EAST-ADL Native Behavior Specification

Concept Presentation

2011

Motivations

- To address some fundamental issues in systems engineering
  - **Requirements engineering**
    - Are the requirements and their assumed operation situations correctly stated?
    - Are the requirements consistent/complete in regard to each other or other derived requirements?
  - **Architecture design and specification**
    - How to specify contracts of functions/components in regard to behavior bounds and invariants?
      - e.g. data trajectories, value invariants and transfer equations, states and state transitions.
    - How to specify the impacts of vehicle modes on functions/components and resource deployment?
    - How to support the traceability in regard to behavior concerns from requirements to design solutions at multiple abstraction levels?
    - What are the semantics of feature-links and function realizations?
  - **Analysis for functionality and nonfuntionalities**
    - Do behaviors at different abstraction levels conform to each other? What are the effect of emergent behaviors at a lower level?
    - What are the compostionality and composability of functions/components?
    - Is the system deadlock free according to the chosen execution scheme?
    - How does a system react to faults/failures in combination with nominal stimuli? How to support fault-injection?
  - **Verification and validation**
    - How to derive test cases as well as the coverage criteria?

Motivations (Cont.)

- In particular, the following language support is considered important for FEV (Fully Electrical Vehicles)
  - precise definitions of temporal characteristics for the definition and analysis of safety constraints
  - assessment of completeness and correctness of the safety requirements
  - descriptions of driving profiles, physical dynamics, power management procedures, fault tolerance design
  - generation and precise definition of test cases

- It is seldom the case that a single tool would cover all these issues.
  - EAST-ADL as a common framework for the integration of external mature formalisms and architecture design specification
    - **Declarations and management of architectural concerns** vs **The definitons of analytical models for analysis leverage**

EAST-ADL native behavior specification

- Supporting three categories of behavior constraints.

Attribute Quantification Constraint	relating to the declarations of value attributes and the related acausal quantifications (e.g., U=I*R).
Temporal Constraint	relating to the declarations of behavior constraints where the history of behaviors on a timeline is taken into consideration.
Computation Constraint	relating to the declarations of cause-effect dependencies of data in terms of logical transformations (for data assignments) and logical paths.

- It is up to the users of EAST-ADL, in their particular design and analysis contexts, to decide the exact types and degree of constraints to be applied.

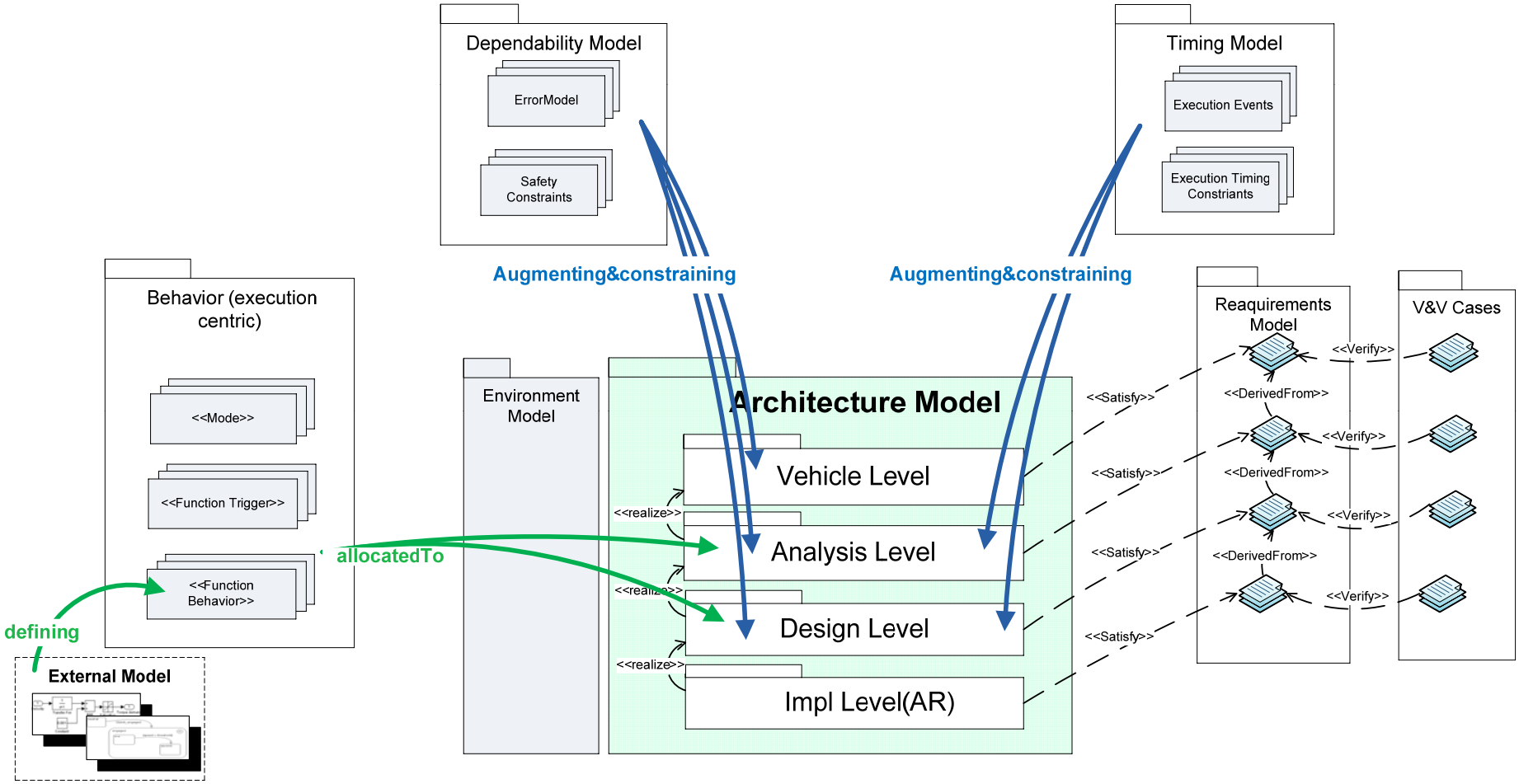
Roles of native behavior constraints - I

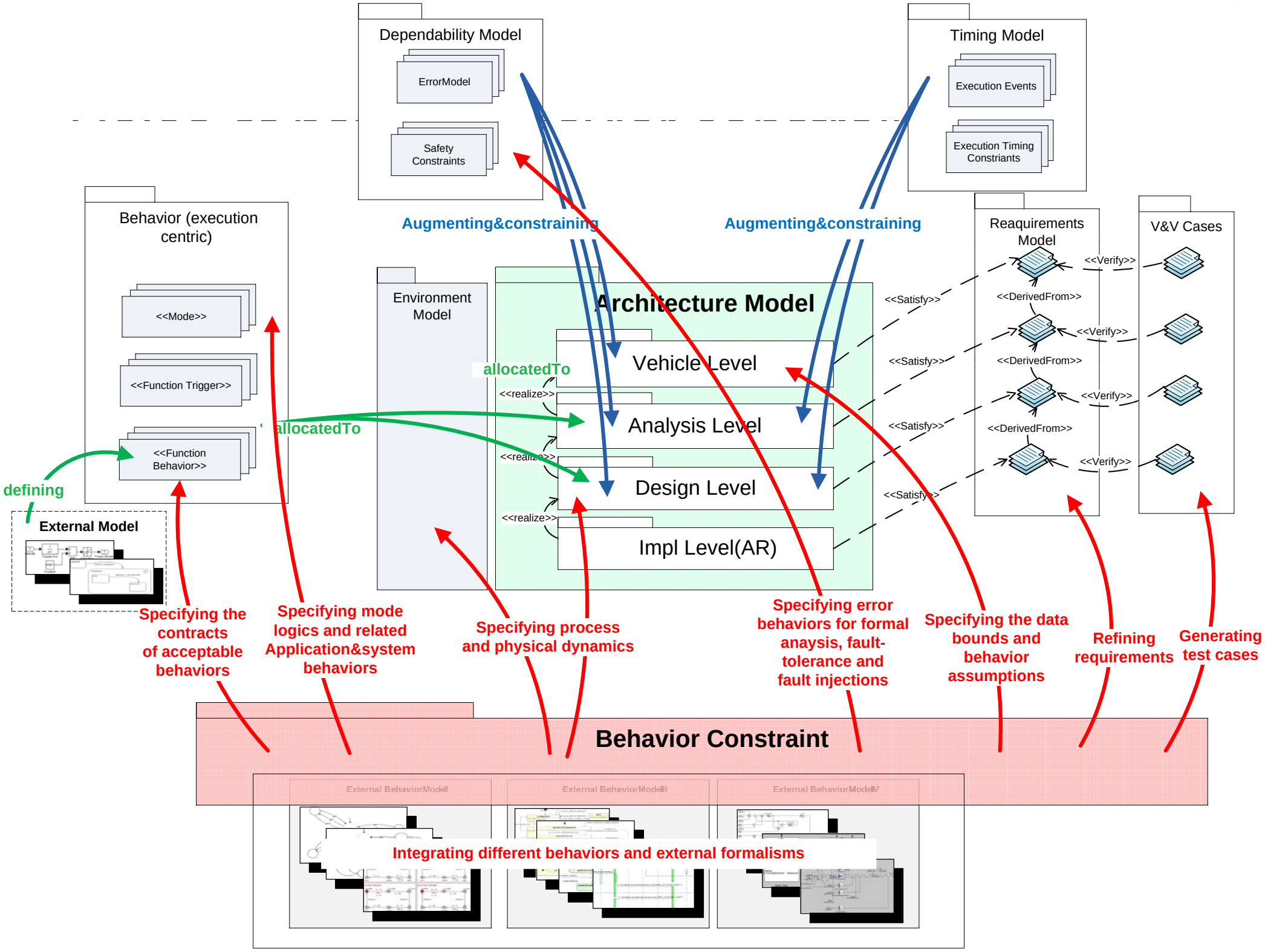
To provide enhanced EAST-ADL support in regard to the following tasks

- Refining textual statements of **requirements** and the assumed **operation situations** for safety engineering and test case generation.
- Specifying **the data and behavior assumptions of vehicle features** for a more precise reasoning about feature configuration.
- Specifying **the contracts of acceptable behaviors of system functions** together with their execution policies.
- Specifying **process and physical dynamics** in environment and hardware platform
- Specifying **mode logics** and the related application behaviors and system services.
- Specifying **faulty conditions, erroneous states and transitions** for fault-tolerance design and fault injections.

Roles of native behavior constraints - II

- An overview of the baseline EAST-ADL support for behaviors and non-functionalities

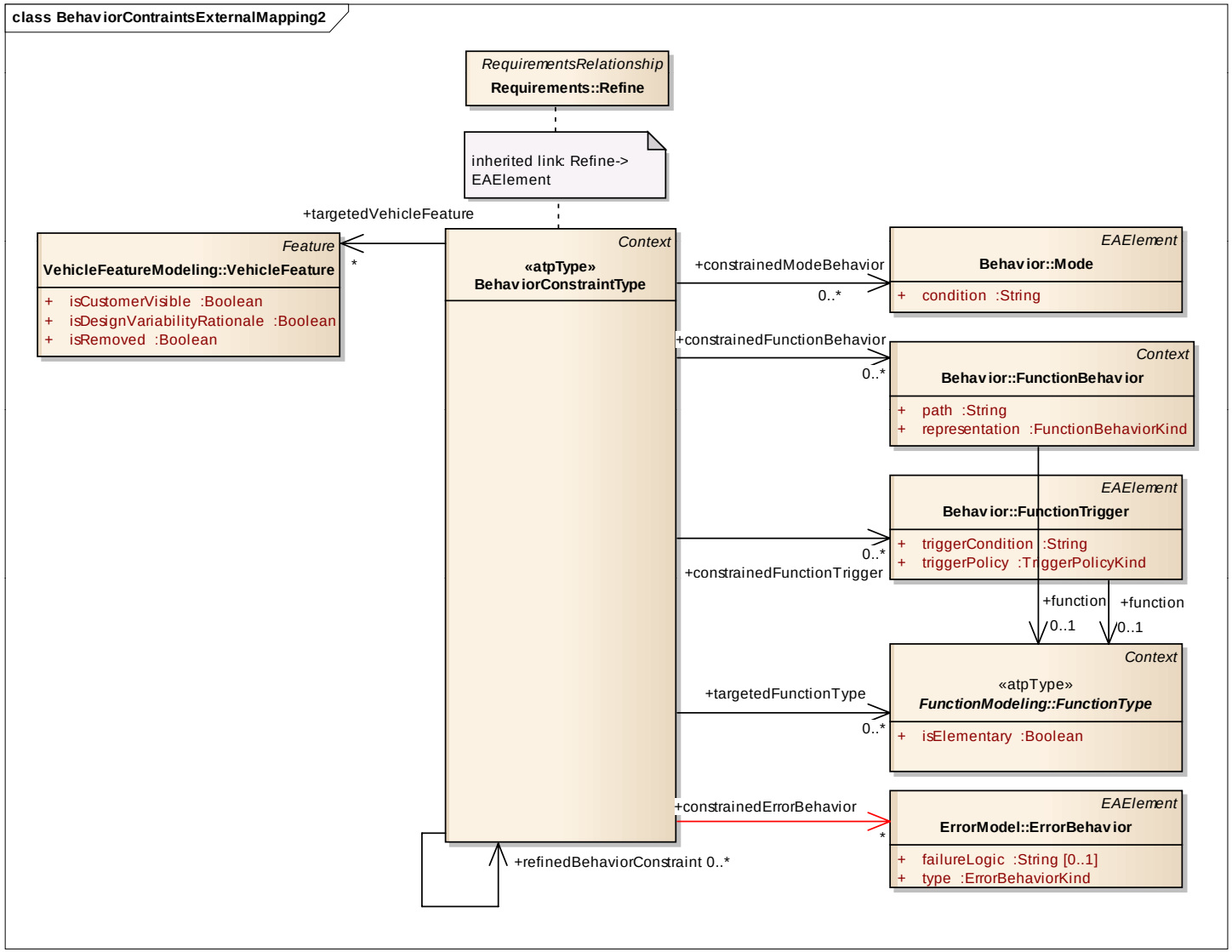



Dependability Model
ErrorModel
Safety Constraints
Timing Model
Execution Events
Execution Timing Constriants
Behavior (execution centric)
<<Mode>>
<<Function Trigger>>
<<Function Behavior>>
Augmenting&constraining
Augmenting&constraining
Environment Model
Architecture Model
Vehicle Level
Analysis Level
Design Level
Impl Level(AR)
allocatedTo
allocatedTo
<<realize>>
<<realize>>
<<realize>>
Reaquirements Model
V&V Cases
<<Satisfy>>
<<Satisfy>>
<<Satisfy>>
<<Satisfy>>
<<DerivedFrom>>
<<DerivedFrom>>
<<DerivedFrom>>
<<Verify>>
<<Verify>>
<<Verify>>
<<Verify>>
defining
External Model
Specifying the contracts of acceptable behaviors
Specifying mode logics and related Application&system behaviors
Specifying process and physical dynamics
Specifying error behaviors for formal anaysis, fault-tolerance and fault injections
Specifying the data bounds and behavior assumptions
Refining requirements
Generating test cases
Behavior Constraint
External BehaviorModelI
External BehaviorModelII
External BehaviorModelV
Integrating different behaviors and external formalisms

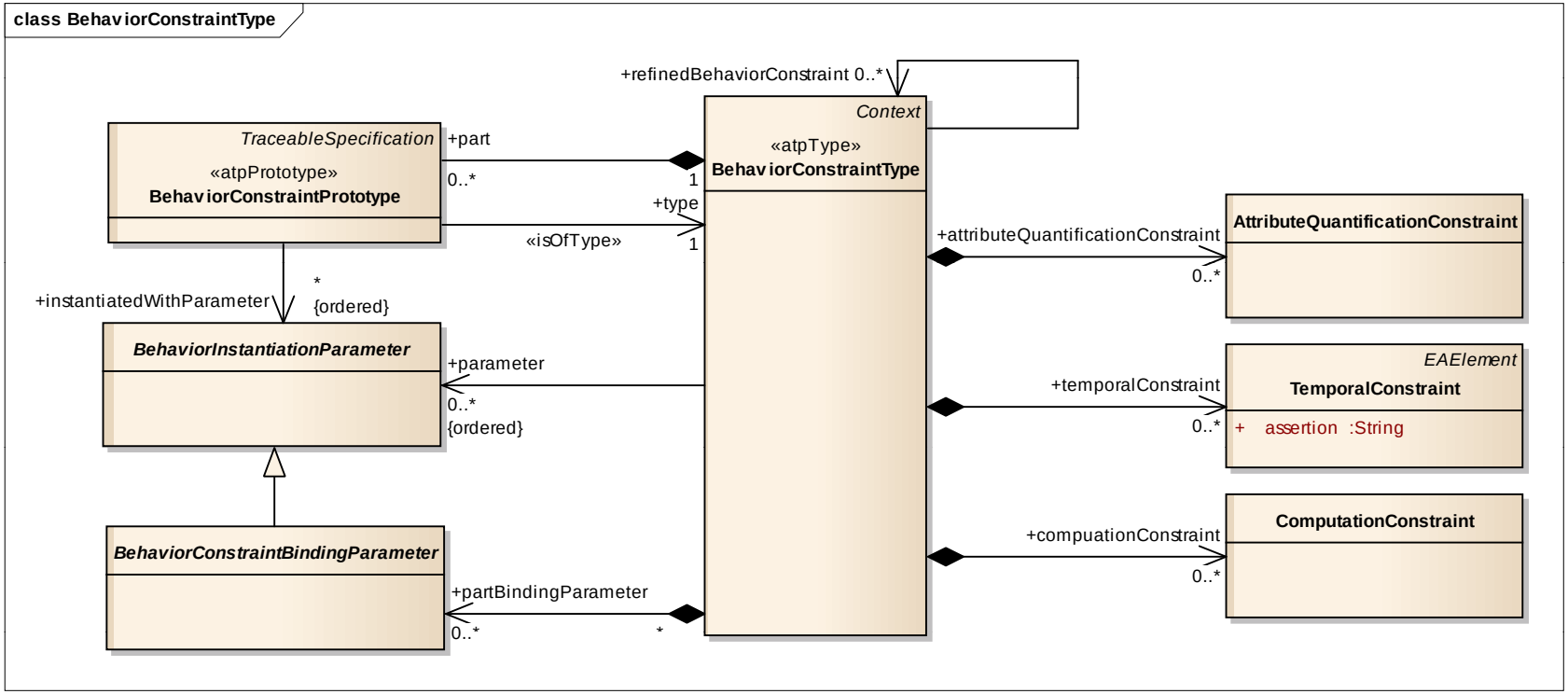
Modeling constructs for behavior constraints and their targets

Behavior constraints for declaring, merging, and tracing, different behavior concerns:

- Functional vs Execution specific
- Nominal vs Erroneous
- Required vs Provided
- Physical vs logical
- Cross-level realization



Modeling constructs for the internal definitions of behavior constraints

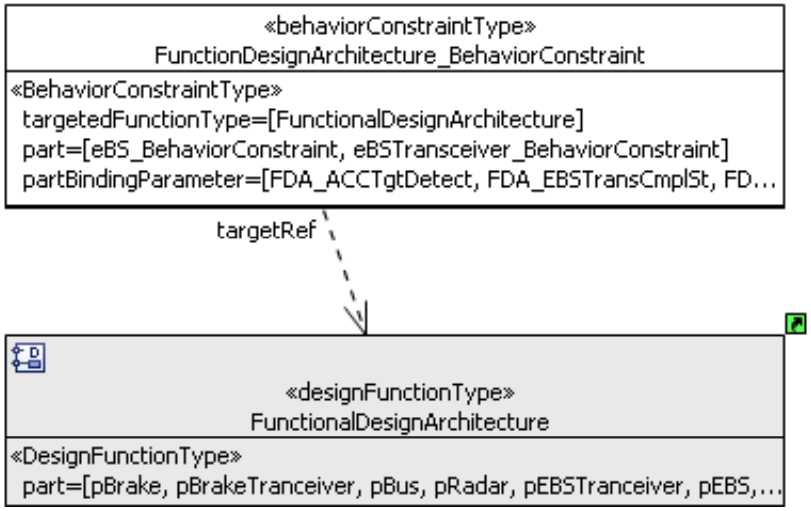


A behavior constraint consists of :

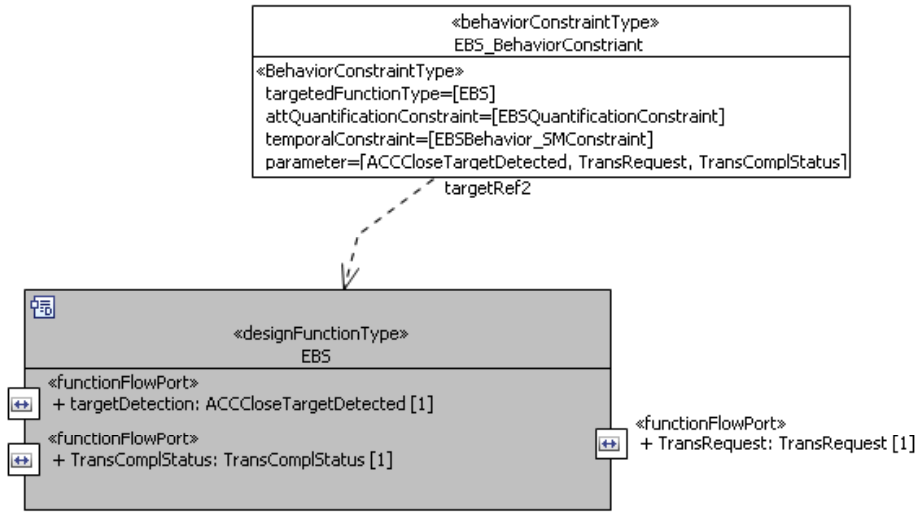
- § Declarations of attribute quantification restrictions;
- § Declarations of temporal restrictions;
- § Declarations of computational restrictions;
- § Declarations of parts and parts-bindings;
- § Declarations of instantiation parameters.

Example – Declaring Behavior Constraints

- Behavior constraint applied to a system, declaring:
  - Structure target
  - Parts
  - Part-bindings



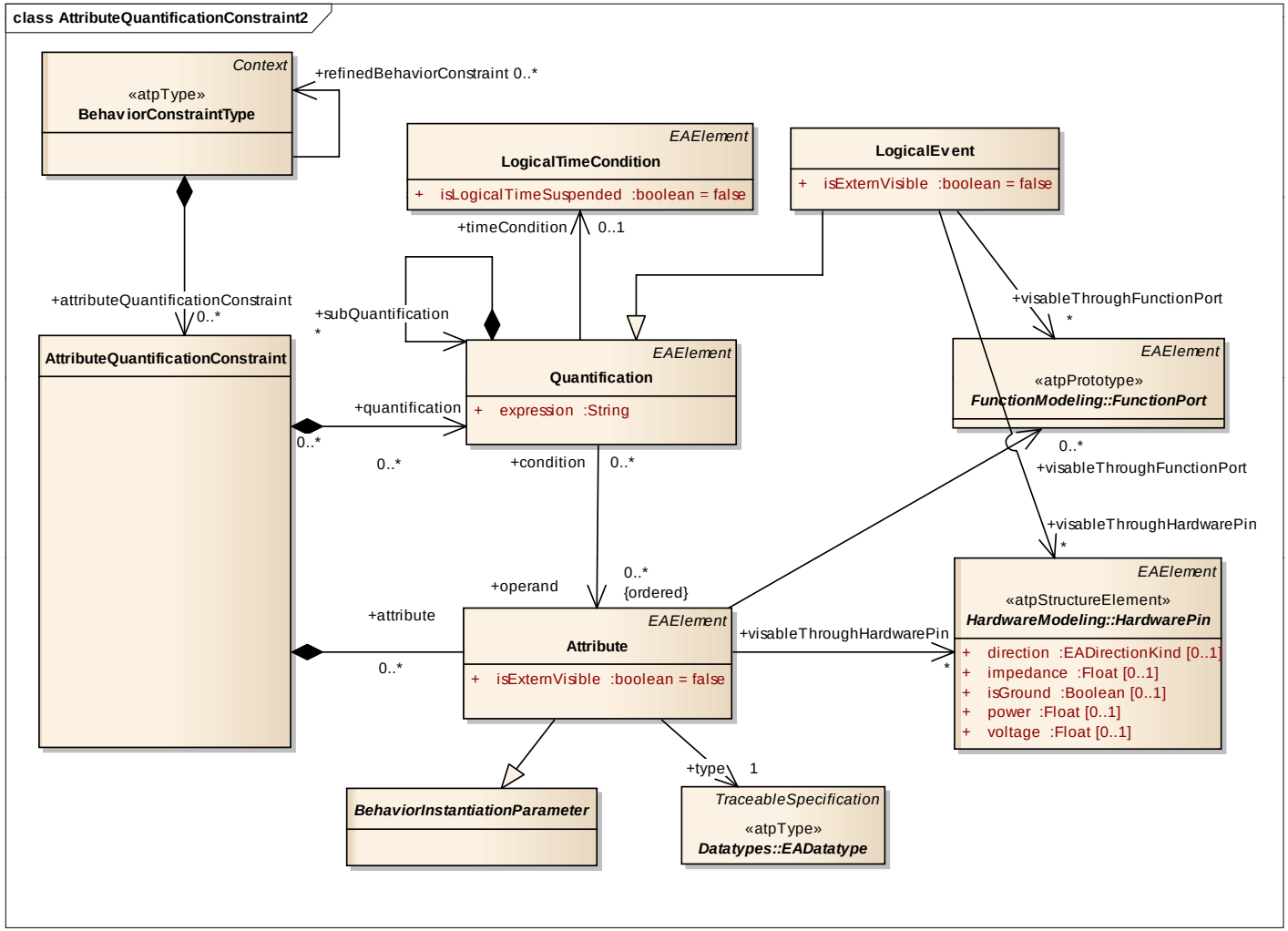
- Behavior constraint applied to a system function, declaring:
  - Structure target
  - Quantification & state-machine restrictions
  - Instantiation parameters



Modeling constructs for (acausal) quantification constraints

A quantification constraint consists of :

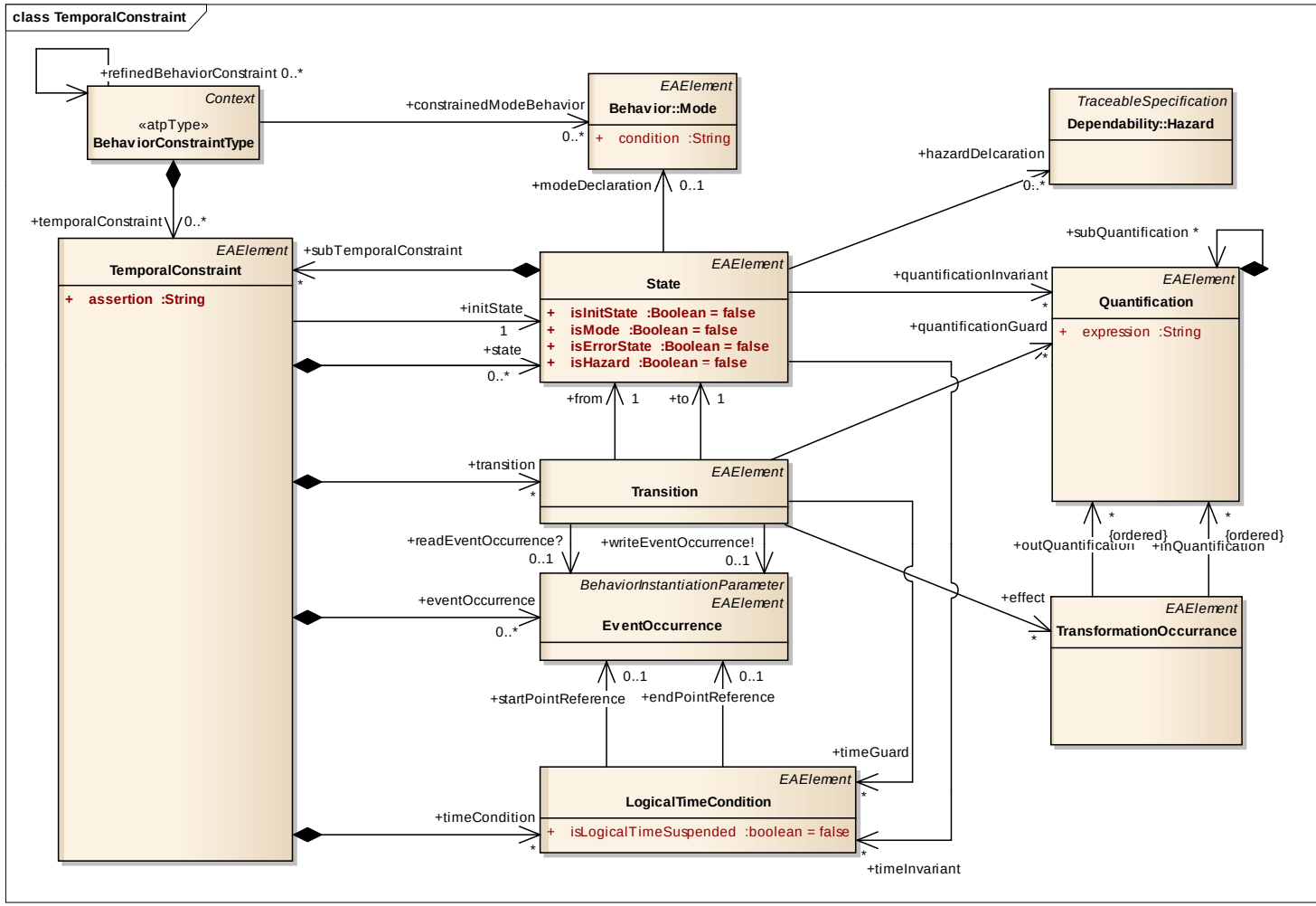
- § Declaration of attributes
  - ❑ data type
  - ❑ corresponding structural elements for external access.
- § Declaration of the quantifications
  - ❑ expressions stating the value bounds, or the logical and arithmetical relations of attributes
  - ❑ related time conditions (e.g. time instances or durations);
  - ❑ any sub-quantification statements
  - ❑ Logical events, which are value conditions that may trigger state transitions when fulfilled.



Modeling constructs for temporal constraints

A temporal constraint consists of :

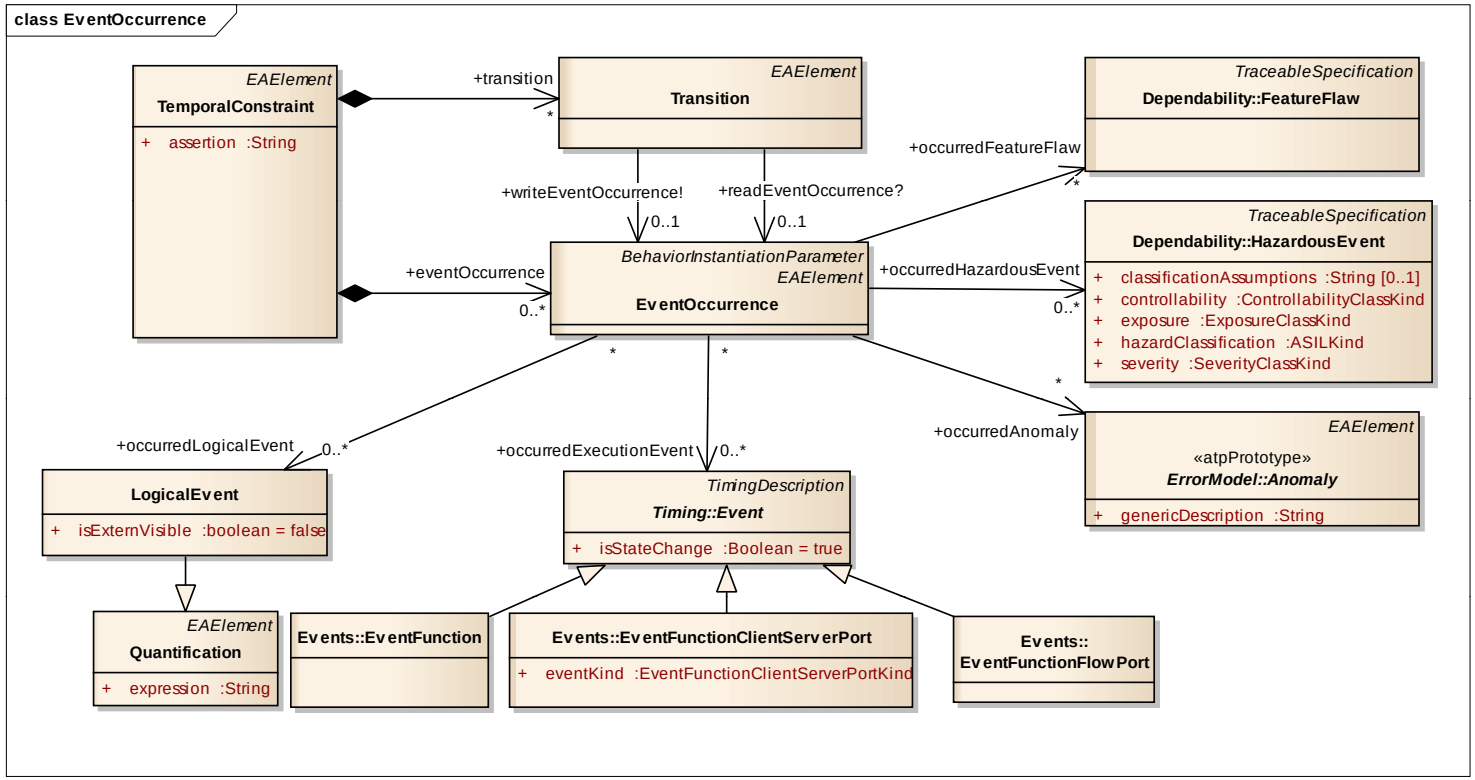
- § Statements of assertions in temporal/modal logics
- § Declaration of states
  - ❑ value invariants (quantifications).
  - ❑ time invariants
  - ❑ Corresponding hazard, mode declarations
  - ❑ Sub temporal constraints
- § Declaration of transitions
  - ❑ Linked states
  - ❑ quantification guards
  - ❑ time guards
  - ❑ read&write event occurrences
  - ❑ effects
- § Declaration of event occurrences
- § Declaration of logical time conditions



Event Occurrences

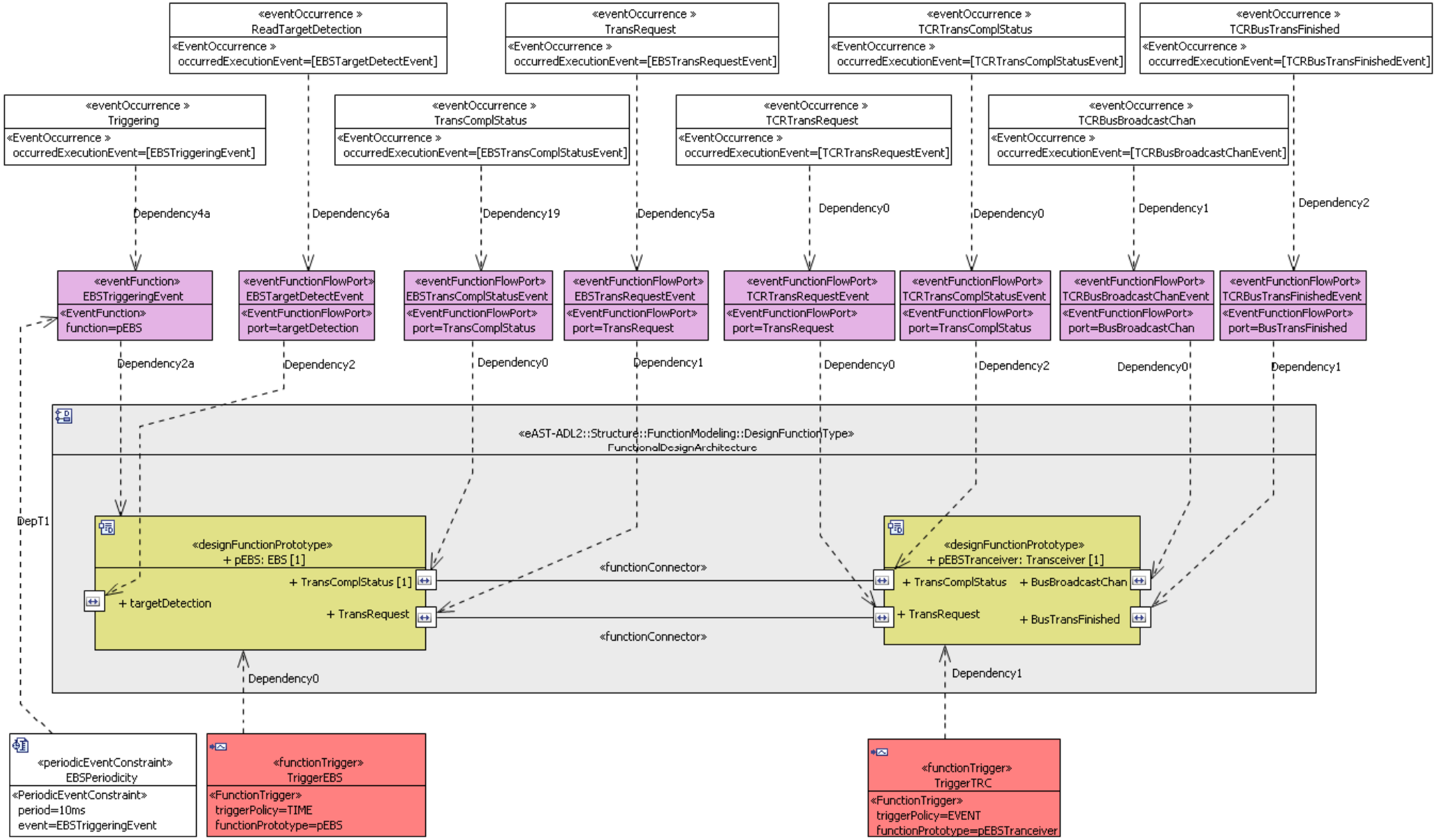
The declarations of events that take place in a running system. Such events can be

- logical events
- execution specific events (in terms of triggering (i.e. EventFunction), data sending & receiving), or
- fault and failure related events (in terms of feature flaws, function anomalies, or hazard events)



- Note: while **events** describe the types and characteristics of condition changes, the **event occurrences** provide support for describing how such condition changes would affect system behaviors when taking place in a running system.

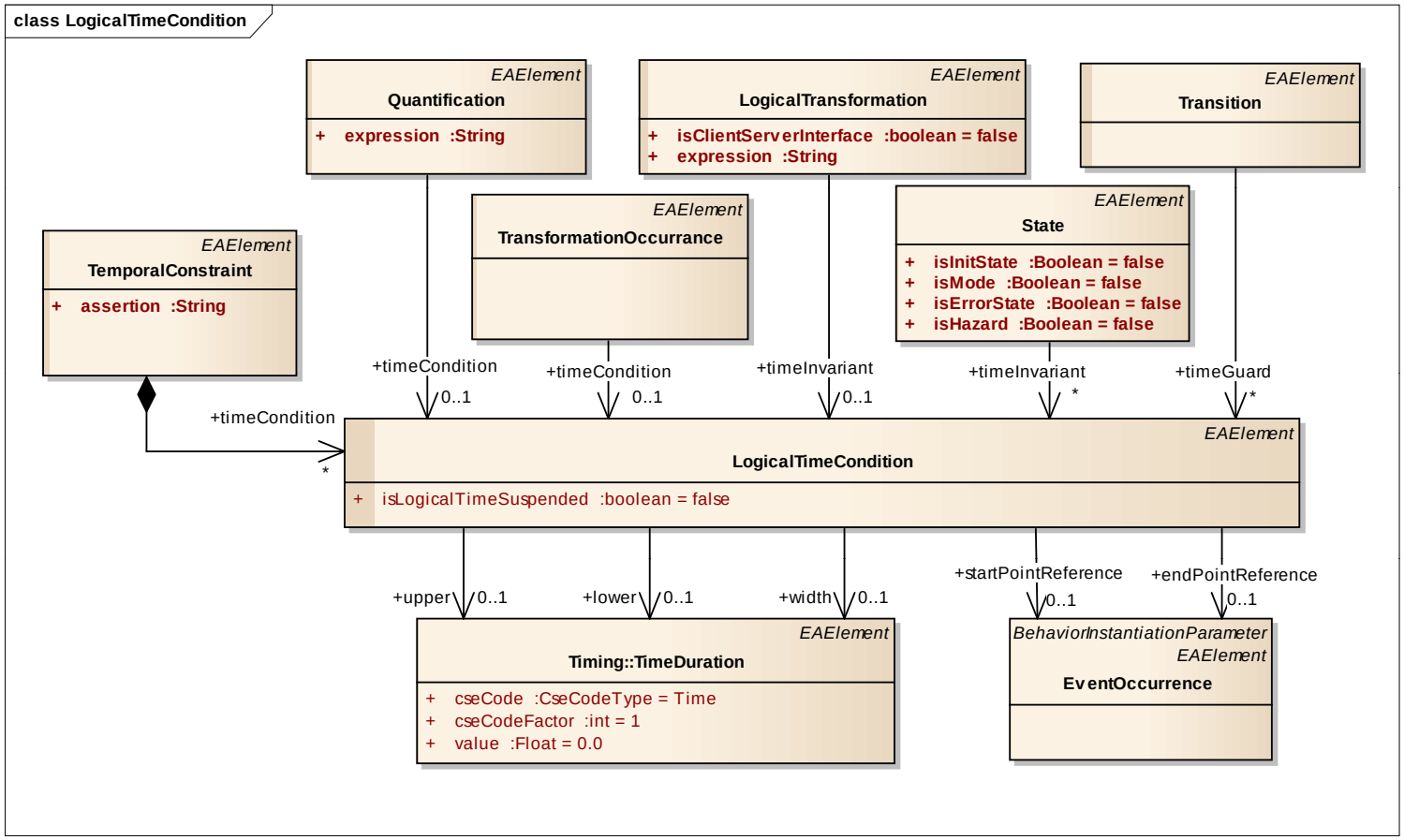
Example – Declaring the occurrences of execution events for two system functions



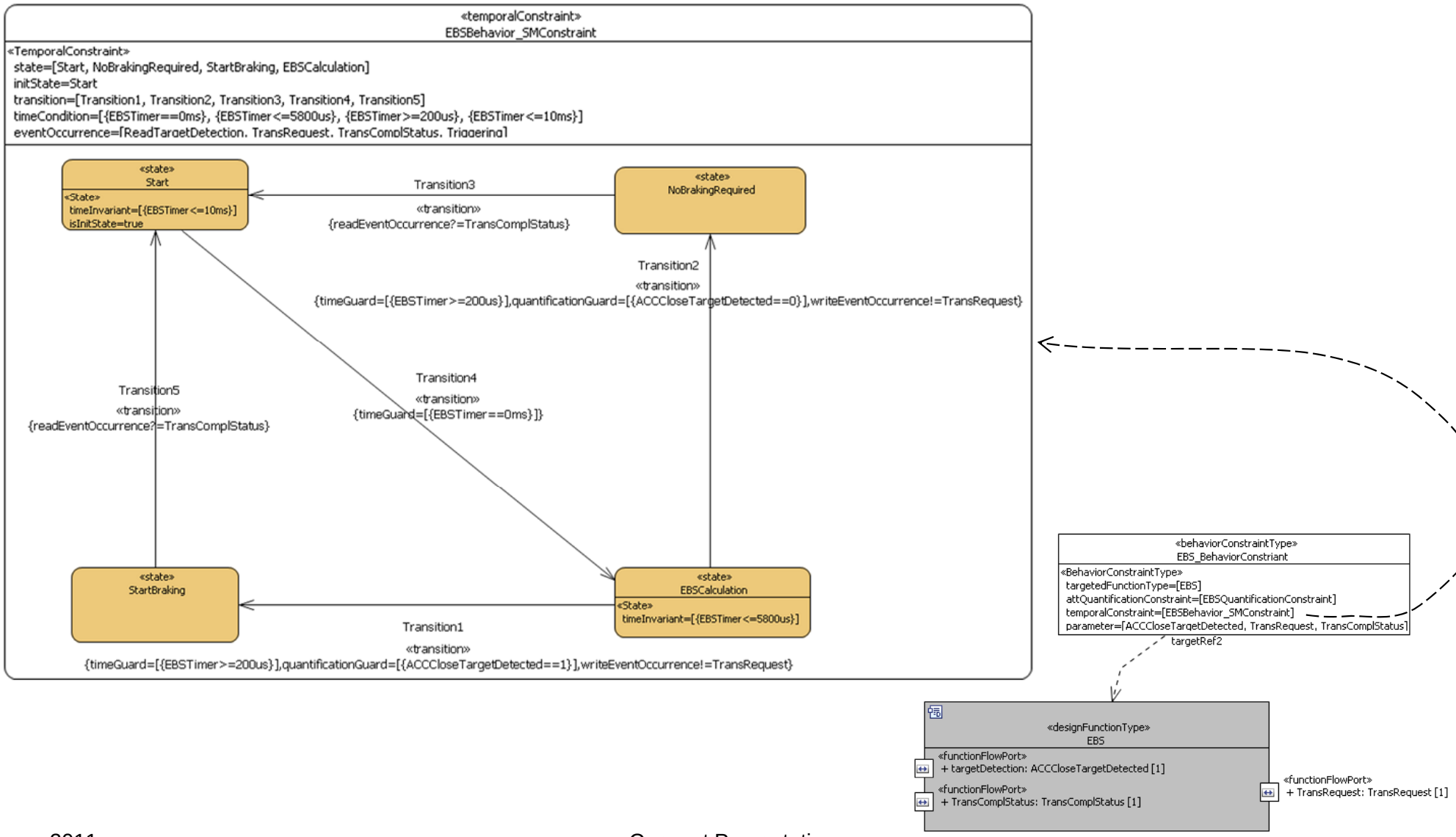
Logical Time Condition

An abstraction of real time for behavior declarations

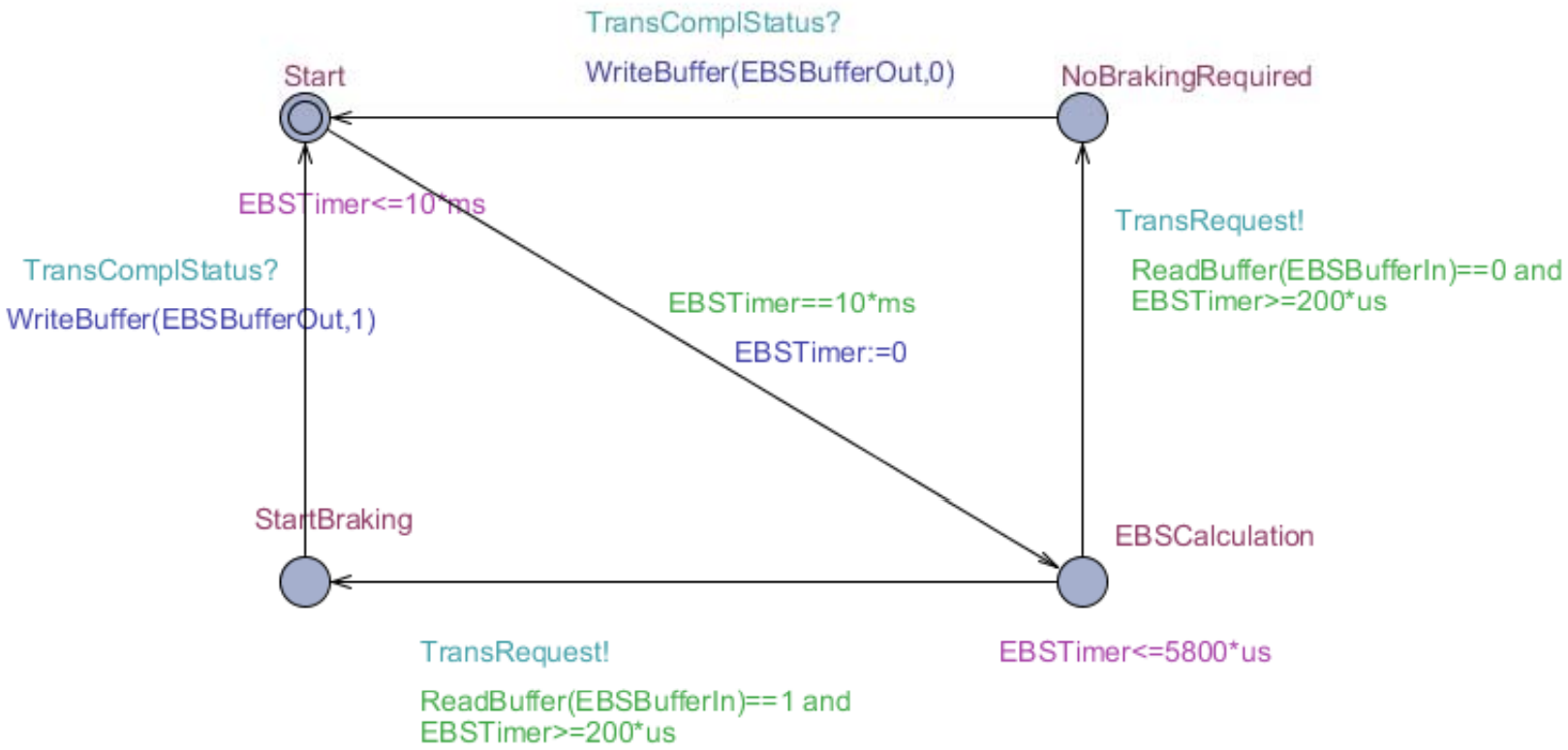
- Based on a time duration specification in the format of CseCode
- Semantics given by the associated occurrences of execution events (e.g., the triggering event of a function).



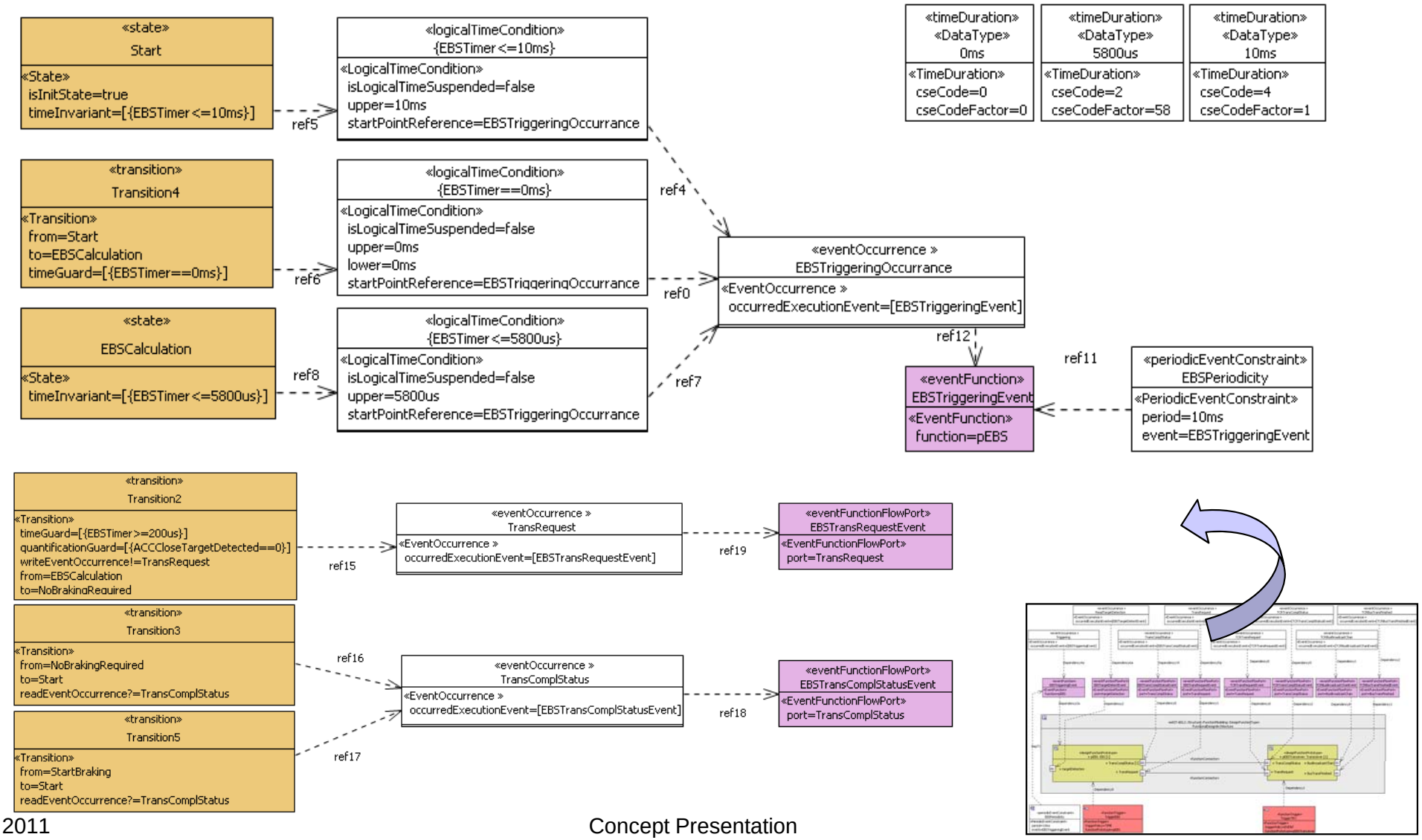
Example – Declaring the temporal constraint of a system function



Example – The corresponding analytical model in UPPAAL



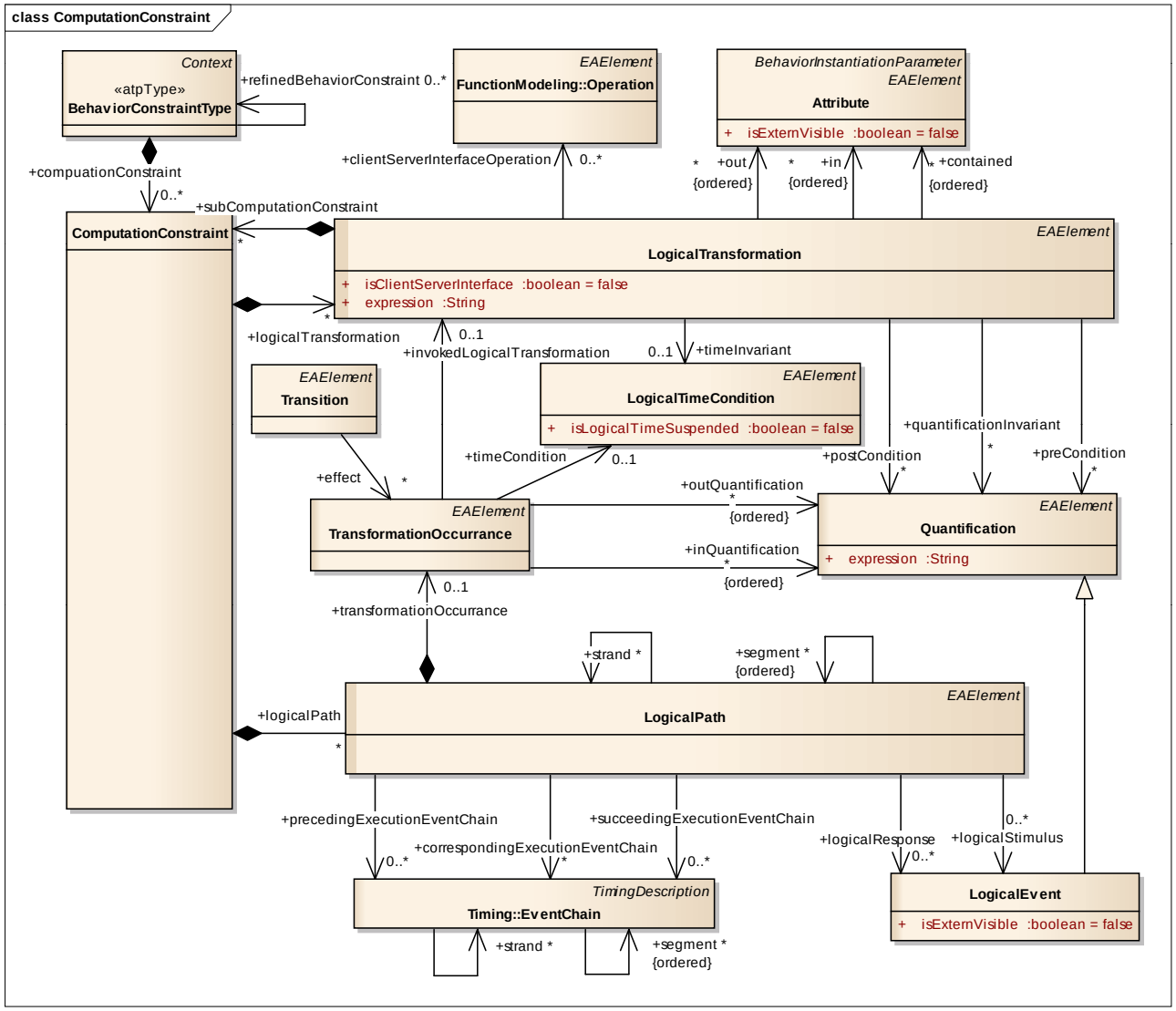
Example – Declaring the related logical time conditions and event occurrences



Modeling constructs for computation constraints

A quantification constraint consists of :

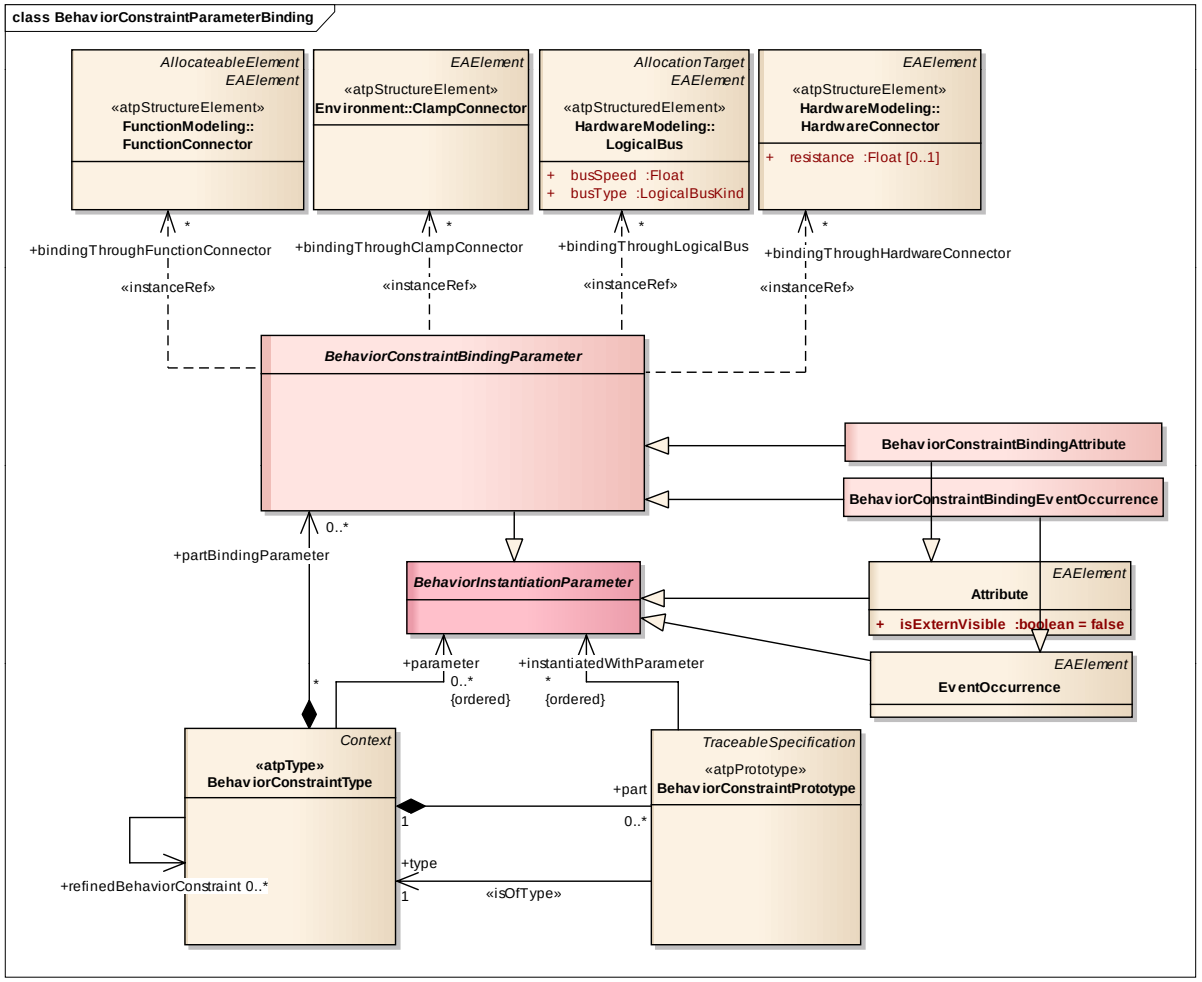
- § Declaration of logical transformations for data processing
  - ❑ out-data (out), in-data (in) and local-data (contained)
  - ❑ value bounds in terms of pre-, post-, and invariant conditions
  - ❑ time invariants (i.e., duration)
  - ❑ any subordinate computation constraints
- § Declaration of expected cause-effect paths/sequences
  - ❑ connecting execution events, logical transformations, and logical events (a merge of the internal causality of functions/components with the related external execution events)
  - ❑ Compositions of such paths in parallel (strand) or in sequence (segment)



Behavior constraint types and their instantiations in prototypes

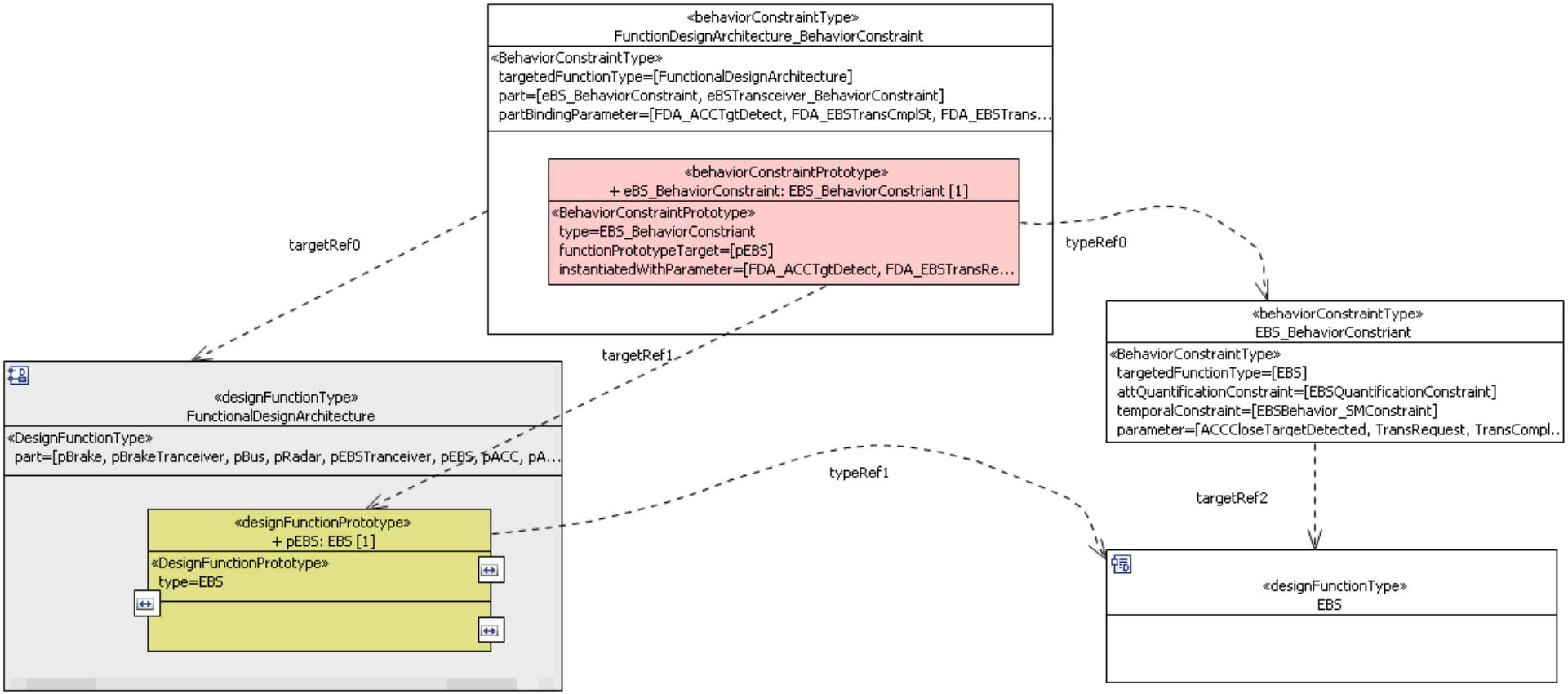
An type instantiation is supported by:

- § Declaration of behavior constraint prototype
- § Declaration of assignments that bind the behavior constraint type's parameters to some contextual parameters (instantiatedWithParameter)



While a type definition provides the template for a range of behaviors, a prototype definition specifies a particular behavior instance in a context

Example – Instantiating a behavior constraint type in an architecture context



Declaring the binding of behavior constraint prototypes in an architecture context