EAST-ADL Introduction

EAST-ADL
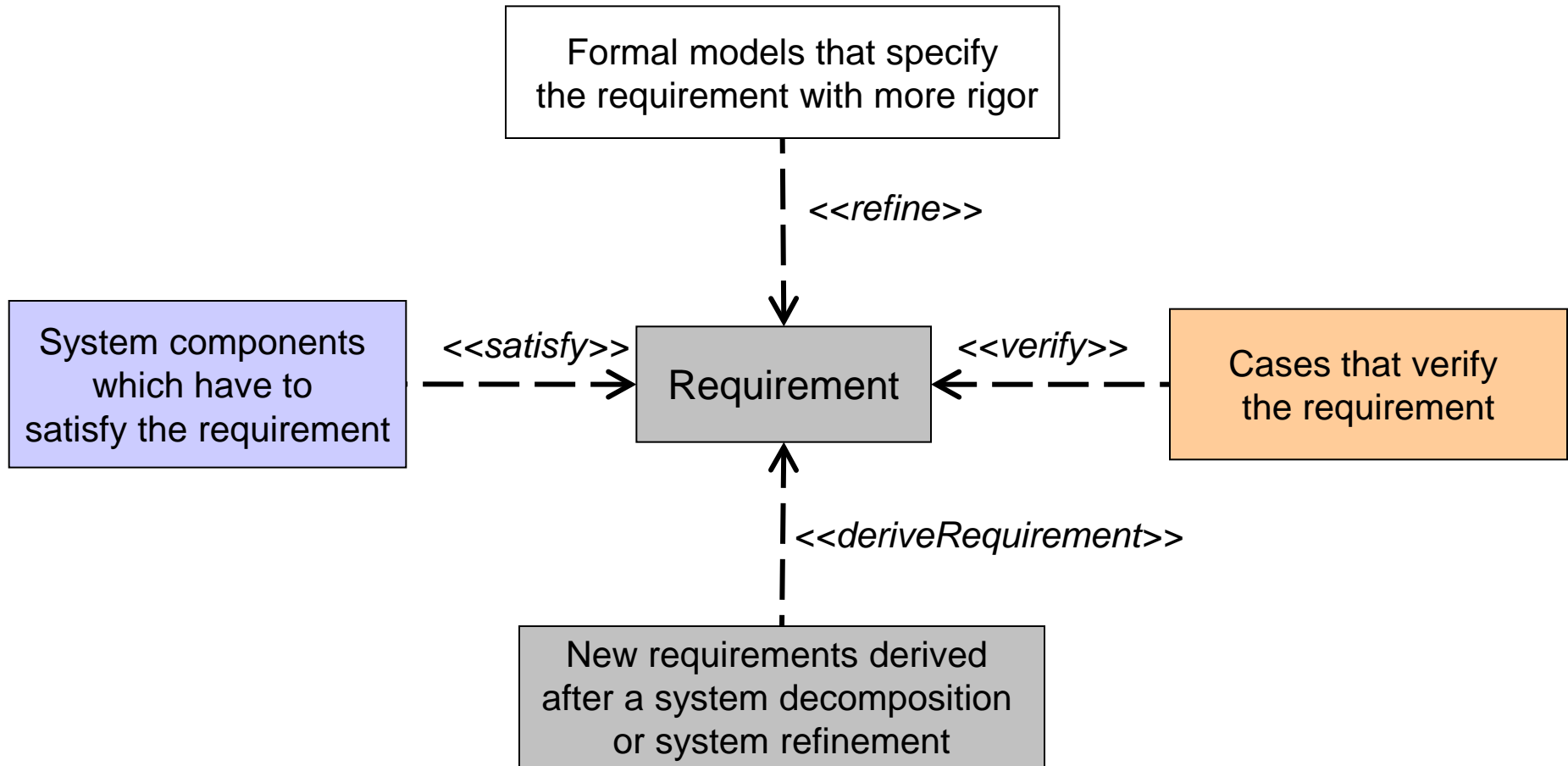Requirements, Constraints and Verification & Validation
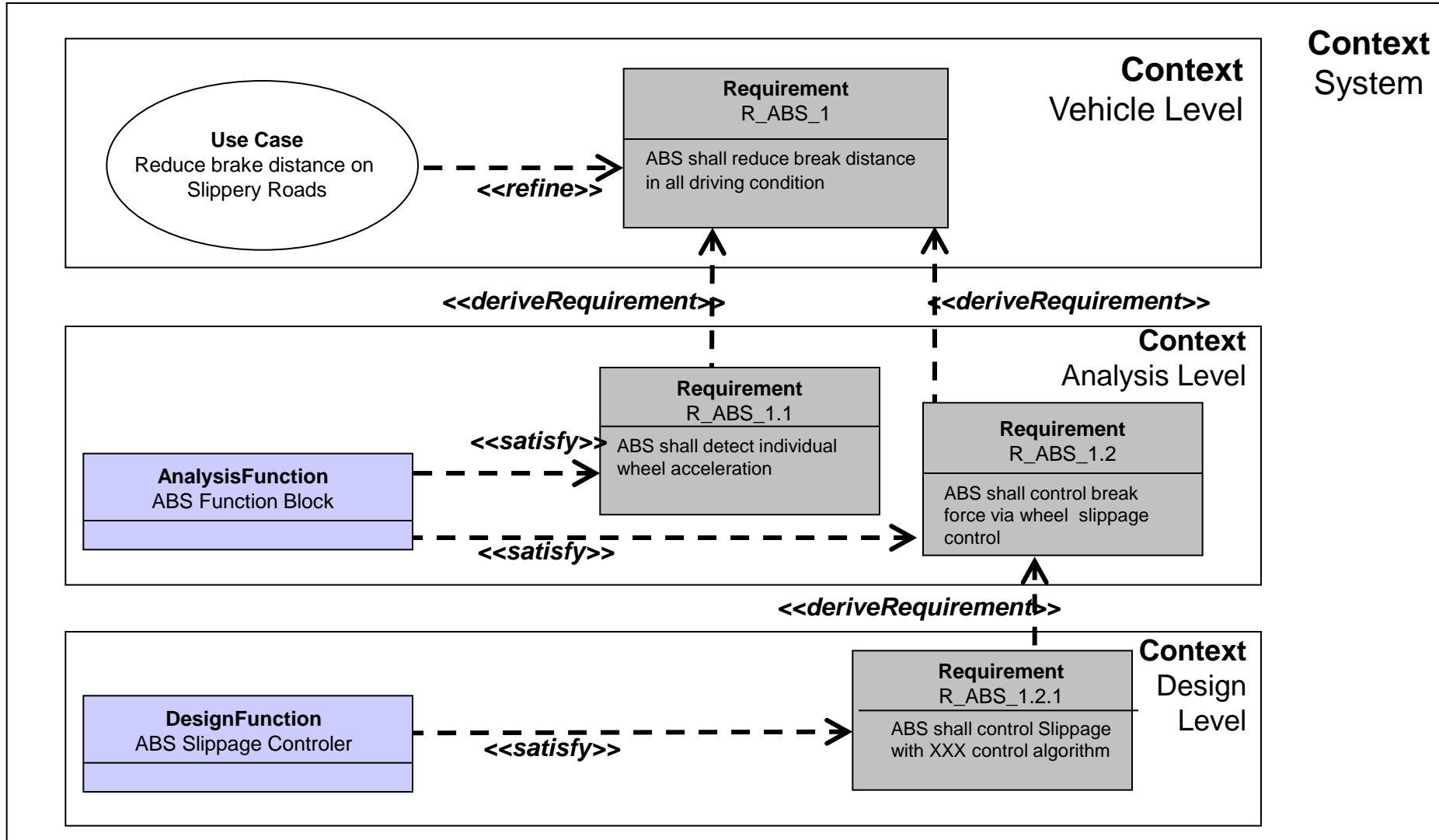
Requirements introduction

Requirements express conditions or capabilities that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed properties.

Requirements and Architecture

- The core EAST-ADL model captures basic structure of system

- Requirement elements annotate model to identify mandatory aspects

- Requirement elements relate requirements engineering information to architectural model

Requirements – basic relations

Formal models that specify
the requirement with more rigor

<<refine>>

System components
which have to
satisfy the requirement

<<satisfy>>

Requirement

<<verify>>

Cases that verify
the requirement

<<deriveRequirement>>

New requirements derived
after a system decomposition
or system refinement

EAST-ADL Introduction: Requirements

Context System

Context Vehicle Level

Use Case Reduce brake distance on Slippery Roads

Requirement R_ABS_1

ABS shall reduce break distance in all driving condition

<<refine>>

<<deriveRequirement>>

<<deriveRequirement>>

Context Analysis Level

Requirement R_ABS_1.1

ABS shall detect individual wheel acceleration

AnalysisFunction ABS Function Block

<<satisfy>>

<<satisfy>>

Requirement R_ABS_1.2

ABS shall control break force via wheel slippage control

<<deriveRequirement>>

Context Design Level

DesignFunction ABS Slippage Controler

<<satisfy>>

Requirement R_ABS_1.2.1

ABS shall control Slippage with XXX control algorithm

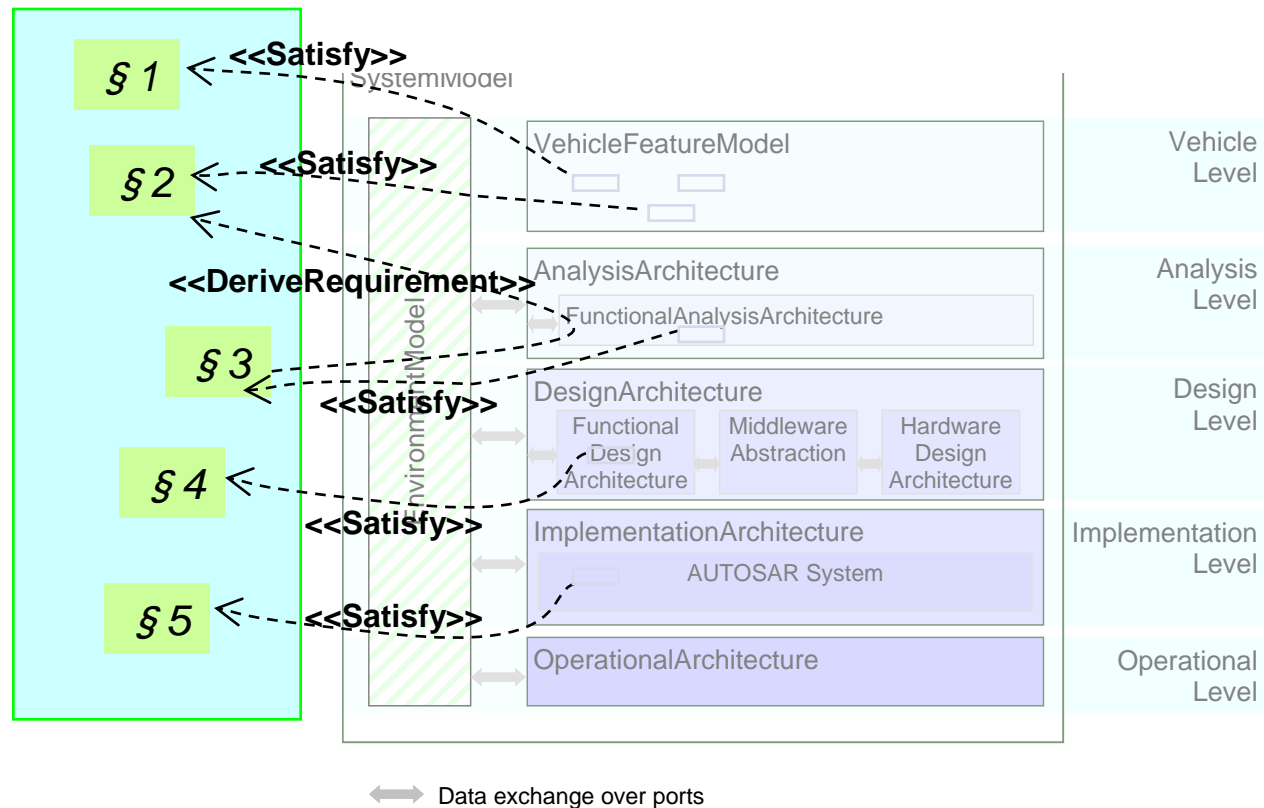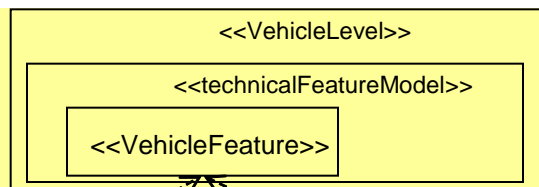EAST-ADL Introduction: Requirements

Requirements

Requirements are normally defined in a central repository.

Satisfy relates requirements to features, functions and components.

Derive relates a derived requirement to its original.

§ 1 <<Satisfy>> SystemModel

§ 2 <<Satisfy>> VehicleFeatureModel — Vehicle Level

<<DeriveRequirement>> AnalysisArchitecture — Analysis Level
FunctionalAnalysisArchitecture

§ 3 <<Satisfy>> DesignArchitecture — Design Level
Functional Design Architecture | Middleware Abstraction | Hardware Design Architecture

§ 4 <<Satisfy>> ImplementationArchitecture — Implementation Level
AUTOSAR System

§ 5 <<Satisfy>> OperationalArchitecture — Operational Level

EnvironmentModel

Data exchange over ports

EAST-ADL Introduction: Requirements

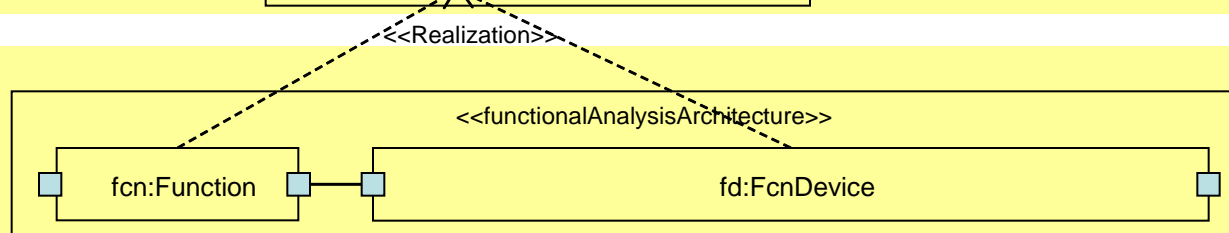# Elements Realizing More Abstract Elements

Vehicle level

<<VehicleLevel>>

<<technicalFeatureModel>>

<<VehicleFeature>>

<<Realization>>

Analysis level

<<functionalAnalysisArchitecture>>

fcn:Function — fd:FcnDevice

Design level

<<functionalDesignArchitecture>>

fcn:Function — ldm:LDM

<<hardwareDesignArchitecture>>

a:ActuatorTF
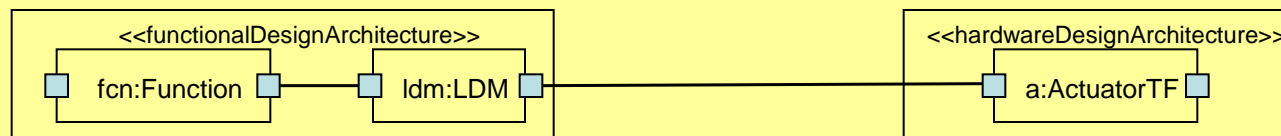
Requirements associated to elements and between abstraction levels

Vehicle level

<<VehicleLevel>>

<<technicalFeatureModel>>

<<VehicleFeature>>

<<Requirement>> ←---- <<Satisfy>> ----

Analysis level

<<DeriveRequirement>>

<<Realization>>

<<functionalAnalysisArchitecture>>

<<Requirement>> ←---- <<Satisfy>>

fcn:Function

fd:FcnDevice

<<DeriveRequirement>>

Design level

<<Requirement>> ←--<<Satisfy>>

<<functionalDesignArchitecture>>

fcn:Function   ldm:LDM

<<hardwareDesignArchitecture>>

a:ActuatorTF

EAST-ADL Introduction: Requirements
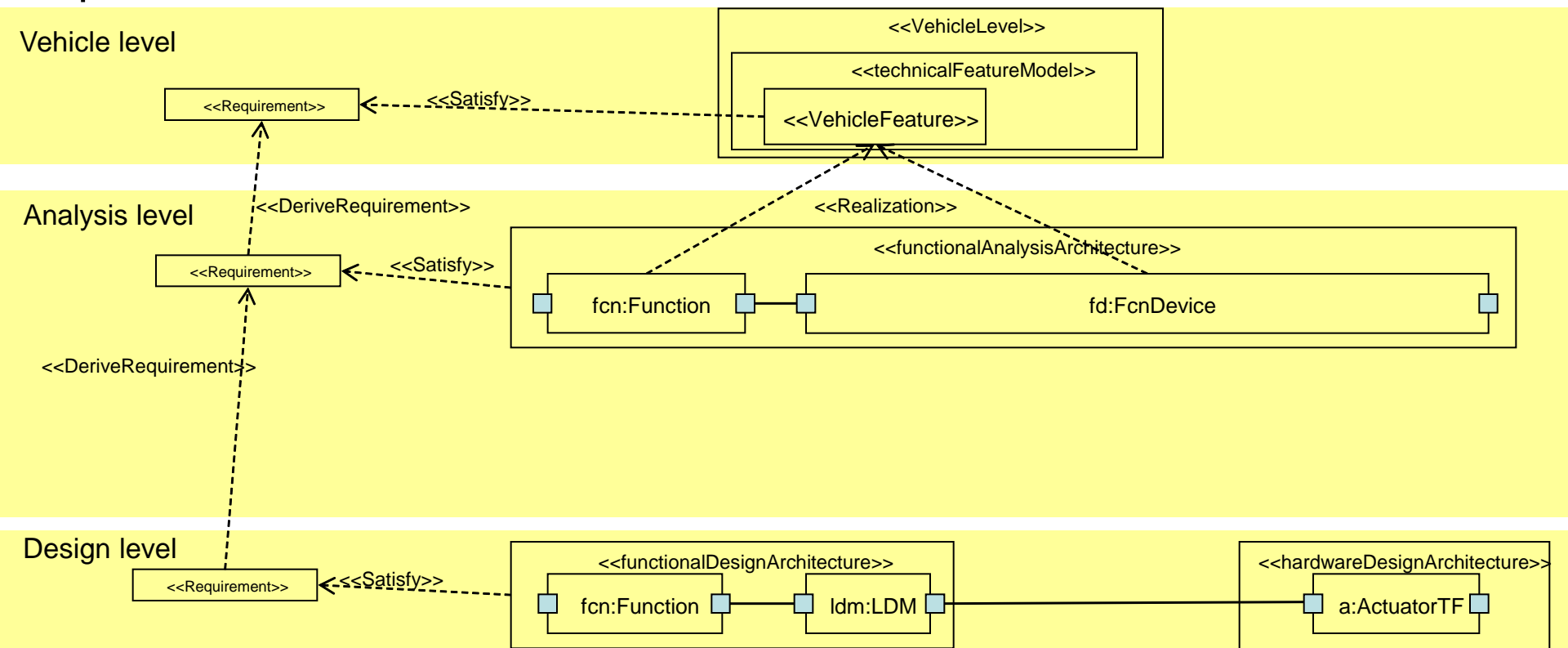
Refining requirements with formalized constraints

Vehicle level

<<VehicleLevel>>

<<technicalFeatureModel>>

<<VehicleFeature>>

<<Requirement>> ⇐ — <<Satisfy>> — — — — — —

Analysis level

<<DeriveRequirement>>

<<Realization>>

<<functionalAnalysisArchitecture>>

<<Requirement>> ⇐ — <<Satisfy>> — —

fcn:Function   fd:FcnDevice

port   port

<<DeriveRequirement>>

<<Refine>>

<<EventFunctionFlowPort>>   <<EventFunctionFlowPort>>

stimulus   response

<<ReactionConstraint>>
jitter = 2.0
nominal = 10.0
mode = startup

scope   <<EventChain>>

Design level

<<functionalDesignArchitecture>>   <<hardwareDesignArchitecture>>

<<Requirement>> ⇐ <<Satisfy>> — —   fcn:Function   ldm:LDM   a:ActuatorTF

<<Refine>>

inFlowPort   outFlowPort

<<EventFunctionFlowPort>>   <<EventFunctionFlowPort>>

<<ReactionConstraint>>
jitter = 2.0
nominal = 10.0
mode = startup

stimulus   response

scope   <<EventChain>>

<<ExecutionTimeConstraint>>
jitter = 2.0
nominal = 10.0
mode = startup

EAST-ADL Introduction: Requirements
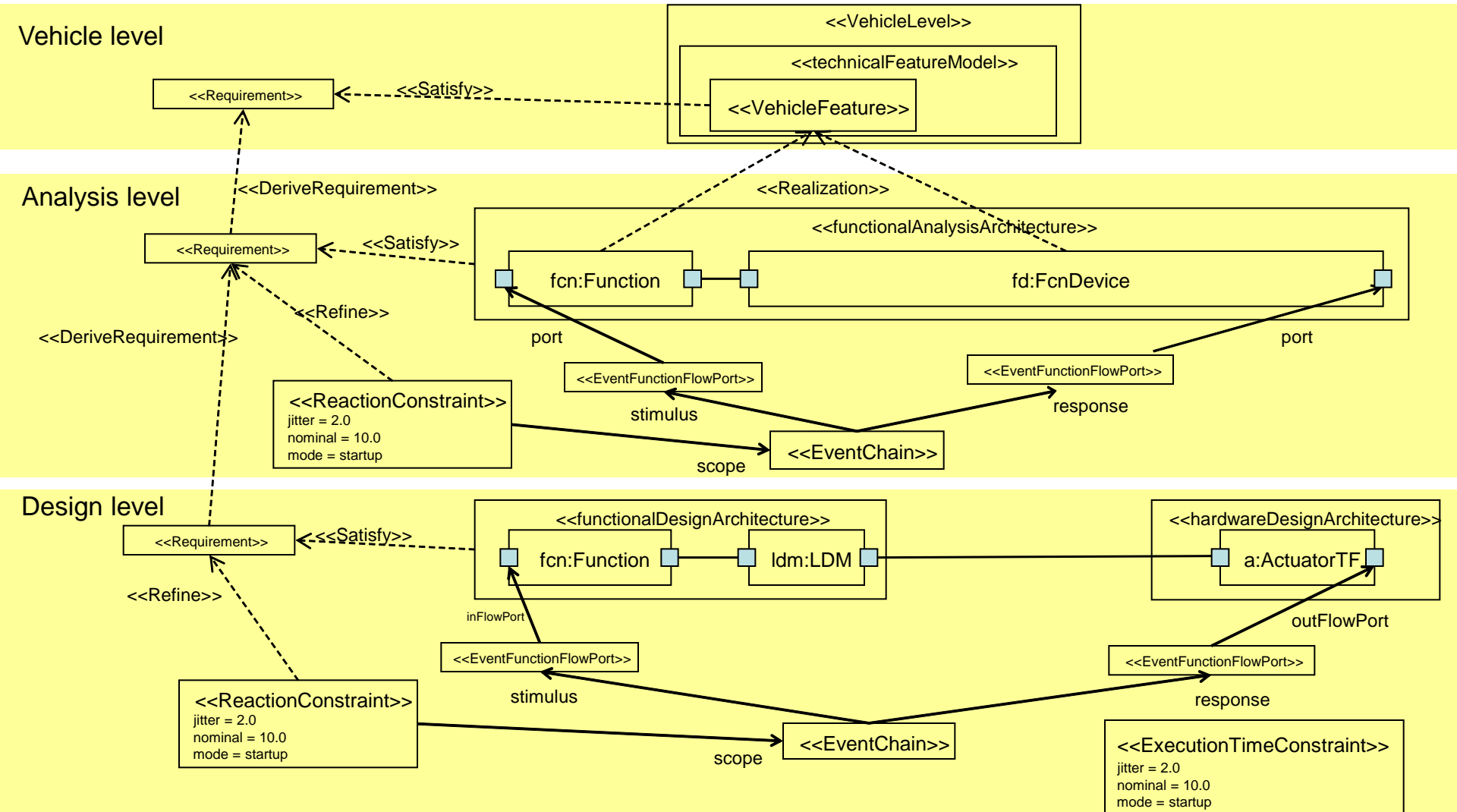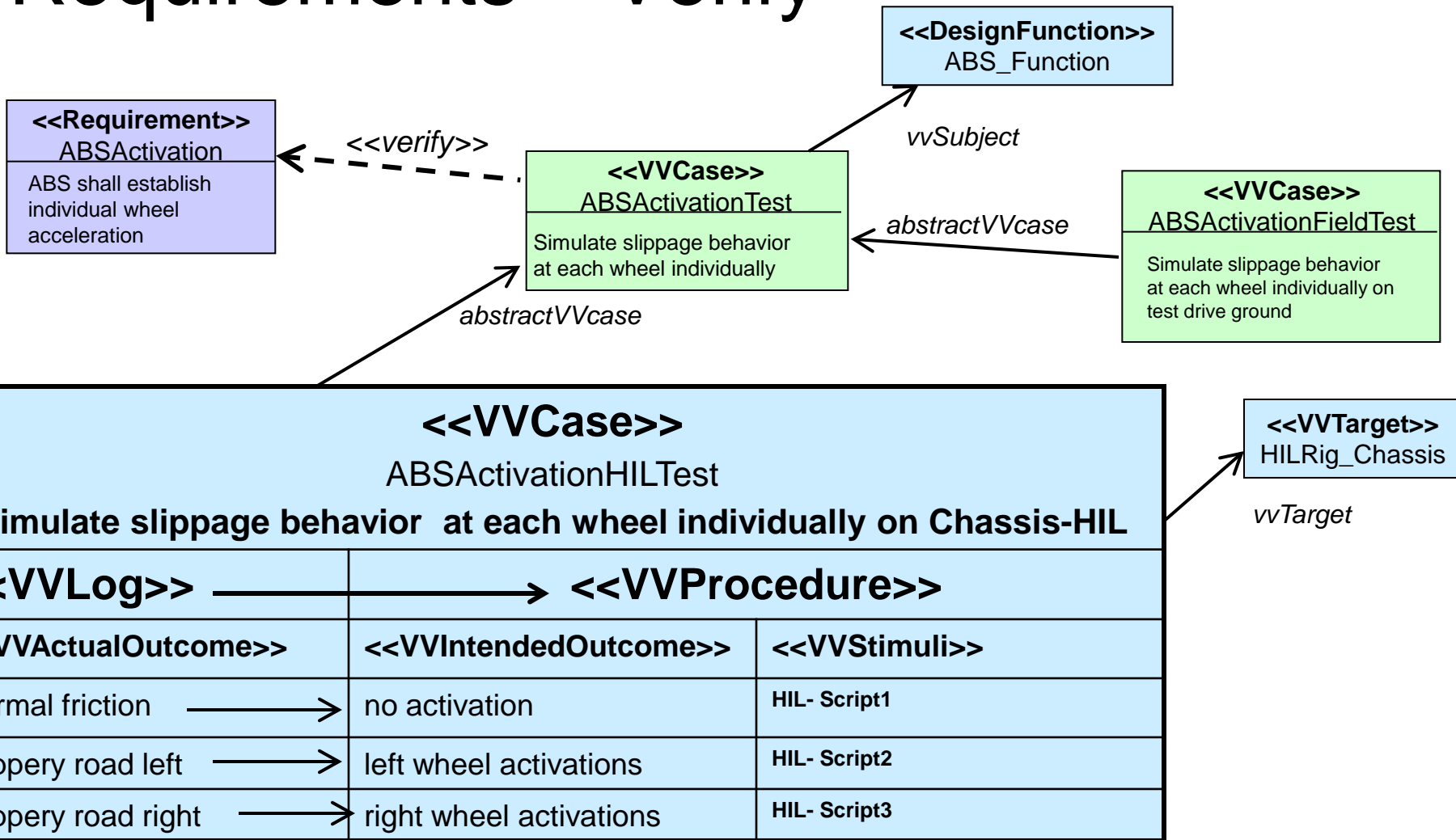
Verification & validation support – basic concepts

- VVCase = a certain, overall V&V effort of varying scope and intention
  - core concept of V&V support in EAST-ADL
  - Safety analysis; specification, design or implementation review; analysis or design level simulation, SIL-testing, HIL-testing, vehicle testing
- VVProcedure = individual task in the context of an overall V&V effort (i.e. a VVCase), which has to be performed in order to achieve that effort's overall objective.
- VVTarget = concrete testing environment in/on which a particular V&V activity (i.e. VVProcedure) can be performed
  - can be physical hardware or pure software (e.g. design level simulations)
- VVLog = captures outcome of an actual execution of a V&V activity

Requirements – Verify

<<Requirement>>
ABSActivation

ABS shall establish individual wheel acceleration

<<verify>>

<<DesignFunction>>
ABS_Function

vvSubject

<<VVCase>>
ABSActivationTest

Simulate slippage behavior at each wheel individually

abstractVVcase

<<VVCase>>
ABSActivationFieldTest

Simulate slippage behavior at each wheel individually on test drive ground

abstractVVcase

<<VVCase>>

ABSActivationHILTest

Simulate slippage behavior at each wheel individually on Chassis-HIL

<<VVTarget>>
HILRig_Chassis

vvTarget

<<VVLog>>	**<<VVProcedure>>**	
<<VVActualOutcome>>	**<<VVIntendedOutcome>>**	**<<VVStimuli>>**
Normal friction	no activation	**HIL- Script1**
Slippery road left	left wheel activations	**HIL- Script2**
Slippery road right	right wheel activations	**HIL- Script3**

EAST-ADL Introduction: Requirements

Summary

- Requirement elements are textual (but may use a formal notation)

- Requirements are linked to architectural elements

- Requirements are linked to derived requirements

- Formalizations can be linked to requirements (Constraints and behavioural models)

- Requirements are linked to V&V constructs

- V&V constructs represent test cases and test results