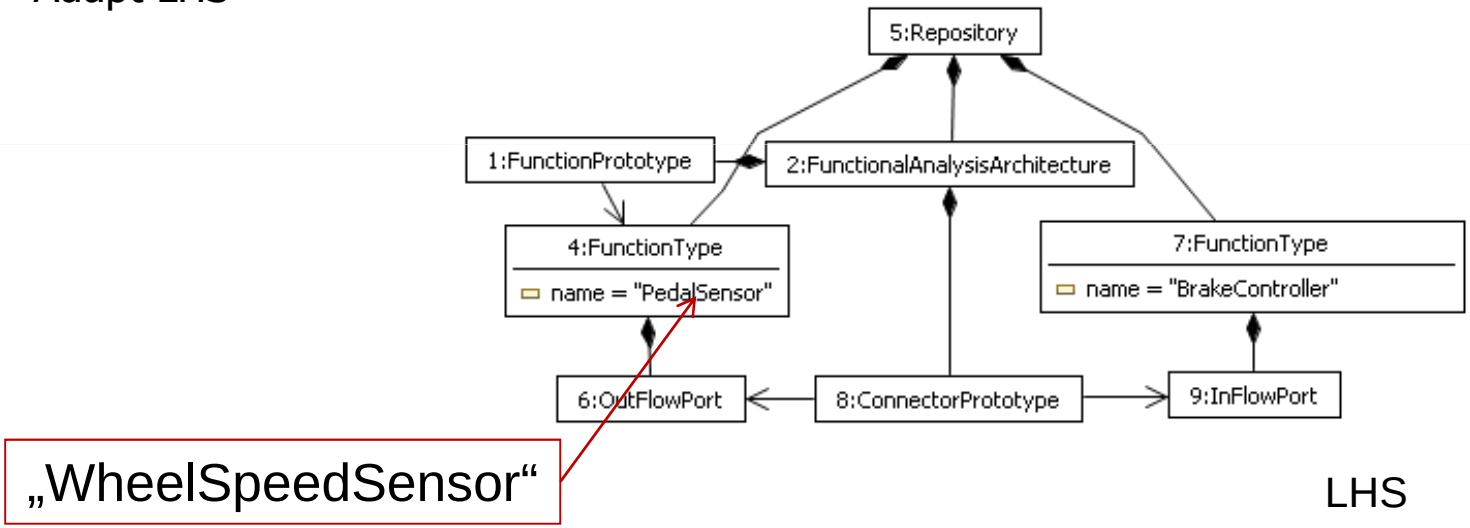


Design Decision Patterns

- Executable design decisions are reusable
- Reuse design decision in a different context
  - adapt LHS
- Change the outcome of the design decision
  - adapt RHS

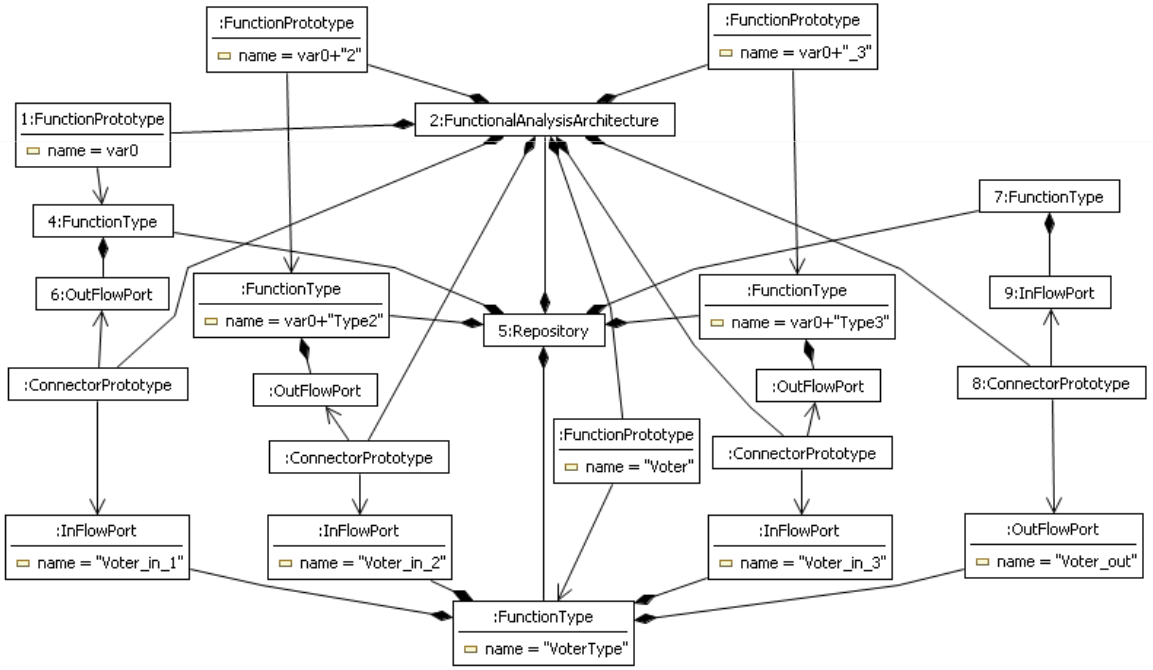
Design Decision Patterns:
Reuse design decision in a different context

- Apply design decision on WheelSpeedSensor instead of PedalSensor
  - Same RHS
  - Adapt LHS



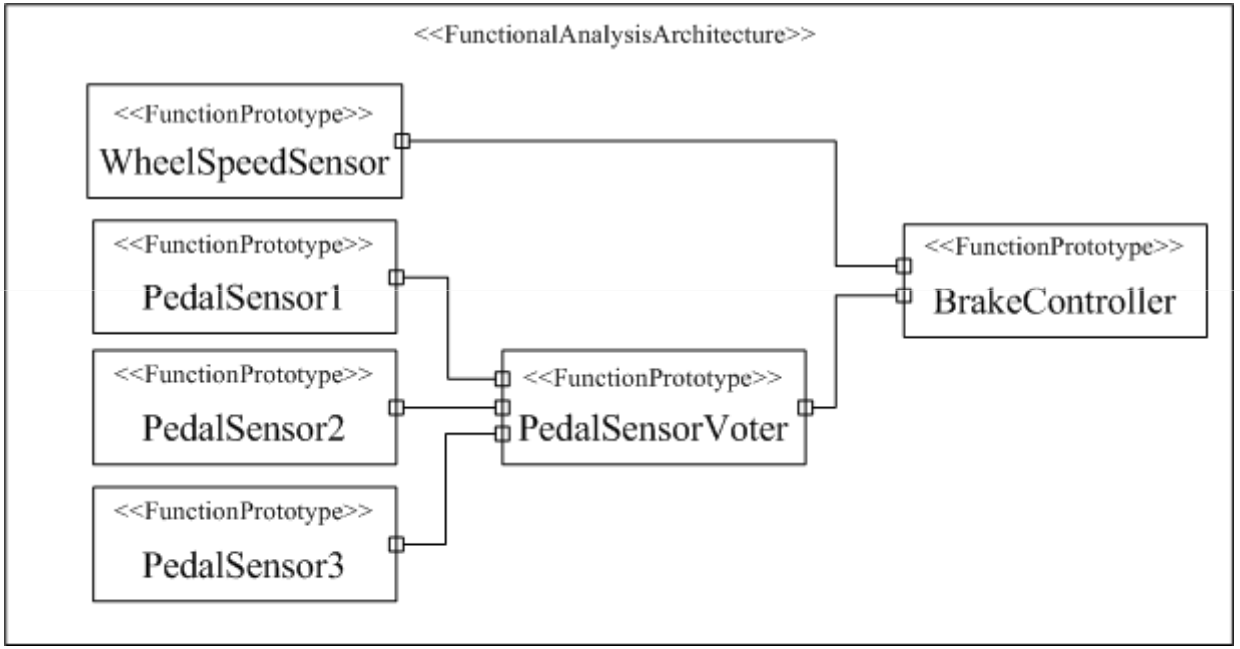
Design Decision Patterns:
Change the outcome of the design decision

- Design decision: use triple redundancy + voter
  - Same LHS as for double redundancy
  - Adapt RHS



New RHS

Brake-by-wire Architecture with Triple Redundancy + Voter instead of Double Redundancy



Summary: Documenting Model Refinement

- Problem: Double effort required for capturing design decisions
  - Both the architecture needs to be adapted
  - and the design decision needs to be documented
- Representation: Executable Design Decision
  - Executable specification using model transformations
  - Additional design decision rationale and „bookkeeping" data
- Result
  - Reduction of **capture overhead**: context and outcome calulated automatically
  - Applicability with **any metamodel**
  - **Consistency** between model and documentation through **automation**
  - **Link** between documentation and model through automated **traces**