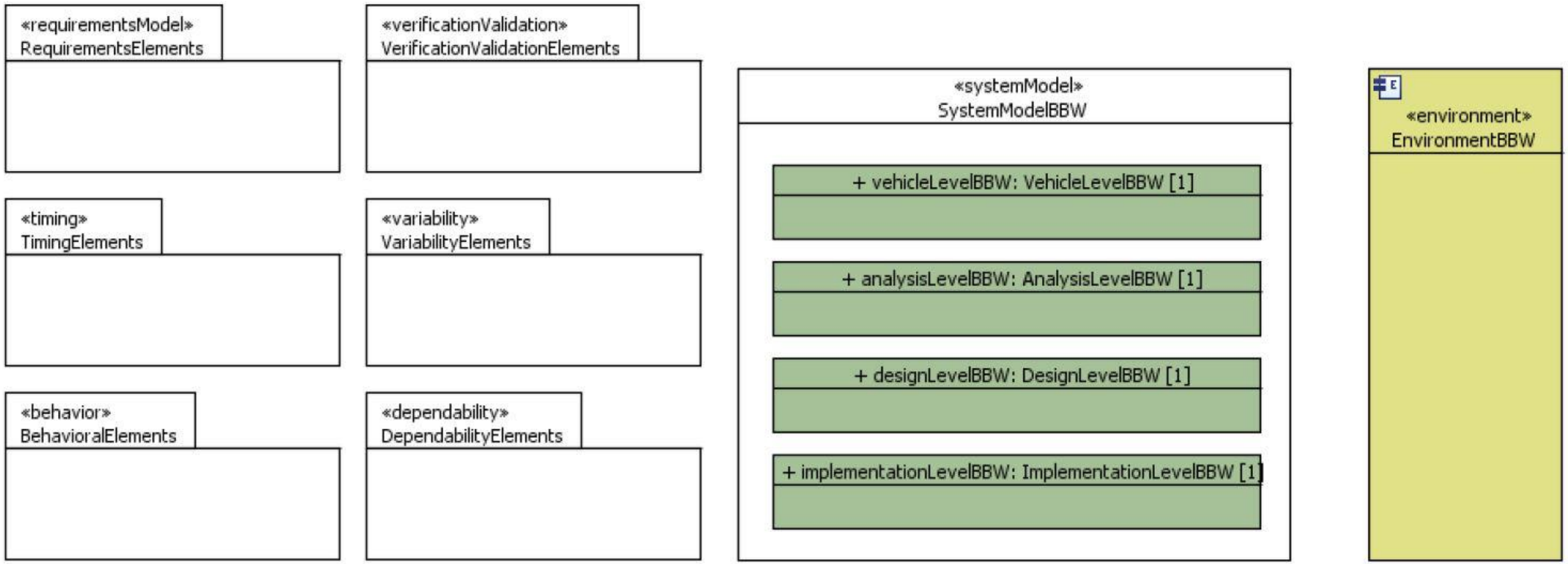
Role of Structural Model

- FunctionalAnalysisArchitecture and FunctionalDesignArchitecture represent functional structure and behavior of embedded system
- Functions in EnvironmentModel represent functional structure and behavior of plant/environment

Role of Structural Model

- Execution semantics is synchronous
  - read-execute-write
- Ports have single-buffer-overwrite semantics
- Trigger Characteristic is defined by
  - FunctionTrigger
  - Timing Constraints on EventFunction
- Transfer function from input to output is contained in FunctionBehavior

Figure 9: The braking electrical/electronic system and its environment in Papyrus.


«requirementsModel»
RequirementsElements
«verificationValidation»
VerificationValidationElements
«timing»
TimingElements
«variability»
VariabilityElements
«behavior»
BehavioralElements
«dependability»
DependabilityElements
«systemModel»
SystemModelBBW
+ vehicleLevelBBW: VehicleLevelBBW [1]
+ analysisLevelBBW: AnalysisLevelBBW [1]
+ designLevelBBW: DesignLevelBBW [1]
+ implementationLevelBBW: ImplementationLevelBBW [1]
«environment»
EnvironmentBBW

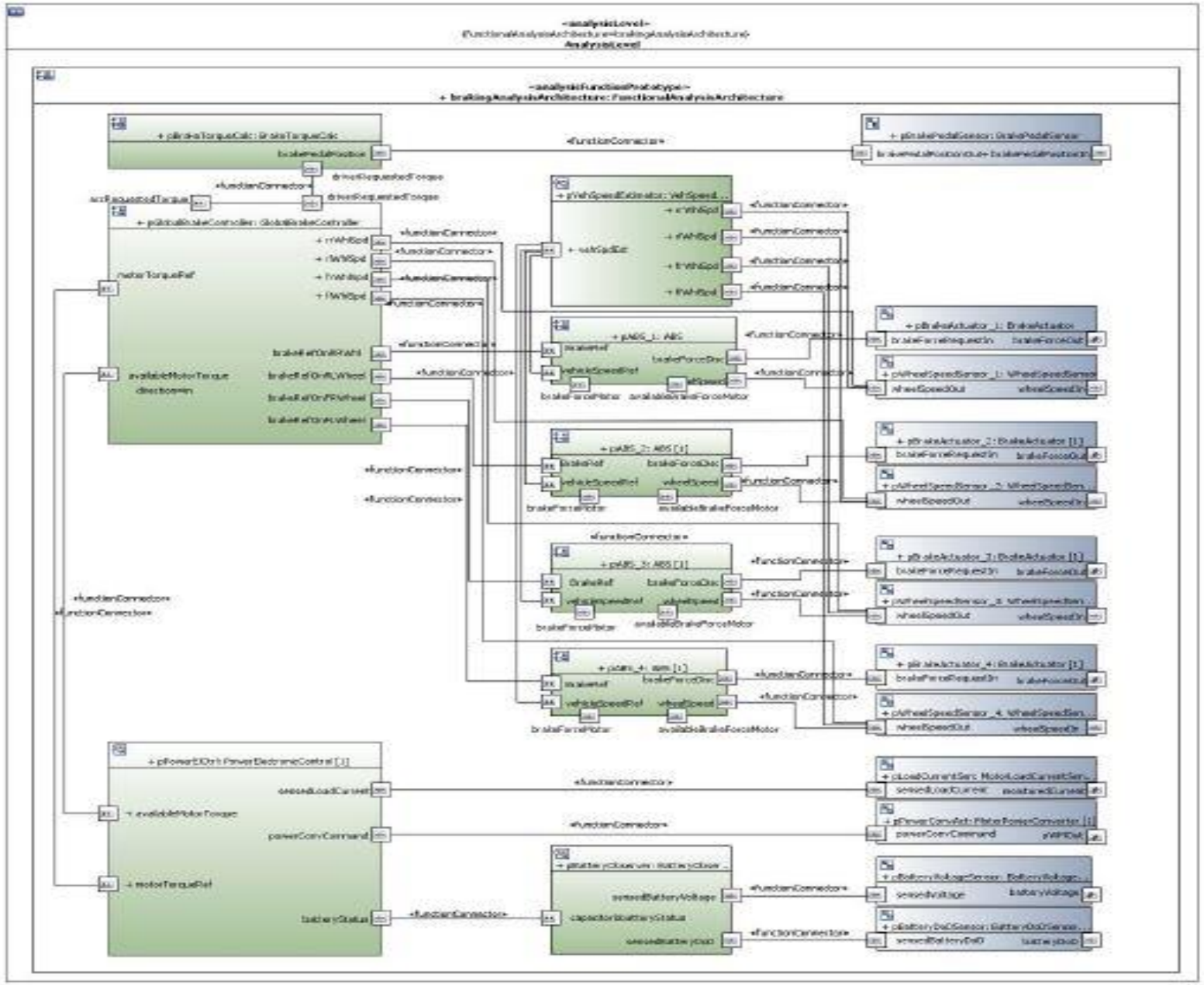
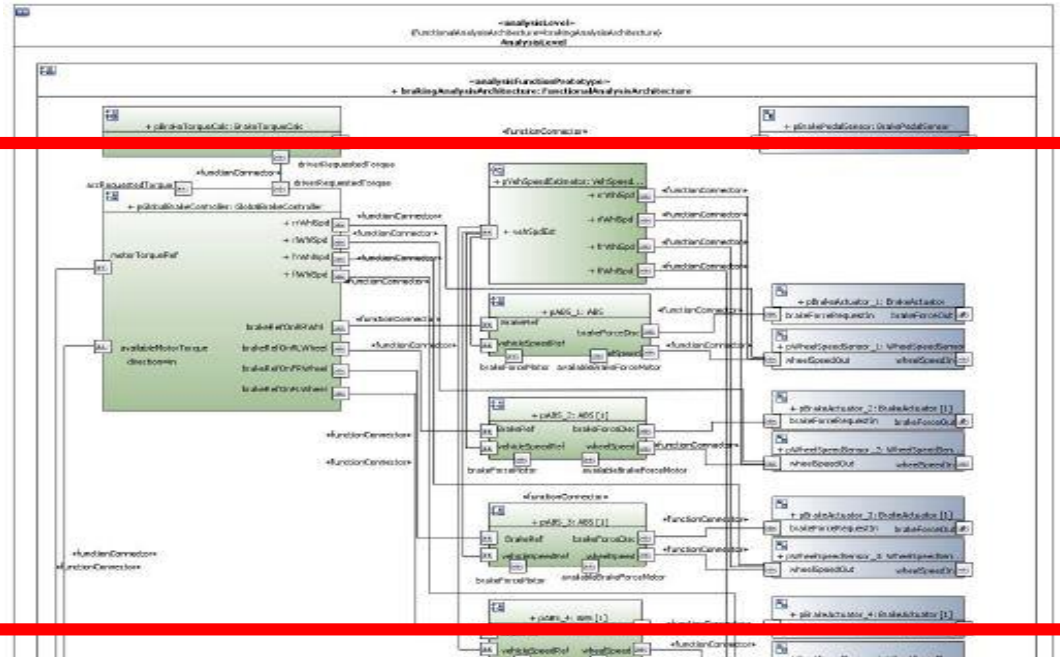


Figure 16: Functional Analysis Architecture specification of the Regenerative Braking System in Papyrus.



«analysisLevel»
{functionalAnalysisArchitecture=brakingAnalysisArchitecture}
AnalysisLevel

«analysisFunctionPrototype»
+ brakingAnalysisArchitecture: FunctionalAnalysisArchitecture

+ pBrakeTorqueCalc: BrakeTorqueCalc

brakePedalPosition

«functionConnector»

+ pBrakePedalSensor: BrakePedalSensor

brakePedalPositionOut+ brakePedalPositionIn

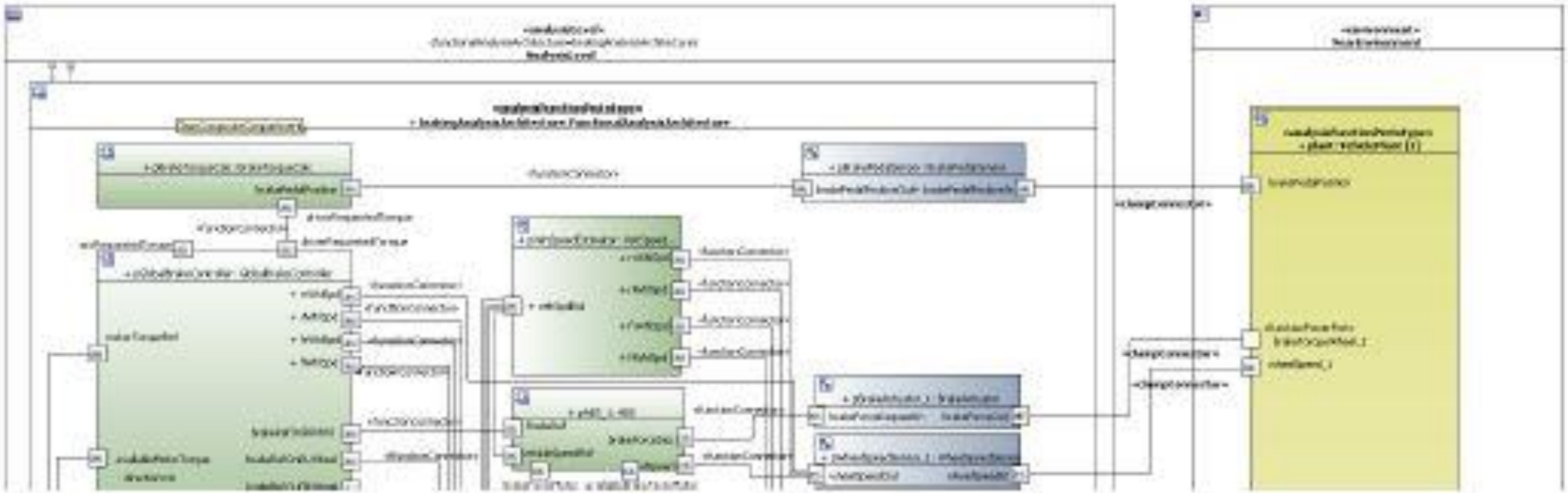


Figure 17: Connecting Analysis Functions with Environment in Papyrus.

Defining Behavior

- FunctionBehavior defines transfer function from input to output
- FunctionBehavior is invoked each time the Function is trigged.
- FunctionBehavior can be defined in any tool and notation provided it
  - Uses input data corresponding to input ports
  - Provides output data corresponding to output ports
  - Respects run-to-completion semantics

FunctionBehavior

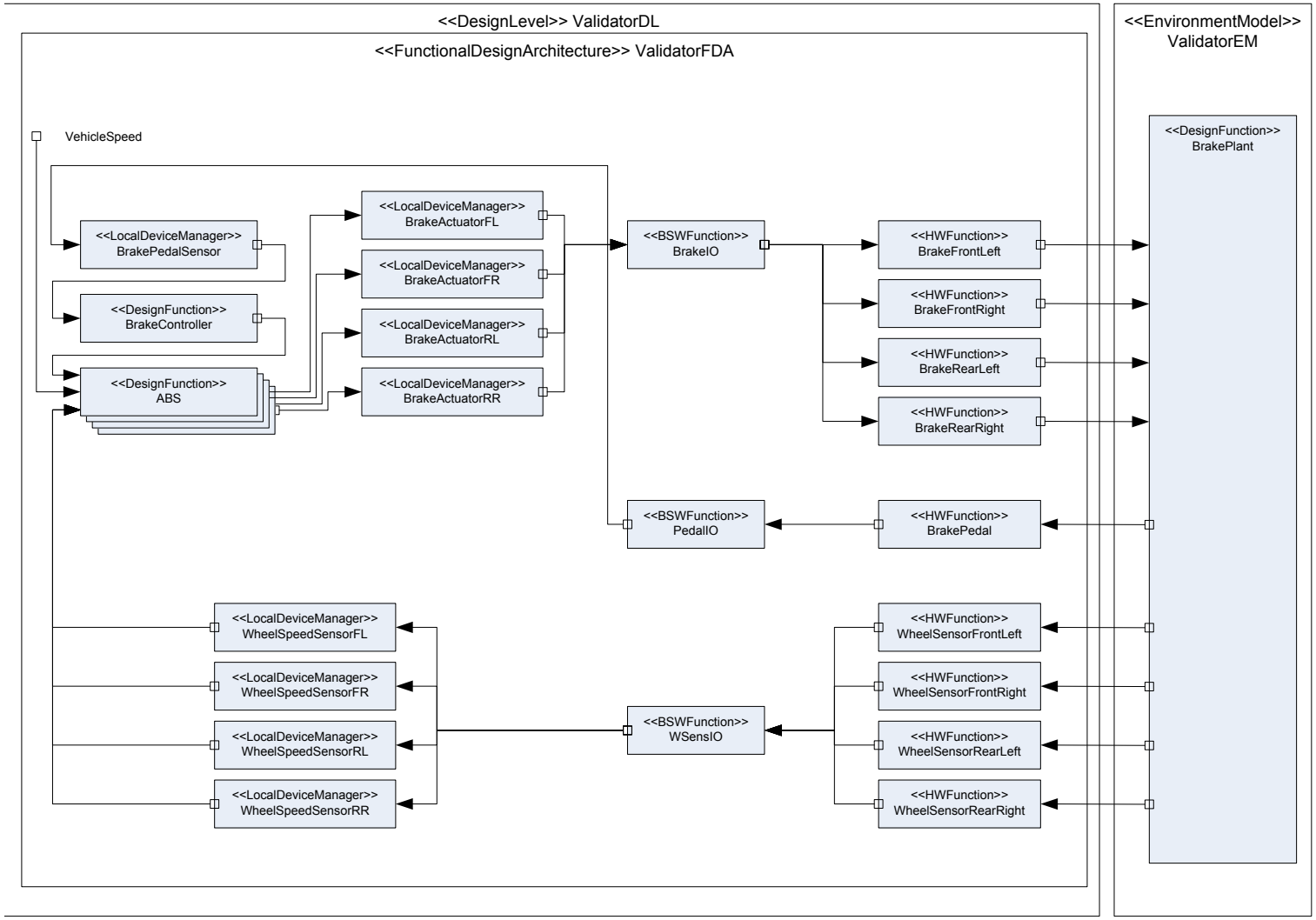
- FunctionBehavior-Path identifies external model that captures transfer function
- FunctionBehavior-Representation declares external notation

Behavioral Semantics

The rules explained until now assumes functions are elementary.

For composite functions, FunctionBehavior represent the (intended) composed behavior of the contained set of elementary functions (their triggering, transfer functions, data flow, etc.)

Example – Functional architecture



Example – Functional architecture

- FunctionBehavior may be defined for each FunctionType
- Simulink/Scade/Ascete/StateCharts… may define the behavior of each FunctionBehavior
- A combined behavior would compose all behaviors of each function according to the
  - Behavioral model defining the transfer functions
  - The timing constraints and triggering defining when each funcion executes
  - The connectors that define data and control flow

Behavioral Tools

- Used for ”black-box” behavioural definition of leaf functions.
- Tool interfaces should be able to compose hierarchical models for tool, based on individual leaf function behaviors and the triggering and dataflow defined in EAST-ADL
- Tool interfaces should be able to compose hierarchical EAST-ADL models based on model hierarchy in tool. Behaviors of leaf EAST-ADL functions should remain in the external tool
- Tools may generate code for simulation purposes or for use on implementation level.

Summary of Behavioral Semantics

- Functional Architecture Capture Structure
- Functions have synchronous execution semantics
- Function ports have single-buffer-overwrite sementics
- Functions can be annotated with timing and triggering
- FunctionBehavior can be linked to Function to define transfer function
- FunctionBehaviors have run-to-completion semantics

Summary of Tool Integration

- EAST-ADL behavioral semantics is defined to allow different tools and notations to interoperate
  - Provided EAST-ADL behavioral semantics is respected.
- Exchange with behavioural tools can preserve model hierarchy and behavioural definitions of leaf functions.

Note

- Exercise: Simulink and Stateflow combined examples

  http://www.mathworks.com/help/simulink/examples/power-window-control-project.html

- 04/20 and 04/27: 20 mins presentation in English

- Score: 20% out of 40% (Team work)